Un modelo para la simulación híbrida de la producción de *software* a medida en un entorno multiproyecto

Departamento de Lenguajes y Sistemas Informáticos

Memoria de tesis doctoral presentada para la obtención del grado de Doctor por la Universidad de Sevilla

Autor:

Javier Navascués Fernández-Victorio

Dirigida por:

Prof. Dra. Isabel Ramos Román

Sevilla, Julio de 2008

Departamento de
Lenguajes y Sistemas Informáticos
Universidad de Sevilla

"The computer comes to feel like an organic extension of your consciousness, and you may feel like an extension of the computer itself."

Donna Haraway. "A Manifesto for Cyborgs: Science, Technology, and Socialist Feminism in the 1980s", 1985.

Índice general

Índice de cuadros

Índice de figuras

Agradecimientos

Son muchas las personas que me han ayudado para llevar a buen término esta tesis. Desde luego, Isabel Ramos, mi directora, y Miguel Toro deben aparecer en el primer lugar por su generosidad, buenos consejos y una elevada dosis de paciencia. Volviendo la vista atrás, es preciso mencionar también a Chema Bueno y Juantxo Larrañeta que me iniciaron respectivamente en los arcanos de la simulación y la investigación operativa, hace ya muchos años. Más recientemente, Pablo Cortés me facilitó ponerme al día en temas que tenía bastante olvidados y Luis Onieva me orientó en un momento de gran incertidumbre.

Manuel Gutiérrez, Kevin Guzmán y Juan Navascués han facilitado extraordinariamente el trabajo para poder contar con datos reales y, especialmente, para comprender los métodos y prácticas que no aparecen en los manuales pero que explican cómo se consigue acabar un proyecto incluso cuando el cliente cambia quince veces los requisitos.

Las personas más próximas son siempre protagonistas de alguna u otra forma de un esfuerzo de esta clase. Manoli y Jaime, mis padres, que han enfrentado y superado muchas veces dificultades ante las que palidece este trabajo, la última muy recientemente. Mi hija Paulita que me ayudó mucho a organizar la bibliografía, además de ofrecerme siempre su cariño. Y Paula, mi compañera, que me ha apoyado en todo momento y me ha aportado más de lo que se puede aquí recoger.

Resumen

La presente tesis propone un modelo de simulación de *la producción de* software *a medida en un entorno multiproyecto*. El hecho de que el sistema que se modela sea multiproyecto es muy relevante según todas las referencias tanto en el campo de la producción de *software* como en el de la gestión de proyectos en general. Esto se debe a que la complejidad inherente a la gestión del desarrollo del *software* se multiplica extraordinariamente en presencia de varios proyectos simultáneos que se realizan por una misma organización que debe repartir sus recursos entre ellos.

Dado que la gestión multiproyecto ha sido estudiada desde hace tiempo por la disciplina de la Gestión de Proyectos (*project management*), es oportuno analizar las aportaciones que ese campo puede hacer al problema que nos ocupa. En este sentido, la tesis emplea metodologías y modelos de la Ingeniería de Proyectos para descomponer jerárquicamente el problema multiproyecto, generar planes en condiciones de limitación de recursos y hacer frente al riesgo y la incertidumbre.

La simulación del proceso *software* es un campo de investigación ampliamente visitado. La disciplina de la Ingeniería del *Software* propone enfoques y metodologías de modelado que se emplean en esta tesis para representar el problema estudiado. Estos resultados hacen referencia al carácter a la vez continuo y discreto de los procesos, a la mejora del proceso *software* y al empleo de técnicas avanzadas para la gestión de los recursos.

A partir de lo anterior, la tesis propone un modelo de simulación híbrida al que se le ha dado el nombre **SimHiProS** cuya finalidad es comprender mejor la dinámica de interacción entre recursos, tareas, tiempo y calidad en este entorno y ayudar a la toma de decisiones de gestión.

El modelo, construido en el lenguaje *Modelica*, permite reproducir diferentes modelos de proceso y realizar experimentos con diferentes supuestos de perturbación de las previsiones iniciales para analizar las respuestas. Dicho modelo se valida con su aplicación a un caso real.

Capítulo 1

Introducción

El objeto del presente trabajo es modelar el problema de la planificación y gestión de la producción de una organización dedicada al desarrollo de *software* a medida para evaluar la aplicación de las técnicas y métodos de la Ingeniería de Organización, en particular la Gestión de Proyectos, y de la Ingeniería del *Software* a dicho problema.

En particular, el enfoque se dirige a organizaciones de tamaño medio, dedicadas al desarrollo, implantación y mantenimiento de sistemas de información bajo pedido. La acotación, por tanto, se produce en dos planos: el producto y el tamaño.

En cuanto al *producto*, el trabajo no tiene en cuenta la producción de *software* denominado "comercial" que, si bien tiene características en común con el segmento seleccionado, presenta especificidades tanto en el proceso productivo como en el ciclo de vida del producto que escapan de la investigación realizada. Lo mismo ocurre con el software especializado de comunicaciones, control de procesos en tiempo real, y otras aplicaciones en las que las similitudes que indudablemente existen no pueden obviar el peso de los determinantes técnicos que la especialidad impone.

En cuanto al *tamaño* de la empresa, el trabajo se enfoca lo que comúnmente se conoce como PYME (pequeña y mediana empresa), dejando a un lado tanto a las "microempresas" (pequeños grupos de programadores, consultores independientes, ...) como las grandes empresas, con más de 300 empleados.

El ámbito objeto del estudio pues, viene a corresponderse con el subsector empresarial de pequeños y medianos productores independientes de *software* de gestión bajo pedido y de almacenamiento, tratamiento y difusión de información. El tipo de producto y el tamaño de empresa fija unos límites al número máximo de productos simultáneos que se pueden estar desarrollando, lo que a su vez permite acotar la dimensión de los problemas a modelar.

1.1. Presentación del problema

El proceso de negocio en las organizaciones objeto de esta tesis puede describirse desde el siguiente esquema básico:

- la organización es, básicamente y en primer lugar, *un conjunto de recursos especializados* con una capacidad productiva, un nivel tecnológico y unos procedimientos operativos determinados,
- la organización capta pedidos a través de su acción comercial, que se convierten en *proyectos* a desarrollar e implementar,
- la organización *planifica la producción de esos proyectos* asignándoles recursos para su ejecución en función de su capacidad, de los compromisos adquiridos con los clientes y de la carga global de trabajo,
- la organización dispone de *un sistema de control* interno a través del cual intenta gestionar la ejecución y modificar las asignaciones para hacer frente a las perturbaciones externas o internas que se produzcan en el curso de la misma,
- la organización lleva a cabo *otros procesos de infraestructura y apoyo al proceso principal* dentro de los cuales es crítica la actualización constante de la capacidad productiva vía el reclutamiento y la formación de nuevos recursos humanos.

Este esquema se ve afectado de características que son específicas del tipo de producto y lo diferencian de otras actividades productivas:

- La primera es que, a diferencia de otros negocios, el resultado final del proyecto, el producto *software*, es un *artefacto evolutivo*, en el sentido de que sus características no están totalmente definidas "a priori" sino que se van conformando a lo largo de su propio ciclo de vida. De este modo, la propia evolución del producto es fuente de perturbaciones que afectan a la definición del proceso pendiente y, por tanto, obliga a la revisión constante de la planificación.

- La segunda, consecuencia de la anterior, es que no es fácil determinar cuando ese proceso se acaba. El momento de la implementación y puesta en operación de un sistema no es nunca el final, sino un hito más en la evolución del sistema. Junto a la corrección de errores imputables a las fases anteriores aparecen nuevas perturbaciones posibles derivadas de la integración con otros sistemas, de la incorporación de bases históricas, de cambios en las operativas de usuario, ...

Como consecuencia de todo ello, los tres objetivos que corrientemente se señalan a un proyecto, *plazos*, *coste* y *calidad*, son mutuamente contradictorios en la dinámica de su propia ejecución.

Para abordar este problema en esta tesis se consideran dos aproximaciones:

- Los modelos normativos: *¿Cómo deben gestionarse los proyectos?* Para ello se considerarán las aportaciones del área de la Ingeniería de Proyectos y de la Ingeniería del *Software*.

- La simulación (modelos descriptivos): *¿Cuál es el efecto de las decisiones de gestión?* Para ello se consideran las soluciones existentes y se propondrán fórmulas para su implementación y mejora.

El resultado final del trabajo realizado es un modelo dinámico de simulación de un sistema de producción multiproyecto sobre el que se pueden evaluar los siguientes aspectos:

- La planificación y el seguimiento de los proyectos desde el punto de vista del cumplimiento de plazos y requisitos.

- La aplicación de políticas dinámicas de asignación de personal y secuenciación de actividades en relación con lo anterior.
- El impacto potencial de la mejora de procesos en la controlabilidad y predecibilidad del sistema.
- Las necesidades de medición y control que todo ello plantea.

1.2. Hipótesis de partida

Las hipótesis de partida de esta tesis, que se justifican en los apartados siguientes, son:

1. Las técnicas y modelos para la asignación de recursos y programación de proyectos en un contexto de restricciones propios de la Gestión de Proyectos - *project managament* - son útiles y relevantes para mejorar los resultados de la gestión de la producción de *software* y, por tanto, para el éxito de los proyectos *software* siempre que se tenga en cuenta el carácter *multiproyecto* del problema en estudio.
2. La implementación de los estándares y modelos de mejora de procesos *software* constituye, además de una guía para posibilitar ese éxito, un conjunto de tareas con contenido de trabajo, coste y consumo de recursos que deben ser incorporadas explícitamente en el análisis de los proyectos como variables a gestionar.
3. La simulación dinámica permite evaluar los resultados de las decisiones derivadas de los dos puntos anteriores siempre y cuando se tengan presentes las características, a la vez continuas y discretas, de la producción de *software*.

1.2.1. La perspectiva de la Gestión de Proyectos

El problema de conjugar plazos, costes y calidad es un problema común en la gestión de proyectos. Desde un cierto punto de vista, el proceso de

producción de *software* a medida puede verse como el de un *proyecto unitario*; un conjunto de actividades y tareas lógicamente ordenadas a la obtención de un fin único. Desde ese punto de vista, la gestión de la producción de este tipo de *software* es fundamentalmente una *aplicación especializada de la gestión de proyectos unitarios*.

Dentro de este campo, el problema crucial es la administración de los *recursos* de la organización por cuyo uso "compiten" varios proyectos concurrentes. Estamos, pues, ante un problema de gestión *multiproyecto* en un entorno de *restricciones temporales y de capacidad*. Esta clase de problemas es precisamente el dominio del área de la secuenciación y programación de proyectos - *project scheduling*- que a su vez es parte de un dominio más amplio, el de los problemas combinatorios.

La Gestión de Proyectos - *project management* - es una de las disciplinas de mayor crecimiento en la actualidad dentro del área de la gestión empresarial. Ello es debido a que la propia idea de proyecto ha trascendido a su interpretación tradicional asociada típicamente a la construcción de un producto o un sistema técnico singular, y se extiende a subconjuntos cada vez mayores de la actividad empresarial hasta el punto de que se habla, a veces, de una *orientación a proyectos* de todos los procesos de trabajo [MBCDH06] e incluso de la *sociedad orientada a proyectos*[Gar01].

La creciente generalización de esta orientación a proyectos da lugar a entidades agregadas y compuestas de proyectos que se denominan comúnmente *programas* y *portafolios* o *carteras de proyectos*. Convencionalmente se denomina *programa* a un conjunto de proyectos que tienen una finalidad común, "un grupo de proyectos relacionados cuya dirección se realiza de manera coordinada para obtener beneficios y control que no se obtendrían si fueran dirigidos de forma individual" [Ins04]. Según la misma fuente, una *cartera* de proyectos es un "conjunto de proyectos cuyo único nexo común es que se desarrollan por una misma organización".

La gestión de programas y de carteras de proyectos plantea problemas adicionales a la simple superposición de los proyectos. Un grado superior de complejidad que se deriva de las dependencias que se establecen, bien por los objetivos en el caso de los programas, bien por la necesidad de hacer com-

patible el empleo de los recursos, en ambas situaciones. En cualquiera de los casos, a ese entorno más complejo lo denominaremos un *entorno multiproyecto*.

En el terreno específico de los proyectos de Sistemas de Información, en un estudio publicado por [WKR04] se analiza mediante una correlación logística la relación entre las desviaciones al alza en plazos y costes y una serie de instrumentos de gestión. La conclusión principal es que la variable que mejor distinguía entre proyectos desviados y no desviados era *la existencia y calidad de los procedimientos de seguimiento y control, incluyendo la anticipación de medidas correctivas*. Le seguían en relevancia las variables relativas a la existencia de métodos ajustados de estimación y el contar con buenas especificaciones y requisitos. El estudio concluye que la inversión en herramientas, técnicas y capacitación especialmente en las áreas citadas es la mejor defensa frente a la variabilidad de los proyectos.

A pesar de lo anterior, investigaciones recogidas en [Her05] indican que el grado de utilización de las herramientas basadas en los modelos de redes provenientes de la Investigación Operativa aún siendo elevado entre los directores de proyecto, *se limita a fines relativamente poco sofisticados* en comparación con el potencial que dichos modelos teóricamente encierran. El uso principal, en el mejor de los casos, es la planificación temporal y posterior seguimiento. En muchas ocasiones *se limita a la representación y comunicación* entre las partes interesadas. Los paquetes *Microsoft Project* y *Primavera Project Planner*, que son los más populares en la profesión, poseen funcionalidades para la gestión de las restricciones ligadas a los recursos y para el equilibrado del empleo de los mismos que raramente se emplean.

Resulta paradójico que el desarrollo de modelos para la programación de actividades bajo diversos tipos de restricciones de recursos es objeto de una intensa actividad de investigación académica a pesar de recibir poca atención por parte de los profesionales. La propia guía del Project Management Institute [Ins04] apenas destina un párrafo de 20 líneas al problema del equilibrado y secuenciación de recursos. De hecho, Goldratt [Gol97] llega a argumentar que los problemas de secuenciación y programación tienen escasa importancia debido a que "*el impacto en la duración de los proyectos es*

mínima".

En el artículo arriba citado [Her05] se argumenta que este tipo de razonamiento es erróneo y conduce a subestimar los problemas ocasionados por los conflictos en el empleo de recursos. Como más adelante se muestra con el caso empleado como referencia empírica en esta tesis, en ocasiones es la *administración de recursos la que explica la realización de los proyectos* y no a la inversa.

1.2.2. La perspectiva de la Ingeniería del *Software*

Una perspectiva complementaria parte de la hipótesis de que en una organización los proyectos están formalmente constituidos por actividades que ofrecen - o pueden ofrecer - un grado significativo de *similaridad* y *repetitividad*. El esfuerzo desde este punto de vista se enfoca en la definición de *proceso* como unidad repetible y, por tanto, mejorable en su articulación interna y susceptible de programarse en relación con otros procesos [Sca01].

Este es el punto de vista de la *Ingeniería de los Procesos Software*, un área de conocimiento del campo de la *Ingeniería del Software* que se orienta a la aplicación de métodos y modelos para conocer, racionalizar y en último extremo mejorar la gestión de las actividades que forman parte del *ciclo de vida* de un producto *software*. Desde el punto de vista que nos interesa, se trata de una aplicación a la Ingeniería del *Software* del paradigma de la reingeniería de procesos y de la mejora continua [FBS+01].

El interés por estudiar el proceso *software* nace de la necesidad de mejorarlo. La idea que subyace es que un proceso mejorado da lugar a un mejor producto. Un éxito que sólo se basa en la disponibilidad de unas personas específicas no constituye una base sólida para mejorar la productividad y la calidad a largo plazo en todo el ámbito de la organización.

Para extender a toda la organización esta capacidad, el paradigma vigente en materia de gestión de la calidad se basa en el supuesto de que *la calidad del producto está directamente vinculada a la calidad de los procesos que conforman su ciclo de vida y esta*, a su vez, *a la madurez de la organización* que los desarrolla.

En este campo, los resultados más operativos son los *estándares de madurez*. Los estándares de madurez en la producción de *software* son tecnología de dominio público a escala internacional difundida a través de diferentes instituciones y aplicada con carácter general, en particular en grandes organizaciones. La aplicación de estos estándares a las PYMEs es objeto de investigación, tanto mediante el estudio de casos como mediante las construcción de propuestas metodológicas que tengan en cuenta las características específicas de cada tipo de empresa [CMPP03, LY01, Ric01].

Estos estándares, orientados a las capacidades a nivel de organización (CMM y CMMI)[1] o a nivel de proceso (ISO/IEC 15504)[2], proporcionan objetivos genéricos para la mejora de la calidad de los procesos y guías para su evaluación, si bien no indican vías unívocas para su puesta en aplicación en un caso concreto. La razón de ello se en que la gran variedad de tecnologías, modelos de desarrollo y tipologías organizacionales existentes determina el carácter dependiente del camino (*path dependent*) de las estrategias posibles de mejora, incluso aunque se utilice un modelo estándar como referencia.

1.2.3. La simulación de proyectos *software*

La aplicación de modelos de simulación al estudio de diversos problemas relacionados con la gestión de los proyectos de desarrollo de productos *software* es un campo al que se dedica un significativo esfuerzo de investigación. La razón para ello es doble: de una parte la necesidad de hacer más predecible y controlable una actividad que cada vez tiene más importancia; de otra, la naturaleza compleja y no lineal del problema que dificulta el empleo de modelos analíticos [Sca01].

En un artículo publicado en 1999, como conclusión del primer taller PROSIM, Keller, Madachy, Raffo [KMR99] proponen una visión panorámica del

[1] CMM - *capability maturity model*- y CMMI-*capability maturity model integration*- se definen como una "aproximación a la mejora de procesos que proporciona a las organzaciones los elementos esenciales de procesos efectivos". Han sido elaborados por el *Software Engineering Institute* de la Universidad Carnegie Mellon [Pau95].

[2] La ISO/IEC 15504 es una norma de la Organización Internacional de Normalización (ISO) y la Comisión Electrotécnica iInternacional (IEC) que proporciona un "marco para métodos de evaluación" para medir la capacidad de los procesos [Rou03].

trabajo realizado en la simulación de procesos *software* basada en tres preguntas *¿Con qué objetivo modelar? ¿Qué modelar?* y *¿Cómo modelar?*

Objetivos de la simulación: ¿Para qué modelar?

Entre los objetivos se señalan los siguientes:

Gestión estratégica de los procesos *software*. Los modelos de simulación permiten, según los autores, recoger las consecuencias que la forma general de organizar el proceso *software* tiene en la evolución estratégica de organización desarrolladora, en términos de creación de valor, posicionamiento de mercado, etc. Así, las prácticas de deslocalizar, subcontratar, desarrollar en base a componentes o mediante aplicaciones "ad hoc", etc. pueden evaluarse en términos de su impacto en el posicionamiento general de la empresa.

Planificación de los procesos. La simulación posibilita también la planificación proporcionando estimaciones de carga de trabajo y esfuerzo, utilización de recursos, análisis de riesgos, etc. La planificación no se limita a simular en el momento previo al inicio de un desarrollo sino que puede ayudar al seguimiento y evaluación, así como a eventuales cambios en la planificación.

Gestión operativa de los procesos. Con un nivel de granularidad más fino, los modelos de simulación pueden también servir para la gestión operativa. Así los modelos de simulación por el método de Monte Carlo pueden servir para controlar desviaciones, estimar tasas de errores y necesidades de corrección, etc. A su vez, los modelos basados en la integración de sistemas de ecuaciones diferenciales pueden servir para evaluar el resultado de las decisiones y las reglas de gestión "ex-ante" a través del análisis de las propiedades de los bucles de realimentación.

Mejora de procesos y adopción de tecnología. Las decisiones de modificar los procesos o adoptar determinadas herramientas u otros elementos pueden ensayarse sobre modelos de simulación antes de ponerse

en práctica, midiendo las consecuencias inmediatas pero también las consecuencias de segundo orden que pueden derivarse, sin incurrir en los costes de una decisión errónea.

Comprensión de los procesos. La construcción de un modelo de simulación y el análisis de escenarios y resultados implica por si misma un esfuerzo por desentrañar los mecanismos que actúan en la práctica del proceso *software* del cual se puede obtener un mayor grado de comprensión de la realidad, descubriendo por contraste con los datos que la misma arroja los aspectos que permanecen ocultos.

Entrenamiento y aprendizaje. Por último, los modelos de simulación son un instrumento potencialmente utilizable en el aprendizaje y entrenamiento en un área de conocimiento donde la experimentación puede resultar prohibitiva.

Todos estos aspectos se refuerzan si se tiene en cuenta que los modelos analíticos que se emplean para estudiar los procesos presentan el inconveniente de que, en general, resultan demasiado abstractos y obvian las características específicas de cada organización de desarrollo. Estas no dejan de ser sistemas sociotécnicos[3] complejos donde a veces los datos numéricos no reflejan más que muy parcialmente la realidad, no se pueden aplicar técnicas estadísticas y a veces no hay manera de entender las desviaciones entre lo predicho por el modelo analítico y lo que ocurre en la práctica, ya que el modelo analítico no ofrece una buena guía para ello.

Alcance de la simulación: ¿qué modelar?

Desde el punto de vista del alcance, los modelos de simulación pueden abarcar desde una parte del ciclo de vida de un producto concreto hasta la evolución de la organización a largo plazo, pasando por un proyecto, varios

[3]El término sistema sociotécnico - *sociotechnical system* - se debe a Emery y Trist, del Tavistock Institute de Londres, y se empleó por primera vez para designar la interación hombre-máquina en un entorno de producción industrial. Hoy en día se ha extendido su alcance para abarcar las interaciones entre las tecnologías y las personas, [ETT74].

proyectos o la evolución en el tiempo de una línea o líneas de producto a través de sus sucesivas versiones.

Desde el punto de vista de los resultados perseguidos, un modelo de simulación puede informar sobre esfuerzo empleado, costes, recursos asignados o tasas de error hasta variables más globales relativas al valor conseguido, la evolución de las plantillas de personal o el nivel de calidad global. De manera muy particular, los modelos de simulación pueden informar sobre el riesgo en los diferentes niveles expuestos en relación con el alcance del modelo; desde el de no cumplir un hito hasta la exposición global al riesgo de la empresa en el mercado.

Para ello los modelos de simulación utilizan diferentes niveles de abstracción de los procesos modelados. Éste dependerá del interés que guía el modelo. Así se pueden identificar tareas y recursos clave, o enfatizar el flujo de objetos a lo largo del proceso, las interdependencias entre tareas, las relaciones y bucles de realimentación, etc.

Para obtener resultados relevantes es igualmente necesario proporcionar los parámetros de entrada adecuados, que pueden ser medidas de tamaño, productividad, ... o parámetros de los procesos organizativos como el ciclo de contratación de personal, los retrasos en la incorporación efectiva de éste, los intervalos de medida y (re)programación, etc.

Metodologías de simulación: ¿cómo modelar?

Las metodologías de simulación empleadas son bastante diversas aunque predominan la simulación continua (Dinámica de Sistemas) y la simulación de eventos discretos. Ambas metodologías se prestan específicamente a representar y estudiar facetas diferentes del proceso de producción de *software*.

La primera permite analizar la estructura de interrelaciones complejas entre las diferentes variables de estado del sistema en estudio, especialmente cuando estas relaciones son fuertemente no lineales y los métodos analíticos resultan poco adecuados. Una organización dedicada al desarrollo de productos *software* es un sistema sociotécnico complejo,en los que las interacciones humanas, los flujos de comunicación y la presencia de objetivos diferentes y,

hasta contradictorios, requieren este tipo de representación.

La metodología orientada a eventos discretos, por su parte, posibilita el estudio del ciclo de vida de entidades individuales diferenciadas y su circulación a través de las diferentes componentes del sistema. La producción de *software* está plagada de este tipo de entidades. El propio producto no es simplemente un "bloque de código" más o menos grande sino una unidad compleja con su funcionalidad propia y diferente de la de otros productos. Cada una de esas unidades da lugar a una multiplicidad de entregables y otro tipo de artefactos que se generan en la producción de *software*. Los propios recursos que se emplean en la producción no son simplemente acumulables; el recurso principal son personas, cada una con sus diferentes capacidades, conocimientos y habilidades. Si bien el tratamiento del ciclo de vida de objetos individuales en un sistema es objeto de estudio desde hace mucho tiempo, típicamente con la Teoría de Colas y los procesos de Markov, aquí lo mismo o más que en el análisis de relaciones estructurales en los sistemas continuos, los métodos analíticos no son suficientes para tratar suficientemente el problema.

En [ZL04] se cita el dato siguiente: revisando los *proceedings* de los talleres PROSIM entre 1998 y 2006, el 54 % de los modelos presentados son de Dinámica de Sistemas (DS) y el 27 % restante, simulación de eventos discretos (SED). A gran distancia en el número de casos que se recogen en la literatura aparece un segundo grupo de metodologías. Estas lo conforman las redes de Petri y la simulación basada en agentes.

Las redes de Petri (y otros modelos anteriores basados en redes estocásticas o en otro tipo de representación de estados y transiciones) tienen una primera aplicación a la simulación estocástica tipo Monte Carlo que aparentemente no se diferencia gran cosa de la simulación de eventos discretos, aunque posee un grado de formalización mayor. Esta formalización tiene potencialidades adicionales que han motivado su empleo específico, tanto en el campo del tratamiento analítico como su posibilidad de *instanciamiento* del propio modelo [dSP05].

La metodología basada en agentes es la más novedosa de las citadas porque permite capturar el carácter distribuido de los procesos y simular

procesos de negociación entre partes y reglas de decisión implementadas en distintos niveles y no globalmente embebidas en el sistema.

1.2.4. Madurez de los procesos y viabilidad de la simulación

Un cambio en los procesos incluyendo la incorporación de técnicas y modelos sofisticados de gestión de los recursos debe traducirse en una mejora de la situación de la organización en términos de sus objetivos estratégicos. Pero ese cambio ni es fácil de implementar ni es evidente la relación entre las medidas concretas y las mejoras deseadas. Ello se debe al carácter complejo e interrelacionado de la propia actividad y del entorno en que se desarrolla [EB00].

Esto añade una dificultad específica a los efectos de esta tesis. Un proceso poco maduro y estructurado, con empleo reducido de técnicas de medición y de métodos cuantitativos en general, resulta extremadamente difícil de modelar de manera formal. Los mecanismos que explican los resultados permanecen ocultos en un conglomerado de prácticas informales y poco transparentes. Por lo tanto el esfuerzo para construir el modelo en si mismo es un esfuerzo que ayuda a diagnosticar las prácticas y áreas susceptibles de mejora pero puede estar limitado a la hora de construir dicho modelo formal.

Estos hechos son conocidos por la comunidad investigadora y así han sido señalados desde hace tiempo [Chr99] por lo que existen varias referencias sobre la interacción entre modelos de simulación y estándares de mejora de procesos que expresan diferentes puntos de vista de los que seleccionamos dos casos extremos para ilustrar la cuestión planteada.

En [ZL04] los autores sostienen que el grado de madurez determina el tipo de paradigma de modelado para la simulación aplicable. Así, utilizando el modelo CMMI por etapas, plantean que la correspondencia entre metodologías de simulación y nivel de madurez es la que aparece en el cuadro 1.1.

Sin negar la relación evidente entre el nivel de formalización de los procesos y las posibilidades de construir modelos de simulación, este planteamiento

Nivel de madurez	Paradigma de Simulación
CMM1	Simulación cualitativa
CMM2	Simulación semicuantitativa
CMM3	Dinámica de Sistemas
CMM4	Simulación de Eventos Discretos
CMM5	Simulación híbrida

Cuadro 1.1: Correspondencia entre niveles de madurez CMMI y metodologías de simulación según [ZL04].

denota una excesiva rigidez. De hecho uno de los problemas característicos de la aplicación de las metodologías implícitas en los modelos de evaluación y mejora es el señalado por Day en [Day00] según el cual la traslación de la metáfora "ingenieril" a la gestión del proceso *software*, provoca paradójicamente la renuncia a emplear los métodos de la ingeniería con la excusa de que el proceso no está suficientemente estructurado como para poderse medir.

Por el otro lado, en [MHSZ04] los autores presentan un estudio empírico realizado entre empresas de desarrollo australianas que, según ellos, muestra que no hay correlación entre las prácticas de gestión y la madurez del proceso. Aún más, postulan que los modelos de madurez de los procesos de desarrollo probablemente no sean ni adecuados ni trasladables al dominio de la gestión de proyectos por su orientación a "procesos de flujo de taller" (*industrial-like flowshop processes*). Por otra parte, las referencias a la sobrecarga (*overhead*) provocada por las prácticas requeridas por los modelos de evaluación y madurez son una barrera importante a la adopción de estas por muchas entidades desarrolladoras como se cita en repetidas ocasiones [ACC+04, Ric01, EB00, SdOC03].

Teniendo presente lo anterior, otros autores [MV06] proponen la incor-

poración explícita de las actividades asociadas a los procesos de mejora a la descomposición en tareas, *work breakdown structure (WBS)*, de los proyectos. De hecho, el modelo CMMI desde sus primeros niveles requiere la existencia de una descomposición en paquetes de trabajo de todas las actividades. En concreto, en el modelo por etapas es preciso incorporar *todas las tareas*, incluyendo las de medida del esfuerzo en la WBS desde el nivel 1. Para el nivel 2 se requiere implementar prácticas de planificación de recursos lo mismo que otras actividades "no constructivas" como la gestión de la configuración (*configuration management*). Esta es la perspectiva adoptada en esta tesis, sin negar la parte de verdad que corresponde a la otra posición.

1.3. Objetivos de la tesis

Esta tesis intenta conjugar las aproximaciones reseñadas en los apartados anteriores de la forma siguiente: se trata de simular la planificación y gestión de la producción de una organización dedicada al desarrollo de *software* a medida para evaluar la aplicación de las técnicas y métodos de la Ingeniería de Organización y de la Ingeniería del *Software* a dicho problema.

Para ello este trabajo ha consistido en:

- Construir un único modelo dinámico híbrido que simule las variables continuas y discretas representativas del proceso de producción de *software* en un entorno multiproyecto.

- Implementar en ese modelo algoritmos de asignación dinámica de recursos a los proyectos.

- Realizar experimentos de simulación a partir de "hechos estilizados"[4] recogidos de un caso real que se empleará para calibrar el modelo.

El modelo de simulación persigue servir para responder a preguntas del siguiente tipo:

[4]En ciencias sociales, especialmente en Economía, se denomina un ***hecho estilizado*** (*stylized fact*) a una representación simplificada de un hallazgo empírico.

- Cómo afectan las dependencias entre fases de ciclos de desarrollo más complejos que el clásico modelo en cascada.

- Cómo afectan dichas dependencias a la marcha general del conjunto de una organización desarrolladora.

- Cómo afectan las decisiones de reasignación de recursos a nivel multiproyecto a los proyectos individuales.

- Cómo afectan las decisiones individuales de cada Jefe de Proyecto a la marcha general del conjunto.

- Qué repercusiones tiene la aparición o modificación de los requisitos en cada proyecto y en el conjunto.

- Qué repercusiones tiene la adopción de determinadas prácticas de medida y de mejora de los procesos en cada proyecto y la marcha general del conjunto.

Desde el punto de vista de los **objetivos** el modelo obedece a los de *Planificación*, *Gestión Operativa* y, por supuesto, *Comprensión* de los procesos según lo señalado en el apartado 1.2.3. Desde el punto de vista del **alcance** el modelo abarca el ciclo de vida de los proyectos *software* que una organización desarrolla de manera simultánea; tiene un alcance *multiproyecto*. Desde el punto de vista de la **metodología** de simulación, es un modelo *híbrido continuo-discreto* basado en la integración numérica de variables continuas definidas a través de ecuaciones diferenciales (base de la Dinámica de Sistemas) conjuntamente con la simulación de eventos discretos que están condicionados por y a la vez afectan a las citadas variables continuas. Puesto que el modelo simula mecanismos de toma de decisión a diferentes niveles jerárquicos, en parte puede considerarse también como un modelo orientado a *agentes*.

1.4. Metodología seguida

La metodología seguida en la tesis se basa en el denominado "paradigma de investigación de la ingeniería" [THP93]. Este paradigma proporciona un marco viable y adecuado para el desarrollo, evaluación y posterior mejora de una propuesta. Tomando en cuenta las sugerencias críticas de Humphrey [Hum02]:

> "*Every time software people have faced a new problem, instead of building on prior work, we have invented some new language, tool, or method. This forces us to start over, and it also forces our users to start over.*"

La metodología adoptada ha pretendido eludir ese riesgo y por lo tanto ha seguido la siguiente secuencia:

1. Observación de las soluciones existentes.
2. Propuesta y desarrollo de una mejor solución a partir de las anteriores (*"buinding on prior work"*).
3. Medida, análisis y validación de la solución propuesta.

1.4.1. Observación de las soluciones existentes

Cuando el campo investigado corresponde en su integridad a un área de conocimiento determinada, la primera fase requiere establecer criterios de evaluación de las soluciones existentes para determinar sus características y limitaciones puesto que el objetivo es profundizar en la dirección en que apuntan las soluciones preexistentes.

En este caso se trata más de extender que de profundizar, es decir de incorporar e integrar los avances de dos áreas de conocimiento, la de la Ingeniería del *Software* y la de la Ingeniería de Proyectos. Ello obliga a establecer los criterios que posibilitan el empleo de los resultados alcanzados en la Ingeniería de Proyectos en la simulación de procesos *software*. Dichos requisitos afectan a ambas áreas de conocimiento. Ello significa:

- Identificar las soluciones implementadas hasta ahora en los modelos de simulación de procesos *software*.
- Identificar los modelos de asignación de recursos que propone la Ingeniería de Proyectos que son:
  - Adecuados al campo de la producción de *software*.
  - Susceptibles de ser implementados en un modelo de simulación.

Los factores que se estiman críticos y con los que se contrastan las soluciones existentes son:

- Desde el punto de vista de los modelos de gestión de proyectos:
  - La representación del entorno multiproyecto.
  - Los algoritmos de asignación de recursos susceptibles de modelado dinámico.
  - El tratamiento del riesgo.
- Desde el punto de vista de los modelos de simulación existentes:
  - La capacidad de representar los aspectos continuos y discretos de los proyectos *software* así como la superposición y retroalimentación entre fases y proyectos.
  - La implementación de métodos de asignación de recursos.
  - La capacidad de representar y medir los efectos de los procedimientos de mejora de procesos.

Esta primera fase del trabajo se ha centrado básicamente en la revisión de literatura y el análisis de la misma a la luz de los criterios citados.

1.4.2. Propuesta y desarrollo de una solución

Para la propuesta de una solución se han tomado los elementos más interesantes y adecuados detectados en la fase anterior. La amplia producción

en el campo de los modelos de simulación de proyectos *software* hace aconsejable tener en cuenta la recomendación de Humphrey antes citada y no partir desde cero sino reutilizar al máximo las soluciones existentes en la medida en que sean adecuadas al fin que se persigue.

Sin embargo, la construcción del modelo obliga, en primer lugar, a elegir un lenguaje de simulación adecuado y "traducir" al mismo aquellos elementos que se van a reutilizar de modelos preexistentes. El lenguaje adecuado debe tener las características siguientes:

- Capacidad de simulación híbrida.
- Capacidad algorítmica para implementar los modelos de asignación.
- Orientación a objetos para posibilitar la jerarquización y el encapsulamiento de los diferentes componentes del modelo.
- Flexibilidad y potencial gráfico para permitir la realización de experimentos.

El lenguaje de simulación elegido es *Modelica*, un lenguaje en código abierto que se adapta a los requisitos señalados [Ass08, Fri04].

Al modelo de simulación que se ha construido se le ha denominado **SimHiProS** (**Si**mulador **Hí**brido de la **Pro**ducción de **S**oftware). Se trata de un modelo jerárquico que diferencia las decisiones comenzando a partir de los procesos elementales de trabajo. Para implementar esta jerarquía se han definido tres módulos; *paquete*, *proyecto* y *multiproyecto*. A su vez cada uno de ellos tiene definidos tres componentes o submódulos cada uno de los cuales representa y simula la actividad de *producción*, las decisiones de *asignación* y la *medida* y control dentro de cada nivel. El modelo se completa con un módulo de *entorno* que representa la interacción del sistema con lo externo a él y un módulo *funcional* que implementa distintas funciones y algoritmos que son llamados desde los anteriores.

Obviamente la metodología empleada en esta fase es la característica de un desarrollo *software* en si mismo. Para ello se ha empleado una orientación a prototipos reutilizables, partiendo de un modelo simple e incorporando

sucesivos refinamientos conforme se iban verificando operacionalmente los modelos previos.

1.4.3. Análisis, validación y explotación

La tercera fase de la elaboración de la tesis se ha destinado a la validación y explotación del modelo construido. Para ello se han realizado las siguientes tareas:

- Estimación de parámetros y validación mediante el contraste con datos históricos.
- Simulación de experimentos.
- Análisis de los resultados.

Para la validación y estimación de parámetros se ha empleado una base de datos de un caso real que se ha tomado como referencia para la elaboración de los trabajos previos a la presente tesis.

La información precedente se ha utilizado para la simulación inversa, estimándose así los parámetros que ha sido posible. Se dispone de un subconjunto de proyectos cuyo tamaño en puntos-función se conoce lo que permite estimar la información relativa a cargas de trabajo. Igualmente existe una memoria "post-mortem" de los proyectos en cuestión, haciendo posible analizar determinadas incidencias que han podido afectar al perfil de desarrollo de cada uno de ellos.

Para concluir la tesis se ha realizado una serie sencilla de experimentos con las finalidades siguientes:

Planificación de los procesos. A partir de las estimaciones de carga de trabajo y esfuerzo se simula la utilización de recursos teniendo en cuenta las diferentes estrategias de descomposición en tareas de los proyectos.

Gestión operativa de los procesos. A partir de perturbaciones sobre la planificación inicial se simulan las decisiones de asignación y reasignación de recursos en función de su impacto intra e inter-proyecto.

Mejora de procesos y sistemas de medida. Las decisiones que llevan a modificar los procesos o adoptar determinadas herramientas u otros elementos de medida y control se simulan midiendo las consecuencias inmediatas en términos de coste.

El análisis de estos experimentos permite extraer las conclusiones finales del trabajo realizado.

1.5. Principales resultados

El principal resultado de esta tesis ha sido la construcción del modelo **SimHiPros**, un modelo híbrido para la simulación de la producción de *software* que presenta las siguientes características:

- Representación los aspectos *discretos y continuos* de la producción de *software*.
- Representación *jerárquica* del entorno multiproyecto empleando tres niveles, de abajo a arriba: *paquete*, *proyecto* y *multiproyecto*.
- Capacidad para representar *diferentes modelos* de proceso *software*: en cascada, incremental, evolutivo, ...
- Obtención de *métricas de proyecto* según el modelo EVM (*earned value management*).
- Capacidad para incorporar *modelos de asignación* de recursos a nivel de proyecto y multiproyecto, habiéndose implementado algunas variantes sencillas.
- Capacidad para medir el esfuerzo en actividades "no constructivas" como una primera aproximación a la simulación del *coste de implantación de la mejora* de procesos *software*.

El modelo se apoya en una propuesta de marco jerárquico para la caracterización del problema de la planificación y asignación de recursos a la

producción de *software* que proviene del campo de la Ingeniería de Proyectos. A partir de ese marco, la tesis propone la tipología de problemas de asignación de recursos y programación de tareas que mejor corresponde a los proyectos *software* estudiados y para los que el modelo ha sido diseñado. Alguna de las variantes más sencillas de los algoritmos de resolución de esos problemas se han implementado en la versión actual del modelo. Éste admite en principio otros algoritmos más sofisticados.

El modelo incorpora también resultados previos del área de la simulación de procesos *software* que aparecen reseñados en la literatura que se presenta en el capítulo 3. Pero hasta donde se ha podido verificar, en ninguno de los modelos anteriores se integran los aspectos discretos y continuos en la forma en la que aquí se propone. Únicamente en [CBK06] se presenta una aplicación del formalismo DEVS, *Discrete Event System Specification*, propuesto por Ziegler [ZKP00] que posibilita representar con naturalidad un sistema híbrido aplicado a la producción de *software*. Sin embargo, por lo menos hasta lo que se ha publicado, en el caso señalado no se implementan sistemas complejos de asignación de recursos sino que el modelo se limita a emular el funcionamiento de los modelos continuos de la saga del modelo de AbdelHamid y Madnick [AM91].

Por último, son muchos los modelos que se hacen eco del efecto de la mejora de procesos; desde su propia estructura, como es el caso del modelo de Ruiz Carreira [Car03], hasta la implementación explícita de la misma, como el modelo *IMMoS* [PR02] que combina a tres niveles, con Dinámica de Sistemas, modelos de procesos y el empleo de modelos cuantitativos basados en la medida. El modelo que se presenta en esta tesis se corresponde más con el segundo tipo, pero a diferencia de él, consigue modularizar la descripción de los procesos y representar los procesos de obtención de métricas sin necesidad de un esfuerzo excesivo de readaptación a cada caso.

1.6. Contenido de la tesis

El Capítulo 2 de esta tesis presenta una síntesis del trabajo realizado de análisis de soluciones existentes en el campo de la Ingeniería de Proyectos.

Se presenta el tratamiento que en la literatura se le da a la situación multiproyecto. Posteriormente se revisan diferentes modelos para la asignación de recursos y las estrategias para la gestión del riesgo.

La revisión de las aportaciones de la Ingeniería del *Software* se concentra en los modelos de simulación y se realiza en el Capítulo 3. Dentro de estos, el capítulo se centra aquellos que prestan más atención al carácter híbrido de los procesos, a los que representan procesos multiproyecto, a los que atienden a la mejora de los procesos y a los que incorporan modelos de Investigación Operativa para la asignación de recursos y secuenciación de tareas.

El Capítulo 4 se dedica a presentar el caso de aplicación elegido y sus particularidades. En concreto se describe las reglas de gestión actualmente empleadas y la base de datos histórica disponible así como el tratamiento que se ha hecho de la misma. Se identifican a través del análisis "post-mortem" algunas de las incidencias que caracterizan a los proyectos a fin de emplear dichas incidencias para el contraste del modelo.

El Capítulo 5 analiza las soluciones existentes en base a su adecuación al caso de aplicación y sus singularidades. Se propone la descomposición jerárquica del problema de programación de recursos, lo que permite caracterizar dos problemas distintos, un problema *táctico* y otro *operacional*. Se proponen modelos de la Investigación Operativa adecuados para cada problema y la gestión del riesgo en ambos casos. Por último se propone el modelo de simulación que se presenta en el capítulo siguiente, en base a las aportaciones analizadas en el Capítulo 3 y a los aspectos a integrar de las mismas.

El Capítulo 6 describe el modelo propuesto y sus componentes. Explica su operativa y las variantes que se han considerado así como algunas cuestiones que no se han desarrollado pero que podrían incorporarse con facilidad. Mediante un ejemplo sencillo se muestra el comportamiento del modelo.

El Capítulo 7 presenta las conclusiones del trabajo realizado y los desarrollos posteriores. Los Apéndices complementan el texto con los aspectos más técnicos y numéricos.

Capítulo 2

Trabajo relacionado: planificación de proyectos

2.1. Introducción y contenido del capítulo

La dsiciplina de la Gestión de Proyectos (*project management* es un área de aplicación de la ingeniería cuyas bases se pusieron en los años 50 del siglo pasado con la aparición de los primeros modelos de *redes* de proyectos: el PERT (*program evaluation and review technique*) y el CPM (*critical path method*). Una rama específica de la Investigación Operativa se ha concentrado en resolver los problemas específicos de optimización de redes de proyectos, es decir, de encontrar secuencias o programaciones (*schedule*) que, compatibles con los requisitos tecnológicos y económicos, optimizan uno (o varios) valores representativos de una magnitud relevante del proyecto, sea esta el coste, la duración total o algún otro. En general, todos los problemas de este tipo son NP-duros por lo que los métodos exactos de solución sólo suelen ser útiles en problemas de escasa entidad.

Una parte de la investigación se ha centrado en la producción de algoritmos para la obtención de soluciones aproximadas para estos problemas. La cantidad de modelos y propuestas producidas es muy importante. Esta producción se ha visto enriquecida en las últimas décadas con las aportaciones de la Inteligencia Artificial, que ha proporcionado nuevos métodos heurísti-

cos para la obtención de soluciones de calidad, así como técnicas, como la propagación de restricciones (*constraint propagation*), para acotar el espacio de búsqueda.

Por otra parte, otras ramas de la disciplina, unas más ligadas a la práctica profesional y otras más enfocadas en la reflexión epistemológica sobre la misma, han relativizado la idoneidad de las soluciones más sofisticadas propuestas desde el área antes citada y enfatizan más los aspectos organizativos, la estructura de la toma de decisiones y el riesgo que rodea a la ejecución de un proyecto. En esta tesis no se pretende intervenir en este debate sino que nos limitamos a tomar nota del mismo viendo de aprovechar las aportaciones que interesan al objeto de la misma.

Con esta filosofía, en este capítulo se describen las aportaciones más relevantes que se han detectado en relación con los siguientes temas:

- El tratamiento del problema multiproyecto cuando los diferentes proyectos que lleva a cabo una organización no guardan especial relación entre si salvo en la medida en que comparten los mismos recursos. Se caracteriza la situación como una en la que existen diferentes niveles de decisión y, por tanto, diferentes problemas.

- Los modelos de programación de proyectos que se proponen para cada uno de los niveles citados. La existencia de niveles y ciclos de decisión diferentes da lugar a la existencia de objetivos distintos y, por tanto, de soluciones diferentes. De estas, se han estudiado aquellas que son susceptibles de implementación en un modelo de simulación.

- El tratamiento del riesgo y la incertidumbre más adecuado para el problema en cada caso. No es igual la estrategia de hacer frente al riesgo de un proyecto concreto en su ejecución que la adecuada para planificar varios proyectos que van a coexistir durante un plazo más o menos largo.

2.2. Caracterización del problema multiproyecto

2.2.1. La organización de la empresa y la gestión de proyectos

Una vez que una empresa tiene que acometer un proyecto debe decidir de qué manera se relaciona el mismo (entendido en el sentido organizacional) con el conjunto de la compañía. Según [Mer06] hay básicamente tres maneras:

El proyecto integrado en una unidad funcional . Este caso sólo se da cuando la dimensión del proyecto es lo suficientemente pequeña y sus características son tales, que una unidad funcional es capaz de asumir íntegramente las actividades del proyecto con sus propios recursos.

La organización por proyectos . Cuando el proyecto contiene todos los recursos necesarios para desarrollar las actividades. En este caso se obtiene la ventaja de que el proyecto es autónomo y no plantea conflictos con otros proyectos o con departamentos funcionales. Por contra, se corre el riesgo de duplicar esfuerzos en áreas funcionales e infrautilizar los recursos que podrían ser compartidos.

La organización matricial . En este caso se produce una solución intermedia entre los dos casos extremos expuestos previamente. El proyecto posee una cierta autonomía pero emplea recursos compartidos. Este es el caso más general, si bien el rango de opciones es elevado. Este caso puede modelarse desde un punto de vista de procesos como un taller: el trabajo es realizado por una serie de estaciones/servidores que se corresponden con las unidades funcionales mientras que los proyectos son esos trabajos fluyendo entre estaciones [AMNS95].

2.2.2. Caracterización del entorno multiproyecto

Si el punto de vista del análisis es la totalidad de la organización el primero de los tres casos anteriores (los proyectos embebidos dentro de una única

unidad funcional) no es propiamente un entorno multiproyecto. Este aparece cuando hay que tomar decisiones que afectan a los recursos generales de la organización, bien aféctandolos al menos temporalmente en exclusiva a un proyecto (organización por proyectos), bien compartiéndolos entre varios (orgaización matricial).

En [Her05] los autores proponen un marco para establecer diferentes categorías de entornos multiproyecto en función de las características de los proyectos y su relación entre si. Señalan para ello dos características determinantes, la *variabilidad* y la *dependencia*.

Por *variabilidad* se entiende la incertidumbre que se deriva de la falta de información suficiente sobre las actividades requeridas al inicio del proyecto, incertidumbre que se irá desvelando conforme avanza el desarrollo, compuesta con la propia incertidumbre operacional en la ejecución de las propias actividades. La *dependencia* sería la medida en la que las actividades de un proyecto se ven afectadas por influencias externas al mismo. Estas influencias pueden deberse a actores externos (clientes, proveedores, ...) o a factores internos, típicamente los conflictos sobre recursos compartidos.

El marco resultante es el que recoge el cuadro 2.1. Se plantea en términos de cuatro casos extremos si bien cabe todas las combinaciones intermedias de manera continua. Esos cuatro casos son:

- Caso BB (*baja variabilidad, baja dependencia*). Se trata de una situación en la que los proyectos son rutinarios y están bien especificados los procedimientos por lo que la incertidumbre es baja. Un ejemplo típico serían trabajos de mantenimiento preventivo “in situ”. Los recursos están asignados desde el nivel jerárquico superior al que gestiona el proyecto, normalmente en una organización orientada a proyectos que mantiene compartimentos relativamente estancos.

- Caso BA (*baja variabilidad, alta dependencia*). Se trata de una situación en la que recursos de propósito general se emplean por diversos proyectos por otra parte bien especificados y de baja incertidumbre pero que pueden colisionar en sus demandas. Un ejemplo típico sería la producción o el ensamblaje de conjuntos de poca complejidad ba-

jo pedido. Típicamente se trata de una organización matricial en la que se busca equilibrar la presión de cada proyecto sobre las diferentes unidades funcionales.

- Caso AB (*alta variabilidad, baja dependencia*). Se trata de una situación en la que proyectos relativamente complejos y con incertidumbre derivadas de variables externas no controladas como las condiciones meteorológicas o ambientales de proyectos complejos de ingeniería civil o la volatilidad de las especificaciones. En una empresa orientada a proyecto, los recursos están dedicados a los proyectos individuales por lo que se producen pocos conflictos en torno a su asignación.

- Caso AA (*alta variabilidad, alta dependencia*). Se trata de una situación en la que se producen junto con las incertidumbres propias de cada proyecto las que se componen por la dependencia entre proyectos que compiten por recursos. Se trata de una situación característica de una organización matricial.

	Baja dependencia	*Alta dependencia*
Baja variabilidad	BB	BA
Alta variabilidad	AB	AA

Cuadro 2.1: Variabilidad y dependencia en el entorno multiproyecto según [HL04b].

2.2.3. Jerarquía de decisiones en la gestión multiproyecto

La clasificación anterior en términos de dependencia y variabilidad resulta incompleta sino se tiene en cuenta que una parte de las decisiones a

tomar en la gestión multiproyecto precisamente conduce a acotar, en la medida de lo posible, esas contingencias. Así, la opción por *una organización matricial conduce a una mayor dependencia* mientras que la adopción de *una organización por proyectos tiene como finalidad precisamente reducir aquella.*

Estas *configuraciones no son naturales sino que son consecuencia de decisiones que se adoptan en los niveles estratégicos.* La primera opción estratégica, precisamente, es la configuración que adopta una organización multiproyecto.

Adler *et al.* [AMNS95] en un artículo muy citado sugieren adoptar un punto de vista de procesos para tratar el problema de la gestión multiproyecto. Sostienen que la actitud que habitualmente prevalece está enfocada sobre los proyectos individuales en lugar de considerar la complejidad de asignar cargas de trabajo en un sistema con una capacidad dada. Se trataría, según ellos, de introducir un enfoque de *gestión por proyectos* que, partiendo de los objetivos y entregables concretos de cada proyecto individual, los integrara la operación global de la empresa. A pesar de que el enfoque de gestión por proyectos es muy citado, en [HHLW07] se sostiene que las aproximaciones reales son escasas.

La jerarquía de las decisiones

A juicio de Hans *et al* [HHLW07] el problema se debe en parte a la *confusión entre los diferentes niveles de responsabilidad y decisión.* En términos de la gestión de una cartera de proyectos es necesario diferenciar entre los criterios empleados para priorizar y seleccionar de proyectos, que corresponde a los niveles *estratégicos* de la dirección, y las decisiones *tácticas* y *operacionales* sobre la asignación de capacidad y el control de la programación de tareas.

Así en [LK02] se proponen métodos para la aceptación de proyectos en un modelo dinámico que considera la asignación de recursos en función de diversos criterios. En [AIG03] se proponen mecanismos de control de proyectos que limitan el número de proyectos activos con el objetivo de asegurar una carga de trabajo constante (CONWIP, *constant work in process*).

Según los mismos autores de [HHLW07] es preciso también *distinguir el caso de los programas* como un caso particular de la gestión multiproyecto, en la medida en que los componentes de un programa concurren a un *objetivo común* y pueden verse como un *proyecto único complejo*, *del caso más general de la coexistencia de proyectos independientes*, cada uno de ellos con sus propios objetivos.

En el caso multiproyecto más general diversos proyectos compiten por los mismos recursos limitados. Para eludir los conflictos que se plantean en la práctica sobre la asignación de los mismos se puede recurrir a una planificación agregada con capacidad fija. Sin embargo esto hace perder flexibilidad al nivel operacional. Cuando los conflictos se plantean en este nivel, es corriente que los planes agregados se subviertan debido al diferente tamaño de los proyectos, la incertidumbre propia de cada uno de ellos, la naturaleza dinámica de la propia cartera de proyectos y el hecho de que sus componentes pueden tener directores diferentes, con criterios distintos y poder diferente en la organización.

Por estos motivos, sostienen, la gestión multiproyecto adecuada requiere ser considerada simultáneamente *en los diferentes niveles de planificación* teniendo presente que dichos niveles tienen diferentes objetivos, niveles de agregación, restricciones y flexibilidad. Por ejemplo, a nivel operacional es prioritaria la adecuada secuenciación de tareas para cumplir los compromisos de plazos. Sin embargo, en el nivel táctico lo es el empleo adecuado de los recursos sin tener momentos de capacidad ociosa frente a momentos de sobrecarga, el típico problema del equilibrado de recursos considerado a escala de la empresa. En todos los niveles, disponer de planes robustos y estables es una necesidad. Así, la gestión multiproyecto debe ser capaz de tratar con estos objetivos de manera diferenciada y teniendo en cuenta la *jerarquía* de los mismos.

Planificación jerárquica

La planificación con un enfoque jerárquico ha sido muy estudiada en el ámbito de la planificación de la producción. Los sistemas MRP y todos

sus sucesores se basan en la idea de la obtención de un *Plan Maestro de Producción*, correspondiente al nivel *táctico* de decisión, que determina los límites de la programación detallada al nivel de taller [LOL88].

Una primera extensión al terreno de la gestión de proyectos aparece en [HL89]. En ella, los autores proponen una planificación jerarquizada en *dos niveles*. En el primero se agregan las actividades de los proyectos que presentan perfiles de empleo de recursos semejantes y se asignan fechas de entrega. Dadas las fechas de entrega de los proyectos en curso y las fechas tentativamente asignadas a los nuevos proyectos, a través de un programa lineal se minimiza el coste descontado de los recursos no utilizados. Una vez se obtiene un perfil adecuado de empleo de recursos, las fechas asignadas se emplean para secuenciar cada proyecto individualmente.

Otras aproximaciones al problema pueden verse en [866] donde se minimizan los costes totales derivados de los retrasos. En [SV93] se agregan las actividades definidas a nivel operacional en programas a nivel táctico. Este enfoque es criticado por [HS96] que demuestra con contraejemplos que se alcanzan soluciones subóptimas. Yang y Sum [YS97, YS93] proponen una estructura dual para gestionar el uso de recursos en un entorno multiproyecto. Una autoridad central negocia los plazos con los clientes y asigna los recursos a los proyectos de modo que los proyectos críticos tienen prioridad, determinado así un calendario de inicio de proyectos. Los responsables operacionales programan las actividades de sus respectivos proyectos dentro de las restricciones así impuestas. Examinan el comportamiento de reglas heurísticas de asignación de recursos y fijación de plazos, incluyendo el caso en el que el desplazamiento de recursos entre uno y otro proyecto impone costes.

En [Boe98] se propone un marco jerárquico para la planificación en las organizaciones orientadas a proyecto. Argumenta que es necesaria una descomposición jerárquica para llegar a un proceso mejor gestionado. Menciona igualmente que la incertidumbre es un factor relevante en un entorno multiproyecto. Si esta es demasiado amplia, los canales de información en la estructura jerárquica se saturan. Para evitarlo propone cuatro estrategias: a) la creación de holguras relajando objetivos; b) la creación de tareas "auto contenidas", es decir, grandes agrupaciones de tareas a desempeñar por equi-

pos multidisciplinarios; c) la creación de vínculos laterales empleando diseños organizativos matriciales o equipos especializados; y, d) la inversión en sistemas de información verticales. Asegura que estas vías son adecuadas para enfrentar los problemas de la incertidumbre en la organización orientada a proyectos. No obstante, y en la misma línea que la mayoría de los autores antes que él, acaba desembocando en un marco determinista de planificación en el cual no hay comunicación explícita para hacer frente a las variaciones entre los diversos niveles de la jerarquía.

En [NS98b] se propone un modelo más sofisticado que se desarrolla en tres planos jerárquicamente organizados suponiendo una cartera de proyectos que deben acometerse dentro de un horizonte de planificación de 2 a 4 años. Cada proyecto posee una fecha de inicio, una fecha máxima de terminación y una descomposición en tareas y actividades.

En el nivel más alto, a largo plazo, todos los proyectos se integran en una única red que los contiene como actividades agregadas. Las fechas de inicio y final se programan utilizando relaciones de precedencia generalizada (ver más adelante). Las actividades se programan restringidas por los recursos clave. La duración estimada de una de tales actividades agregadas equivale a la duración del camino crítico del proyecto correspondiente más un *buffer* que anticipa la duración calculada al tercer nivel de desagregación y que se estima utilizando métodos de la teoría de colas. La capacidad en términos de recursos clave viene dada por la estrategia general de la empresa. La función objetivo es la maximización del valor actual neto de la cartera de proyectos.

Esto proporciona una programación en la que se le asigna una duración máxima a cada proyecto y el perfil de empleo de recursos resultante que actúa como restricción de los proyectos individuales al segundo nivel de planificación. En éste segundo nivel, referido al medio plazo, cada proyecto se secuencia en términos de actividades desagregadas considerando los recursos no-clave como ilimitados. Aquí el objetivo es equilibrar el empleo de dichos recursos. Todo lo anterior proporciona el marco de restricciones para el tercer y último nivel en el que se minimiza la duración de los proyectos, ahora descompuestos en las tareas elementales.

En [Kol00] se propone un marco jerárquico para diferenciar entre los

diferentes procesos de gestión en la fabricación bajo pedido, del tipo MTO (*manufacturing to order*). Se diferencia entre tres niveles: la aceptación de pedidos, la planificación de la producción y la secuenciación de operaciones. Un marco que se corresponde con los que ya se han comentado.

2.3. Aproximaciones al problema de planificación táctica de capacidad

El problema de planificación táctica de capacidad es un problema relativamente poco estudiado. En el plano estratégico existe una literatura muy abundante, en general en relación con el extenso campo de las cadenas de suministro, la producción flexible, las redes de transporte, etc. La planificación de capacidad en el plano estratégico se basa en estimaciones agregadas de la demanda sin tener en cuenta la especificidad de los pedidos de los clientes.

En el plano operacional, la secuenciación *on-line* ha sido ampliamente estudiada como una aplicación del problema RCPSP estocástico [MSSU03, MS00, Uet01] en el mismo contexto de las cadenas de suministro, la secuenciación de tareas en talleres *job-shop* y *flow-shop* y también en aplicaciones relacionadas con la gestión de redes y recursos informáticos (procesadores). En este plano, el principal factor de incertidumbre es el ritmo de llegada de las tareas o proyectos.

El problema de la etapa de planificación táctica que aquí se investiga no consiste en que el ritmo de aparición sea una fuente de disrupción permanente de la programación puesto que en el nivel estratégico previo existe un filtro que "admite" o no nuevos proyectos. El problema es que se dispone sólo de una aproximación muy general al contenido de trabajo de los proyectos que van llegando.

2.3.1. El problema táctico como un problema de programación dinámica

Una aproximación al problema táctico se recoge en [AT08], en el que se emplea un modelo de programación dinámica para un horizonte finito de intervalos discretos de tiempo y se busca minimizar los costes totales de hacer frente a una demanda estocástica mediante una política de recursos permanentes y recursos contingentes (personal eventual y horas extraordinarias). Se trata de producir para inventario, por lo que el modelo se adapta poco al caso de estudio. También en un contexto de producción para inventario se pueden reseñar [NS98b] y [CLT02].

2.3.2. El problema táctico como un problema de asignación

Otro conjunto de aproximaciones proviene del campo del problema generalizado de asignación (GAP) en el que se trata de determinar una asignación óptima (generalmente de coste mínimo) de una serie de tareas a agentes. Se trata de un problema derivado de un problema tradicional de taller, la asignación de m trabajos a n máquinas, con restricciones de todo tipo: limitación de recursos, ventanas de tiempo, etc. Un problema parecido se trata a la hora de confeccionar horarios, asignar personal a flotas de transporte o programar intervenciones quirúrgicas en un hospital [BJN+98, Nag02, GC03, CR06, LTN06, SW06, TGR05, FCMA08].

En un contexto de proyectos las principales contribuciones detectadas en esta línea corresponden a [HGdVZ02b, GS05, HGdVZ02a, Boe98] y [BS99]. Se trata de modelos en los que el horizonte de planificación es finito y está definido en términos discretos. A partir de ahí, el problema se plantea como uno de *programación lineal mixta* continua-entera. Las restricciones ligadas al tiempo dan lugar al carácter entero del problema a través de la asignación de variables binarias a los intervalos discretos de tiempo. Las restricciones de recursos, de carácter continuo, limitan la cantidad de trabajo que puede hacerse en un intervalo de tiempo.

Este tipo de problemas de asignación ha sido ampliamente estudiado y existen diversas propuestas de solución, algunas de las cuales se describen en el apéndice A. La idea general es que al dividir el horizonte de planificación en T intervalos de tiempo discretos de duración igual, hay que definir cuánto de cada proyecto se realiza en cada intervalo. Los proyectos que la organización debe llevar a cabo en ese horizonte están descompuestos en n *actividades agregadas*, definida. Las relaciones de precedencia entre esas actividades así como las restricciones temporales (fechas de entrega, hitos, ...) se expresan mediante *ventanas de tiempo* admisibles para cada actividad.

Una solución está formada por una lista ordenada de $n \times T$ componentes, x_{jt}, con t de 1 a T y j de 1 a n que indica la fracción de la actividad j que se ejecuta en el periodo t.

La función objetivo a considerar puede adoptar dos formas:

- *Problema restringido por el tiempo:* se trata de cumplir con un horizonte temporal para cada proyecto al menor coste de oportunidad que es el derivado del empleo de recursos no regulares.

- *Problema restringido por los recursos:* se trata de minimizar la duración de los proyectos.

Sea como fuere, la solución táctica, que no es más que la secuencia de asignación de recursos a cada proyecto en los intervalos de tiempo previamente definidos, determina el marco de las restricciones en el problema operacional que es el que se trata en el apartado siguiente.

2.4. Modelos de programación operacional de proyectos

El modelo clásico de un proyecto [Tav02] es un modelo relativamente simple de Investigación Operativa, un *grafo dirigido y acíclico* definido por:

- Un conjunto finito de *actividades* o tareas que hay que programar o secuenciar a fin de completar el proyecto

- Unas *relaciones de precedencia* entre las actividades representadas de forma unívoca por el conjunto de las inmediatamente anteriores a cada una de ellas. A esta relación se le denomina Final-Inicio aunque en ocasiones esas relaciones pueden ser *generalizadas*, es decir, no necesariamente vinculan a la actividad precedida con la precedente por la fecha de terminación de esta que debe ser menor o igual a la de iniciación de aquella, sino que pueden ser de diversos tipos como:

  - *Inicio-Inicio*; cuando el inicio de la actividad sucesora requiere que se ha iniciado la predecesora
  - *Final-Final*: cuando la predecesora debe terminar en tiempo menor o igual que la sucesora
  - *De realización parcial*: cuando la sucesora sólo pude iniciar una vez que la predecesora se ha completado en un porcentaje arbitrario
  - *Con retrasos*: cuando debe satisfacerse un retraso arbitrario entre los inicios y finales en cualquiera de sus combinaciones

- Un conjunto finito y discreto de *atributos de cada actividad*, representando propiedades relevantes del proyecto a efectos de su gestión, tales como duración, coste, consumo de recursos, ...

- Un conjunto discreto y finito de *criterios* que expresan las preferencias del (de los) decisor (decisores) tales como la duración total, el coste total, la relación coste-beneficio, el valor actual, ...

Este modelo básico se ha refinado y enriquecido progresivamente con las aportaciones de la comunidad académica, especialmente del campo de la Investigación Operativa y, más recientemente, de la Inteligencia Artificial. Dichos avances se han producido básicamente en tres terrenos:

- *El modelo de la red*; incorporando topologías más complejas que la de simples actividades relacionadas de manera determinista por relaciones de precedencia. De este modo hay precedencias basadas en operaciones lógicas, estocásticas, incluso ciclos en la red.

- *Los modos de ejecución*; tanto en lo relativo a los atributos de las actividades y sus restricciones temporales como a los recursos que pueden ser deterministas, estocásticos o borrosos y que pueden estar funcionalmente relacionados como en el caso del problema denominado *multimodo* en el que el consumo de recursos determina la duración de una actividad.

- *Los métodos de solución*; optimización algorítmica por métodos exactos o aproximados, basados en simulación, en agentes, en el tratamiento analítico del espacio de las soluciones, en el análisis de la topología de la red, etc.

Una descripción detallada del estado de la cuestión bastante actualizado puede encontrarse en [Tav99b] y [DH02]. Textos básicos de Investigación Operativa sobre el problema son [AMO93], [ME78] y [Elm70]. Escritos por profesores de la Universidad de Sevilla se pueden señalar [Lar87] y [Lar60].

2.4.1. Gestión de proyectos con recursos limitados; el problema RCPSP

La tipología de problemas que pueden plantearse con estos modelos es muy extensa. A fin de identificarlos de forma unívoca se han propuesto diversas *taxonomías* [BDM+99, DH02] . Básicamente consisten en generalizaciones de la clasificación tradicional de los problemas de secuenciación de tareas en talleres de máquinas. Existe además una nomenclatura basada en las siglas del nombre estándar del problema, algo menos precisa que las propuestas en las referencias, pero que es la que seguiremos en este trabajo por tratarse de la más extendida.

El punto de partida es el problema denominado RCPSP (*resource constrained project scheduling problem*). En términos generales se formula como un problema de programación matemática en el que:

- Las *variables de decisión* son las fechas en que se programa el inicio de las actividades

- Las *restricciones* incorporan tanto las *relaciones de precedencia* como la *disponibilidad* en términos de cota superior (e inferior) de los recursos.

- La *función objetivo* describe criterios tales como la minimización de la duración total, la nivelación del empleo de recursos, la minimización del riesgo de retraso, la maximización del valor actual neto así como otros indicadores coste-beneficio. En muchas ocasiones la función objetivo es una combinación ponderada de estos criterios. A su vez puede implementarse en términos deterministas o como funciones estocásticas en términos de valor esperado o cuantiles extremos.

Una función objetivo se denomina *regular* si no es decreciente con la duración de las actividades. Los problemas de minimización de la duración total, minimización del retraso de las actividades, etc. son problemas con una función objetivo regular. Una función objetivo es *no-regular* en caso contrario. Los problemas ligados a la minimización del coste o del valor actual y aquellas que contemplan una relación inversa entre duración y empleo de recursos, son ejemplo de funciones objetivo no regulares [DH02].

Existe una amplísima variedad de modelos a partir de las diferentes combinaciones posibles cómo ya se ha señalado. Pero siguiendo a [Tav99a] se puede caracterizar dicha variedad de una manera simplificada atendiendo a tres dimensiones:

- El criterio que se sigue

- Las características de las actividades

- Las características de los recursos

Conviene señalar alguna de las cuestiones de entre estas variantes que resultan clave a la hora de tratar los diversos problemas:

- La ejecución de las actividades puede seguir un único patrón en términos de empleo de recursos por unidad de tiempo, debe seleccionarse uno entre varios, o se admiten variaciones en la intensidad de la misma en cada intervalo de tiempo. El primer caso es el problema *unimodal*, el segundo el *multimodal* y el tercero el caso *continuo*.

Dimensión	**Indicador**	**Variantes**
Criterios	Temporal Empleo de recursos Nivelado de recursos Valor actual neto u otro criterio económico	Cualquier composición de los anteriores
Actividades	Duración Precedencias Modo Posibilidad de interrupción	Deterministas o estocásticas
Recursos	Monetarios No monetarios	Renovables o no Deterministas o estocásticos

Cuadro 2.2: Elementos que determinan los diferentes tipos de problemas de programación de proyectos según [Tav02].

- La segunda variante respecto de las alternativas es la *posibilidad de suspender y posteriormente retomar* las actividades (*preemptive*) o no (*non-preemptive*).

- En tercer lugar, las variables formuladas (duración y/o requerimiento de recursos pueden ser magnitudes *deterministas* o, alternativamente, variables aleatorias o *estocásticas*.

- En cuarto lugar, los recursos pueden ser *renovables* o no *renovables*; en el primer caso estarán disponibles al inicio de cada intervalo de tiempo mientras que en el segundo su empleo los irá agotando.

- Por último, el proceso de toma de decisiones puede ser *estático* o *dinámico*, es decir empleando una programación y asignación de tareas y recursos predeterminada desde el inicio del proyecto o posibilitando modificaciones a lo largo del ciclo de vida del proyecto.

Evidentemente, los casos más complejos contemplan actividades multimodo o en modo continuo, duraciones y/o disponibilidades de recursos no deterministas y la posibilidad de la reprogramación dinámica a lo largo del ciclo de vida.

El problema RCPSP con ventanas de tiempo

El problema RCPSP con ventanas de tiempo, también denominado TC-PSP - *time constrained project scheduling problem*- es una variante del problema RCPSP en el que las actividades solo pueden ejecutarse en un intervalo de tiempo determinado, la *ventana de tiempo*, de manera análoga a como se ha descrito el problema de planificación táctica de capacidad restringido por el tiempo.

A su vez es un caso particular del denominado problema RCPSP con relaciones de *precedencia generalizadas*(*RCPSP-GPR)*. Cualquier problema con relaciones de precedencia generalizadas se puede modelar empleando un *retraso mínimo y/o máximo* entre las fechas de inicio de dos actividades que se encuentran ligadas por una de estas relaciones. Como en cualquier proyecto se incorporan dos actividades ficticias, una inicial y otra final, la imposición de una ventana de tiempo absoluta a cualquier actividad se puede resolver simplemente imponiendo un retraso mínimo y uno máximo (descontando la duración de la actividad implicada) respecto de la actividad ficticia inicial.

El problema multiproyecto - RCMPSP

El problema multiproyecto admite básicamente dos tratamientos diferentes:

- La incorporación de todos los proyectos en una única red, es decir, modelar el conjunto de proyectos como un *metaproyecto* en el que por medio de actividades ficticias se conectan entre si las redes parciales de cada proyecto independiente. Este tratamiento es adecuado en un entorno de *programas* y desde el punto de vista de la existencia de un objetivo común a todos los proyectos.

- La consideración explícita de los proyectos de manera individualizada. Este tratamiento es adecuado cuando los objetivos de los proyectos son distintos, al menos al nivel operacional. En este caso la principal diferencia con los casos de un sólo proyecto se establece en la formulación de *la función objetivo.*

Este segundo caso es el que nos interesa pues el primero se reduce a los casos ya descritos anteriormente.

Entre las formulaciones de función objetivo adecuadas para el problema multiproyecto [BY06] señalan las siguientes:

- la minimización del tiempo de terminación de los proyectos.
- la minimización del retraso acumulado de los proyectos.
- la minimización del retraso medio de los proyectos.
- la minimización del coste total.
- la minimización del coste (penalización) del retraso.
- maximizar el empleo de los recursos (penalizar los recursos ociosos).
- maximizar el valor descontado de los proyectos.
- ...

Como es lógico, la selección de la función objetivo depende básicamente del problema en estudio y también de la posición jerárquica desde la cual se aborda. En el caso que nos ocupa, la posición jerárquica es el nivel operacional. Se parte de una previa asignación de recursos y fechas límite que proviene de la planificación táctica. Por ese motivo los objetivos pueden verse a dos niveles:

- Desde la dirección de cada proyecto individual el objetivo es minimizar el retraso relativo de *cada* proyecto
- Desde la dirección conjunta el objetivo es minimizar el retraso de *todos* los proyectos de la cartera

Puesto que en la etapa de planificación táctica se han atribuido a los proyectos una fechas límite en función de la asignación de recursos establecida, los retrasos relativos deberán ser con relación a las duraciones establecidas en esas fechas límite. De esta manera, podemos plantear los objetivos de minimización de los retrasos como:

- El retraso relativo de cada proyecto es la desviación respecto a la duración prevista en la planificación táctica como un porcentaje de la misma
- El retraso relativo de la *cartera* de proyectos es el retraso máximo de los proyectos en curso

Así pues, en una cartera con P proyectos, se define una actividad ficticia para cada uno de ellos y_{pt} que vale 1, si el proyecto p termina en el momento t y vale 0 de los contrario. Cada proyecto tiene una fecha asignada de finalización f_p que proviene de la planificación táctica.

El problema de minimización del retraso relativo de cada proyecto p es:

$$\text{mín } \frac{t \cdot (y_{pt} - f_p)}{f_p} \tag{2.1}$$

El problema de minimización del retraso *medio* es:

$$\text{mín } \frac{1}{P} \sum_{p=1}^{p=P} \frac{t \cdot (y_{pt} - f_p)}{f_p} \tag{2.2}$$

Y el problema de la minimización del máximo retraso en la cartera:

$$\text{mín } \max_{p=1}^{p=P} \left[\frac{t \cdot (y_{pt} - f_p)}{f_p} \right] \tag{2.3}$$

Otras variantes del problema

El número de variantes del problema es prácticamente interminable. Existen:

- El problema multiproyecto con múltiples modos de realización, *MRCMPSP*
- El problema de ventanas de tiempo con múltiples modos de ejecución, *MRCPSP/max o MRCPSP-GPR*
- El mismo que el anterior pero en contexto multiproyecto, *MRCMPSP/max o MRCMPSP-GPR*

- y muchos otros.

De la literatura revisada, el problema con más complejidad y opciones encontrado es el MRCMPSP-GPR/EXP, siglas que corresponden a *multi-mode resource constrained multi-project scheduling problem with generalized precedence relations and expediting resources* en el cual se añade la posibilidad del empleo de recursos extraordinarios para acelarar la ejecución de actividades o proyectos críticos [Fre01].

2.4.2. Métodos exactos para la resolución de los problemas RCPSP

El problema RCPSP es un problema combinatorio de extraordinaria complejidad computacional. Se trata de un problema *NP-duro* como se demuestra en [DH02]. Los métodos exactos para la solución del problema RCPSP se basan generalmente en procedimientos de enumeración implícita (*branch and bound*). Estos procedimientos parten de una solución inicial y van generando un árbol de particiones del espacio de las soluciones (ramas) y acotándolo. Existen diversas estrategias para generar el árbol de exploración y las cotas que aparecen descritos en la literatura. Básicamente se agrupan en dos tipos: búsqueda en profundidad (*depth-first*) y búsqueda en la frontera (*best-first* o *frontier search*). Los algoritmos van seleccionando los nodos que se exploran y ramificando a partir de ellos de acuerdo con la estrategia adoptada.

En [HD98] se analizan los diversos criterios para acotar y las reglas de dominancia que permiten reducir el espacio de exploración. Se presentan tres alternativas para la solución del MRCPSP. En [HRD98], [LWT01] y [FMK01] se revisan también los diferentes métodos aplicados al problema RCPSP.

Entre las aplicaciones más recientes relacionadas con la aplicación de estos métodos a la gestión de proyectos que se han localizado se encuentran las siguientes:

- En [BdOBW08] se aplica los métodos *branch and bound* a generar equipos de trabajo para proyectos de desarrollo software

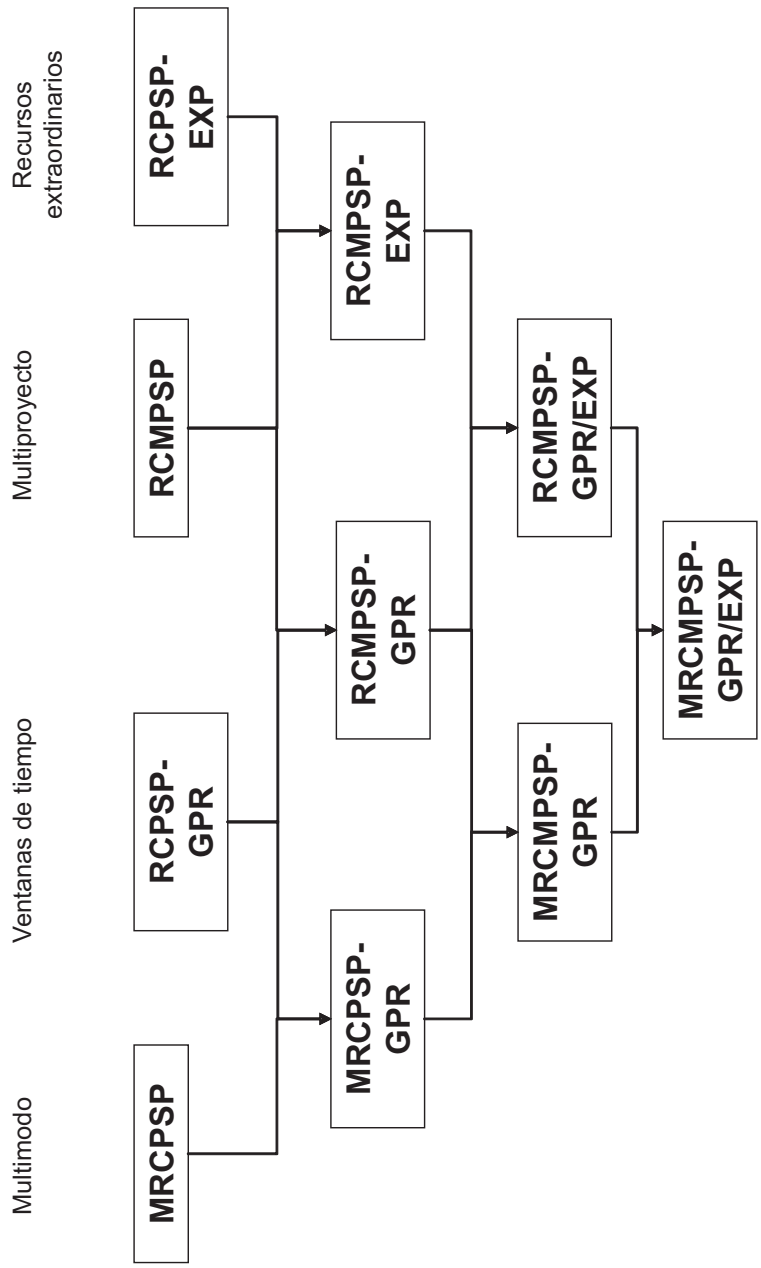


Figura 2.1: Algunas extensiones del problema RCPSP.

- En [SM07b] se construye un modelo para decidir sobre la aceptación o no de pedidos para su posterior secuenciación
- En [BZ06] se toma los métodos *branch and bound* como punto de partida para un heurístico de secuenciación de tareas en un entorno de manufactura virtual

Esta última aplicación, en la que el método es el punto de partida de un procedimiento heurístico, es significativa porque, si bien la potencia del *branch and bound* es grande, la complejidad de los problemas RCPSP ha llevado a que la investigación se centre en métodos aproximados en lugar de perseguir los métodos exactos. Esto es lo que se trata en los siguientes apartados. Las variantes sofisticadas del problema (MRCPSP, MRCMPSP, ...) son todavía más complejas, por lo que prácticamente todas se abordan con metodologías heurísticas.

2.4.3. Métodos heurísticos y metaheurísticos para la solución de los problemas RCPSP

Los métodos heurísticos aplicados en el problema RCPSP caen básicamente dentro de dos categorías, los métodos *constructivos* y los métodos de *mejora*.

Los métodos *constructivos* se centrar en construir una secuencia de actividades factible, es decir que cumpla las restricciones tanto de precedencia como de recursos. Para ello se parte de una primera secuencia "vacía" a la que se van incorporando actividades de acuerdo con una serie de *reglas de prioridad* y cumpliendo con las restricciones para así contar con una *secuencia viable*.

Los métodos de *mejora* se centran en obtener mejores soluciones a partir de contar con esa secuencia viable de actividades. Para ello realizan una *búsqueda local* en torno a la misma que conduzca a un óptimo local. Con el fin de eludir que éste sea subóptimo en términos del espacio total de soluciones, a veces los procedimientos de búsqueda permiten "deteriorar" temporalmente la solución. Esto, a su vez, puede dar lugar a la aparición de *ciclos*. Con el fin

de evitarlo se emplean técnicas *metaheurísticas* desarrolladas en el terreno de la inteligencia artificial como algoritmos genéticos, búsqueda tabú, recocido simulado, etc.

Construcción de secuencias

Una secuencia o solución S_n es un vector de $n+1$ fechas s_i de inicio o de final f_i de las *n+1* actividades del proyecto, incluyendo las actividades ficticias inicial, 0, y final, *n*. Se denomina longitud o duración de la secuencia, *T(S)*, al plazo de terminación del proyecto si se sigue esa secuencia, es decir, a la fecha de terminación de la actividad final. Una *secuencia factible* es aquella que respeta las restricciones de tiempo, precedencia y recursos. Dada una secuencia *S*, se denominan actividades *activas* en un momento *t* al conjunto *A(t)* de actividades que están ejecutándose en ese momento de seguirse esa secuencia.

Para la construcción de secuencias se utilizan los denominados *esquemas de generación se secuencias* (SGS)[Bal03]. Los esquemas de generación de secuencias se agrupan básicamente en dos categorías, *serie* y *paralelo*. Las primeras parten de una secuencia vacía y van iterando *incorporando actividades*. Las segundas, partiendo también de una secuencia vacía realizan una *iteración temporal*.
El algoritmo general de secuenciación en serie se realiza en tantas etapas como actividades hay que secuenciar. En cada etapa se diferencian dos conjuntos; *S*, actividades ya secuenciadas y *E*, conjunto de actividades elegibles, es decir actividades que aun no han sido secuenciadas y cuyas predecesoras ya lo han sido. Igualmente se determina en cada momento *t*, *D*, la disponibilidad de recursos, es decir, la cantidad de recursos que no está siendo usada por las actividades activas en ese momento. Por último se determina *F*, el vector de los tiempos de finalización de las actividades comprendidas en *S*.

Las diferentes implementaciones de este algoritmo se diferencian por el criterio de selección de la siguiente actividad elegible.

Por su parte el algoritmo de secuenciación en paralelo se basa en iterar en el tiempo. A cada paso de interación le corresponde un tiempo t_g. En cada

momento hay una serie de actividades ya completadas, C, otras activas, A, y otras elegibles definidas como en el caso anterior. Una vez más, aquí las diferentes implementaciones dependen del criterio elegido para incorporar la nueva actividad.

Cualquiera de los dos procedimientos permite obtener secuencias óptimas en el problema no restringido en los recursos. Sin embargo no es así en el caso general.

Teniendo en cuanta la limitación de recursos, las secuencias obtenidas se denominan *secuencias activas* o *justificadas a la izquierda* si ninguna de las actividades que comprende puede iniciarse antes de la fecha secuenciada sin provocar un retraso en otra actividad.*Justificada a la derecha*, en cambio, es el caso en el que el retraso de cualquiera provoque un retraso en alguna de las demás. Si se permite la suspensión temporal de una actividad activa, una secuencia *sin retraso* es aquella que incluso en ese caso da lugar a un retraso si se adelanta. En [Kol96] se demuestra que el algoritmo serie construye secuencias activas (justificadas a la izquierda) mientras que el paralelo construye secuencias sin retraso.

El conjunto de todas las secuencias sin retraso en un subconjunto de las secuencias activas por lo que "a priori" el algoritmo paralelo construye un espacio de búsqueda para la mejora posterior de una cardinalidad menor. El problema radica en que se demuestra que puede dejar fuera la(s) solución(es) óptima(s) para un problema con función objetivo regular, cosa que no ocurre con el algoritmo serie. De ahí que hayan aparecido diversos algoritmos mixtos, serie-paralelo, que buscan aprovechar las ventajas de ambos métodos [KP01] [KH06].

Reglas de prioridad

Tanto en el algoritmo serie como en el paralelo se requieren reglas para seleccionar las actividades elegibles que se incorporan a la secuencia. El modo más habitual de hacerlo es mediante las *reglas de prioridad*.

Una regla de prioridad se caracteriza por los siguientes elementos:

- Una función real para asignar un valor a cada actividad, el *valor* de la

prioridad

- Un criterio de *extremos*, máximo o mínimo
- Un procedimiento para el *desempate* entre dos actividades con igual valor de la prioridad, que habitualmente es simplemente el valor de la etiqueta numérica asignado a cada actividad

Las reglas, a su vez, pueden ser:

- *Locales o globales*, según se calculen con valores exclusivamente asociados a la actividad en consideración o empleen valores asociados a otras actividades
- *Estáticas o dinámicas*, según se mantengan constantes o varíen en el tiempo de ejecución del algoritmo
- Referidas a la *topología* de la red del proyecto, a los *tiempos* o a las *recursos*
- Por último, algunas son aplicables a los dos tipos de algoritmos, serie y paralelo, y otras son adecuadas sólo para alguno de ellos.

Entre las reglas más corrientes se encuentran las que se presentan en la tabla 2.3, extraída de [LTB06].

La representación de las soluciones

Con el fin de poder aplicar los procedimientos heurísticos basados en reglas de prioridad así como los métodos más avanzados que se exponen más adelante, se hace necesario contar con una forma de representar las soluciones del problema RCPSP.

En la literatura [Bal03] se proponen diversos modos de representación, cada uno de los cuales se adapta en mejor o pero grado a las diferentes metodologías de generación y mejora existentes. Básicamente son las siguientes:

Lista de actividades. Una lista de actividades es una permutación de las actividades $\lambda = (j_1, j_2, ..., j_n)$ que es compatible con las prioridades. En

Regla	Extremo	Valor de la prioridad
Activity number (AN)	min	j
Earliest finish time (EFT)	min	EFT_j
Earliest start time (EST)	min	EST_j
Greatest rank positional weight (GRPW)	max	$dj + (\sum d_i \forall i \in IS_j)$
Greatest resource demand (GRD)	max	$dj(\sum r_{jk})$
Longest processing time (LPT)	max	d_j
Latest finish time (LFT)	min	LFT_j
Latest start and finish time (LSTLFT)	min	$LST_j + LFT_j$
Latest start time (LST)	min	LST_j
Minimum free slack (MINFREE)	min	$(EST_i \forall i \in IS_j) - EFT_j$
Minimum total slack (MINSLK)	min	$LST_j - EST_j$
Number of immediate successors (NIS)	max	IS_j
Remaining work (RWK)	max	$d_j + (\sum mind_i \forall i \in AS_j)$
Shortest processing time (SPT)	min	d_j

Cuadro 2.3: Algunas de las reglas de prioridad más conocidas.

general una secuencia activa puede admitir más de una representación en términos de lista de actividades.

Vector de prioridades. Un vector de prioridades o *random key* es un vector que asigna a cada actividad un valor real correspondiente a la prioridad asignada por la regla correspondiente. Los algoritmos de secuenciación serie van eligiendo actividades de acuerdo con esos valores (máximo o mínimo, según el caso). Cada secuencia admite infinitas representaciones en términos de vector de prioridades.

Vector de reglas de prioridad. Un vector de reglas de prioridad aplica a cada actividad a ser secuenciada una regla de prioridad distinta. Este modelo se adopta cuando se verifica que según las características del proyecto no existe una regla de prioridad que domine claramente a las demás. Sin embargo esta representación puede no incluir a todas las secuencias posibles y, por tanto, excluir las óptimas.

Vector de traslación. Un vector de traslación, *shift vector*, asigna a cada actividad un retraso a su fecha más temprana de inicio respecto de la de finalización de la última predecesora. El problema de esta representación radica en que en algunos casos proporciona soluciones no factibles en términos de recursos. Por otra parte sí recogerá a todas las soluciones y, dentro de ellas, a las óptimas.

Vector de tiempos de inicio. En este caso la representación de la solución es la solución misma. Por tanto existe una representación unívoca de todas las soluciones.

Esquema de secuencias. Un esquema de secuencias contiene cuatro conjuntos disjuntos de pares de actividades. Las *conjunciones* son los pares de actividades en los que la primera debe finalizar para que comience la segunda. Las *disyunciones* son aquellos pares de actividades que no pueden coincidir en el tiempo. Las *relaciones paralelas* son aquellas en las que las actividades deben coincidir al menos en un periodo de tiempo y las *relaciones flexibles* todos aquellos pares que no están sometidos a

ninguna restricción. Esta representación exhaustiva del proyecto puede originar un problema NP-duro simplemente para determinar si existe una secuencia factible dentro de un esquema de secuencias.

Métodos heurísticos y metaheurísticos para la solución del problema RCPSP

Dadas las características de complejidad del problema, el RCPSP es objeto de constante investigación en la aplicación de todo tipo de métodos y técnicas para su resolución. En la literatura existen amplias revisiones de los métodos aplicados tales como [KP01] [KH06] [HRD98] [DH02]. Nos limitaremos en este punto a comentar los métodos más empleados y poner algunos ejemplos.

Métodos multi-paso. Un método de mejora de las soluciones obtenidas por la generación de secuencias empleando reglas de prioridad es el de someter al grafo del proyecto a una secuenciación múltiple que a su vez puede producirse de dos formas:

Secuenciación hacia adelante y hacia atrás. Una vez generada la secuencia en el modo descrito anteriormente, bien serie o paralelo, con una determinada regla de prioridad, se secuencia el grafo inverso a partir de una cota superior de la duración, justificando hacia la derecha y, desplazando luego la solución hacia el momento t=0 ya que, generalmente, la secuenciación hacia atrás dejará una holgura.

Secuenciación bidireccional. Se secuencian la actividad inicial a 0 y la final a una cota superior del problema. En cada paso de la iteración se determina si secuenciar hacia adelante o hacia atrás y la actividad que debe secuenciarse aplicando una regla de prioridad.

Muestreo aleatorio. En lugar de aplicar una regla de prioridad estricta, la actividad a secuenciar se obtiene mediante muestreo de las actividades elegibles, a su vez, en varias formas posibles:

Muestreo aleatorio puro. Se extrae aleatoriamente una actividad a secuenciar entre todas las posibles.

Muestreo sesgado La probabilidad de que una actividad sea seleccionada de acuerdo con una probabildiad que es proporcional al valor de la prioridad en una regla elegida de entre las habitualmente empleadas

Muestreo sesgado con penalización. La probabilidad se basa no en el valor de la prioridad de cada actividad sino en el coste de oportunidad - *regret* - de no seleccionarla.

Metaheurísticas clásicas. Entre las metaheurísticas clásicas que se exponen en la literatura se encuentran varias muy conocidas como:

Búsqueda local. Los métodos de búsqueda local evalúan la función objetivo en un entorno definido en el espacio de las soluciones en relación a una primera solución factible. Se "desplazan" a cualquier otro punto de ese entorno en el que la función objetivo mejore. Ese desplazamiento puede determinarse de varias formas: bien a la mejor solución posible después de realizar una exploración completa del entorno (*best fit strategy*) o bien a la primera que se encuentre que mejore al valor actual de acuerdo con un orden de búsqueda predeterminado (*first fit strategy*). En ambos casos el desplazamiento se detiene cuando no existe posibilidad de mejorar.

Recocido simulado. Esta técnica se denomina así por analogía con el tratamiento térmico al que se refiere su nombre. Básicamente es una mejora de la estrategia *first fit* con la variante de que admite la posibilidad de desplazarse a una solución peor que la actual con una probabilidad dada. Esa probabilidad se va reduciendo conforme el algoritmo va ejecutándose, por lo que se le denomina *temperatura* prosiguiendo con la analogía. El permitir que se opte por soluciones que no mejoran el objetivo pero que abren nuevos entornos a la evaluación persigue evitar que el algoritmo quede atrapado en un óptimo local. El descenso de la temperatura persigue acotar la recursión.

Búsqueda tabú. Esta técnica es a su vez una mejora de la estrategia *best fit*. Elije la mejor solución en un entorno de la de partida aunque aquella sea peor que esta primera. Con el fin de no entrar en ciclos mantiene una lista de las soluciones ya visitadas que pasan a estar prohibidas en los próximos desplazamientos. Esta lista tabú es la que da el nombre al procedimiento. La lista tabú es limitada por lo que una solución puede "salir" de ella cuando hayan pasado tantas iteraciones como se establezca para determinar la longitud de la lista. Además de esta memoria a corto plazo se mantiene también una memoria de plazo largo en el que se conservan soluciones indicativas de regiones "atractivas" ya exploradas y de regiones inexploradas que permiten diversificar la búsqueda. Un procedimiento denominado reencadenamiento de trayectorias permite orientar los desplazamientos hacia esas regiones.

Algoritmos genéticos. Esta técnica, al contrario que las técnicas descritas anteriormente donde se itera de solución en solución, opera con una población de soluciones. A partir de una población inicial se cruzan los distintos pobladores y se someten a mutaciones análogamente a los procesos de la evolución biológica. Una función de adaptación - *fitness* - evalúa a cada elemento y determina los supervivientes de la siguiente generación que vuelven a recombinarse y mutar. La función de adaptación se basa, obviamente, en la función objetivo del problema. Los algoritmos genéticos pueden ser muy diversos según la forma de codificación de las soluciones - genes - y los procedimientos de combinación y mutación que se adopten.

Búsqueda dispersa. La búsqueda dispersa - *scatter search* - genera una colección de soluciones diversas que almacena en un conjunto de soluciones de prueba. Posteriormente con técnicas de mejora local genera una o varias mejoras de esas soluciones iniciales. A continuación extrae de entre ellas un subconjunto reducido de las mejores soluciones que se denomina conjunto de referencia. Particiona el conjunto de referencia en varios subconjuntos, normalmente de dos elementos, para generar soluciones combinadas. De manera análoga a los algoritmos genéticos

combina dichas soluciones para obtener un nuevo conjunto de soluciones de prueba.

Colonias de hormigas. El método de la colonia de hormigas se basa en modelos provenientes de la vida artificial. Una generación de hormigas artificiales explora el espacio de las soluciones siguiendo criterios probabilísticos y, eventualmente, heurísticos específicos del problema. Las aristas que conducen a las mejores soluciones son marcadas por *feronomas* de manera que las sucesivas generaciones de hormigas retoman la búsqueda en las regiones más prometedoras exploradas por las generaciones precedentes.

Otras metaheurísticas descritas en la literatura. La literatura sobre métodos de solución del problema RCPSP es extensísima. Además de las ya citadas metaheurísticas se presentan aplicaciones no convencionales basadas en la combinación de varias de las ya descritas y de otras modalidades tales como las técnicas de enjambre - *swarm* - , la relajación lagrangiana, el heurístico electromagnético y, por supuesto, heurísticos basados en búsqueda limitada a partir de los métodos exactos de *branch and bound* y cortes.

Resultados computacionales

Una amplia revisión de los diversos métodos aparece en [KH06]. Contrastados con una librería estándar de problemas generados aleatoriamente, denominada PSPLIB, que se usa como referencia entre los estudiosos del problema, los mejores algoritmos resultan ser los que reúnen estas características:

- Generación de secuencias en *serie*,
- Representación de las soluciones como *lista de actividades*,
- Metaheurísticos *mixtos* que combinan algoritmos genéticos u otros mecanismos basados en la población con otros sistemas de mejora (multipaso, autoadaptación, ...),

Referencia	Técnica	Problema
[GMR08]	Algoritmo genético	RCMPSP
[MGR08]		RCPSP
[KJR06]		RCMPSP
[LCM04]		RCPSP
[AM01]		RCPSP
[Har98]		RCPSP
[APH04]		RCPSP
[VBQ08]		RCPSP
[BZ06]	Branch and Bound	RCPSP
[LTN06]		RCPSP
[RRK08]	Búsqueda dispersa	RCPSP
[DRLV06]		RCPSP
[PAM04]	Búsqueda local	RCPSP
[Wal08]	Busqueda tabú	DCFPSP
[PHC08]		RCPSP
[MWW08]		MRCPSP
[DER98]		RCPSP
[SM07a]	Colonias de hormigas	RCPSP
[TC06]		RCPSP
[LJF04]	Cortes mínimos	RCPSP
[DV06]	Electromagnetismo	RCPSP
[Sho06]	Muestreo aleatorio	RCMPSP
[VBQ05]	Multipaso	RCPSP
[TL03]		RCPSP
[CH08]	Multipaso, muestreo aleatorio sesgado	RCPSP
[GC03]	Primal-dual	RCPSP
[JMR+01]	Recocido simulado	MRCPSP
[Boc96]		RCPSP
[ASA06]		RCPSP
[BL03]		MRCPSP
[XMNU08]	Reglas de prioridad	RCPSP
[LTB06]		MRCPSP
[HK05]		MRCPSP
[JDSR08]	Swarm	RCPSP

Cuadro 2.4: Referencias de metaheurísticos en la literatura para diversos problemas del tipo RCPSP.

Para problemas de dimensión más reducida (hasta 1.000 secuencias posibles) el sistema de secuenciación hacia adelante y hacia atrás (*forward-backward improvement* combinado con el muestreo aleatorio puro proporciona resultados muy semejantes a los obtenidos con los metaheurísticos clásicos siendo un método mucho más sencillo. De hecho, para el subconjunto de problemas J120 (120 tareas) resulta incluso mejor que los mejores metaheurísticos.

En cuanto a las reglas de prioridad, muy empleadas en la práctica profesional para problemas de dimensión más reducida donde las técnicas metaheurísticas no se emplean más que en casos muy contados, en [BY06] se presenta un estudio en el contexto RCMPSP que arroja los siguientes resultados:

- Para proyectos *poco complejos*, es decir con una relación arcos/nodos relativamente reducida, la regla SASP ofrece los mejores resultados en términos de *retraso medio* de los proyectos que componen la cartera.
- Si el *nivel de complejidad es elevado* la mejor regla es TWK-LST siempre hablando desde el punto de vista del *retraso medio*.
- Si lo que se pretende es *minimizar el máximo retraso*, que es el punto de vista de los responsables de la cartera de proyectos, frente al punto de vista anterior que correspondería a los jefes de los proyectos individuales, la mejor regla resulta ser la MINWCS.

Las reglas citadas como más eficientes, que corresponden al entorno multiproyecto y por tanto son más complejas que las antes expuestas referidas al entorno de un proyecto único, se aplican siempre con esquemas de generación de secuencias *serie* a *lista de actividades* y su contenido es el siguiente:

SASP ,*shortest activity from the shortest project*; se selecciona de entre las actividades elegibles la actividad i del proyecto l que tiene el menor valor de $CP_l + d_{il}$. En caso de empate se elige la de número más bajo.

TWK-LST , *total work contain + latest early start*; se selecciona la tarea con más contenido de trabajo pendiente, desempatando si es necesario con la que tiene la fecha más tardía de inicio menor.

MINWCS ,*minimum worst case slack*; se selecciona la tarea que cumple la siguiente relación:

$$\text{mín}\,(LS_i - \text{máx}\,[E_{ij} \forall (i,j) \in AP_t])$$

donde E_{ij} es el tiempo más temprano que puede empezar la actividad j si la actividad i ha empezado en t y AP_t es el conjunto de todos los pares de actividades que pueden empezar en t y no están programadas. Esta regla, en ausencia de restricción de recursos equivale a la regla de mínima holgura, MINSLK.

2.5. El problema del riesgo

2.5.1. Riesgos externos y riesgos internos

Los métodos de solución del problema RCPSP descritos en el capítulo anterior corresponden al caso determinista. Sin embargo la mayoría de los problemas reales son problemas en un entorno de *incertidumbre*. Elmaghraby [Elm05] sostiene que la incertidumbre en el terreno de los proyectos se arraiga en *dos dominios* diferentes. En primer lugar está el riesgo *externo* a las actividades en casos tales como las condiciones climatológicas, el absentismo laboral o el fallo de la maquinaria en un proyecto de construcción. El segundo dominio es el del riesgo *interno*, el derivado del desconocimiento de todos los extremos concernientes a las actividades a realizar y, en particular, el problema de estimar acertadamente el contenido de trabajo, *el esfuerzo*, que conllevan. Según este autor lo tradicional ha sido prever para el primer tipo de contingencias dando por supuesto que las estimaciones que se hacen son suficientemente exactas. Sin embargo en muchos proyectos, especialmente con aquellos de elevado contenido de trabajo intelectual creador o complejo (tales como los proyectos de I+D, de desarrollo de nuevos productos, o, nuestro caso, el desarrollo de *software*), son los factores internos los que desempeñan el papel dominante.

La manera de afrontar comúnmente este tipo de proyectos intenta estimar

por intervalos el esfuerzo requerido por las diferentes tareas (entre x e y persona-mes, p.e.). Si no se dispone de mejor información se puede adoptar el criterio de que el esfuerzo así estimado es una variable aleatoria que se distribuye uniformemente en ese intervalo.

Con esta incertidumbre por delante, el responsable del proyecto aún tiene que decidir sobre los recursos que deberá destinar al proyecto. *La duración del proyecto acaba siendo pues no la fuente de la incertidumbre sino la consecuencia de las decisiones de asignación.* Este punto de vista determina una primera manera de abordar el proyecto en condiciones de riesgo: *se trata de decidir la asignación de recursos a las tareas que asegura el cumplimiento de los plazos, preferiblemente a coste mínimo.*

2.5.2. Estrategias para hacer frente al riesgo de los proyectos

El reconocimiento de que la incertidumbre se encuentra en el corazón de la propia planificación ha dado lugar a diferentes líneas de investigación en el terreno de la Gestión de Proyectos en condiciones de incertidumbre. Básicamente existen dos enfoques [HL04b]:

- aquellos en los que no se cuenta con una programación inicial sino que las actividades se van secuenciando conforme van "apareciendo"
- aquellos en los que, partiendo de una programación inicial, se plantea una estrategia específica para tener en cuenta la incertidumbre. Tradicionalmente en este segundo caso las estrategias han sido de dos tipos:
  - estrategias *proactivas*
  - estrategias *predictivas-reactivas*

Una estrategia *proactiva* implica el desarrollo de un programa base, o inicial, *baseline schedule*, que esté protegido lo más posible frente a las interferencias y disrupciones que puedan producirse durante la ejecución del proyecto. Por el contrario, una estrategia *reactiva* se basa en la revisión, y eventual reprogramación, del proyecto cuando un evento altera la secuencia

o duración inicialmente previstas. Una estrategia *predictiva-reactiva* es una estrategia que programa el proyecto previendo la posibilidad de que tenga que ser reprogramada.

En el caso en que no se cuenta con programa base nos encontramos ante *un caso extremo de estrategia reactiva*, pues en la práctica equivale a reprogramar -implícita y parcialmente - el proyecto ante la aparición de cada nueva actividad. Por analogía con los problemas de taller, a esta práctica se le llama una regla o política de despacho, *dispatching rule*.

Un concepto fundamental en el contexto en el que se cuenta con un programa base es el de *secuencia crítica* [WL77] o *cadena crítica* [Gol97] que de algún modo generaliza el concepto de camino crítico - *critical path* - propio de los modelos tradicionales PERT/CPM. Se trata de una secuencia de actividades interrelacionadas, bien por relaciones de precedencia bien por relaciones impuestas por la asignación los recursos, de tal manera que la misma determina la duración del proyecto total. De este modo se habla de la *criticalidad* de una tarea como la probabilidad de que pertenezca a una cadena crítica. La programación óptima de un proyecto restringido por los recursos calculada en condiciones deterministas puede resultar manifiestamente subóptima en la ejecución real a consecuencia de la existencia de este tipo de secuencias críticas. En tal caso se dice que es una solución *poco robusta*.

La *robustez*, también denominada *estabilidad* de una solución es una medida de la insensibilidad del tiempo de comienzo de las actividades a las variaciones de los datos de entrada. El término robustez es empleado en dos contextos: *robustez en el espacio de las soluciones* y *robustez en el espacio de los objetivos*. En el segundo caso se le denomina también *robustez cualitativa*. Una estrategia proactiva persigue construir soluciones robustas.

Un concepto próximo al de robustez es el de *flexibilidad* de las soluciones. Una solución es flexible si puede ser *reparada con facilidad* sin merma de su calidad en términos de los objetivos [gdf02]. Cuando se emplea una estrategia predictiva-reactiva, la flexibilidad de la solución obtenida es la característica más deseable.

En función de lo descrito hasta aquí existen diversas metodologías para enfocar el problema que se describen en los apartados que siguen a continua-

Enfoque	Estrategias frente al riesgo
Proactivo	Cadena Crítica Generación de secuencias estables Generación de secuencias robustas
Predictivo - Reactivo	Reparación de secuencias Reprogramación Secuenciación contingente Aceleración de actividades (crashing)
Totalmente Reactivo	Secuenciación estocástica Multi-armed Bandits Teoría de Colas generalizada

Cuadro 2.5: Estrategias frente al riesgo.

ción.

2.5.3. Enfoques proactivos

El enfoque proactivo se basa en generar una programación base que incorpore un grado de anticipación suficiente para hacer frente a la variabilidad que pueda surgir en la ejecución del proyecto. Es el campo de interés de muchos esfuerzos de investigación tanto en el terreno académico como en el profesional.

La Cadena Crítica.

La programación basada en la cadena crítica es la aplicación de la teoría de las restricciones - *theory of constraints*- de Goldratt [Gol97] a la gestión de proyectos es una de las líneas más conocidas. También conocida como *buffer management*, consiste en síntesis en aplicar los siguentes pasos:

- Construir una programación base *agresiva*, es decir, sin conceder holgura ni margen a la duración de las actividades, empleando la media o la mediana de dicha duración,

- No tener en cuenta ni hitos intermedios ni fechas-límite,

- No programar en modo multitarea, es decir, no permitir que un recurso se dedique a más de una actividad a la vez
- Programar las actividades del camino crítico ajustadas a la derecha, es decir, a partir de la fecha más tardía de inicio,
- Minimizar el trabajo en curso, WIP, *work in progress*, desplazando a la izquierda las actividades si es preciso para eliminar los conflictos sobre recursos,
- Identificar la *cadena crítica* como la secuencia de actividades que determina la duración del proyecto
- Colocar al final de la cadena crítica la reserva de tiempo que no fue considerada a la hora de construir la programación, esa reserva de tiempo se denomina el *buffer* del proyecto y se suele fijar en el 50 % de la duración total,
- Incluir un *buffer* de alimentación, *feeding buffer*, en el enlace entre actividades no críticas y la cadena crítica, también calculado normalmente como el 50 % de la duración de la secuencia que "desemboca" en la cadena crítica,
- Por último, incluir un *buffer* de recursos, *resource buffer*, en cada nodo de la cadena crítica en la que se produzca un cambio en los recursos empleados; este servirá a modo de aviso.

Tras esta programación base se construye una *programación de proyecto* basada en el inicio más temprano posible, ajustando a la izquierda, y sin considerar los *buffers*. La implementación del procedimiento consiste en desarrollar la programación de proyecto utilizando la programación base como referencia para tomar medidas correctivas. De abí la denominación de *buffer management*. En [HLD02] se analiza críticamente esta metodología señalando sus ventajas e inconvenientes.

En un *entorno multiproyecto* la metodología que se aplica, basada en los principios generales de la teoría de las restricciones, es la siguiente:

1. Priorizar los proyectos de la organización,
2. Programar cada proyecto de acuerdo con el procedimiento antes indicado en función del recurso más restrictivo, el "cuello de botella",
3. Escalonar los proyectos de acuerdo con la prioridades y el recurso más restrictivo,
4. Insertar *tambores*, reservas o *buffers* del recurso más restrictivo entre proyectos.

De esta forma la gestión, y de manera análoga al caso de un sólo proyecto, se convierte en una gestión de los diferentes *buffers*, de tiempo y de recursos. En la literatura hay diversas discusiones sobre la idoneidad o no de concentrar las holguras mediante esta política de *buffers*. Veáse, por ejemplo, [dVDHL05].

Generación de secuencias robustas.

Al margen del procedimiento de la cadena crítica, el más conocido en medios profesionales, existen otros procedimientos que se describen en la literatura que se basan en el concepto de *estabilidad*. Cuando la restricción de recursos no opera rígidamente, es decir, cuando se pude recurrir de forma extraordinaria a recursos adicionales en caso de una disrupción de la programación base, la estabilidad es el objetivo deseable con el fin de reducir la excepcionalidad.

La medida de la estabilidad puede plantearse de dos maneras. Una de ellas es la *estabilidad promedio*, medida como la esperanza matemática de la suma ponderada de las desviaciones del tiempo de comienzo de las actividades cuando se produce una variación en el plazo de inicio de una de ellas. El problema de minimización que se plantea se pude abordar de varias formas diferentes:

- Como un problema de flujo a coste mínimo [HL04a].
- Como un problema de programación lineal paramétrica [TFC98].

En algunos casos, lo que puede interesar es asegurar la estabilidad en *el peor de los casos*, es decir, asegurar la *calidad* de la programación mediante una cota superior a la máxima disrupción. Este es el procedimiento que se describe en [gdf02].

En la situación citada, se puede recurrir a recursos de forma extraordinaria y el problema siempre se resuelve resecuenciando a partir de la alteración a la fecha más temprana. Cuando eso no es posible se comprueba la existencia de soluciones factibles una vez producida la disrupción utilizando un modelo análogo al empleado en la programación de la producción *just in time* [LN94].

Generación de secuencias con lógica borrosa

. Una tercera aproximación al problema de generar secuencias robustas es el de modelar el problema como un problema "fuzzy". Aunque aparentemente semejante, la idea que hay detrás de la aplicación de la lógica borrosa no es que las duraciones sean variables aleatorias, sino que la incertidumbre es *epistémica*: no se conoce el contenido de trabajo con precisión suficiente. La *función de pertenencia* de Zadeh se emplea tanto para modelar aquellas variables que no se conocen con precisión como aquellas otras que son imprecisas en si mismas, como la productividad, por ejemplo. Los métodos del álgebra "fuzzy" permiten tratar los problemas generalizando muchos de los métodos diseñados para los problemas *crisp*.

En [LO08] se presenta un modelo que relaciona el método de la cadena crítica con la lógica borrosa, utilizando esta para determinar el *buffer* de proyecto. En [KYY$^+$05] los autores proponen un algoritmo genético híbrido con un controlador de lógica borrosa para el problema multiproyecto - RCMPSP. En [San00] se presenta un modelo de apoyo a la decisión - DSS - que emplea la lógica borrosa para asignar recursos, en este caso ingenieros, a tareas con diferentes niveles de complejidad y duración "fuzzy", en función de la habilidad de los recursos y la complejidad de las tareas, también modeladas como variables borrosas.En [OA01] se propone un modelo parecido.

2.5.4. Enfoques predictivos-reactivos

Los enfoques predictivos-reactivos, al contrario que los anteriores, se basan en la modificación de la programación base como respuesta, reacción, a las disrupciones. Hay diferentes estrategias que van desde la más simple, la *reparación de secuencias*, hasta la *reprogramación completa* del proyecto a la vista de la disrupción.

Reparación de secuencias.

La reparación de secuencias en su modalidad más simple se basa exclusivamente en el desplazamiento hacia la derecha de las actividades que se ven afectadas por las disrupciones, bien por verse afectadas por el recurso que ha fallado o bien porque están condicionadas por las relaciones de precedencia. Es una estrategia reactiva que genera resultados pobres en general.

En [AR00] los autores plantean una modalidad más sofisticada del problema de reprogramar un proyecto cuando se produce una disrupción en un modelo multimodo a partir del problema de insertar óptimamente una nueva actividad inesperada en un determinado momento del tiempo, de manera que el tiempo total de terminación varíe lo menos posible. Utilizando una representación en forma de grafos de recursos, generan cortes en el grafo que posibilitan seleccionar la mejor inserción empleando un sistema de generación de secuencias en serie - SGS - y una regla MINSLACK. Posteriormente en [AMR03] mejoran la solución con un procedimiento de búsqueda tabú.

Reprogramación.

La reprogramación o *regeneración* de secuencias, como se denomina en el ámbito de los problemas de taller, emplea una medida determinista de los objetivos, típicamente el tiempo hasta la terminación, para reconstruir una nueva programación a partir de la situación de hecho en la que se produce la disrupción. En cierto sentido, la reparación de secuencias que se ha comentado en el punto anterior es como un *paso heurístico* en un algoritmo de reprogramación.

Dado que en este caso la programación resultante puede diferir extraordinariamente de la solución inicial, el problema se plantea como uno de *inestabilidad*. En el contexto de una organización multi-proyectos, la estabilidad es necesaria para eludir el problema que en el campo de los problemas de taller se denomina el *nerviosismo de la secuencia*. De ahí que en la práctica sea preferible muchas veces reparar la programación antes que alterarla completamente.

Es posible modelar explícitamente este objetivo de prevenir la inestabilidad buscando una *estrategia de mínima perturbación*. Para ello se puede minimizar bien la diferencia entre las fechas de inicio previstas inicialmente y las reprogramadas o el número de actividades a realizar por diferentes unidades de recursos. Cabe distinguir además dos momentos en la reprogramación que dan lugar a situaciones distintas. En unos casos se trata de reasignar una fecha de inicio a actividades que no han comenzado. En otros hay que reasignar recursos y tareas sobre la marcha, una vez iniciadas las actividades.

Una metodología empleada para enfrentar este problema se basa en la programación por metas - *goal programming* - aplicada al problema MRC-PSP. Se asignan metas a clases de actividades agrupadas por prioridad. Una de las metas es minimizar el número de actividades que se modifican. El problema se resuelve aplicando la búsqueda tabú [CDM$^+$02].

Secuenciación contingente.

Puede ocurrir que sea la propia dirección del proyecto la que realice cambios en la programación del mismo en el tiempo de su ejecución. En este caso, el problema interesante es presentar los cambios posibles con menor impacto en el resultado final en términos de duración. Una forma de abordarlo es la de generar para cada recurso una *secuencia de grupo*, es decir, un conjunto de tareas total o parcialmente ordenadas vinculadas a ese recurso. A partir de cada secuencia de grupo se pueden construir programaciones del proyecto total de modo que la dirección del proyecto no cuenta con un único programa posible sino con varios[BDR02].

Aceleración de actividades - *activity crashing*.

Una alternativa para tomar acciones correctivas en la ejecución del proyecto es la de acelerar la misma concentrando recursos en las actividades que van retrasadas. Este problema ha sido estudiado en la literatura como el problema del *trade-off* tiempo-coste. En [DH02] está ampliamente tratado. En el contexto de los proyectos de sistemas de información, y en otros, se trata extensamente sobre las paradojas que pueden derivarse en el conocido problema del *mythical man-month*.

2.5.5. Enfoques totalmente reactivos

Los enfoques totalmente reactivos son el campo del problema SRCPSP - *stochastic resource constrained project scheduling problem*. No se parte de una programación base, sino que se plantea como un *problema de decisión multietapa* que utiliza *reglas*, *políticas* o *estrategias de secuenciación* para decidir sobre la asignación de recursos a las actividades en el momento del inicio del proyecto y a la conclusión de cada actividad en base al pasado observado y al conocimiento a priori de que se dispone.

El SRCPSP no es más que una clase de la denominada programación estocástica, *stochastic scheduling*. El objetivo comúnmente considerado, la minimización del tiempo de terminación del proyecto para una clase de reglas o estrategias de secuenciación, es un caso particular del problema más general de asignar recursos a tareas con características aleatorias que demandan dichos recursos.

Teóricamente estos problemas pueden formularse en términos de *programación dinámica*, pero la aplicación directa de esta técnica resulta muy limitada en términos de efectividad debido a la denominada *maldición de la dimensionalidad*, termino acuñado por Richard Bellman, el "padre" de la programación dinámica [Bel72].

En la práctica eso hace que los modelos disponibles se hayan desarrollado para problemas muy específicos y que los resultados obtenidos no puedan generalizarse con facilidad a otros dominios, salvo algunas consideraciones muy generales, según Niño-Moura [NM01]. Según este autor los modelos más

estudiados pueden clasificarse en tres grandes categorías que han evolucionado autónomamente. Estas serían: las aplicaciones para la *secuenciación de tareas de duración estocástica*, los denominados *multi-armed bandit problems* y los *modelos generalizados de la Teoría de Colas*. En cada una de las áreas se observa la aparición de resultados en tres etapas:

- Un primer grupo de resultados en los que se definen un conjunto de *reglas de índices de prioridad* sencillas y fáciles de computar que se demuestra que permiten ofrecer resultados óptimos para problemas relativamente simples. La característica principal de esas reglas es la de que su valor para cada tarea depende de la propia tarea y su clase y no del resto.

- Un segundo grupo de resultados se ha alcanzado intentando generalizar esas políticas sencillas a casos más generales introduciendo supuestos técnicos adicionales y verificando la optimalidad en ese nuevo contexto.

- El tercer tipo de resultados se refiere a modelos más complejos para los que la optimización no parece estar al alcance de procedimientos y reglas sencillas. En estos casos los investigadores persiguen diseñar heurísticos que sean relativamente fáciles de implementar y que, generalmente, se basan en las reglas de los modelos más simples. La investigación del grado de (sub)optimalidad de estos heurísticos se realiza o bien de modo analítico o bien, de manera más corriente, mediante simulación.

2.5.6. La iteración en las redes de los proyectos, la arquitectura de los procesos y el riesgo en la estructura de descomposición en tareas

Las aproximaciones al problema del riesgo en los proyectos, la *variabilidad*, a la que antes se ha hecho referencia fundamentalmente parten de la idea de que la incertidumbre proviene del desconocimiento del contenido en trabajo de las actividades de los proyectos. Se trata, por lo tanto, de *acotar*

las consecuencias de ese desconocimiento en función de las posibilidades que la dirección de proyecto tiene de acotar su propio desconocimiento.

Las estrategias proactivas afirman la posibilidad de hacerlo, bien a través del juego de *buffers* del método de la cadena crítica, bien diseñando programaciones *estables* que absorban con poco coste la demanda de recursos extraordinarios para recuperar una programación que ha sido perturbada externamente. Las estrategias predictivas-reactivas reprograman, parcial o totalmente el proyecto en respuesta a una de esas perturbaciones. Las estrategias puramente reactivas renuncian a una programación base a priori e intentan, estudiando las características fundamentalmente estocásticas de las eventuales perturbaciones, encontrar reglas de prioridad robustas, que en este caso son más bien reglas de despacho de tareas.

Pero la complejidad de las interrelaciones entre las actividades en los proyectos, especialmente en los dominios en los que la ejecución cualitativa de las actividades afecta a todas sus sucesoras y la propia gestión del proyecto influye sobre esa calidad, exige dar un paso más para abordar el problema del riesgo. La manera más evidente en la que se produce esa complejidad es la que se deriva de la existencia de *iteraciones* en la ejecución del proyecto que *desmienten el carácter acíclico del grafo* con el que se modelan e invalidan, así, gran parte de las soluciones hasta ahora presentadas, salvo, lógicamente las puramente reactivas.

La segunda fuente de distorsión de los modelos de redes convencionales es la existencia de *eventos* que pueden dar lugar a *bifurcaciones* en la propia red. Según sean los resultados de una actividad, las siguientes deberán de hacerse de un modo u otro. A veces surgirán actividades nuevas no previstas inicialmente. Evidentemente, un programa inicial puede contemplar "a priori" todos esos posibles cambios en la topología de la red, pero a costa de aumentar extraordinariamente la complejidad de la misma y sin que sea evidente como deben tratarse actividades que a lo mejor deben hacerse o a lo mejor no con los métodos convencionales.

Básicamente se trata de incorporar al flujo natural de las actividades un *flujo complementario de información*. La decisión de "volver atrás" en la red para corregir errores o defectos en una actividad, o la de "optar" por un arco

u otro, se produce a partir de información sobre el propio proceso de ejecución del proyecto o sobre cambios sobrevenidos en su entorno. No es por tanto sólo un problema de desconocimiento o aleatoriedad en las variables que rigen la ejecución sino una dimensión adicional que hace que la propia ejecución de las tareas modifique la estructura de descomposición del proyecto en tareas. Desde el propio campo de la Gestión de Proyectos y desde otros campos se han hecho contribuciones a este problema que denominaremos el *riesgo* (derivado de la complejidad) *de la estructura de descomposición en tareas* - WBS. Señalaremos básicamente tres líneas que merecen ser reseñadas:

1. Los modelos de redes estocásticas y su evolución "natural", las Redes de Petri.

2. Los modelos basados en la *Design Structure Matrix* (DSM) provenientes del campo del desarrollo de producto (*product development*).

3. El modelo "cuántico" que emplea una analogía con el concepto de *densidad de probabilidad* de la mecánica cuántica para caracterizar el riesgo de los proyectos.

Las dos primeras líneas están orientadas básicamente a la simulación estocástica, esto es, a caracterizar el riesgo de los proyectos a partir de métodos no analíticos. Sin embargo presentan también un potencial analítico (alcanzabilidad de los estados en las redes de Petri, análisis estructural de las DSM, ...) que potencialmente puede ser de interés a la hora de concebir y diseñar la arquitectura de los procesos, en particular de la WBS.

La tercera línea, apenas una propuesta individual, introduce una representación del riesgo nada convencional que puede ser de interés en el mismo sentido, en el del riesgo inherente a la estructura. En el apendice C se describen con más detalle.

Capítulo 3

Trabajo relacionado: la simulación del proceso *software*

3.1. Introducción y contenido del capítulo

la simulación del proceso *software* es un fértil campo de investigación y propuestas de modelado. La complejidad de los procesos implicados se presta poco a un abordaje analítico por lo que se han aplicado diversas metodologías de simulación para estudiar toda clase de problemas. En los siguientes apartados se comentan con más detalle algunas de las aplicaciones más interesantes de estas metodologías. Están clasificadas en cinco grupos:

1. Aplicaciones más o menos basadas en el modelo de Abdel Hamid y Madnick [AM91] que aportan aproximaciones nuevas, bien por el área de interés que focalizan, bien por el esfuerzo de sintetizar los mecanismos básicos que gobiernan la dinámica de un proyecto *software*.

2. Aplicaciones que proponen modelos de simulación híbridos.

3. Aplicaciones orientadas específicamente al entorno multiproyecto.

4. Aplicaciones que tienen en cuenta la mejora de los procesos *software*.

5. Aplicaciones que incluyen alguno de los métodos de la gestión de proyectos revisados en el capítulo 2.

Referencia	Descripción
[LCWB08]	Diseño iterativo
[NZBS07]	Desviaciones de coste en líneas de producto *software*
[PAER07]	Estabilidad en el lanzamiento de versiones
[LM04]	Cadena Crítica y DS
[RHB03]	Participación y compromiso en equipos de desarrollo
[ARRRR02]	Reglas de gestión a partir de la simulación
[BWT02]	Six Sigma
[LKR02]	Complejidad en la evolución del *software*
[WH02]	Desarrollo global de *software*
[KLRW01]	Complejidad en la evolución del *software*
[RRT01]	Diseño colaborativo
[SG01]	Diseño colaborativo
[HH00]	Desarrollo de grandes sistemas
[MT00]	Estudios de caso
[PL00b]	Construcción de modelos
[PL00a]	Impacto de la volatilidad de los requisitos
[RKP+99]	Métodos empíricos y DS
[Abd93a]	Reutilización de componentes

Cuadro 3.1: Referencias de Dinámica de Sistemas no comentadas en el texto.

Además de las referencias descritas en lo que sigue, se han revisado las que aparecen en la tabla 3.1.

3.2. Modelos de simulación del proceso *software*

En la tesis de Mercedes Ruiz Carreira [Car03] se recoge una relación de trabajos de simulación del proceso *software* entre los años 1991 y 2000. En el origen de todos ellos está el trabajo de Abdel Hamid y Madnick [AM91] que supuso un exhaustivo intento de modelar el proceso de inspección formal con un grado de detalle muy elevado. Reúne las actividades relacionadas con la gestión de recursos humanos, la producción, la planificación y el control en un modelo de Dinámica de Sistemas que se ha convertido en una referencia en este campo de investigación.A partir de ahí en la misma fuente se

Referencia	Descripción
[KG07]	Mejora de procesos
[Pad05]	Programación dinámica y simulación
[Pad04b]	Simulación de colas
[Pad02a]	Comparación de políticas de secuenciación de tareas
[Pad02b]	Programación dinámica

Cuadro 3.2: Referencias de simulación de eventos discretos no comentadas en el texto.

Referencia	Descripción
[HBH+06]	Redes de Petri
[KC06]	Redes de Petri
[dSP05]	Redes de Petri - constructivo
[EKR95]	Work-Flow (analogía)
[XOZ+07]	Agentes
[SCFR06]	Agentes - código abierto
[RRT+03]	Agentes
[XMQ+04]	Agentes adaptativos
[NC03]	Agentes distribuidos
[CDSC05]	DSM

Cuadro 3.3: Referencias de otras metodologías de simulación.

recogen referencias a una serie de trabajos que han ido enfocando diferentes perspectivas y problemas dentro del marco general antes citado.

Con posterioridad a la publicación de este trabajo, la producción no ha cesado, enriqueciéndose el campo de la producción de modelos y acometiéndose esfuerzos de modelado híbrido dentro del campo de la simulación e integrándolo con métodos y herramientas de otros campos, contemplandose:

- El empleo simultáneo de varios paradigmas de modelado (continuo y discreto, fundamentalmente)
- La integración entre la simulación y la aplicación de métodos y modelos de optimización del campo de la Investigación Operativa y de la Inteligencia Artificial
- La integración de la simulación dentro de herramientas de ayuda a la decisión en tiempo de ejecución básándose en la metodología de agentes
- La integración de la simulación en modelos generales de representación y explotación del conocimiento

Desde el punto de vista académico y profesional la referencia fundamental es la serie de talleres PROSIM (*Workshop of Software Process Simulation and Modeling*) iniciada en 1998 y fusionada a partir de 2006 con el taller internacional SPW (*Software Process Workshop*), que se celebra en el marco de las conferencias ICSE (*International Conference on Software Engineering*) organizadas por la SCM y el IEEE.

En lo que sigue de este apartado se exponen algunos de los trabajos más interesantes revisados, remitiendo para una relación más exhaustiva a la ya citada tesis.

3.2.1. El modelo dinámico reducido (MDR)

La acumulación de experiencias en construcción de modelos va decantando una serie de *comportamientos típicos* de los proyectos que aparecen recurrentemente. Para identificar estos comportamientos no es necesario un modelado exhaustivo caso por caso sino que pueden platearse a modo de

hechos estilizados una serie de pautas básicas y de mecanismos que las explican. A raíz del trabajo de la tesis doctoral de Isabel Ramos [Ram99] se pone en marcha una línea de trabajo que se presenta en [RRT01] donde se propone un *modelo dinámico reducido* (MDR).

El MDR se obtiene manteniendo la estructura en subsistemas del Modelo de Abdel Hamid y Madnick [AM91], y se implementa en el entorno de simulación *Vensim*. La creación de modelos simplificados, a partir de otros ya existentes, se viene realizando en otros campos de conocimiento, fundamentalmente, a partir de los trabajos de simplificación de modelos dinámicos realizados por Robert L. Eberlein [Ebe89]. Estos trabajos están centrados en la reducción de modelos dinámicos de un cierto tamaño, para obtener un modelo más pequeño en los que se eliminan los bucles de realimentación que se consideran que no son esenciales para el comportamiento que se quiere analizar. Según Eberlein, la simplificación de modelos dinámicos grandes y/o complejos no solamente es útil, sino que además es un trabajo que debe de realizarse siempre para conocer los bucles de realimentación básicos.

El modelo reduce sustancialmente el tamaño del modelo original identificando cuatro bucles de realimentación básicos que gobiernan la trayectoria global del proyecto:

- En primer lugar está el mecanismo que regula el empleo de recursos, modelado en este caso como *contratación de personal*, en función de las estimaciones de carga de trabajo que van surgiendo,

- En segundo lugar se modela el mecanismo a través del cual la *presión del plazo* incrementa la posibilidad de errores en la codificación, de manera que el trabajo se amplifica por la necesidad de corregirlos posteriormente,

- En tercer lugar se plantea un bucle de realimentación que modela el impacto de la incorporación de nuevo personal al equipo de desarrollo, incorporación que se traduce en un incremento de la *sobrecarga de comunicación* derivada del tamaño creciente del equipo, y el *efecto aprendizaje* que reduce la productividad efectiva del conjunto,

- El último bucle contemplado refleja el efecto no-lineal de la presión del plazo sobre la *productividad*.

Con este modelo reducido se puede conseguir una visión sintética de algunos de los mecanismos fundamentales que provocan la incertidumbre inherente al ciclo de vida de un proyecto, recogiendo las no-linealidades características de un sistema socio-técnico.

A partir de aquí el modelo se ha utilizado para ganar en la comprensión del efecto de las diferentes *reglas de gestión* sobre la dinámica del proyecto y la capacidad mayor o menor de alcanzar los objetivos del mismo en tiempo y plazo. Para ello se han desarrollado métodos para la inferencia de reglas a partir de algoritmos evolutivos y minería de datos que se presentan en [ARRRT01] y [MRGT08].

3.2.2. La integración de la simulación con otras áreas de la ingeniería del *software*, el modelo *IMMoS*

El modelo *IMMoS* [PR02] combina el modelado a tres niveles, con Dinámica de Sistemas, modelos de procesos y el empleo de modelos cuantitativos basados en la medida. La motivación básica es la "falta de procedimientos bien definidos y repetibles para generar o empelar información producida en la práctica real que sea adecuada para la construcción de modelos de DS", de ahí el nombre de la metodología, *integrated measurement, modeling and simulation*.

Según las conclusiones de los autores, el resultado alcanzado requiere un esfuerzo de modularización para hacer posible la reutilización del modelo sin un esfuerzo excesivo de readaptación a cada caso. Por otra parte, se señalan objetivos en relación con el análisis coste-beneficio del empleo de estos modelos frente a otros métodos basados en la construcción de experimentos.

3.2.3. Análisis de sensibilidad para identificar los factores de riesgo en los proyectos

Un grupo de la Arizona State University [HMC01] readpata el modelo de Abdel Hamid y el de Tvedt [TC95] para evaluar seis factores de riesgo:

- El crecimiento incontrolado de los requisitos,
- La inexactitud en la estimación de los costes,
- La presión del plazo,
- La falta de compromiso y la baja moral en el equipo de desarrollo,
- La inestabilidad de la plantilla de desarrollo,
- La falta de compromiso de la Dirección.

Para cada una de estas dimensiones, los autores identifican una serie de variables significativas y simulan parametrizando dichas variables para obtener una estimación por intervalos de confianza del plazo de realización y del coste total de los proyectos.

Este mismo grupo de investigadores analiza el *comportamiento* de diferentes modelos frente a esas mismas variables. En [HFC+01] aportan una interesante reflexión sobre como las opciones para modelar el comportamiento de determinadas variables y relaciones se manifiesta en los resultados del análisis de sensibilidad. Aconsejan utilizar este análisis para criticar también las suposiciones sobre las que los modeladores han ido construyendo.

3.2.4. Simulación de la interacción entre grupos humanos en un proyecto *software*

En [SG01] se presenta una aplicación de la Dinámica de Sistemas al proceso de elicitación cooperativa de requisitos dentro de la filosofía del modelo *WinWin* de Boehm. Para ello los autores tienen que trasladar al modelo a simular, además de los formalismos de los procesos objetivos, representaciones de fenómenos sociales y de comportamiento humano.

3.3. Modelos híbridos de proyectos *software*

No obstante el potencial explicativo de los modelos continuos, en particular para el análisis de las interaciones estructurales entre los diferentes mecanismos que gobiernan el ciclo de vida de un proyecto *software*, la dinámica de stos proyectos contiene una dimensión *discreta* que los modelos continuos no capturan.

Esa dimensión discreta se manifiesta básicamente a dos niveles:

- Los *artefactos* que van apareciendo a lo largo del ciclo de vida de un proyecto *software* son entidades lo suficientemente diferenciadas como para que a veces su agregación de lugar a unas abstracciones que pierden muchas características relevantes.

- Las *decisiones* así como los procesos de *medida* y *control* son típicamente discretas. Ningún sistema de control de un proceso de trabajo humano es continuo ni siquiera aproximadamente y menos en un proceso de trabajo de contenido básicamente intelectual como es la producción de *software*. Máxime cuando las decisiones y las medidas se refieren a artefactos que, a su vez, son discretos.

Todo ello ha dado lugar a la aparición de modelos de simulación *híbridos* que buscan compaginar metodologías continuas con metodologías orientadas a eventos discretos. Este apartado presenta algunas de las referencias destacadas en este sentido.

3.3.1. Simulación para la fiabilidad; el modelo de Rus, Colofello y Lakey

En [RCL99] se enfoca el problema específico de la fiabilidad del *software* y su calidad, evaluando el efecto de diferentes políticas de calidad y dando apoyo a la gestión y planificación de los proyectos *software* desde esta perspectiva.

En este modelo se introduce la *simulación de eventos discretos* como consecuencia de la necesidad detectada de considerar los atributos individuales

Referencia	[RCL99]
Objetivo	Estimación y gestión de proyectos
Lenguaje	Extend
Enfoque básico	Realimentación en cada actividad Dividir el trabajo e iterar cinco veces
Modelo Discreto	Inicio y final de cada actividad Entradas discretas con atributos Tamaño y la calidad se pasan a la siguiente actividad
Modelo Continuo	Bucles de realimentación dinámicos que calculan determinados valores de factores del producto y el proceso
Temporización	No hay avance del tiempo, el calendario viene dado por la programación

Cuadro 3.4: El modelo de Rus, Colofello y Lakey

de las entidades. Así, por ejemplo, los objetos de diseño y de código difieren entre si en tamaño, complejidad y modularidad. La duración de las actividades y el tiempo para generar y producir dichos objetos puede variar.

El resultado es un *modelo híbrido* en el que aparecen bucles de realimentación entre los factores continuos del proceso (mano de obra, programación y presión del plazo, sobrecarga de comunicación, ...) y los factores discretos del producto (tamaño, calidad y complejidad).

3.3.2. El modelo híbrido de Donzelli y Iazeolla

Este modelo [DI01] aborda el modelado del proceso *software* a tres niveles, como un sistema continuo aplicando dinámica de sistemas, como un sistema guiado por eventos, es decir de forma discreta, y por último tratándolo como un problema analítico.

A partir de un modelo de proceso *en cascada* el modelo se desenvuelve en *dos niveles de abstracción*, uno superior en el que el proceso se modela como una cola de objetos que van desplazándose entre estaciones como objetos discretos. El nivel inferior modela la dinámica interna de cada uno de esos movimientos como procesos continuos o a partir de magnitudes promedio, empleando los modelos analíticos de la teoría de colas.

Este modelo tiene como principal inconveniente el que no se establecen relaciones de realimentación entre ambos niveles sino que los mecanismos de realimentación se mantienen confinados a la explicación de la dinámica de los procesos a la escala más fina. Evidentemente, más en un entorno multiproyecto, las relaciones entre los diferentes niveles, es decir, la capacidad de que un proyecto influya en la ejecución de todos los demás, es una faceta que la simulación debe ser capaz de representar.

Referencia	[DI01]
Objetivo	Predicción de inestabilidad en los requisitos
Lenguaje	QNAP2 package (SIMULOG)
Enfoque básico	Modelos analíticos o continuos en una cola Entorno discreto
Modelo Discreto	Inicio y fin de cada actividad Recepción y entrega de un artefacto Dinámica determinada por eventos de artefactos que se mueven entre actividades
Modelo Continuo	Consumo de recursos y errores descubiertos
Temporización	El tiempo avanza de forma discreta a partir de una estimación tipo COCOMO

Cuadro 3.5: El modelo de Donizelli y Iazola.

3.3.3. La visión inversa: el modelo híbrido de Martin y Raffo

En [RM00] se presenta una aplicación conjunta de simulación discreta y continua. El modelo está implementado en EXTEND y persigue evaluar los cambios potenciales en el proceso, modelando las etapas del proceso como entidades discretas con sus atributos propios lo que permite calcular el tiempo, esfuerzo y tasa de error. Estos datos son pasados a un modelo de DS que representa la evolución continua del entorno del proyecto. A su vez, el entorno proporciona las variables donde evolucionan las entidades discretas.

Cada entidad computa su propio tiempo de evolución pero solamente en intervalos discretos δ definidos para la simulación de eventos. Eso hace que la

Referencia	[RM00]
Objetivo	Evaluación de cambios potenciales
Lenguaje	Extend
Enfoque básico	Modelos discretos en un marco dinámico Entorno continuo
Modelo Discreto	Componentes y pasos de iteración discretos Los detalles del proceso son discretos Calcula duración, esfuerzo y errores
Modelo Continuo	El modelo de DS pasa los valores del entorno al modelo discreto
Temporización	Incremento temporal predefinido

Cuadro 3.6: El modelo de Martin y Raffo

evolución de las actividades no pueda calcularse dinámicamente. Por el contrario, las decisiones de asignación de recursos se toman de manera discreta lo cual resulta más realista que la asignación continua del modelo tradicional de AbdelHamid, aunque al nivel del entorno del proyecto se mantenga una evolución continua de los factores que afectan a la fuerza de trabajo.

3.3.4. Simulación híbrida para las compañías de *software* global

En [SWR07] se presenta un trabajo que emplea la simulación híbrida para modelar el problema de la organización de la producción de *software* según la estrategia denominada "seguir al sol", *follow-the-sun*, practicada por las grandes compañías para acelerar los desarrollos. El modelo se asemeja bastante al descrito en el apartado anterior. La principal variación radica en que añade una dimensión más para recoger la existencia de diversas localizaciones geográficas. En ese sentido es un modelo *jerárquico*.

Así pues el modelo contiene cuatro niveles:

- A nivel local:
  - un modelo discreto que representa las tareas y los objetos entregables,

- un modelo continuo que representa la capacidad en términos de mano de obra disponbile así como las otras magnitudes relacionadas al entorno de trabajo (aprendizaje, presión de plazos y productividad, etc.).

- A nivel global:

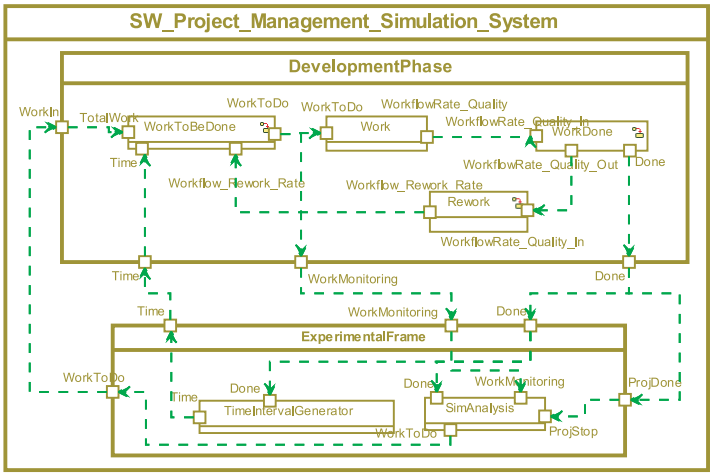
  - un modelo discreto que representa el movimiento de los artefactos entre las diferentes localidades "siguiendo al sol",
  - un modelo continuo que recoge la gestión global de los recursos y de la mano de obra, la planificación y el control sobre la ejecución.

Este modelo permite a los autores formular hipótesis sobre la complejidad que emerge del trabajo conjunto de equipos con diferentes culturas, lenguas y actitudes. Estas hipótesis se contrastan con datos empíricos provenientes de estudios realizados en otras áreas de conocimiento viéndose validadas las hipótesis formuladas.

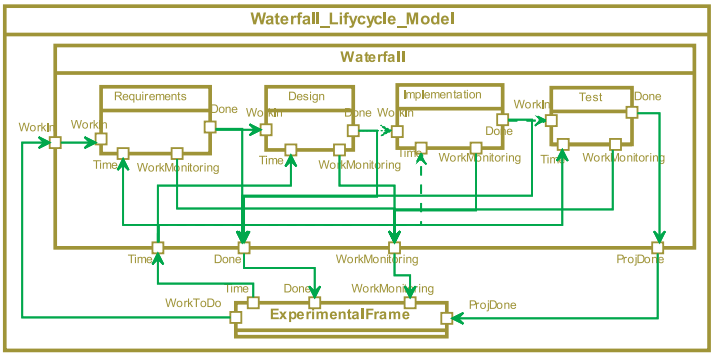
3.3.5. Integrando simulación discreta y continua con el formalismo DEVS

En [CBK06] se presenta una aplicación del formalismo DEVS, *Discrete Event System Specification*, propuesto por Ziegler ([ZKP00]) que posibilita representar con naturalidad un sistema híbrido al gestionar conjuntamente una simulación continua con un manejo muy flexible de la lógica de eventos y los flujos de información asociados al inicio y finalización de tareas y a la monitorización de las mismas. Este formalismo presenta además la ventaja de que el modelo básico, basado en un sistema muy simple denominado el DEVS *atómico* y en un acoplamiento elemental de dos átomos, es escalable y fácil de representar con la tecnología orientada a objetos. Esto permite generar una clase de modelos fácilmente parametrizables y reutilizables.

El ejemplo que presentan es una "relectura" del modelo clásico de Abdel Hamid y Madnick [AM91] en un proceso con ciclo de vida en cascada en el que se pueden diferenciar, y a la vez mantener coherentes, los procesos



a)



b)

Figura 3.1: Modelo citado en [CBK06]

continuos del entorno de la organización y del propio desarrollo de las tareas elementales, sujetas al tipo de interacciones que sintetiza el MDR, y los procesos discretos tanto de flujos materiales (entregables, módulos, ...) como de medida, control y decisión.

La figura 3.1 muestra, en su mitad superior, la arquitectura de un bloque constructivo compuesto de dos módulos: el de una Fase de Desarrollo - *DevelopmentPhase*- y un Marco de Medida (*ExperimentalFrame*). Cualquier fase de desarrollo, bien de un modelo de ciclo de vida en cascada o incremental, puede modelarse acoplando bloques constructivos del tipo mostrado a través

de puertos de entrada y/o salida específicos, como *Time*, *WorkMonitoring* y *Done*.

El módulo *WorkToBeDone* almacena el trabajo pendiente y lo envía al módulo *Work*. Este contiene diversos submódulos que calculan el flujo de trabajo, la calidad, etc. El módulo *WorkDone* integra el flujo de trabajo para obtener el trabajo terminado y el módulo *Rework* determina el trabajo a rehacer o iterar y lo reenvía a *WorkToBeDone*.

El módulo *ExperimentalFrame* desempeña el papel de un sistema de medida a semejanza de un osciloscopio. Genera una señal que se envía al sistema observado y recibe y analiza los datos de la simulación. Genera el mensaje inicial *WorkToDo*. Este mensaje será procesado por *DevelopmentPhase* quien devuelve el mensaje *WorkMonitoring* a *ExperimentalFrame*.

El módulo *TimeIntervalGenerator* dirige la simulación generando un evento *Time* a intervalos constantes lo suficientemente pequeños dando así lugar a la simulación híbrida.

Todos los mensajes en este modelo incluyen las variables de nivel y flujo, así como las variables auxiliares típicas de la Dinámica de Sistemas. El mensaje *WorkMonitoring* contiene las variables del estado de la simulación que se almacenan y analizan por *SimAnalysis* dentro de *ExperimentalFrame*. El mensaje *Done* generado por el módulo *WorkDone* detiene la simulación.

La parte inferior de la figura 3.1 muestra el modelo de un ciclo de vida en cascada construido a partir de los bloques anteriores. Para ello se acoplan secuencialmente las distintas fases del proceso. El modelo se activa cuando recibe la entrada *WorkIn* y se detiene cuando le llega *ProjDone*. El bloque *Requirements* realiza su trabajo y emite la señal *Done*. Este mensaje contiene todas las variables dinámicas de la fase de Requisitos que son pasadas a la fase de Diseño (*Design*). El mensaje *WorkMonitoring* de cada fase es enviado al módulo *ExperimentalFrame* que analiza explícitamente el comportamiento de cada elemento discreto. Así se consigue que el modelo incorpore la realimentación propia de la Dinámica de Sistemas y haga que las actividades discretas sean coherentes con las actividades de las variables ambientales continuas.

3.4. Simulación del proceso software en el entorno multiproyecto

El entorno multiproyecto, con sus elevadas dependencias, está naturalmente inclinado a multiplicar los efectos cruzados de los diferentes bucles de realimentación que gobierna la evolución del estado de la organización de desarrollo. Es significativo, por tanto, que el esfuerzo por la simulación de este tipo de entorno sea relativamente reducido en comparación con el caso de un único proyecto.El propio Abdel Hamid en su artículo [Abd93b] llama la atención sobre lo inapropiados que pueden ser los métodos y modelos de estimación cuando no tienen en cuenta las implicaciones a escala de toda la organización. En los apartados siguientes se presentan los dos modelos más interesantes que se han detectado.

3.4.1. Abdel Hamid: desmintiendo la Ley de Brooks

En el artículo citado [Abd93b] se presenta un trabajo en el que se modela específicamente la transferencia de personal entre varios proyectos que se están desarrollando en paralelo. La transferencia obedece a una serie de reglas relacionadas con la presión del plazo. Se realizan dos experimentos: uno de aceleración de los plazos y otro en el que se emula la regla MINSLK (ver aparado 2.4.3 para asignar recursos de la organización a los proyectos que compiten por ellos.

El primer experimento corrobora lo señalado sobre la necesidad de incorporar en las estimaciones del coste de acelerar los plazos, los efectos que esa decisión tiene no sólo sobre el coste *directo* del proyecto afectado sino también el coste de *oportunidad* que no se manifiesta directamente sino a nivel de toda la organización.

El segundo experimento proporciona la oportunidad de relativizar la percepción generalizada sobre la denominada “Ley de Brooks”. La adición de nuevos recursos a un proyecto, si bien reduce la productividad promedio del equipo de desarrollo a través de las *sobrecargas de comunicación* y del *efecto aprendizaje*, puede aumentar la productividad total, acelerando así los plazos,

siempre que se den las condiciones que Boehm denomina *proyectos orgánicos*, es decir proyectos de un tipo semejante a otros que se desarrollan en la organización.

Conviene repetir que, a pesar de la recomendación del autor, el trabajo posterior en el entorno multiproyecto es bastante escaso.

3.4.2. Powell *et al*: El desarrollo incremental y multiproyecto

En [PMB99] se presenta un modelo de dinámica de sistemas en el que se modela un ciclo de vida incremental en términos que recuerdan al entorno de proyectos. El ciclo de vida incremental conlleva relaciones de concurrencia entre diferentes paquetes de trabajo que, a otra escala, equivalen a la competencia por recursos de varios proyectos simultáneos, con la complicación adicional que dicha interrelación se establece no sólo por esa competencia por los recursos sino también por la información que fluye de un paquete de trabajo a otro, acoplándolos por tanto también a través de ese mecanismo.

Los autores proponen un modelo sencillo, una vez más basado en una estrucutra simplificada del modelo de Abdel Hamid, que relaciona recursos, tiempo y esfuerzo con el paquete de trabajo. Esta sencilla estructura se denomina el *triángulo del proceso*. Este triángulo puede definirse, modularmente, a diferentes niveles jerárquicos: paquetes de trabajo, fases, entregas, proyectos y organización en su conjunto.

Para acoplarlos diferentes módulos se emplea tanto la asignación de recursos a módulos como el acoplamiento de los trabajos. El primero es una regla de reparto de recursos entre paquetes de trabajo (o módulos de nivel superior) que recuerda a la RWK tratada antes (ver apartado 2.4.3): en base a prioridades y presiones de plazo. El acoplamiento de los trabajos implica tratar el trabajo de *corrección de errores* y las *transiciones*. La corrección de errores puede aparecer dentro de un mismo paquete de trabajo o en otro subsiguiente. Trasladando la corrección de errores a una entrega posterior se *externalizan* estas tareas. Las transiciones se producen cuando una fase madura lo suficiente como para iniciarse la siguiente. Las transiciones pueden

producirse de diversas formas: mediante *hitos* o por *unidades* de trabajo que al concluirse se convierten en *inputs* de las siguientes. Esto permite modelar la superposición de fases. El objetivo último del modelo es analizar las estrategias de desarrollo incremental, tanto a nivel previo (planificación) como su implementación y su modificación en tiempo de ejecución.

3.4.3. Lee y Miller: escenarios y cadena crítica

En este trabajo ([LM04]) se propone un modelo de simulación para analizar las interaciones en el entorno multiproyecto. Emplea la Dinámica de Sistemas y modelos de gestión basados en la Cadena Crítica gobernados por diversos escenarios.

Para construir el modelo se genera una red de dependencias que se describe en los términos de la Cadena Crítica (CCPM). Sobre esta base el modelo de Escenarios y el de DS permiten simular diferentes políticas de gestión. El modelo de Escenarios describe los diferentes eventos que se pueden producir en la ejecución de los proyectos. La red CCPM controla las dependencias entre proyectos en términos de asignación de recursos, secuenciación de actividades y plazos de terminación. El modelo dinámico continuo utiliza la información proporcionada por los escenarios y la red para modelar los bucles de realimentación que gobiernan en entorno de desarrollo.

Se presenta un caso de aplicación en el que dos proyectos, uno en cascada y el otro incremental, se desarrollan conjuntamente. Se analizan dos escenarios en los que un proyecto sufre un incremento repentino del 40 % en su contenido de trabajo. En el primer escenario se permite que se deslicen las fechas de entrega mientras que en el segundo no. El primer escenario simula la transferencia de recursos al proyecto retasado mediante una reconstrucción de la Cadena Crítica. El segundo se resuelve a través del mecanismo convencionalmente modelado en DS de contratar nuevos recursos en función de las previsiones de carga de trabajo y los plazos, mecanismo que aparece bien sintetizado en el modelo MDR (ver 74).

Los autores plantean que los resultados de la simulación pueden utilizarse para construir una base de conocimientos retulizable, aunque reconocen que

el modelo se encuentra en una fase de desarrollo aún inicipiente.

3.5. Simulación y mejora de procesos *software*

Como se ha indicado más arriba, una de las aplicaciones de la construcción de modelos para la simulación del proceso *software* es la de evaluar los cambios en el propio proceso. La puesta en marcha de programas para la mejora del proceso *software* en las organizaciones de desarrollo se ha convertido, por tanto, en un área de interés natural para la simulación. Un cambio en los procesos debe traducirse en una mejora de la situación de la organización en términos de sus objetivos estratégicos. Pero ese cambio no es ni fácil de implementar ni es evidente la relación entre las medidas concretas y las mejoras deseadas. Ello se debe al carácter complejo e interrelacionado de la propia actividad y del entorno en que se desarrolla.

Existe además una relación inversa entre ambas cuestiones. Un proceso poco maduro y estructurado, con empleo reducido de técnicas de medición y de métodos cuantitativos en general, resulta extremadamente difícil de modelar de manera formal. Los mecanismos que explican los resultados permanecen ocultos en un conglomerado de prácticas informales y poco transparentes. Por lo tanto el esfuerzo para construir el modelo en si mismo es un esfuerzo que ayuda a diagnosticar las prácticas y áreas susceptibles de mejora.

Estos hechos son conocidos por la comunidad investigadora y así han sido señalados desde hace tiempo [Chr99] por lo que existen varias referencias sobre la interacción entre modelos de simulación que sintetizaremos en dos por referirse una a cada uno de los estándares más generalizados: la familia CMM del SEI (*Software Engineering Institute*) [Pau95] y la norma ISO/IEC 15504([Rou03]).

3.5.1. El modelo de Stallinger para mejora de procesos

En [Sta00] se presenta un esfuerzo por modelar el proceso táctico de mejora del proceso *software* mediante un modelo generalizado de DS que se

combina con el estándar de evaluación ISO/IEC TR 15504-5 (en su versión de *Technical Report* de 1998), intentando representar así los principios del paradigma de la calidad total, TQM-(*Total Quality Management*).

El objetivo es dar soporte a la planificación del proceso de mejora a un nivel táctico caracterizando el impacto de un conjunto de medidas programadas de mejora en la totalidad del proceso. Eso lleva al modelo a identificar, de un lado, el comportamiento de los objetivos estratégicos de la mejora: el tiempo de realización, el coste, la calidad, etc. De otro lado implica identificar los costes del propio proceso de mejora, principalmente a través del coste mismo de los recursos empleados y de las nuevas actividades que aparecen como consecuencia de la formalización creciente de los procesos.

La filosofía es que el modelo sea aplicable tanto a organizaciones con un bajo nivel de madurez como a aquellas que ya están más avanzadas. En el primer caso, el modelo opera a la manera de *un simulador de vuelo* mediante una presentación de escenarios de mejora en relación con la situación inicial de la organización que ha sido sometida a una evaluación previa. En el segundo caso, en organizaciones con mayor nivel de madurez, en las que las medidas para la efectividad de los procesos ya están implementadas, la aplicación modela explícitamente la implementación de las mismas como una tarea más dentro del proceso.

3.5.2. Marco dinámico integrado para la mejora de los procesos *software* (DIFSPI)

Frente a esta propuesta en la que la simulación se emplea como herramienta para apoyar el proceso de mejora, en [RRT02] se presenta, bajo el título *un marco dinámico integrado para la mejora de los procesos* software (DIFSPI, *Dynamic Integrated Framework for Software Process Improvement*) ,una metodología y un entorno de trabajo más "bidireccional" en el sentido de que es el grado de madurez el que a su vez determina las posibilidades de la simulación. Este trabajo es un resultado de la investigación presentada en [Car03]. Una aproximación semejante se recoge en [RVM99]

Uno de los objetivos principales de DIFSPI consiste en facilitar la evolu-

ción del nivel de madurez de las organizaciones de desarrollo de acuerdo con el modelo CMM. El propio diseño y construcción del marco y de los modelos dinámicos que lo integran permite, en sí mismo, la definición de métricas que, por un lado son necesarias para la inicialización y validación del modelo y que, por otro lado, sirven para la definición de un programa de recogidas de métricas concretas dentro de la organización. Esta recogida definida y sistemática de métricas persigue no sólo la utilización de modelos dinámicos sino además mejorar el conocimiento real del estado de los procesos dentro de la organización.

Tras la definición de los componentes del sistema de métricas, éste puede implementarse en la organización, traduciéndose en la definición y desarrollo de una base de datos histórica. Estos datos de carácter histórico serán los que se utilizarán para la simulación y validación empírica del modelo construido. Una vez demostrada la validez de dicho modelo, los valores ofrecidos por la simulación permiten instanciar una base de datos que permita realizar análisis acerca de los efectos de diferentes acciones o mejoras.

Un aumento de la complejidad de las acciones que se pretenden evaluar, se traduce en un aumento de la complejidad del modelo dinámico que vuelve a diseñar un nuevo programa de recogida de métricas para los nuevos módulos de simulación, cerrándose, con ello, el ciclo. Si el nivel de madurez de la organización es bajo, estos datos permiten la observación de las tendencias gráficas que muestran la evolución del proyecto a partir de determinadas condiciones iniciales (establecidas por los valores de los parámetros de inicialización); cuando el nivel de madurez aumenta, la definición y conocimiento de los procesos de desarrollo también, por lo que los resultados de la simulación se convierten en estimaciones cuantitativas reales que predicen, de acuerdo con la incertidumbre de sus parámetros, los resultados futuros del proyecto.

En segundo lugar, la posibilidad de definir y experimentar diferentes escenarios sin asumir coste o riesgo alguno, es una de las principales cualidades de la simulación de modelos en cualquier área. También la gestión de proyectos *software* puede verse favorecida por esta cualidad. Es posible, por tanto, definir y experimentar diferentes mejoras de procesos, simular el modelo, y

analizar los resultados de manera que la mejora o mejoras que se implementen de manera efectiva en el proyecto sean aquellas que arrojaron los mejores resultados durante la simulación. En tercer lugar, los modelos de simulación también se pueden utilizar la estimación de costes del proyecto; costes que pueden estar relacionados con algún sector concreto del mismo, como por ejemplo el coste de la calidad o el de las actividades de revisión, o con el proyecto completo.

3.5.3. La simulación de las actividades de gestión y mejora de procesos

Teniendo presente lo anterior, otros autores [MV06] proponen la incorporación explícita de las actividades asociadas a los procesos de mejora a la descomposición en tareas, WBS, de los proyectos. De hecho, el modelo CMMI desde sus primeros niveles requiere la existencia de una descomposición en paquetes de trabajo de todas las actividades. De hecho, en el modelo por etapas es preciso incorporar *todas las tareas*, incluyendo las de medida del esfuerzo en la WBS desde el nivel 1. Para el nivel 2 se requiere implementar prácticas de planificación de recursos lo mismo que otras actividades "no constructivas" como la gestión de la configuración (*configuration management*).

El ciclo de vida total puede descomponerse en tres fases principales: la previa al proyecto, la ejecución del proyecto y la posterior al proyecto, en las cuales el esfuerzo es estimado inicialmente, medido, controlado y reestimado posteriormente, y al final evaluado y analizado. Aparte de las propias actividades implicadas estas tres fases incluyen también tareas relativas a las actividades "constructivas" y flujos de información entre unas y otras.

Durante la fase previa, el proyecto se programa y se pone a punto. La programación del proyecto debe incluir las actividades propias del desarrollo pero también las subactividades de gestión y control del proyecto. El seguimiento del proyecto durante la ejecución normalmente contempla la recogida de información sobre el esfuerzo invertido en las actividades constructivas pero debe *también* contemplar el esfuerzo de las propias actividades de ges-

tión. Por último, el análisis "post mortem" del proyecto debe incluir también el análisis de las actividades de gestión.

El modelado explícito de estas tareas tiene dos virtualidades:

- Proporciona mayor precisión en el cálculo del esfuerzo requerido que no es sólo el trabajo directo de análisis y construcción sino que incluye, además de la propia gestión, las tareas anexas a ésta que corresponden a lo que genéricamente se llama la sobrecarga de comunicación (*communication overhead*),
- Arroja un mapa efectivo para la mejora de los procesos al permitir contrastar el coste de la misma con el beneficio esperable.

3.6. Simulación e investigación operativa en los proyectos *software*

Los modelos que se presentan a continuación son modelos híbridos que emplean técnicas de la *investigación operativa y simulación de forma conjunta*. En los casos que se han revisado aparecen básicamente tres enfoques:

- generar una solución a partir de un modelo de investigación operativa y simular luego las consecuencias de su aplicación, generalmente por medio del método de Monte Carlo,
- generar el espacio de estados a través del proceso de simulación y luego aplicar un heurístico de la investigación operativa para optimizar el estado del sistema,
- modelar el decisor explícitamente como un agente dotado de las reglas heurísticas; en este caso se trata de una herramienta de apoyo a la dirección del proyecto para que evalúe las consecuencias de las decisiones adoptadas de forma dinámica.

Los heurísticos que se han detectado en la literatura son de diferentes tipos:

- reglas simples de prioridad,

- programación dinámica (métodos aproximados),
- algoritmos de búsqueda *greedy* (miopes),
- algoritmos genéticos,
- *squeaky wheel optimizaton*; un algoritmo de búsqueda en el espacio de las estrategias de prioridad.

Los casos más interesantes que se han encontrado en la literatura son los que se describen a continuación.

3.6.1. Antoniol *et al*: Teoría de colas para la asignación de personal a las tareas de desarrollo

Los autores [ACLP04] presentan una aproximación al problema basada en la teoría de colas y la simulación estocástica para el apoyo a la planificación, gestión y control de la asignación de personal a un proyecto de mantenimiento multifase.

Se trata de la adaptación al efecto 2000, *Y2K*, de un gran sistema de información de una entidad financiera desarrollado en COBOL/JCL que requiere la constitución de diferentes equipos trabajando en varios centros geográficamente distribuidos. Se estudia una configuración centralizada desde el punto de vista del cliente así como otras configuraciones más desagregadas. Los métodos de *Teoría de Colas* y la *simulación estocástica* se emplean para evaluar las políticas de asignación de personal a los equipos y la posibilidad de alcanzar los hitos previstos para la ejecución del proyecto.

En otro segundo artículo [APH04] se emplean los *algoritmos genéticos* para generar soluciones para el problema de colas teniendo en cuenta la necesidad de corregir errores y de reprogramar y abandonar algunas de las tareas inicialmente previstas. La simulación de Monte Carlo de las colas junto con los algoritmos genéticos permiten generar secuencias y asignar personal a los equipos. El impacto posible del trabajo de corrección de errores, el abandono de tareas y los errores e incertidumbres en las estimaciones iniciales

se caracterizan con funciones de distribución que se utilizan como entradas para la simulación.

3.6.2. Padberg: Programación dinámica y simulación de reglas de asignación

En varios artículos [Pad02b, Pad04a, Pad04b, Pad03, Pad05] este autor presenta su trabajo en el que se modelan como problemas *markovianos* de decisión en los que se tienen en cuenta, además de la variación estocástica de la duración de las actividades, la posibilidad la existencia de ciclos y retrasos. Para ello se generan políticas de secuenciación con un algoritmo aproximado de *programación dinámica*.

Con un conjunto de problemas similares pero que se diferencian en la necesidad mayor o menor de recursos especializados y el grado de integración entre los componentes del producto, se simula la aplicación de la política óptima generada por el algoritmo con las políticas clásicas de listas baadas en reglas de prioridad. El modelo comprueba que cuanto mayor es la necesidad de recursos especializados y cuanto menor es el grado de integracion entre los componentes, más destaca la solución propuesta sobre las reglas de prioridad.

3.6.3. Browning, Meire y Yassine: Simulación y algoritmos genéticos en el desarrollo de producto

El modelo que se presenta por estos autores [MYB07] se basa en la DSM ya descrita en el capítulo anterior. La varianza en la duración y el coste en los proyectos de desarrollo de producto son atribuibles según su criterio en gran medida a las iteraciones, algo que es común también en el proceso *software*. Dichas iteraciones pueden o no ocurrir dependiendo del comportamiento de una serie de variables que el modelo simula como variables estocásticas cuya probabilidad de ocurrencia depende de la existencia de determinados paquetes de información que activan un proceso de reelaboración (*rework*). Un proceso de reelaboración puede, a su vez, desencadenar otros sucesivos afectando así al proyecto en su totalidad. Para hacer frente a este problema

utilizan el método de Monte Carlo para simular políticas de secuenciación robustas generadas por un algoritmo genético.

3.6.4. Özdamar: Simulación de reglas de prioridad en proyectos definidos con el formalismo *fuzzy*

Este trabajo [OA01] no consiste realmente en un modelo de simulación sino que se trata de una aplicación del MRCPSP a los proyectos *software*. Presenta un heurístico para la programación de las actividades de un modelo de desarrollo en cascada que se modela utilizando la lógica borrosa. la duración de las actividades depende del modo de ejecución seleccionado representado por una función de pertenencia trapezoidal. También se modela la disponibilidad de los recursos diferenciando entre dos categorías, una de personal más cualificado que comparte varias tareas y - por tanto - asignable de forma continua, y otra categoría de personal que no realiza multitareas y que se presenta en unidades discretas.

El objetivo del modelo es minimizar el tiempo de realización. Para ello se emplea un algoritmo de generación de secuencias en serie y se ensayan diferentes reglas de prioridad entre las más sencillas empleadas en los problemas de secuenciación. El modelo permite la resecuenciación dinámica para escoger las variaciones en las estimaciones a lo largo de la ejecución del proyecto. Posteriormente el modelo se analiza mediante la simulación de Monte Carlo.

3.6.5. Joslin: simulación basada en agentes

En este trabajo [JP05] se presenta un modelo basado en agentes que simula de forma dinámica la reasignación de personal a las tareas en un proyecto con diferentes funcionalidades que deben entregarse en un plazo fijo.

Se trata de un sistema de ayuda a la decisión (DSS) basado en un heurístico denominado *Squeaky Wheel* que explora en el espacio de las estrategias en lugar de en el espacio de las soluciones. El simulador puede implementar diferentes reglas de asignación si bien la versión que se presenta en el artículo

sólo presenta los primeros resultados para reglas muy sencillas.

La mecánica básica es la de reasignar el personal entre tareas conforme se aproximan los plazos y se detecta el riesgo de no cumplirlos. El modelo contempla el efecto aprendizaje cuando se produce un cambio de tarea y también el efecto de la sobrecarga de comunicación por lo que incorpora no linealidades el tipo de las que se presentan en los modelos clásicos de dinámica de sistemas. Los autores anuncian la futura implementacion de reglas más sofisticadas, como las basadas en algoritmos genéticos.

Capítulo 4

El caso de aplicación

4.1. Introducción y contenido del capítulo

Para la construcción y validación del modelo propuesto se ha empleado el caso de una mediana empresa dedicada a la producción de *software* a medida. Se trata de una empresa con una vida relativamente prolongada en el sector y que cuenta con un sistema de información de gestión que posibilita la obtención de métricas históricas de proyecto, permitiendo así una base empírica para el trabajo realizado en la tesis.

En la literatura existen referencias de muchas clases que se centran en empresas y organizaciones de desarrollo *dedicadas*, es decir, cuyo producto o línea de producto principal gira en torno al desarrollo de aplicaciones integradas en grandes sistemas o a la producción de versiones, extensiones y mejoras sucesivas de un mismo producto. Las razones para ello son diversas, desde casos en los que las organizaciones implicadas están habituadas al empleo de métodos y modelos cuantitativos de gestión en otras áreas funcionales hasta otros donde la propia evolución de su línea de producto en un entorno muy controlado incentiva el registro de información y la explotación ulterior de la misma.

Otra proporción importante de la literatura está dedicada a los grandes proyectos, en los que el alcance y enfoque está precisamente en esos mismos proyectos. Sin embargo, y como ya se ha indicado en el capítulo anterior,

la dinámica de la producción multiproyecto aparece relativamente mucho menos.

No obstante, las organizaciones cuyo negocio es el desarrollo simultáneo de sistemas y aplicaciones muy variados para diferentes clientes son una parte importante del sector de la producción de *software*. Las razones por las que se produce esta falta de referencias en la literatura académica probablemente tenga que ver con la opacidad (deseada o involuntaria) de estas organizaciones a la hora de facilitar datos empíricos, con la proliferación de métodos y procedimientos "ad-hoc" desarrollados internamente y difícilmente codificables o, simplemnte, porque la dinámica guiada por el mercado de estas organizaciones no se presta con facilidad a establecer una relación con el mundo académico e investigador.

Pero la falta de referencias no es un indicador de falta de interés de los problemas específicos de este tipo de organizaciones, sino que, al contrario, lo que indica es una laguna en el alcance del trabajo de investigación[1]. Una de las motivaciones básicas de esta tesis es que la complejidad de las interrelaciones que se producen en una empresa que desarrolla varios proyectos diferentes a la vez afecta a cada uno de ellos y a la marcha global de la empresa. Esta motivación no es original, sino que ya el propio "padre" de la aplicación de los modelos de simulación, Abdel Hamid afirmaba en 1993 ([Abd93b]) que:

> *"The implication of the above results is clear: interproject management policies do matter. They influence project behavior in real and measurable ways and, in turn, project cost and schedule performance. Yet, such interproject policy variables are being totally overlooked; they have yet to be incorporated into any software estimation tools. (...) Furthermore, because such managerial policies vary from one software development organization to another, the portability of software cost estimation models will improve if such variables are explicitly incorporated in the models' formulations."*

[1]Frente a otros subsectores dentro del *software*, la producción "a medida" es casi en su totalidad valor añadido.

En este capítulo se presenta un caso de organización multiproyecto que se ha empleado como guía en la construcción del modelo al que se refiere esta tesis.

Tras una descripción general del proceso productivo, se pasa a presentar la información disponible a partir del sistema de control de gestión existente. Esta información permite, tomando como punto de partida el registro de trabajo del personal, reconstruir el ciclo de vida de los proyectos y estimar una serie de métricas de los proyectos.

La reconstrucción del ciclo de vida permite obtener una serie de *perfiles-tipo* de empleo de recursos por parte de los proyectos. La problemática de la interación multiproyecto se reconoce en dichos perfiles. A partir de ahí se obtiene una orientación para la construcción del modelo que se propone en el capítulo 6. En el apéndice E se presenta el trabajo realizado con la información disponible para la estimación de los parámetros del modelo.

4.2. Descripción del caso

4.2.1. Características de la producción

La empresa en cuestión desarrolla sistemas de cierta envergadura siendo sus clientes habituales organismos públicos tanto estatales como autonómicos y locales. En una gran mayoría de los casos se trata de sistemas orientados a la gestión administrativa pero también se ocupa en otros más especializados como Sistemas de Información Geográfica y sistemas de monitorización ambiental en tiempo real.

Con el advenimiento de la denominada *e-administración*, la empresa ha ido evolucionando a sistemas basados en Internet donde se compatibilizan procesos *front-office* de interación con los usuarios (por ejemplo, tramitación electrónica) con sistemas *back-office* que a su vez deben comunicarse con muy diversas plataformas de generaciones anteriores.

Las características del producto y la tipología de los clientes obliga a una forma de desarrollo que encaja mal con los modelos clásicos de proceso. De todas las especificidades que rodean al tipo de proyecto que habitualmente

se desarrolla caben destacar dos:

La "volatilidad" de los requisitos . Dado que los clientes son organizaciones complejas en las que se combina la jerarquía orgánica clásica de la administración con la diversidad funcional, es bastante corriente que los requisitos no estén formulados con precisión desde el primer momento sin que ninguna metodología de elicitación pueda asegurar lo contrario. El usuario directo, imprescindible para delimitar las funcionalidades requeridas, no es, ni mucho menos, el único interlocutor. Además, el proceso de información va mucho más allá de los sistemas informáticos y está sujeto a requisitos administrativos y legales que dificultan precisar de partida muchos aspectos del sistema a desarrollar. El resultado para la organización encargada del desarrollo es una elevada frecuencia de cambio, aparición y modificación de requisitos.

Evolución de los sistemas . Los clientes son usuarios de sistemas de tratamiento de la información desde hace ya tiempo y emplean todo tipo de plataformas por lo que casi ningún sistema nace *ex-novo*. Por el contrario, debe coexistir con otros e, idealmente, integrarse con ellos. Cualquier innovación tecnológica desencadena, una vez adoptada, un proceso muy prolongado de migración de datos y aplicaciones. Por otra parte, la decisión de incorporar la innovación en nuevas funcionalidades (otra vez el ejemplo puede ser el de la administración electrónica) sólo puede hacerse de manera incremental para no interrumpir la operación de los sistemas heredados, que a su vez se han consolidado de forma incremental. Todo esto tiene un efecto añadido en la fase de implantación que se prolonga de forma notable debido a la complejidad de la integración con los sistemas preexistentes. Un factor añadido de incrementalidad es la necesidad de abordar en algunas ocasiones un proyecto que de otro modo sería unitario, descomponiéndolo en varios por razones de ciclo presupuestario.

Estas circunstancias son abordadas en el caso de referencia siguiendo básicamente una estrategia basada en la *estabilidad* de los equipos y la plantilla:

- De una parte se intenta mantener estables los equipos a lo largo de la ejecución del proyecto, especialmente en lo que se refiere a analistas e ingenieros, sin que quepa separar fases de análisis, de un lado, y de diseño y construcción, de otro, con personal diferente. La complejidad de las organizaciones cliente recomienda, además, intentar especializar a las personas en clientes y grupos de clientes.

- De otra parte, y reforzado por lo anterior, se intenta mantener estables las plantillas. La opción de la empresa en cuestión es que la estabilidad en la plantilla, aunque supone renunciar a la posibilidad de ajustar rápidamente la capacidad de producción, es la que le permite mantenerse competitiva a través del conocimiento y el compromiso con las necesidades del cliente.

La trayectoria histórica de la empresa justifica estas opciones pero, a cambio, da lugar a un proceso de gestión de proyectos bastante accidentado por la combinación de la presión de los plazos, los cambios imprevistos y la necesidad de adscribir personal a proyectos durante periodos muy prolongados.

Como es lógico una plantilla sobredimensionada, que permitiría absorber estas variaciones, no es económicamente viable. La solución está pues, en el empleo del personal en varios proyectos a la vez y la administración de las holguras (y, a veces, todo sea dicho, la tolerancia de los clientes).

4.2.2. El proceso actual de gestión de proyectos

El proceso actual de gestión de los proyectos, entendiendo por estos normalmente el conjunto de actividades incluidas en un *único contrato*, se basa en unas reglas muy simples que combinan la elaboración de *previsiones de plazos y recursos* necesarios para completar un proyecto con el *control del empleo de los recursos* en cada momento.

Inicialmente se designa un Jefe de Proyecto que elabora una estimación del programa de desarrollo del proyecto en base a su experiencia previa. Dicho programa incluye unas demandas a plazo de determinados recursos durante determinados periodos de tiempo. Con base en esas demandas se constituye

un *equipo* para el desarrollo del proyecto asignándose personas al mismo, personas que entrarán a trabajar en el proyecto en los plazos estimados y por el periodo de tiempo igualmente estimado.

Esto proporciona una programación del empleo de los recursos humanos en el tiempo y permite, por exclusión, ir asignando recursos a proyectos que vayan apareciendo posteriormente. Semanalmente se evalúa el cumplimiento de los programas y se actualizan las previsiones, de modo que se pueden producir reasignaciones de recursos a tareas eventuales o a acelerar el cumplimiento de las previsiones.

Los recursos ociosos se destinan a tareas internas, salvo cuando estas son repetitivas - en cuyo caso la capacidad utilizada se descuenta de la total - o de magnitud tal que son consideradas como un proyecto más. Con este heurístico tan simple se acomoda la producción a la capacidad y se regula - en último extremo - el flujo de las operaciones en la empresa.

4.3. El sistema de control de gestión actual

El sistema de control de gestión implantado tiene por objetivo principal la determinación del coste de los contratos que realiza la empresa con sus clientes en relación con el coste presupuestado en la oferta. El coste se determina por la imputación del coste directo de mano de obra más un coeficiente de absorción de los gastos indirectos predeterminado. En el caso de que el proyecto incorpore suministros de equipos o cualquier otro elemento adquirido en el exterior - incluso la subcontratación excepcional de partes del desarrollo - se acumula su coste a la cifra anterior.

La organización mantiene un sistema de bases de datos que se orienta a la estimación del coste total de un proyecto a través del registro del consumo de recursos. El sistema contiene mucha información sobre aspectos administrativos, clientes, etcétera y, de hecho, alguna de sus funcionalidades evoluciona hacia un sistema CRM (*customer relationship management*). Sin embargo, a los efectos de este trabajo la información relevante es la que se basa en el modelo que se describe acontinuación.

4.3.1. Proyectos y productos

La organización distingue entre **proyectos**, que se corresponden biunívocamente con contratos, y **productos**, que se corresponden con la definición convencional en la ingeniería del *software*: un artefacto compuesto básicamente de información codificada y que forma parte de la instancia operativa de un sistema de información. El énfasis en la gestión está puesto en los proyectos.

La relación **proyecto** a **producto** es una relación *n* a *m*. Normalmente, la construcción y evolución de un producto puede deberse a uno o más proyectos, es decir, contratos. La existencia de varios contratos puede deberse a la evolución natural del producto o a requisitos derivados del proceso administrativo, la restricción presupuestaria, etc. Cuando un producto evoluciona, a través de varios proyectos, en un grado significativo, aparece una **versión**. La relación **producto** a **versión** es *1* a *n*.

Cuando existe una relación *1* a *n* de **proyecto** a **producto**, se trata de una crcunstancia excepcional, una relación impuesta por la mecánica de la contratación administrativa. Se produce cuando se han agregado varios desarrollos menores en un único contrato administrativo.

4.3.2. Trabajadores, vidas laborales y categorías

Las entidades representativas de los recursos humanos son la entidad **trabajador** y la entidad **vida laboral**. La entidad **trabajador** se corresponde con una persona, la cual evoluciona en la empresa a través de su **vida laboral**, que indica el coste horario de esa persona para la empresa en cada periodo de tiempo. Así pues hay una relación *1* a *n* entre **trabajador** y **vida laboral**.

Existe una tercera entidad, **categoría**, que tiene una denominación equívoca, pues no es un atributo del trabajador y el periodo temporal sino de la función que el trabajador realiza en un periodo concreto en un proyecto concreto; en el lenguaje convencional del estudio de procesos *software* corresponde más bien a un *rol*. Esta anomalía obedece a dos razones: en primer lugar a que los presupuestos que se elaboran en las ofertas contienen desgloses justi-

ficativos en función de las cargas de trabajo calculadas para cada categoría. En segundo lugar, a la flexibilidad del personal del caso de estudio, lo que hace que una misma persona pueda realizar varias funciones, apareciendo en un mismo proyecto en varias categorías (desempeñando varios roles).

A los efectos del control de costes esta anomalía es irrelevante pues lo que se controla es qué persona estuvo ocupada en qué proyecto y eso da lugar a un determinado coste, con independencia de la función que desarrolló. Pero, junto con otras carencias del sistema a los efectos de esta tesis es una información equívoca que no permite reconstruir un perfil más pormenorizado de los recursos empleados. Ello se debe a que, como la finalidad original del sistema era el control de costes y el dato de coste relevante es la persona (y no el rol) es una parte de la información que no se ha llevado con el suficiente cuidado como para ser utilizable.

4.3.3. El registro básico: trabajador-proyecto

El registro básico del sistema es el registro **trabajador-proyecto**($\mathbf{T}\times\mathbf{P}$). Idealmente, recoge para un determinado periodo de tiempo qué persona estuvo ocupada en qué proyecto y durante cuantas horas. En algunos casos recoge también la actividad concreta que desarrolló, pero no en todos.

La captura de esta información proviene de un procedimiento implantado en la organización que obliga a cada persona a registrar semanalmente el uso del tiempo entre los distintos proyectos en los que está participando. Junto con ese cómputo del tiempo se indica también la **categoría**, es decir, la función que ha realizado pero con la imprecisión ya señalada.

4.3.4. El tiempo

El tiempo está presente en el modelo como atributo de diversas entidades. Aparece registrado:

- en el registro básico **trabajador-proyecto**, como tiempo de inicio y tiempo final del periodo de tiempo descrito,

- en la **vida laboral**: como fecha inicial y final del periodo en el que el trabajador mantuvo el coste horario descrito,

- en el **proyecto**: como fechas contractuales y reales de inicio y fin del trabajo contratado,

- en la **versión**: como la fecha de implantación de la misma.

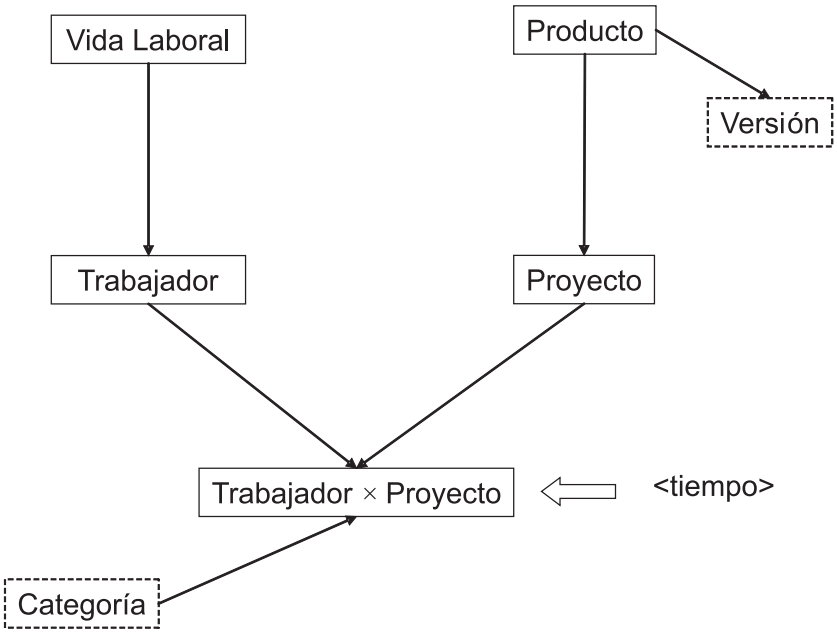


Figura 4.1: Estructura de la base de datos.

En abril de 2008, el sistema de información de apoyo al control de gestión se extendía desde 1993 hasta el momento citado. La fiabilidad de la información disponible, sin embargo, es muy escasa para los datos anteriores a 1999.

El número de filas incluidas en cada una de las tablas principales es el recogido en el cuadro 4.1.

Tabla	Registros
Producto	190
Proyecto	763
Versión	918
Proyecto×Producto	244
Trabajador	130
Vida Laboral	1.555
Trabajador×Proyecto	64.209

Cuadro 4.1: Número de registros.

4.4. Explotación del sistema: métricas de proyecto

Idealmente los datos incluidos en las bases del modelo permitirían reconstruir el ciclo de vida de un producto *software* a partir de la reconstrucción del ciclo de vida de los proyectos que han ido concurriendo en su evolución. No obstante, la existencia de posibles relaciones entre un proyecto y varios productos dificulta este trabajo puesto que las tareas elementales están descritas a nivel de proyecto y no de producto.

4.4.1. Reconstrucción del ciclo de vida

Alternativamente se puede reconstruir el ciclo de vida de un proyecto a partir de los registros **trabajador-proyecto**. De este modo un proyecto no es más que una secuencia de registros de la tabla **T**×**P** cada uno de los cuales indica durante un cierto periodo de tiempo cuánto esfuerzo le dedicó cada trabajador a ese proyecto y cumpliendo qué función (**categoría**). Desgraciadamente la información disponible no nos permite en la mayoría de los casos identificar qué tarea o tareas específicas realizaba dentro del ciclo de vida ese trabajador.

Cada registro contiene la información que se presenta en el cuadro 4.2.

Atributo	Representación	Rango
Trabajador	W_i	De la tabla **trabajador**
Proyecto	P_j	De la tabla **proyecto**
Categoría	C_k	De la tabla **categoría**
Tiempo de inicio	t_{ini}	Del calendario
Tiempo de finalización	t_{fin}	Del calendario
Carga de trabajo (horas)	H	$\in \Re, \geq 0$

Cuadro 4.2: Contenido de los registros de la tabla T×P.

4.4.2. Empleo de recursos

A partir de estos datos se pueden inmediatamente obtener las siguientes métricas del proyecto:

Carga total de trabajo del proyecto. La suma de las cargas de trabajo de los registros asociados a ese proyecto:

$$WL_j = \sum_{i,k} H(W_i, P_j, C_k, t_{ini}, t_{fin})$$

Duración del proyecto. La diferencia entre las fechas extremas de los registros del proyecto:

$$D_j = \max_j t_{fin} - \min_j t_{ini}$$

Intensidad aparente de la carga de trabajo. El cociente de ambas:

$$IA_l = \frac{WL_j}{D_j}$$

Estas métricas pueden también desglosarse por categorías limitando las sumas a cada una de las categorías posibles.

El cuadro 4.3 presentan las métricas obtenidas de una muestra de proyectos donde las columnas son, respectivamente las *horas-persona totales*, la duración en días y el cociente de ambas. La tabla muestra situaciones muy diferentes entre proyectos con una intensidad aparente de trabajo persona-

Proyecto	$\mathbf{WL_j}$	$\mathbf{D_j}$	$\mathbf{IA_j}$
195	467	604	0,772
204	310	291	1,066
205	9.363	1.331	7,034
215	384	412	0,932
221	7.383	1.325	5,572
222	1.458	886	1,646
231	6.938	1.055	6,577
232	2.283	942	2,424
235	589	1.018	0,579
305	2.910	1.090	2,670
237	6.923	1.529	4,528
240	8.028	1.623	4,947
216	624	1.179	0,529
253	2.027	1.187	1,708
260	618	1.054	0,586
266	5.431	307	17,689
269	4.750	958	4,958
272	1.011	485	2,085
283	407	370	1,099
291	777	1.096	0,709
294	1.052	1.014	1,037
297	483	816	0,592
338	2.099	580	3,619
369	1.468	741	1,981
402	812	312	2,603
μ			3,118
σ			3,633

Cuadro 4.3: Intensidad aparente.

hora/día muy bajas y otras bastante más elevadas. Eso indica la presencia de dos tipos de situaciones:

- Proyectos que sufren detenciones, por diversas razones que se exponen más adelante.
- Desplazamiento de personal entre proyectos, de manera que hay equipos que llevan adelante varios proyectos simultáneamente.

4.4.3. Fijación de errores y trabajos de integración

Puesto que cada proyecto tiene una fecha de entrega registrada, te_j, las tareas realizadas con posterioridad a dicha fecha son indicativas de las labores de corrección de errores y trabajos de integración con los sistemas del cliente. Como ya se ha indicado, ésta es una actividad importante en este tipo de desarrollos (ver apartado 4.1).

La empresa intenta conservar registro de estas actividades específicas genéricamente denominadas *extra* así como de la causa de las mismas, es decir, si se deben a errores propios del proyecto o a problemas sobrevenidos en la integración. Esta no es una cuestión sencilla pues, en ocasiones, los problemas sobrevenidos equivalen en la práctica a la aparición de un requisito que no había sido especificado previamente.

Para evaluar este esfuerzo y su impacto temporal, se han determinado las siguientes métricas:

Índice temporal post-entrega. Fracción de tiempo más allá de la fecha de entrega que se ha dedicado a corregir errores (o resolver problemas generados por la integración).

$$ET_j = \frac{D_j}{te_j - \min_j t_{ini}}$$

Índice de esfuerzo post-entrega. Proporción del esfuerzo dedicado a tareas más allá de la fecha de entrega. sobre el esfuerzo total. Evidentemente, no contempla la corrección de errores durante el periodo de

desarrollo previo.

$$EW_j = \frac{WL_j}{\sum_{i,k} H(W_i, P_j, C_k, t_{ini}, t_{fin})} \; \forall t_{ini} \geq te_j$$

El cuadro 4.4 presenta los resultados de esta última métrica para la misma muestra de proyectos. Comparando los estadísticos del trabajo extra, promedio y desviación típica, con los equivalentes del índice temporal post-entrega, 13 % y 29 % respectivamente, se concluye que los trabajos adicionales posteriores a la entrega ocasionan retrasos muy eelvados hasta que el producto desarrollado puede considerase totalmente terminado.

Obviamente existe más de una razón para esta demora. En el análisis concreto de los proyectos afectados se observa que algunas de las deficiencias surgidas en la integración con otros sistemas tardan en aparecer cuando las funcionalidades que interactúan no se ponen en marcha inmediatamente sino en un momento posterior. Hay también un componente, lógico, de "pérdida" de prioridad en un proyecto que ya está "instalado".

Sea como fuere, los datos agregados citados muestran un comportamiento de los proyectos que requiere mejorarse. Profundizando en el ciclo de vida de los mismos se pueden identificar algunos de los factores sobre los que hay que actuar.

4.5. Reconstrucción del perfil temporal de los proyectos

4.5.1. Depuración previa de la información

El perfil temporal de la utilización de recursos humanos en el proyecto puede reconstruirse con la superposición de todas los registros a lo largo del calendario de realización. Para ello ha sido necesario reparar los errores contenidos en las bases de datos de la manera que se expone en el apéndice E. La información registrada en estas tablas es introducida por un lado por los trabajadores de forma semanal especificando la distribución de las horas

Proyecto	**Antes de entrega**	**Después de entrega**	$\mathbf{D_j}$	$\mathbf{EW_j}$
181	312	155	467	33,12 %
186	289	21	310	6,79 %
205	8.437	926	9.363	9,89 %
215	352	32	384	8,33 %
221	5.385	1.998	7.383	27,06 %
222	1.377	81	1.458	5,55 %
231	6.600	338	6.938	4,87 %
232	2.193	91	2.283	3,97 %
235	514	75	589	12,74 %
305	2.710	201	2.910	6,89 %
237	6.413	510	6.923	7,37 %
240	7.655	373	8.028	4,65 %
253	534	90	624	14,42 %
260	1.613	415	2.027	20,46 %
266	607	11	618	1,82 %
269	5.280	151	5.431	2,77 %
270	4.734	15	4.750	0,32 %
272	771	240	1.011	23,70 %
276	391	16	407	3,89 %
283	773	4	777	0,51 %
291	1.038	14	1.052	1,33 %
297	435	48	483	9,88 %
308	1.896	203	2.099	9,67 %
369	1.136	332	1.468	22,61 %
402	764	48	812	5,88 %
μ				9,94 %
σ				8,85 %

Cuadro 4.4: Índice de esfuerzo post-entrega.

de su trabajo entre los proyectos en los que ha participado a lo largo de los últimos cinco días. Otras veces los registros son completados por los Jefes de Proyecto antes, durante y al final del proyecto.

Como consecuencia de este rellenado manual de las tablas se encuentran en las mismas muchos errores, contradicciones, lagunas de información, etc. que deben ser corregidos para poder proceder al análisis de la distribución de las cargas de trabajo de las personas que trabajan en la empresa y que éste arroje unos resultados que se ajusten a la realidad de la misma.

En primer lugar, observando detenidamente los datos del sistema de gestión y mediante entrevista con el creador y mantenedor del mismo, se decidió descartar la información excesivamente antigua. La razón es que el sistema de gestión no se implanta y se comienza a utilizar de forma eficiente hasta el año 2000, lo que motiva que la información correspondiente al periodo anterior tenga un altísimo nivel de errores, que introducirían ruido en demasía para el análisis de los datos.

El proceso de depurado se realiza sobre la tabla T×P, puesto que al ser la que maneja todo el personal de la empresa es la que contiene el mayor número de errores. Una vez finalizado el proceso de depuración de la tabla T×P, lo que queda es una tabla en la que figuran las cargas de trabajo de todos los proyectos seleccionados sin ningún tipo de solapes ni incoherencia, que es la que permite caracterizar el perfil de los proyectos.

4.5.2. Perfiles tipo de proyecto

Los datos muestran que la mayor parte de los proyectos presentan el perfil de empleo de recursos típico, una acumulación de recursos en la primera fase y, posteriormente, una caída en el empleo de los mismos. Este perfil de empleo es el que integrado genera la típica curva de progreso de forma sigmoidal. Así, por ejemplo, el proyecto presentado en la figura 4.2 presenta una acumulación de dedicación en la fase característica de la construcción.

No obstante, en varios proyectos se observan perfiles de ejecución más atípicos. Así, en las figuras siguientes (4.3 y 4.4) se puede apreciar la oscilación en el personal dedicado. El análisis realizado, a partir de las entrevistas a los

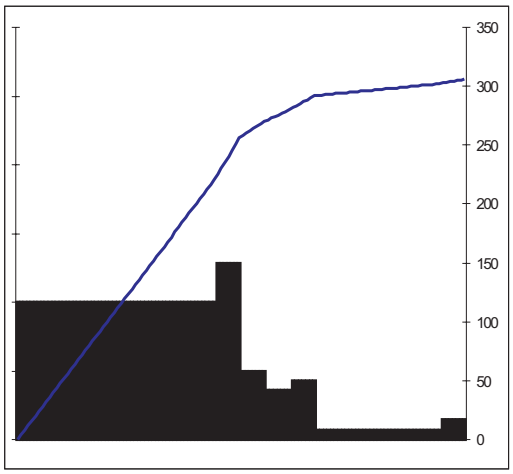


Figura 4.2: Perfil de un proyecto tipo.

directores de los proyectos, apunta a las siguientes causas:

- Después de un fuerte impulso inicial, se retira personal del proyecto para atender a otros de mayor prioridad, hasta que puede volverse a retomar (caso de la figura 4.3).
- Se producen cambios en las especificaciones sobre la marcha, que dan lugar a nuevas puntas de trabajo (caso de la figura 4.4).

Ambos casos, de origen completamente diferente, dan lugar a un comportamiento aparentemente similar. La diferencia está en *la dirección de la causalidad*. En el primer caso el proyecto ha de ceder recursos a los demás. En el segundo, son causas internas del proyecto las que generan la irregularidad.

Un tercer caso es el de los proyectos "urgentes", en los que se debe poner a punto una aplicación, o todo un sistema, en un plazo fijo insuperable. Este es el caso de la figura 4.5, en el que la acumulación de recursos tras un retraso obviamente se realiza a costa de otros proyectos.

4.6. El perfil de los proyectos y el modelo a construir

En todos los casos se puede observar lo siguiente:

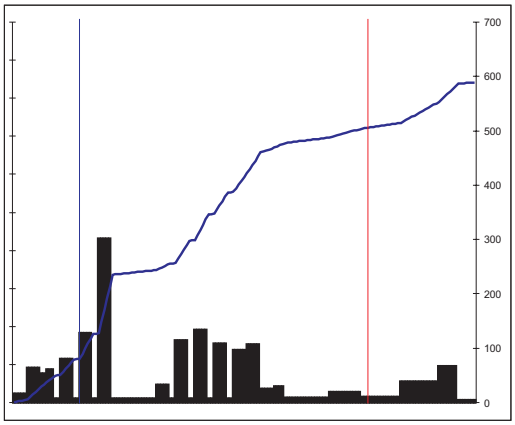


Figura 4.3: Proyecto que cede temporalmente recursos.

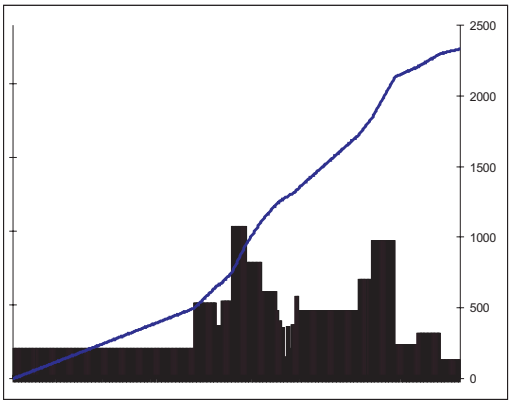


Figura 4.4: Perfil de proyecto con requisitos volátiles.

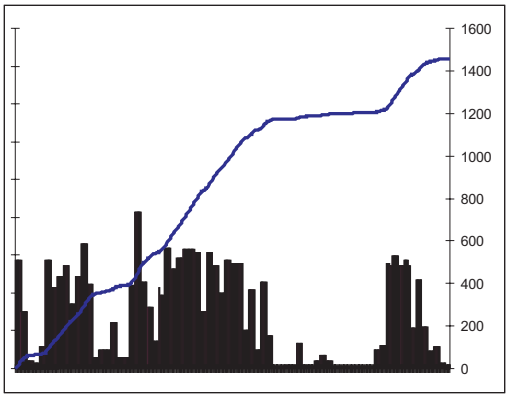


Figura 4.5: Acumulación de recursos para finalizar un proyecto.

- La voluntad de la empresa de mantener equipos estables con los proyectos y clientes se compatibiliza con la necesidad de atender los cambios en los requisitos y otras circunstancias externas que afectan a los proyectos, desplazando personal continuamente de uno a otro proyecto.

- A pesar de que el ciclo de asignación tiene una semana de periodo, se producen cambios a veces hasta diarios. El efecto de estos cambios tan frecuentes en la productividad no puede ser más que negativo.

- En algunos casos estos cambios pueden estar justificados por las cargas de trabajo, las detenciones producidas a instancias (o por culpa) del cliente, o cualquier otra circunstancia justificada. de los proyectos a los que el personal está asignado. Pero en la mayoría de los casos da la sensación de que es la dinámica general de cambio de prioridades y las interrelaciones entre proyectos provocadas por la asignación de personal a varios proyectos a la vez la que determina los cambios citados.

La conclusión que se obtiene es doble:

- De una parte, se precisan criterios de asignación de recursos que puedan ser más parsimoniosos con el fin de preservar el objetivo de equipos relativamente estables.

- De la otra, y en lo que afecta a esta tesis, que los modelos continuos de simulación de proyectos que han sido reseñados en el capítulo anterior tienen dificultades para reproducir está dinámica, que es básicamente discreta. Este es el objetivo del modelo que se presenta en el capítulo 6.

Capítulo 5

Análisis de las soluciones existentes y propuestas para el modelado

5.1. Introducción y contenido del capítulo

En este capítulo se contrastan las soluciones revisadas en los capítulos 2 y 3 con los factores que se estiman críticos que, tal como se indica en 1.4, son:

- Desde el punto de vista de los modelos de gestión de proyectos:
  - La representación del entorno multiproyecto.
  - Los algoritmos de asignación de recursos susceptibles de modelado dinámico.
  - El tratamiento del riesgo.
- Desde el punto de vista de los modelos de simulación existentes:
  - La capacidad de representar los aspectos continuos y discretos de los proyectos *software* así como la superposición y retroalimentación entre fases y proyectos.
  - La implementación de métodos de asignación de recursos.

- La capacidad de representar y medir los efectos de los procedimientos de mejora de procesos.

De este análisis y de las características específicas del caso de estudio se desprenden las especificaciones que se han tenido en cuenta a la hora de construir el modelo objeto de esta tesis. Algunos de los elementos que han sido ya estudiados en los trabajos presentados no se incorporan pero se prevé su posterior inclusión en el modelo.

5.2. Modelo del entorno multiproyecto

Se ha considerado el problema de la gestión multiproyecto en dos marcos de referencia. El primero clasifica los entornos multiproyecto en términos de *variabilidad* y *dependencia*. El segundo diferencia entre los niveles de decisión *estratégico*, *táctico* y *operacional*.

La decisión de aceptar proyectos se ha dejado fuera del ámbito del problema en estudio. Se deriva de criterios estratégicos y de la acción comercial. A los efectos que de este trabajo, los proyectos "aparecen" previamente "marcados" con fechas límite que pueden deberse a condiciones internas o externas.

Según el análisis expuesto, una organización dedicada al desarrollo de *software* a medida es típicamente una organización por proyectos, por tanto *orientada a reducir la dependencia entre proyectos a nivel operacional asumiéndola en el nivel táctico*. En la práctica, lo que se persigue en la medida de lo posible es independizar en términos de recursos cada proyecto individual. Esta independencia será siempre relativa, en función del recurso del que se trate. Habrá por tanto recursos que se pueden compatibilizar entre proyectos y recursos exclusivos de cada uno de ellos.

Los objetivos de gestión entonces se plantean como:

- En el nivel *táctico*; asignar a los proyectos los recursos necesarios para asegurar el cumplimiento de plazos de entrega. Estos recursos deben obtenerse empleando la capacidad productiva de la empresa de la forma más económica posible y dentro de los requisitos de calidad de los

proyectos. Se trata de un problema de *planificación táctica de capacidad* que debe considerar, aparte de las restricciones temporales, las asignaciones previas y el coste de los recursos.

- En el nivel *operacional*; programar los proyectos dentro de los márgenes impuestos desde el nivel táctico de forma que se pueda hacer frente a la variabilidad propia de cada uno de ellos. Se trata de un problema de *programación de proyectos con restricción de recursos* en entorno de incertidumbre y con ventanas de tiempo.

5.3. La asignación de los recursos a los proyectos y la programación de los mismos

Como se ha expuesto en el punto anterior, se ha descompuesto el problema en dos, uno táctico y otro operacional.

5.3.1. El problema táctico

Se ha estudiado el problema de dimensionar los equipos necesarios para abordar los proyectos de manera que se asegure que se cumplen los plazos. Se trata de un problema *restringido por el tiempo*.

Los proyectos se representan como una serie actividades agregadas de manera que la proporción entre los distintos recursos que emplea cada una de ellas es constante. El modelo admite que los recursos sean compartidos por varios proyectos, algo que a este nivel de análisis se corresponde con la realidad del problema en estudio.

En los proyectos de *software* a medida, existen hitos externos caracterizados por entregables que deben hacerse llegar al cliente. Esto se representa de forma adecuada con *las ventanas de tiempo admisibles*.

En el caso que nos interesa, la planificación táctica debe dar lugar a una previsión del personal necesario, por encima de la plantilla disponible. El modelo *restringido por el tiempo* maneja como variables de decisión los recursos extraordinarios a emplear. Atribuyendo coste nulo a los recursos ordinarios

Objetivo	minimizar coste de recursos empleados
Restricciones	cumplimiento de hitos externos relaciones de precedencia
Variables de decisión	Nuevos recursos a adquirir Porcentaje de trabajo hecho por periodo
Referencias	[GS05]

Cuadro 5.1: Modelo propuesto para la planificación táctica.

y minimizando el coste de los recursos emleados, la solución proporciona, además de una asignación de personal a proyectos una indicación sobre la contratación necesaria en el horizonte de planificación.

El modelo elegido presenta algunos problemas para el caso de estudio:

- El modelo asume una productividad lineal, es decir, que el tiempo de desarrollo es inversamente proporcional al número de personas empleadas en las tareas, algo que no es evidente en el caso del desarrollo de *software* cuando el nivel de agregación es alto.

- El modelo es estático, en el sentido de que la aparición de un nuevo proyecto obliga a una reprogramación. Este segundo aspecto se trata más adelante al hablar de la cuestión del riesgo.

La inclusión de un nuevo proyecto generará una nueva solución factible simplemente añadiendo recursos extraordinarios para cumplir los plazos. Este es el punto de partida para una nueva exploración ahora con el nuevo proyecto.

5.3.2. El problema operacional

El problema operacional se ha definido como un problema de *programación multiproyecto y multimodal con restricción de recursos y ventanas de tiempo*, es decir, un problema MRCMPSP-GPR como se ha expuesto en la página 43.

La aplicación de las técnicas *multiproyecto* al problema operacional no se considera adecuada teniendo en cuenta la estrategia de descomposición adoptada. En el nivel operacional, los equipos de cada proyecto han sido definidos desde la planificación táctica de capacidad. En condiciones normales,

para un director de proyecto la variable de decisión es la asignación de los componentes de su equipo a las distintas tareas de su proyecto.

Caben distinguir dos situaciones, la programación inicial del proyecto y la respuesta a los problemas en la ejecución del mismo.

Programación inicial del proyecto

En este caso el objetivo es minimizar la duración total del proyecto individual administrando los equipos de personal disponible. Este es un caso claro de programación *multimodal*. Una vez seleccionado el modo de ejecución, lo normal es mantener éste. En la realización de proyectos *software* la incorporación de personal a actividades que ya se están realizando no resulta aconsejable para evitar los "costes de aprendizaje". Este fenómeno ha sido muy estudiado en la literatura de simulación de proyectos *software* y está expuesto con bastante claridad en el modelo MDR descrito en el capítulo 3.

En la literatura descrita aparecen diversas opciones para solucionar este problema. Dado que la magnitud del mismo no es excesivamente grande, se propone el empleo de *heurísticas* basadas en *reglas de prioridad dinámicas*, generando secuencias en *serie* y representando las soluciones en la forma *lista de actividades*.

A pesar de lo anterior, desde el punto de vista de la supervisión desde el nivel táctico tiene sentido contar con una función objetivo más global. Se propone para ello un seguimiento de la máxima desviación de todo el conjunto de proyectos simultáneos.

Reparación de la programación operativa

La principal diferencia entre la programación operativa inicial y la reparación de la misma una vez surgida una disrupción es que en el primer caso el obejtivo ("a priori") es la minimización de la duración total del proyecto, respetando las ventanas de tiempo. En el segundo, suele producirse que el objetivo más perentorio sea el de cumplir con las limitaciones temporales de una o varias actividades antes que el tiempo de terminación del proyecto concreto.

En un entorno de *compresión* de la programación tiene sentido romper el criterio antes señalado de que una vez elegido un modo de ejecución, éste se mantiene constante hasta la finalización de la actividad afectada. Por el contrario, la respuesta normal es la acumulación de recursos en la tarea que se retrasa.

De este modo, aunque el problema inicial es un problema MRCPSP-GPR, pasa a ser más bien del tipo MRCPSP-GPR/EXP, en el sentido de que se modifica la asignación de recursos.

5.4. El tratamiento del riesgo

Se ha estudiado el riesgo desde dos puntos de vista: el más corriente en términos de la gestión de proyectos y algunas aportaciones desde otros dominios entre las que destaca la metodología de la *design structure matrix* -DSM- originada en el campo del desarrollo de productos que permite captar y tratar los ciclos y las iteraciones, facetas que comparten los procesos de desarrollo de producto y los procesos *software*.

Con un análisis más detallado de la incidencia de la variabilidad y la dependencia en los dos niveles, táctico y operacional, se llega a la conclusión de que:

- Al nivel táctico, las estrategias son las de obtener programaciones *robustas*, es decir, capaces de absorber las variaciones sin grandes desviaciones sobre las fechas de terminación programadas inicialmente.

- Al nivel operacional, las estrategias *predictivas-reactivas* orientadas a la flexibilidad, es decir, a la recuperación de programaciones que se ven alteradas por alguna disrupción aparecen como la mejor alternativa.

5.4.1. Estrategias para obtener programaciones robustas a nivel táctico

En el nivel táctico, las fuentes de incertidumbre son de dos tipos:

Programación	Inicial	Recuperada
Objetivo	Minimizar la duración total	Cumplir fechas críticas
Modo	Constante	Acumulación de recursos
Problema tipo	MRCPSP-GPR	MRCPSP-GPR/EXP
Variables de decisión	Fechas de inicio y modo	Fechas de inicio y recursos empleados
Referencia	[LTB06]	[JP05]

Cuadro 5.2: El problema operacional: solución inicial y reparación de la programación.

Incertidumbre en la carga real de trabajo. En el caso de la planificación táctica los proyectos están definidos a nivel agregado. El conocimiento de la carga de trabajo real es aproximado y, sin embargo, se adquieren compromisos de fechas y entregas. Una estrategia *proactiva*, típicamente del estilo de la *cadena crítica*, se basa en el dimensionado de *buffers* de recursos bien con datos *crisp* o empelando modelos basados en la lógica borrosa. En todos los casos, este tipo de solución contiene inevitablemente un cierto nivel de sobredimensionado de los recursos, lo cual se corresponde poco con el caso estudiado.

Incertidumbre por aparición de nuevos proyectos . Los nuevos proyectos, o la subestimación inicial de un proyecto existente, plantea un problema diferente al anterior. Aquí se trata de "insertar" la(s) nueva(s) actividad(es) en la programación prevista alterando esta lo menos posible. De hecho, el caso anterior de desconocimiento de la carga real de trabajo puede reducirse a este caso, "generando" una actividad nueva que recoja el trabajo recientemente descubierto y las restricciones temporales correspondientes.

La solución propuesta en [AR00] y [AMR03] se basa en el empleo de los denominados *grafos de recursos*. La idea es que la restricción de recursos se represente como un grafo de flujos mediante el cual las actividades agregadas, representadas en notación AEN, "entregan" los recursos que emplean a las actividades que les siguen una vez que han terminado. Así, además de las restricciones temporales, aparecen unos arcos nuevos que representan las restricciones de recursos.

La inserción de una actividad o varias actividades nuevas (un trabajo imprevisto o un nuevo proyecto) equivale a la inserción de uno o varios nodos nuevos en la red del proyecto y un "desvio" del flujo de recursos por esos nuevos nodos. Esto se consigue mediante cortes en el grafo cuya capacidad sea igual a la demanda de recursos requeridos por la actividad insertada. Un corte *dominante* es aquel que minimiza el aumento de la duración total. Los autores reseñados proponen un algoritmo en tiempo exponencial para generar cortes dominantes. Incorporando ese algoritmo a un SGS, se puede

reconstruir la planificación táctica empleando reglas de prioridad.

5.4.2. Reconstrucción de secuencias operacionales

Como se ha indicado más arriba, la reconstrucción de secuencias en el caso de un proyecto concreto, se basa en la idea de incrementar los recursos en una actividad que se retrasa, rompiendo la asignación modal inicialmente prevista, al contrario que en el caso anterior. En la terminología técnica este modo de proceder se denomina "activity-crashing".

El problema real que se intenta representar es el de la detección de un retraso excesivo sobre la programación a consecuencia de un imprevisto, un error e o de la asignación de personal a otro proyecto. En tal caso lo que se persigue es reparar una secuencia reasignando recursos dentro del mismo proyecto si es posible. De lo contrario, detectar la imposibilidad de hacerlo para emitir una petición de recursos adicionales al nivel táctico.

A diferencia de los heurísticos basados en la exploración descritos en la página 53 que exploran el *espacio de las soluciones*, el procedimiento elegido explora el *espacio de las estrategias*. La estrategia, en el caso de un heurístico de secuenciación basado en reglas de prioridad, es la selección de prioridades. La solución propuesta secuencia con un constructor *greedy* y, posteriormente, modifica las prioridades hasta que encuentra una solución aceptable "penalizando" las actividades mal secuenciadas. La solución propuesta recuerda a la metodología de *agentes* en el sentido de que se simula una solución circunscrita a un proyecto, cuyo asignador actúa como un agente autónomo. Para ello, y teniendo en cuenta que la respuesta clásica de un jefe de proyecto a una eventualidad de este tipo es la de reasignar personal de unas tareas a otras para atender a la más crítica, se ha implementado un algoritmo del tipo SWO (*squeaky wheel optimizaton*)[JC99].

La filosofía del algoritmo es, de manera análoga al agente humano, modificar las prioridades que dieron lugar a la primera asignación para incrementar sucesivo de recursos para la actividad ahora retrasada. A partir de ahí se opera con un SGS. Si la solución obtenida es poco satisfactoria, el algoritmo altera aleatoriamente las prioridades y reprograma de nuevo durante un

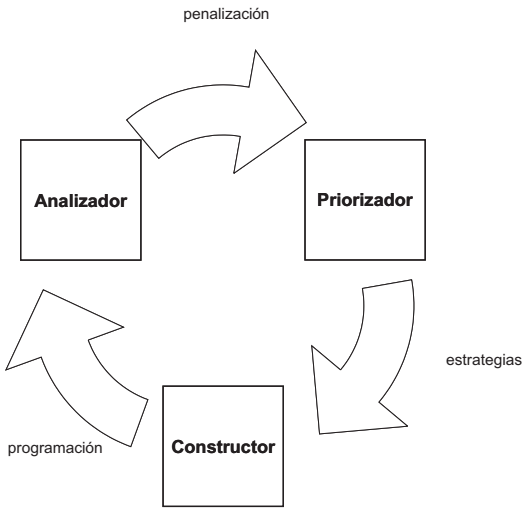


Figura 5.1: Componentes del algoritmo *swo*.

número de iteraciones que se fija previamente.

5.5. Modelos de simulación del proceso *software*

En el capítulo 3 se han revisado diferentes modelos de simulación del proceso *software*. Atendiendo a las diferentes características buscadas en esta tesis, estos modelos pueden clasificarse desde diferentes puntos de vista.

Una primera cuestión es la representación del entorno *multiproyecto*. Caben básicamente dos aproximaciones que hemos denominado respectivamente *contingente* y *jerárquica*. En el primer caso, como ocurre con el modelo de Abdel Hamid [Abd93b], la representación multiproyecto es "ad-hoc", es decir, se modela una situación de este tipo pero el modelo no permite variar la estructura de relaciones con facilidad. En cambio, en la aproximación *jerárquica* el propio modelo es innatamente multiproyecto. El carácter jerárquico y modular de esta aproximación es lo que más interesa a los efectos de esta tesis.

La segunda cuestión es el carácter continuo, discreto o híbrido de los modelos. Esta cuestión está relacionada con la *orientación* de los modelos, entendida esta como el enfoque principal u objetivo de los mismos. Como es

característico en la simulación hay dos tipos de orientaciones, una *estructural* que enfatiza las relaciones entre los diferentes componentes del sistema modelado, siendo esta la aproximación característica de la Dinámica de Sistemas; y una orientación estadística, que emplea el modelo para generar experimentos repetidos mediante el método de Monte Carlo y así caracterizar el sistema en función de la distribución estimada de una o varias variables aleatorias.

Tradicionalmente la orientación estructural se basa en modelos deterministas, utilizándose el *análisis de sensibilidad* para evaluar las diferentes "regiones de comportamiento"[1]. Ello no impide el empleo de variables aleatorias en la simulación, pero la complejidad del comportamiento se deriva de las relaciones estructurales.

En los modelos de orientación estadística lo que se persigue es caracterizar una variable aleatoria cuyas funciones de distribución y densidad, y por tanto sus momentos, no pueden determinarse de modo analítico. El método empleado es la simulación de Monte Carlo empleando generadores pseudoaleatorios para las variables de entrada y realizando un número suficiente de experimentos. La orientación, por tanto, es la caracterización estadística del riesgo.

Las dos últimas cuestiones son si los modelos presentados hacen o no referencia a modelos de asignación de recursos provenientes del área de la Ingeniería de Proyectos y si tienen o no en cuenta los aspectos relacionados con la mejora de procesos *software* y, en general, la medida y el control de los proyectos.

La tabla 5.3 sintetiza estas características lo que permite identificar las aportaciones que se han considerado más útiles para el objetivo de esta tesis y que se presentan en los apartados siguientes.

5.5.1. Modelos multiproyecto

La mayoría de los modelos multiproyecto que se han analizado son del tipo que se ha denominado *contigente* en el sentido de que modelan explíci-

[1] En el análisis cualitativo de sistemas dinámicos no lineales, se trataría del análisis del espacio de las soluciones en función de los parámetros y de las cuencas de atracción [AT93].

Modelo	Multiproyecto	Tipo de Modelo	Modelos de asignación	SPI y medida
IMMoS [PR02]	Contingente	Híbrido		Sí
Rus et al. [RCL99]		Híbrido		
Donizelli y Iazolla [DI01]		Híbrido	Parcialmente	
Martin y Raffo [RM00]		Híbrido		
DEVS [CBK06]	Jerárquico	Híbrido		
Producción global [SWR07]	Jerárquico	Híbrido		
Abdel Hamid [Abd93b]	Contingente	Continuo	Parcialmente	
Powell [PMB99]	Jerárquico	Continuo	Parcialmente	
Lee y Miller [LM04]	Contingente	Continuo	Parcialmente	
Stallinger [Sta00]		Continuo		Sí
DIFSPI [Car03]		Continuo		Sí
Antoniol et al. [ACLP04]		Eventos (Monte Carlo)	Sí	
Padberg [Pad02b]		Eventos (Monte Carlo)	Sí	
Browning et al. [MYB07]		Eventos (Monte Carlo)	Sí	
Joslin [JP05]		Eventos (Monte Carlo)		

Cuadro 5.3: Características de los modelos revisados a efectos de los objetivos de la tesis.

tamente una instancia en la articulación entre los distintos componentes del proceso pero no permiten naturalmente representar cualquier configuración de proceso sino sólo aquella explícitamente modelada.

La principal excepción la constituye el modelo de Powell *et al.* (ver apartado 3.4.2) que modela un ciclo de vida incremental y las relaciones de concurrencia entre diferentes paquetes de trabajo que, a otra escala, equivalen a la competencia por recursos de varios proyectos simultáneos. El denominado*triángulo del proceso* se repite a todos los niveles de la jerarquía: paquetes de trabajo, fases, entregas, proyectos y organización en su conjunto. Para acoplar los diferentes módulos de la jerarquía se emplea tanto la asignación de recursos a módulos como el acoplamiento de los trabajos y así se consigue modelar la superposición de fases del desarrollo incremental.

El modelo DEVS (ver apartado3.3.5), si bien no aparece así documentado, se presta con mucha facilidad a la implementación jerárquica por las propiedades del formalismo en cuestión, que permite la implementación de la jerarquía de forma natural, encapsulada y escalable.

5.5.2. Modelos híbridos

En el capítulo 3 se han revisado diversos modelos, continuos, discretos e híbridos, elaborados desde diversos enfoques y puntos de vista para simular el proceso *software*. La simulación híbrida es una tecnología de uso general en muchos dominios por lo que existen muchas soluciones.

En el problema que aquí se está planteando, la cuestión crítica parece ser la de no "perder" el efecto de la realimentación en el paso de continuo a discreto y viceversa. Desde el punto de vista conceptual, el problema es identificar que procesos tienen lugar de forma continua y qué procesos se producen de forma discreta. Esto está ampliamente documentado en la literatura, Los mecanismos que regulan la productividad, la tasa de errores, el efecto de la presión de los plazos, son típicamente continuos. El ciclo de vida de las tareas y los entregables, sin embargo se manifiesta de forma discreta. Y desde luego la información y las decisiones son discretas.

Los modelos analizados poseen una arquitectura que separa y limita las

relaciones entre la parte continua y la parte discreta de diversas formas:

- Los modelos presentados en 3.3.3 y 3.3.1 modelan como continuo el entorno de trabajo, la productividad y la dinámica de los recursos, mientras que tratan como discreta la evolución de los trabajos. Esto da lugar a que no exista una verdadera realimentación entre la marcha de trabajo y su entorno.

- El modelo presentado en 3.3.2 realmente lo que hace es simular un proceso continuo de ejecución de tareas para caracterizarlo estadísticamente y emplear esta caracterización para tratarla mediante un modelo analítico de teoría de colas, este sí discreto. Una vez más, no hay realimentación entre la marcha del trabajo y el entorno.

- El modelo presentado en 3.3.4 implementa una estructura jerárquica del estilo de la que aquí se persigue, al modelar diferentes equipos trabajando en la producción de *software* para un único proyecto. Sin embargo, el interés del modelo está en más en los aspectos relacionados con las diferencias culturales y el modo de trabajar en localizaciones distintas. Desde el punto de vista de laoperación híbrida coincide con el de Martin y Raffo, ya señalado.

- El modelo presentado en 3.3.5 es el único en el que la simulación híbrida se materializa de forma integrada con un único formalismo, el denominado formalismo DEVS, *Discrete Event System Specification*, propuesto por Ziegler [ZKP00]. Sin embargo, por lo menos hasta lo que se ha publicado, en este caso no se implementan sistemas complejos de asignación de recursos y el modelo se limita a emular el funcionamiento de los modelos continuos de la saga del modelo de Abdel Hamid y Madnick [AM91]. Puede decirse, por tanto, que si bien ofrece una herramienta adecuada para implementar la simulación híbrida, su uso se limita a una integración numérica por un método alternativo al basado en incrementos finitos[2].

[2]Se trata del método denominado QSS - *quantized-state system*- en el que las variables de estado se aproximan con funciones constantes a tramos - *piecewise* - con histéresis

Modelo	Referencia	Ventajas	A mejorar:
Rus *et al.*	[RCL99] 3.3.1	Análisis explícito de los aspectos discretos de artefactos y fases.	Sólo efectúa un número limitado de iteraciones
Donizelli y Raffo	[DI01] 3.3.2	Emplea métodos analíticos junto a la simulación.	No hay realimentación
Martin y Raffo	[RM00] 3.3.3	Evolución dinámica de los recursos y asignación discreta de los mismos.	No determina de forma continua la duración de las actividades.

Cuadro 5.4: Ventajas y aspectos a mejorar en algunos de los modelos híbridos revisados

Esta última aproximación es la que se ha tomado como base para el modelo propuesto en esta tesis. Desde el punto de vista de la implementación el problema de la simulación híbrida lo resuelve la utilización de la integración numérica para la simulación continua. Es, por tanto, un problema de intervalos de tiempo y de gestión de eventos. Salvo que estemos en presencia de mecanismos no lineales que den lugar a fenómenos de bifurcación, los métodos de integración numérica pueden resolver el problema. Queda como tarea para el posterior desarrollo de este trabajo investigar analíticamente los modelos que resulten para caracterizar, aunque sea aproximadamente, el espacio de estados e identificar posibles anomalías.

5.5.3. La implementación de las técnicas de la gestión de proyectos en los modelos de simulación

Los modelos analizados que incorporan explícitamente las técnicas de asignación, tal como se indica en la tabla 5.3, están orientados a la simulación de Monte Carlo de manera que lo que pretenden es caracterizar el riesgo ante la adopción de determinadas reglas de asignación de recursos. En ningún caso consideran la existencia de dos ámbitos de decisión - el táctico y el operacional

definidas para intervalos discretos de la magnitud de la variable - *quantized* - frente a los intervalos discretos de tiempo característicos de la integración numérica convencional[KJ01]. Así, cada evento discreto viene definido no por un cambio en el intervalo de tiempo considerado sino por un cambio discreto en la magnitud de la variable de estado.

- y requieren una caracterización estadística de las entradas que se adapta poco al problema que trata esta tesis.

El modelo de Lee y Miller ([LM04]) analizado en el apartado 3.4.3 si bien explícitamente emplea modelos de gestión de proyectos, en este caso la Cadena Crítica (ver página 61), pero no contempla en su integridad la dimensión multiproyecto a la hora de la asignación ni realiza ésta de forma dinámica sino a través de la técnica de *escenarios*.

A los efectos de esta tesis, el problema es modelar el proceso por el cual los decisores, situados a niveles diferentes en la jerarquía de decisiones, determinan los recursos asignados a proyectos y actividades de forma dinámica. La representación del decisor siguiendo el modelo de agentes permite implementar ese proceso. Esta es la aportación del modelo presentado en 3.6.5; los agentes, de acuerdo con las estrategias que se modelarán, reúnen información aproximada sobre el estado del sistema y formulan reglas de las que se desprenden las decisiones de asignación de recursos y de secuenciación de tareas.

Esto se puede modelar a ambos niveles, al táctico y al operacional. El decisor táctico asigna equipos a una cola de proyectos. Eventualmente decide adquirir nuevos recursos, sustituyéndose así el mecanismo continuo de los modelos estrictamente continuos por paquetes de contratación que se deciden cuando se establecen los planes tácticos. Los recursos evolucionan de forma continua: tanto en su calidad mediante procesos de aprendizaje como a través de una tasa de abandono "natural". El decisor táctico también puede decidir reasignar recursos de un proyecto a otro. Para ello tiene unos umbrales a partir de los cuales se considera esta opción.

El decisor operacional gestiona sus recursos y hace frente a las disrupciones en su programación readaptando ésta o reasignando los recursos de que dispone entre tareas para asegurar que se cumplen los plazos. La aproximación es la de la *secuenciación contingente*: "qué ocurre si ...?" En la literatura hay modelos que permiten representar estos procesos de decisión y sus consecuencias. El heurístico expuesto en 3.6.5 puede explorar el espacio de las estrategias para implementar de forma eficiente la secuenciación contingente.

La integración de ambos niveles y la representación de los procesos de

desarrollo incremental, con sus ciclos entre fases se puede implementar de manera análoga a la que se presenta en 3.4.2. El mecanismo básico que gobierna la evolución al nivel más elemental, previa adaptación, puede ser el de los modelos reducidos expuesto en 3.2.1.

5.5.4. La mejora de procesos y la simulación

Las referencias encontradas a la simulación de procesos *software* en relación con la mejora de esos mismos procesos son abundantes. Tanto para el CMM y CMMI (las más) como para la norma ISO-IEC 15504 (menos). Sin embargo, los modelos consultados caen dentro de dos grandes grupos:

- Unos presuponen la madurez de la organización y modelan de acuerdo con ésta. Es el caso del modelo *DIFSPI* (ver página 89). Este modelo permite evaluar la diferencia entre procesos realizados en condiciones de madurez diferentes.

- Otros pretenden modelar explícitamente los procesos de mejora y control. Son los casos del modelo *INMoS* y del modelo de Stallinger, presentados respectivamente en los apartados 3.2.2 y 3.5.1. Sin embargo, el primero adolece de tratarse de un modelo "ad-hoc", en el sentido de que, como los propios autores reconocen, requiere un esfuerzo de modularización posterior. El segundo caso anuncia un trabajo cuya conclusión no se ha publicado (téngase en cuenta que la referencia es del año 2000) y en el que se dan indicaciones de cuáles pueden ser precisamente las dificultades para llevarlo a término.

El problema de construir modelos que recojan la mejora de procesos es que los estándares orientados a las capacidades a nivel de organización (CMM y CMMI) o a nivel de proceso (ISO 15504), si bien proporcionan objetivos genéricos para la mejora de la calidad de los procesos y guías para su evaluación, no indican vías unívocas para su puesta en aplicación en un caso concreto. La razón de ello se en que la gran variedad de tecnologías, modelos de desarrollo y tipologías organizacionales existentes determina el carácter dependiente del camino (*path dependent*) de las estrategias posibles de mejora,

incluso aunque se utilice un modelo estándar como referencia. O, expresado de otra forma, se conocen los atributos de un grado determinado de madurez pero no la trayectoria para ir de un grado a otro.

Este razonamiento está en la raíz de la discusión ya señalada en el apartado 1.2.4 sobre si la madurez determina los límites de la metodología de simulación. La aproximación al problema que se ha considerado más interesante, por las razones que se exponen en 3.5.3 es la de modelar explícitamente las tareas que la mejora impone. El modelado explícito de estas tareas tiene dos virtualidades:

- Proporciona mayor precisión en el cálculo del esfuerzo requerido que no es sólo el trabajo directo de análisis y construcción sino que incluye, además de la propia gestión, las tareas anexas a ésta que corresponden a lo que genéricamente se llama la sobrecarga de comunicación (*communication overhead*)

- Arroja un mapa efectivo para la mejora de los procesos al permitir contrastar el coste de la misma con el beneficio esperable

Dado que la progresión en la madurez de la organización desarrolladora avanza en el sentido de mejorar la calidad de la gestión, al menos en los tres primeros niveles del marco de referencia del CMMI, esto se modela explícitamente en la mejora de los mecanismos de asignación indicados en el apartado anterior.

5.6. Propuestas de incorporación a las especficaciones del modelo

Como resumen del análisis expuesto del trabajo preexistente y de las características del caso de estudio se concluye que los factores críticos señalados al principio de éste capítulo pueden resolverse empleando las propuestas y modelos analizados de la forma que se describe a continuación.

1. La representación del entorno multiproyecto debe realizarse mediante una arquitectura jerárquica del tipo de la propuesta por Powell

[PMB99] que recogerá la gestión del trabajo, de los recursos y de los plazos. Este mismo modelo posibilita la superposición y retroalimentación entre fases y proyectos y resuelve el problema de la *contingencia* permitiendo modelar diferentes tipos de proceso.

2. Esta misma referencia y el modelo DEVS [CBK06] establecen una diferencia entre el modelado del *trabajo*, el de la *medida del resultado del mismo* y el de la *asignación* que permite separar variables *continuas* de *discretas* y el modelado de *procesos* del de *decisiones*.

3. Los algoritmos de asignación de recursos deben contemplar los diferentes niveles de la jerarquía de manera autónoma. Eso supone que existen ciclos de decisión y estrategias diferentes para asignar recursos dentro de las actividades o fases de los proyectos, en estos y a nivel táctico.

4. Los modelos expuestos basados en reglas de prioridad dinámicas se pueden aplicar al nivel más bajo, es decir, dentro de una actividad y entre las actividades de un proyecto. En el primer caso, el modelo MDR presentado en la página 74 implementa una versión dinámica de la regla RWK que debe discretizarse. En el segundo caso se necesita un modelo capaz de tratar el problema multimodo con ventanas de tiempo (MRCPSP-GPR).

5. El modelo de planificación táctica de capacidad presentado en la tabla 5.1 es el adecuado para el nivel multiproyecto pues permite establecer las necesidades de personal para el horizonte de planificación y, por tanto, las decisiones de contratación.

6. El tratamiento del riesgo es diferente al nivel táctico del operacional. Para el primero se ha visto un modelo de inserción de actividades que puede servir para recoger la eventualidad de un nuevo proyecto en el horizonte de planificación táctica. Otra posibilidad es volver a programar con los nuevos datos. Como la aparición de nuevos proyectos no es un evento corriente, no existe ninguna razón computacional para no emplear incluso ambos métodos.

7. En cualquier caso, al nivel operacional un cambio en la planificación táctica, lo mismo que un error u otro tipo de disrupción interno a un proyecto, tiene una consecuencia inmediata: la posibilidad de que algún plazo no se cumpla. La respuesta a esta situación es una aceleración de las actividades críticas mediante la concentración de recursos en las mismas. Esta reasignación se puede resolver aplicando un algoritmo del tipo *squeaky wheel* que modifica las prioridades.

8. La capacidad de representar los aspectos continuos y discretos de los proyectos *software* se consigue mediante una metodología de simulación híbrida que gestione simultáneamente la integración numérica de sistemas dinámicos continuos con las interrupciones provocadas por los eventos. A pesar de que el único caso detectado en la literatura específica es la metodología DEVS que señala una vía basada en incrementos discretos de tiempo lo suficientemente pequeños, existen entornos de simulación convencionales que mejoran la capacidad computacional de esta propuesta intervalos de tiempo variables que posibilitan "acelerar" la integración numérica en las regiones donde la linealización es una aproximación suficiente y aumentan la discriminación en el entorno de los eventos.

9. Para representar y medir los efectos de los procedimientos de mejora de procesos el modelo seguirá la recomendación expuesta en el apartado 3.2.2 que aconseja modelar explícitamente el esfuerzo dedicado a las tareas de soporte y control.

El cuadro 5.5 resume las perspectivas, conceptos y modelos que se han tomado de los trabajos previos expuestos y que se han utilizado a modo de "especificaciones" para la construcción del modelo que se describe en el capítulo siguiente.

Aspecto	Concepto/modelo	Referencias
Carácter multiproyecto	Modelos jerárquicos encapsulados	[PMB99]
Separación entre procesos y decisiones	Modelado modular con tres ámbitos: trabajo, decisión y medida	[PMB99] [CBK06]
Asignación de recursos dentro de actividades	Proporcional al trabajo discretizado: RWK	[RRT01] [AM91]
Asignación de recursos dentro de proyectos	MRCPSP/GEN MRCPSP/GEN-EXP	[LTB06] [JP05]
Asignación de recursos entre proyectos	Planificación táctica restringida por los recursos	[GS05]
Tratamiento del riesgo Táctico: Operacional:	 Programación robusta Recuperación de secuencias *"activity crashing"* *"squeaqy wheel optimization"*	 [LN94] [AR00] [DH02] [JC99]
Simulación híbrida	Ecuaciones Diferenciales Algebraicas Híbridas	[Fri04]
Medida y mejora	modelado explícito de actividades no constructivas	[PR02] [Sta00]

Cuadro 5.5: Aportaciones de trabajos previos incorporadas.

Capítulo 6

El modelo SimHiProS

6.1. Arquitectura del modelo básico

El modelo que se ha construido, denominado **SimHiProS** (**Si**mulador **Hí**brido de la **Pro**ducción de **S**oftware), simula la jerarquía que caracteriza al problema de la producción de *software* en un entorno multiproyecto a partir de los procesos elementales de trabajo.

Para implementar esta jerarquía se diferencian tres módulos, *paquete*, *proyecto* y*multiproyecto*. A su vez cada uno de ellos tiene definidos tres componentes o submódulos que representan y simulan respectivamente la actividad de producción, las decisiones de asignación y la medida y control dentro de cada nivel. El modelo se completa con un módulo de *entorno* que simula la interacción del proceso de producción con el entorno general de la empresa y un módulo *funcional* que implementa distintas funciones y algoritmos y que es llamado desde los anteriores.

6.1.1. Los módulos

Como se ha dicho, el modelo se organiza jerárquicamente en tres módulos:

Paquete. Es el nivel más bajo al que puede definir un conjunto de actividades medibles y controlables más allá del trabajador individual.

Proyecto. Es el nivel operacional, el conjunto de actividades orientadas

a obtener un producto *software*. Tiene la la posibilidad de describir, además de las fases tradicionales del ciclo de vida en cascada, una desagregación más fina así como la superposición de fases característica de los modelos de desarrollo incremental o evolutivo.

Multiproyecto. Es el nivel táctico, representa el conjunto de todas los proyectos que se realizan a la vez por la organización.

En el nivel más bajo de la jerarquía se sitúa un *paquete de trabajo*, un conjunto continuo de *tareas* homogéneas, que se inicia y termina en función de los recursos (cambiantes) que el paquete tiene asignados y de las condiciones que impone el entorno. La unidad característica es la *tarea*, que se mide en *unidades de tamaño*, (KLOC, puntos función, puntos función objeto, etc.).

A cada paquete se corresponde con un *entregable* definido dentro del ciclo de desarrollo del proyecto *software*. La ejecución de las tareas contenidas en un paquete no esta condicionada externamente más que por la disponibilidad de recursos, siempre que los paquetes anteriores (los artefactos previos necesarios) hayan sido completados. Puede ser, por tanto, una fase tradicional en el modelo de desarrollo en cascada, una iteración en un modelo evolutivo o, como se produce en la práctica en el caso de aplicación, un conjunto mixto de actividades de diseño y construcción que proporcionan un entregable interno o externo con una funcionalidad determinada dentro del proyecto.

El *proyecto* es una combinación de paquetes de trabajo organizada de acuerdo con las *reglas de precedencia lógica y temporal* que definen el ciclo de vida de los artefactos que lo componen y que conduce a la obtención de un producto *software*. Las combinaciones de paquetes permiten representar diferentes modelos de proceso: desde el proceso secuencial en cascada hasta modelos de desarrollo incremental, así como actividades de soporte al proyecto como la gestión de configuración, la documentación, ...

El nivel *multiproyecto* está conformado por todos los proyectos que desarrolla la organización en un determinado instante del tiempo. Los proyectos no tienen ninguna relación entre si aparte de su coincidencia temporal y su dependencia del mismo conjunto de recursos.

6.1.2. Los componentes

Cada módulo está organizado en los siguientes componentes o submódulos que se corresponden con diferentes aspectos o o perspectivas, del proceso *software* definidos a cada nivel jerárquico:

Actividad. Simula los aspectos (continuos) del proceso de trabajo, es decir la construcción, revisión y corrección de los artefactos *software* así como la carga de trabajo ocasionada por las actividades *no constructivas*.

Asignación. Simula las decisiones (discretas) de asignación y reasiganción de recursos a las diferentes actividades desde los diferentes niveles jeráquicos.

Medición y Control. Simula las medidas (discretas) del estado de las actividades que determinan las decisiones de asignación.

El cuadro 6.1.2 sintetiza los tres módulos con sus respectivas componentes o sub-módulos.

6.2. Nivel básico: el paquete

A este nivel el proceso de trabajo se modela partiendo de una variable continua, la *tarea*, que se supone que corresponde a unidades de trabajo homogéneas que deben completarse para obtener un determinado artefacto. El número total de tareas a realizar caracteriza a dicho artefacto desde el punto de vista de la producción (con independencia de su funcionalidad en un sistema, o proyecto) y, por tanto, al paquete.

Los tres componentes o submódulos son los siguientes:

Actividad. Representa el trabajo continuo sobre las tareas.

Asignación. Representa el reparto de recursos a las diferentes fases del submódulo *actividad*.

Medida. Representa la obtención de métricas del avance del trabajo y del consumo de recursos para decidir sobre la asignación e informar al nivel jerárquicamente superior en relación con la planificación.

Módulo	**Actividad**	**Medida**	**Asignación**
Paquete	Tareas elementales	Métricas de: -Trabajo (tareas) -Coste (u.m.) -Tiempo (días)	día-persona a fases
Proyecto	Paquetes	Desviaciones Métricas acumuladas Desviaciones	 día-persona a paquetes
Multiproyecto	Proyectos	Métricas acumuladas Desviaciones	día-persona a proyectos

Cuadro 6.1: Componentes básicos del modelo **SimHiProS**.

6.2.1. Actividad

El módulo **paq.actividad** representa el proceso de trabajo elemental en el mdelo. Tiene una dinámica interna básicamente *continua* y está inspirado en un modelo clásico de Dinámica de Sistemas, el presentado en [NS98a] por Ford y Sterman, dedicado al modelado genérico de procesos de desarrollo.

La lógica es muy similar al modelo de AbdelHamid y Madnick [AM91] y toda la saga subsiguiente. Las tareas pasan por sucesivas *fases* hasta que son terminadas. En el modelo clásico arriba citado, dichas fases son *construcción*, *verificación* y, eventualmente, *iteración* si en la verificación se detecta que deben corregirse, después de lo cual vuelven a verificación. Las tareas que superan la fase de verificación se acumulan como tareas *terminadas*.

En la formulación de Ford y Sterman esto se representa como una sucesión de *niveles*[1] encadenados que se alimentan unos a otros a unos ritmos determinados por los recursos empleados. Cada nivel representa una *cartera* (backlog) *de tareas pendientes* de pasar a la siguiente fase. La transición de una a otra fase está representada por el *flujo*[2] que "vacía" el nivel previo y "llena" el siguiente, salvo el último nivel que representa tareas ya concluidas y por tanto, no tiene flujo de salida. Del mismo modo, el primer nivel en la secuencia no tiene flujo de entrada sino que toma como valor inicial el total de tareas a procesar. Los flujos, a su vez, vienen determinados por los recursos aplicados a cada una de las fases.

Las variantes introducidas en este caso sobre el modelo clásico son dos:

1. La primera es que se ha introducido una fase adicional, denominada *preparación* previo a la construcción; en el modelo de proceso *software* iterativo o incremental que aquí se representa, el diseño detallado de los bloques formados por tareas a construir es una fase diferenciada de la construcción de los mismos, pero sin embargo totalmente vinculada a ésta en el sentido de que no hay una separación entre el diseño y la

[1] En la terminología de Dinámica de Sistemas se denomina *nivel* a las variables de estado continuas, por analogía a un depósito que contiene un fluido cuyo nivel evoluciona dinámicamente.

[2] *Flujo* en la misma terminología y siguiendo con la analogía es un término de la variación temporal - la derivada - de un *nivel*.

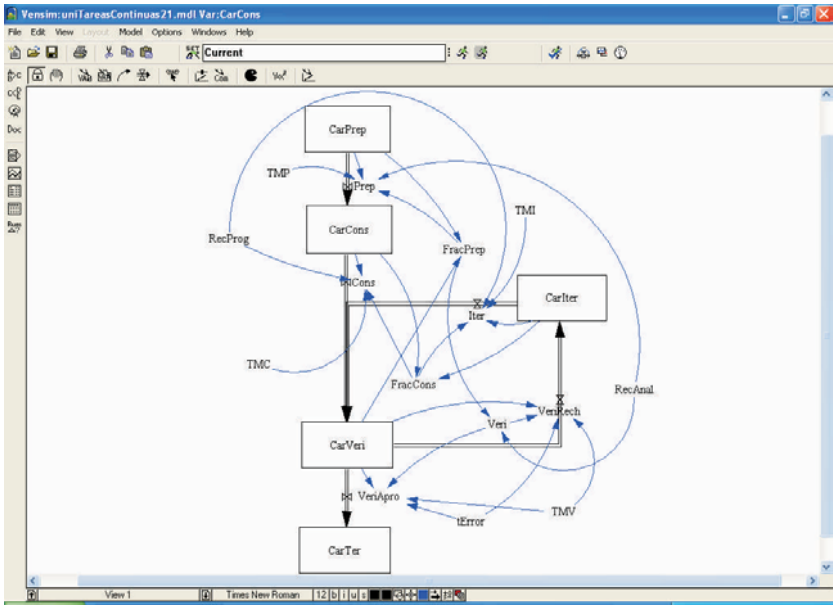


Figura 6.1: Diagrama de Forrester del modelo básico de tareas.

construcción del código que de lugar a un artefacto intermedio.

2. La segunda es que, a consecuencia de este modelo de desarrollo y a la escala de desagregación a la que se construye el modelo, las tareas preparadas pueden pasar directamente a ser construidas sin que haya que reorganizarlas en una determinada secuencia previa, representada en los modelos clásicos como una función no lineal de las tareas pendientes.

En la figura 6.1 se presenta el diagrama de Forrester (representación gráfica convencional de la Dinámica de Sistemas) del modelo implementado.

Entradas y salidas

El componente **paq.actividad** recibe entradas de **paq.asignacion**, del módulo **proyecto** al que pertenece y del **entorno**. Del primero, la asignación de recursos a cada una de las fases (preparación, construcción, verificación e iteración) así como la productividad de estos recursos y la tasa de errores en la construcción. Del proyecto al que pertenece recibe una señal de activación.

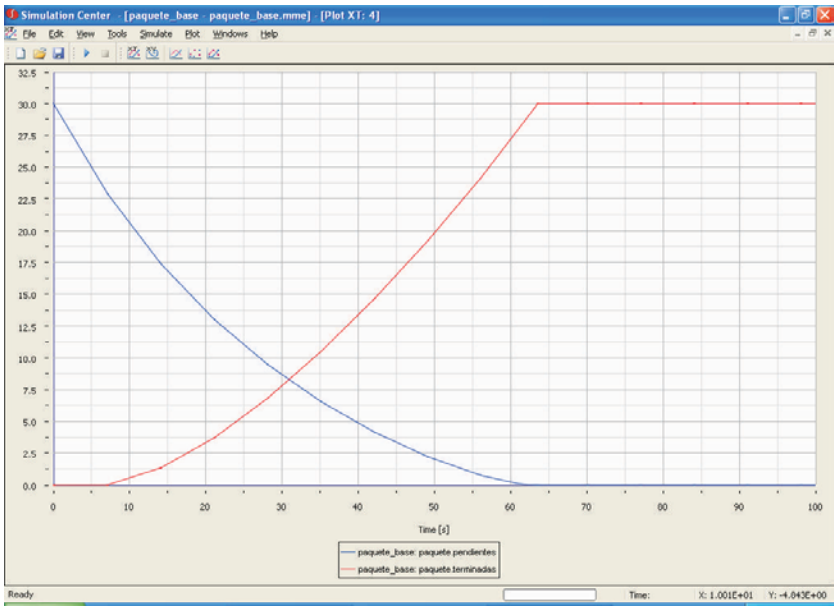


Figura 6.2: Evolución de las tareas en **paq.actividad**

Del entorno recibe los parámetros que fijan los valores iniciales de las tareas a realizar. A su vez, proporciona información sobre su estado al módulo **paq.medida** y transmite su estado al proyecto para posibilitar la activación de los paquetes que le siguen cuando haya concluido.

Funcionamiento

El módulo se activa una vez finalizadas las tareas de los módulos precedentes. Las tareas se van ejecutando de acuerdo con los recursos que les asigna **paq.asignacion**. Eventualmente se puede recibir un incremento no previsto de tareas o puede simularse la detección de un error que obliga a rehacer el trabajo como se expone en 6.7.

6.2.2. Medida

El submódulo **paq.medida** discretiza las señales continuas que recibe del componente **paq.actividad**, muestreándolas a intervalos periódicos que coinciden con el ciclo de medida. Su función principal es proporcionar datos al submódulo **paq.asignacion** así como al nivel jerárquico inmediatamente

Entradas	Tipo	Dimensión	de:	Contenido
Recursos	Discreto	2	**paq.asignacion**	Programadores y analistas
Activador	Lógico	1	**proyecto**	Dispara ejecución
Valores iniciales	Parámetro	7	**modelo**	
Productividad	Continuo	3	**proyecto**	Incluye tasa de error
Salidas	**Tipo**	**Dimensión**	**a:**	**Contenido**
Medidas	Continuo	7	**paq.medida**	Tareas pendientes por fases y coste acumulado
Estado	Lógico	2	**paq.medida** **paq.asignacion** **proyecto**	Activo y Terminado

Cuadro 6.2: Entradas y salidas de **paq.actividad**.

superior, el proyecto.

Entradas y salidas

Este componente recibe los valores continuos de todas las variables de estado del componente **paq.actividad**, en total nueve valores de los que dos son booleanos (el estado de activación y la señal de finalización) y el resto variables continuas.

Igualmente, recibe, como parámetros, los datos de la planificación (perfil previsto de realización del paquete y fechas relevantes) y el periodo de muestreado. Tiene dos salidas, ambas discretas:

- Las denominadas en el modelo *carteras (backlog) de tareas pendientes* de cada una de las fases definidas en el proceso de su producción. Estas son proporcionadas al componente **paq.asignacion** para el reparto de los recursos de acuerdo con el procedimiento que se describe más adelante.

- Los indicadores de *valor conseguido*, *coste* y *plazo* en relación con la planificación determinados de la manera descrita en el siguiente apartado.

Funcionamiento

Cómo se ha indicado, el componente discretiza el estado de la producción muestrandolo a intervalos regulares. Una parte de esa señal discreta es enviada al componente **paq.asignacion** y la otra se emplea para obtener las métricas de *valor conseguido* basadas en el método del *EVM, Earned Value Management* [Cio06].

El EVM es un método de control y seguimiento del coste y grado de avance en la programación de proyectos complejos. Se trata de un estándar puesto en marcha por el Departamento de Defensa estadounidense para el seguimiento de contratos de grandes sistemas y desarrollos. Su empleo va formando parte progresivamente de las prácticas de los clientes públicos en todas circunstancias[Cio06, DF00].

Entradas	Tipo	Dimensión	de:	Contenido
Medidas	Continuo	7	**paq.actividad**	Tareas pendientes por fases y coste acumulado
Estado	Lógico	2	**paq.actividad**	Activo y Terminado
Plan	Parámetro	3	**proyecto**	Recursos y plazos previstos
Salidas	Tipo	Dimensión	a:	Contenido
Métricas	Discreto	9	**proyecto**	Medidas muestradas de trabajo realizado, coste y plazo
Carteras	Discreto	4	**paq.asignacion**	Medidas muestradas de Tareas pendientes por fases

Cuadro 6.3: Entradas y salidas de **paq.medida**.

En este caso se ha implementado una variante basada en la propuesta de Lipke [VV07] que resulta más adecuada para el objetivo que aquí se persigue, la estimación de retrasos con vistas a reconstruir la programación de un proyecto.

A partir de los datos recogidos y por comparación con los datos de la planificación, el módulo elabora unos indicadores que activan el proceso de reasignación de recursos a nivel de proyecto fijando nuevos valores a las prioridades de los paquetes. Al nivel multiproyecto activan un aviso para la reasignación (manual).

Las métricas que se emplean se basan en la comparación entre los datos de la *planificación* y los datos obtenidos en tiempo de *ejecución*.

Los datos de *planificación* están disponibles desde el principio de la ejecución del modelo. Son los siguientes:

Valor Presupuestado Total: unidades de tamaño previstas inicialmente del trabajo completo: VPT. En nuestro caso, tareas.

Duración Presupuestada: número de intervalos discretos de tiempo que inicialmente se prevé que tardará en realizarse el trabajo completo; DT.Se cumple que $VP(DT) = VPT$.

Coste Presupuestado: coste previsto en cada intervalo t; $CP\,(t)$. Horas-persona o unidades monetarias, conocido el coste de aquellas.

Los datos medidos en *tiempo de ejecución* en el momento t son, a su vez:

Valor Conseguido: unidades de tamaño (en términos de valor presupuestado) que se han completado en el momento t; $VC(t)$.

Coste Acumulado: recursos consumidos en hasta el momento t; $CA(t)$.[3]

Para evaluar el grado de avance del proyecto se determina el parámetro denominado **Progreso Temporal Conseguido** (*earned schedule*) en t; momento

[3]En el caso del modelo que nos ocupa, al contrario que en la metodología EVM estándar, se emplea una unidad diferente para el trabajo y para el coste, que en los modelos convencionales se mide en horas-persona. La razón de ello es que aquí no se ha igualado *esfuerzo* a *tamaño* ya que no se entiende que exista una relación lineal entre ambos.

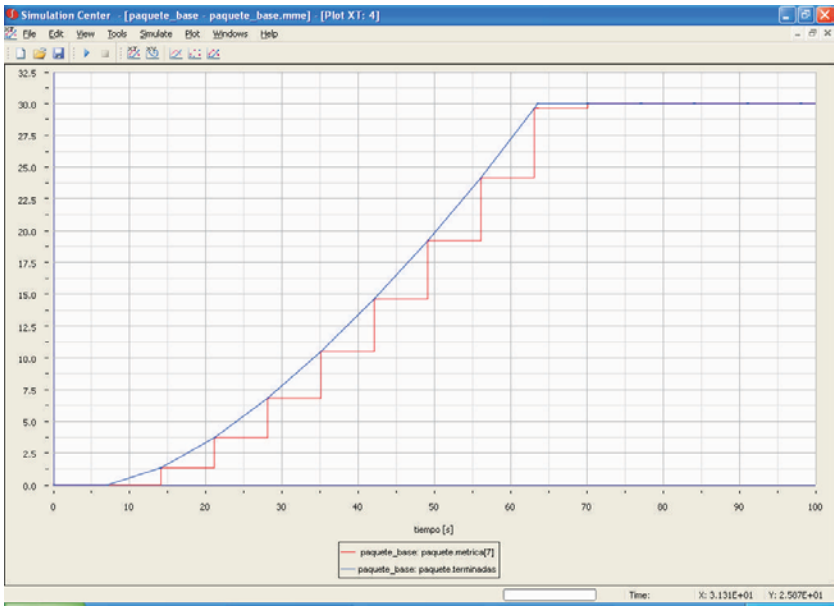


Figura 6.3: Muestreo del Valor Conseguido.

Magnitud	**Índice**	**Varianza**
Trabajo	$IC(t) = \frac{VC(t)}{VP(t)}$	$VT(t) = VC(t) - VP(t)$
Coste	$IC(t) = \frac{CA(t)}{CP(t)}$	$VC(t) = CA(t) - CP(t)$
Progreso Temporal	$IP(t) = \frac{PTC(t)}{t}$	$VP(t) = PTC(t) - t$

Cuadro 6.4: Métricas EVM de paquete.

en el que el Valor Presupuestado debía ser igual al Valor Conseguido. En la práctica como las previsiones se refieren a intervalos discretos se determina como:

$$PTC(t) = k + \frac{VC(t) - VP(k)}{VP(k+1) - VP(k)}$$

donde k es el intervalo de tiempo más tardío en el que el Valor Presupuestado es inferior al valor conseguido (*k+1* el inmediatamente siguiente.

Con este dato se obtienen las métricas denominadas en la metodología EVM *índice* y *varianza* de trabajo, coste y plazo o progreso temporal que aparecen en la tabla 6.4: Los indicadores anteriores permiten hacer previsiones sobre costes y plazos en el módulo superior.

6.2.3. Asignación

El componente **paq.asignacion** reparte los recursos asignados desde el proyecto al paquete a las distintas fases de trabajo de éste. El modelo gestiona inicialmente dos tipos de recursos: recursos de *tipo 1* equivalentes a analistas, ingenieros, etc. y recursos de *tipo 2* que equivalen a programadores y otro personal.

Entradas y salidas El componente recibe de **paq.medida** la señal muestreada de las carteras de trabajo pendientes así como la asignación de recursos de tipo 1 y tipo 2 del proyecto. A partir de esta información devuelve a **paq.actividad** el reparto de los analistas entre las fases de preparación y verificación y de los programadores entre las de construcción e iteración.

Funcionamiento

Puesto que el sistema de ejecución es multimodal, la decisión de asignar recursos a un paquete de tareas hace operativa la decisión de activar el paquete. El número de recursos asignados determinará el ritmo de ejecución y, por tanto, la duración.

Cada paquete se caracteriza por una *carga inicial* de tareas y un límite inferior y otro superior de recursos aplicables. El límite inferior representa el número mínimo de recursos (por defecto 1 de cada tipo) necesarios para ejecutar un paquete. El límite superior representa el máximo número de recursos que es razonable asignar a un paquete.

Las decisiones de asignación de personal dentro de un mismo bloque de tareas se realiza con un periodo corto y en función de la carga de trabajo pendiente en cada una de las carteras de tareas de manera que se aprovechen al máximo los recursos existentes.

Se han planteado dos variantes de asignación interna:

- Realización de las tareas secuencialmente siguiendo una *disciplina de colas* con un servidor único; es decir se van completando las diferentes fases con todos los recursos disponibles y estos no se dedican a otra

Entradas	Tipo	Dimensión	de:	Contenido
Carteras	Discreto	4	**paq.medida**	Medidas muestradas de Tareas pendientes por fases
Asignados	Discreto	2	**proyecto**	Recursos asignados
Estado	Lógico	2	**paq.actividad**	Activo y Terminado
Salidas	Tipo	Dimensión	a:	Contenido
Recursos	Discreto	2	**paq.actividad**	Programadores y analistas
Productividad	Continuo	3	**paq.actividad**	incluye tasa de error

Cuadro 6.5: Entradas y salidas de **paq.asignacion**.

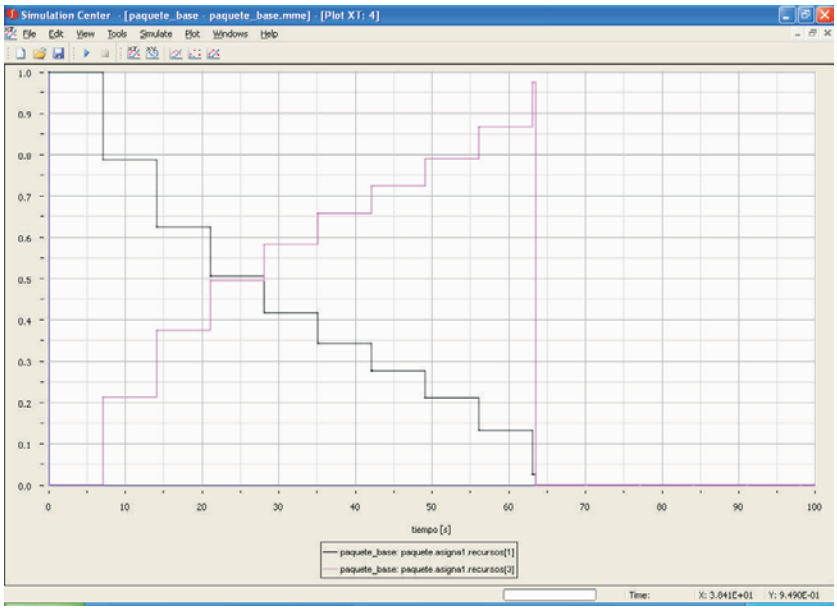


Figura 6.4: Asignación proporcional de recursos.

fase hasta la finalización de la anterior. En cierto modo, esta variante de asignación es semejante a la regla MINSLK de la tabla 2.3.

- Realización de las tareas asignando *dinámicamente* el personal en *proporción a las carteras pendientes*; por defecto esta en la implementación adoptada, que equivale á la aplicación dinámica de la regla RWKde la misma tabla.

En cualquiera de las dos variantes, la asignación no es continua sino discreta, por ciclos de asignación que por defecto duran un día.

6.2.4. La conexión del paquete con el resto de módulos

A partir de la descripción anterior se desprende que un paquete intercambia con el resto de módulos información a través de las vías siguientes:

Entradas

El paquete recibe:

1. Los recursos asignados por el módulo *proyecto*.

2. La señal booleana de activación del mismo módulo.

3. Las condiciones de productividad, tasa de errores, etc.[4]

4. Las previsiones de la planificación, definidas igualmente a nivel de proyecto.

5. Los valores iniciales, parámetros, etc. definidas a nivel global.

Salidas

El paquete, a su vez, envía las señales siguientes al módulo *proyecto* de quien depende:

1. La señal booleana de "terminado" que posibilita activar a los paquetes que le siguen.

2. La información sobre desviaciones en trabajo, costes y plazos.

En síntesis, la operativa del módulo *paquete* es la siguiente:

- Una vez recibida la señal de activación, **paq.actividad** simula la ejecución de las tareas al ritmo que determinan los recursos asignados por **paq.asignacion** hasta que las concluye emitiendo la señal correspondiente.

- **paq.medida** muestrea tanto el trabajo pendiente como el terminado y el coste acumulado. Compara los valores muestreados con la planificación y envía la información resultante al módulo **proyecto**. Envía igualmente los valores muestreados del trabajo pendiente al componente **paq.asignacion**.

- **paq.asignacion** reparte los recursos disponibles que el proporciona el módulo *proyecto* entre las diferentes fases del trabajo y suministra a **paq.actividad** dicha información.

[4]En esta versión del modelo se han fijado estas variables en cantidades constantes. No obstante se han definido como entradas variables para poder incorporar en su día los fenómenos, ampliamente estudiados en otros modelos de simulación continua, de "presión del plazo", "efectos aprendizaje", la "ley" de Brooks, etc. La razón para ello es aislar los efectos de la asignación dinámicade recursos entre proyectos del resto de efectos.

Entradas	Tipo	Dimensión	de:	Contenido
Asignados	Discreto	2	**proyecto**	Recursos asignados
Activador	Lógico	1	**proyecto**	Dispara ejecución
Productividad	Continuo	3	**proyecto**	Incluye tasa de error
Plan	Parámetro	3	**proyecto**	Recursos y plazos previstos
Salidas	Tipo	Dimensión	a:	Contenido
Métricas	Discreto	9	**proyecto**	Medidas muestradas de trabajo realizado, coste y plazo
Estado	Lógico	2	**proyecto**	Activo y Terminado

Cuadro 6.6: Entradas y salidas del módulo **paquete**.

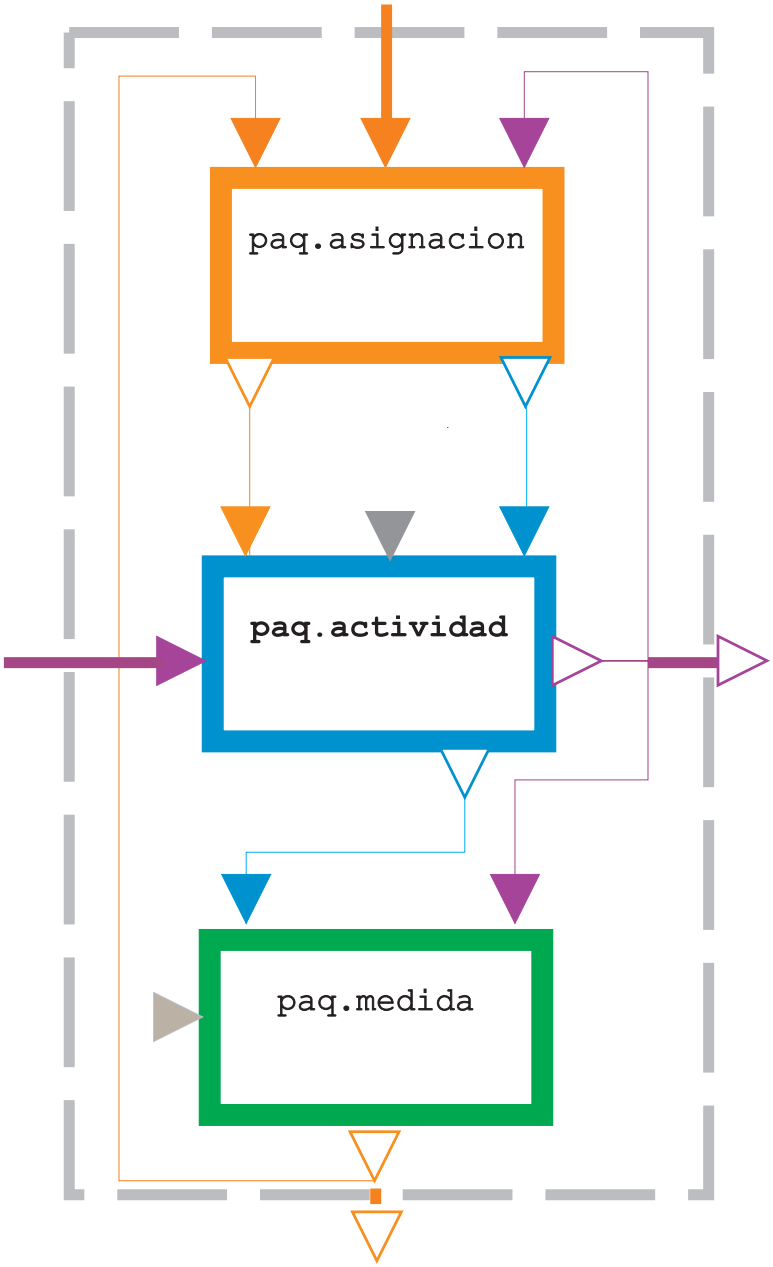


Figura 6.5: Módulo **paquete**.

6.3. Nivel operacional: el proyecto

El proyecto es un conjunto de paquetes ordenados lógica y temporalmente con el fin de obtener un producto *software*. Este conjunto está inicialmente dotado de una *planificación operacional* derivada a su vez de la asignación de recursos que proviene de la *planificación táctica*. Al igual que el módulo anterior posee tres componentes, denominados aquí **pro.actividad**, **pro.medida** y **pro.asignacion**.

6.3.1. Actividad.

En el módulo *proyecto*, la componente que simula el trabajo tiene la configuración de una red de tareas ordenadas mediante las relaciones de precedencia características de una red de proyectos. Para ello las tareas que componen los proyectos se conectan a unos nodos booleanos cuya lógica interna gobierna refleja las relaciones de precedencia existentes entre las tareas.

Esta componente es la única que realiza alguna interacción con los módulos del nivel jerárquico inferior, los paquetes.

Si bien se puede con facilidad representar unas *relaciones generalizadas de precedencia* (ver 2.4), se ha optado por la implementación más sencilla, la de las relaciones *principio-final*. Es decir, una tarea, para activarse requiere que se hayan completado todas las precedentes. Mediante las utilidades que se han implementado en el módulo *funcional* es posible forzar un retraso entre las actividades precedentes y cualquiera de las siguientes.

La evolución del trabajo en un proyecto es, por tanto, la de los paquetes que lo componen. Pero además se ha implementado en la componente **pro.actividad** el registro del trabajo acumulado en la propia actividad de medida y asignación como una primera aproximación a la medida del esfuerzo en *tareas no constructivas* [MV06]. Este indicador, aunque sea rudimentario, permite una estimación de la sobrecarga de trabajo relacionada no sólo con la medida sino con otras actividades como documentación, control de calidad, actividades de orientación y soporte y otras que guarden proporción con la duración (tamaño de los paquetes), complejidad (número de paquetes) y las incidencias en el desarrollo (desviaciones producidas en los paquetes).

El registro de este esfuerzo no es más que un *contador* conectado al reloj del ciclo de medida que acumula el número de veces que éste se activa por el número de paquetes activos además de las veces que se lanza la reasignación de recursos. Estimando un coste en horas-recurso para cada una de las unidades acumuladas se tiene una aproximación al coste de estas actividades no constructivas.

Entradas y salidas

El componente **pro.actividad** recibe la asignación de recursos a los paquetes que conforman el proyecto de **pro.asignacion**.

De los paquetes recibe las señales indicativas del estado de ejecución de cada uno de ellos. Estas son las métricas de cada uno de ellos y de su estado de activación, tal y como se han descrito en el apartado anterior. El propio ciclo de recepción de estas señales discretas se emplea para el cálculo del trabajo no constructivo.

A su vez envía a cada paquete los recursos asignados por el componente **pro.asignacion** y al componente **pro.medida** el estado de cada uno de los paquetes y el registro del esfuerzo en actividades no constructivas.

Funcionamiento

El funcionamiento de este componente es relativamente trivial, limitándose a transmitir la información que recibe. La única elaboración que hace es la acumulación del esfuerzo de medida.

6.3.2. Medida

El componente **pro.medida** elabora los datos agregados del proyecto a partir de la información que viene de los paquetes y realiza estimaciones sobre esos datos, en comparación con los datos de la planificación.

Entradas	Tipo	Dimensión	de:	Contenido
Asignados	Discreto	2n	**pro.asignacion**	Recursos asignados
Estado	Lógico	2n	**paquete**	Activo y Terminado
Métricas	Discreto	9n	**paquete**	Medidas muestradas de trabajo realizado, coste y plazo
Salidas	Tipo	Dimensión	a:	Contenido
Asignados	Discreto	2n	**paquete**	Recursos asignados
Activador	Lógico	n	**paquete**	Dispara ejecución
Productividad	Continuo	3	**paquete**	Incluye tasa de error
Plan	Parámetro	3	**paquete**	Recursos y plazos previstos
Métricas	Discreto	9n	**pro.medida**	Medidas muestradas de trabajo realizado, coste y plazo
Esfuerzo	Discreto	n	**pro.medida**	Esfuerzo en medida y control

Cuadro 6.7: Entradas y salidas de **pro.actividad**.

Entradas y salidas

La información que recibe este componente es únicamente la señal discreta de las medidas del estado de ejecución de los paquetes y su desviación, que le envía **pro.actividad**. Por su parte, envía a **pro.asignacion** esta información elaborada para su empleo en los procedimientos de reasignación de recursos.

Adicionalmente envía al nivel superior (multiproyecto) los datos agregados de la ejecución de todos sus paquetes y del esfuerzoen actividades no constructivas.

Funcionamiento

El componente **pro.medida** elabora dos tipos de información:

- Las métricas acumuladas de Valor Conseguido, Coste Acumulado y Progreso Total, a partir de la comparación entre la planificación y los datos medidos.
- Las estimaciones de coste y plazo para cada una de las tareas y para el proyecto en su conjunto que se registran y se emplean para la asignación de recursos o, eventualmente, para recurrir a recursos suplementarios al nivel superior.

Las estimaciones citadas, basadas igualmente en la metodología EVM, son:

Previsión de costes Se plantean tres previsiones de costes diferentes, según suponga que la productividad y la eficacia se recupera o no. Así hay tres posibilidades:

1. Coste Estimado para Terminar con eficiacia y productividad normales:

$$CET_1(t) = CA(t) + (VPT - VP(t)) \times \frac{CP}{VPT}$$

2. Coste Estimando para terminar con la eficacia alcanzada hasta el momento:

$$CET_2(t) = CA(t) + \frac{VPT - VP(t)}{IT(t)} \times \frac{CP}{VPT}$$

Entradas	Tipo	Dimensión	de:	Contenido
Métricas	Discreto	9n	**pro.actividad**	Medidas muestradas de trabajo realizado, coste y plazo
Esfuerzo	Discreto	n	**pro.actividad**	Esfuerzo en medida y control
Salidas	Tipo	Dimensión	a:	Contenido
Desviaciones	Discreto	3n	**pro.asignacion**	Desviaciones en trabajo, coste y plazo de paquetes
MétricasAc	Discreto	9	**multiproyecto pro.asignacion**	Acumulado del proyecto
EsfuerzoAc	Discreto	1	**multiproyecto**	Esfuerzo en medida y control acumulado

Cuadro 6.8: Entradas y salidas de **pro.medida**.

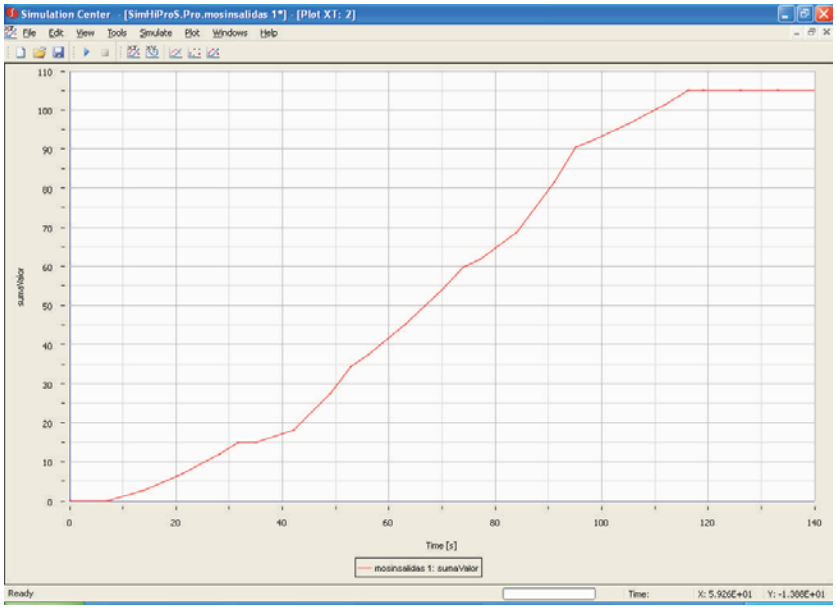


Figura 6.6: Valor Conseguido acumulado.

3. Coste Estimado para terminar con la eficacia y productividad alcanzadas hasta el momento:

$$CET_3(t) = CA(t) + \frac{VPT - VP(t)}{IC(t) \times IP(t)} \times \frac{CP}{VPT}$$

Previsión de plazos Al igual que el caso anterior hay tres previsiones de plazos de terminación:

1. Plazo Estimado de terminación con eficacia y productividad normales:

$$PET_1(t) = t + (DT - PTC(t))$$

2. Plazo Estimado de terminación con la eficacia alcanzada hasta el momento:

$$PET_2(t) = t + \frac{DT - PTC(t)}{IT(t)}$$

3. Plazo Estimado de terminación con la eficacia y productividad alcan-

zadas hasta el momento:

$$PET_3(t) = t + \frac{DT - PTC(t)}{IT(t) \times IP(t)}$$

6.3.3. Asignación

El componente **pro.asignación** gestiona una tabla dinámica que es básicamente una matriz de dos columnas, una por recurso, y tantas filas como paquetes tiene el proyecto. Como el ciclo de asignación y reasignación es discreto, la dinámica de este componente es esencialmente discreta (y en la implementación actual, entera pues los recursos asignados son individuos). Esta dinámica discreta determina la llamada a los algoritmos de reasignación de recursos y a la vez se determina como consecuencia de los resultados de esas reasignaciones.

Entradas y salidas

Las entradas a **pro.asignacion** son de dos tipos:

- Los recursos globalmente asignados al proyecto desde el nivel multiproyecto.
- Los datos de ejecución de los paquetes y los globales del conjunto del proyecto que le son proporcionados por el componente **pro.medida**.

Procesando esta información, el componente obtiene la asignación de recursos que pasa como salida a los paquetes a través de **pro.actividad**.

Funcionamiento

El componente **pro.asignacion** decide sobre la asignación de recursos a los paquetes con el objetivo de hacer mínimo el tiempo de ejecución total del proyecto (al inicio) o asegurar el cumplimiento de los plazos de entrega (en tiempo de ejecución). Para ello genera inicialmente y emplea después los datos que provienen de la programación inicial de cada proyecto y, eventualmente,

Entradas	Tipo	Dimensión	de:	Contenido
AsignadosPro	Discreto	2	**multiproyecto**	Recursos totales
Desviaciones	Discreto	3n	**pro.medida**	Desviaciones en trabajo, coste y plazo de paquetes
MétricasAc	Discreto	9	**pro.medida**	Acumulado del proyecto
Salidas	Tipo	Dimensión	a:	Contenido
Asignados	Discreto	2n	**pro.actividad**	Recursos asignados

Cuadro 6.9: Entradas y salidas de **pro.asignacion**.

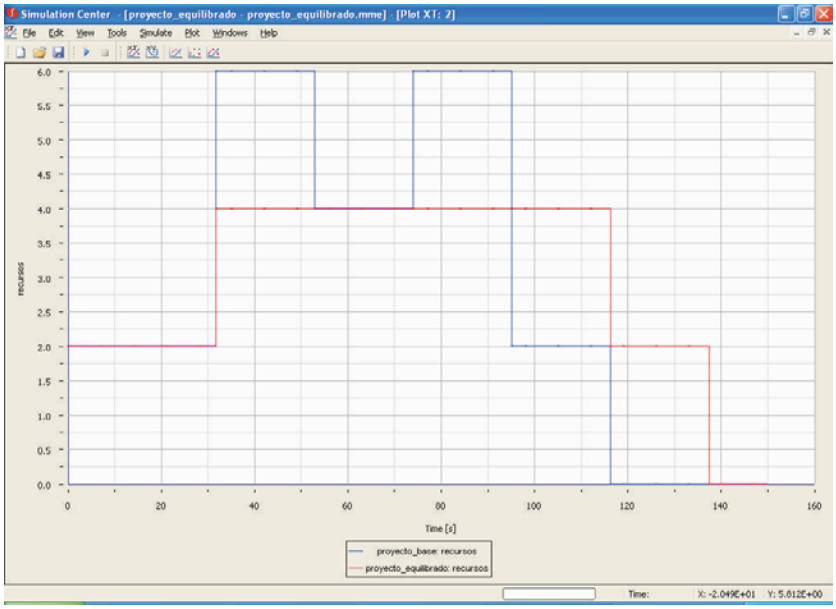


Figura 6.7: Asignación factible y no factible de recursos.

tiene que reparar la misma variando la asignación de recursos para hacer frente a un imprevisto o un error.

Como la duración del paquete depende del modo elegido, es decir, se pueden destinar más o menos recursos a la realización del mismo, es necesario contar con una primera asignación de modos de ejecución para inicializar la secuenciación.

Planificación inicial .

Para obtener esta planificación inicial se ha implementado la generación de secuencias en serie empleando la regla de prioridad RWK (*remaining work*, de modo que a cada paquete se le imputa como prioridad su carga de trabajo más toda la de sus sucesores, al modo más rápido factible, EFFT (*earliest feasible final time* [LTB06]. Una vez seleccionado el modo y secuenciados los paquetes se dispone de una solución factible. Esta solución se mejora posteriormente aplicando el mismo modelo que para la reporgramación (ver apartado siguiente) hasta obtener una planificación inicial.

Evidentemente esta solución no es necesariamente la óptima por lo que caben diferentes vías de mejora de la misma, bien a través de métodos exac-

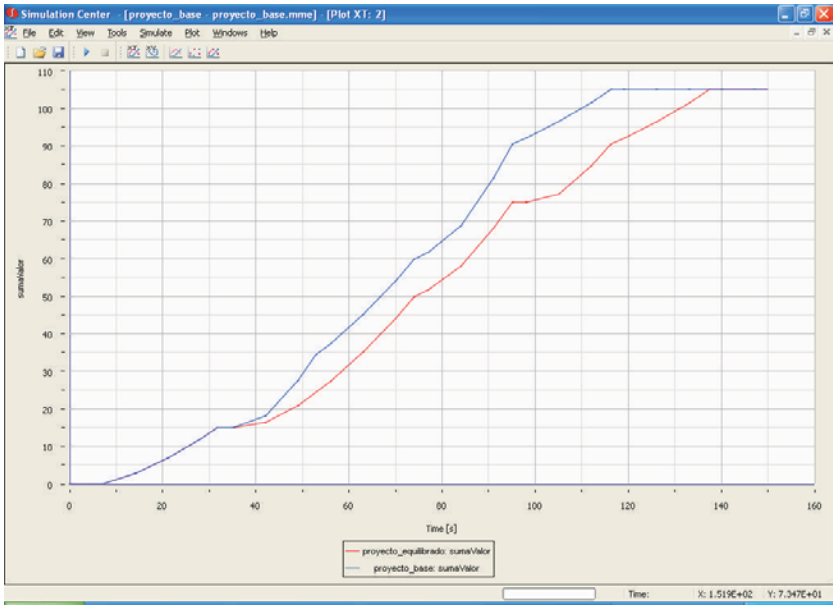


Figura 6.8: Efecto en el valor conseguido de una asignación factible y otra no factible.

tos (*branch and bound* o métodos metaheurísticos como los expuestos en el capítulo 2. De hecho, la reducida dimensión previsible de las redes de un proyecto permite considerar la primera opción. No se han implementado otros métodos más sofisticados por no considerarse relevante para el problema investigado ni prestarse con facilidad el software seleccionado a los algoritmos correspondientes[5].

Así pues, cada proyecto se inicializa con una carga de trabajo prevista, una dotación de recursos a los paquetes que lo componen y una secuencia prevista de activación de paquetes a través de esa dotación de recursos asignados.

Como las prioridades se basan inicialmente en el trabajo pendiente (*remaining work*), la evolución temporal no modifica el orden de secuenciación entre actividades mientras no se produzca ningún evento que varíe la carga de trabajo pendiente de un paquete respecto al valor previsto en cada momento por lo que no hay necesidad de reconstruir dinámicamente la programación y asignación salvo que se produzca una disrupción. Esto vale tanto para la

[5]Para hacer esto de forma eficiente, debe programarse la llamada a funciones externas en C o FORTRAN, lenguajes con mayor potencia algorítmica que Modelica

regla RWK como para cualquiera otra que dependa de la duración (LST, LSTLFT, etc. ver la tabla 2.3).

Reparación de la programación .

El problema real que se intenta representar, con independencia de la idoneidad de la programación inicial, es el de la reacción a un retraso excesivo sobre aquella, a consecuencia de un imprevisto, un error en un paquete o la asignación de personal a otro proyecto. En tal caso lo que se persigue es reparar una secuencia reasignando recursos dentro de un mismo proyecto si es posible. De lo contrario, detectar la imposibilidad de hacerlo para emitir una petición de recursos adicionales al nivel táctico.

La solución propuesta recuerda a la metodología de *agentes* en el sentido de que se simula una solución circunscrita a un proyecto, cuyo asignador actúa como un agente autónomo. Para ello, y teniendo en cuenta que la respuesta clásica de un Jefe de Proyecto a una eventualidad de este tipo es la de reasignar personal de unas tareas a otras para atender a la más crítica, se ha implementado un algoritmo del tipo SWO (*squeaky wheel optimizaton*)[JC99].

El algoritmo debe su nombre a la metáfora que lo inspira: ´´la rueda que chirría, es engrasada"[6].La filosofía es, con las lógicas diferencias, análoga a la actuación habitual del agente humano; modificar las prioridades que dieron lugar a la primera asignación para asignar más recursos al paquete que se ha vuelto crítico. Puesto que el trabajo pendiente evoluciona dinámicamente, cuando en un paquete que está en ejecución se da un evento del tipo señalado, su trabajo pendiente no decrece al tiempo que el de los demás paquetes activos o incluso aumenta[7]. Si en ese momento se llama al algoritmo de secuenciación ésta se hará con las prioridades cambiadas (véase F.8).

Si el algoritmo no es capaz de secuenciar los paquetes en el horizonte previsto, lo incrementa, lo cual ocasiona una señal indicadora que se entrega al módulo multiproyecto.

[6] "The squeaky wheel gets the grease".

[7] Lo mismo ocurriría si en lugar de una regla de prioridad RWK se emplea otra de las dependientes de la duración, ya citadas.

6.3.4. La conexión del proyecto con el nivel multiproyecto

El proyecto se comunica con el nivel multiproyecto a través de los recursos que recibe de este y transmitiendo las métricas del proyecto calculadas por el componente **pro.medida**.

6.4. El nivel táctico: multiproyecto

El nivel táctico está, al igual que los previos, organizado en tres componentes:

- **mul.actividad**
- **mul.medida**
- **mul.asignacion**

6.4.1. Actividad

En la implementación que se ha hecho, este componente se limita a englobar a todos los proyectos, recibir de estos la información sobre su estado y enviarles la asignación de recursos a cada uno de ellos.

6.4.2. Medida

El componente **mul.medida** acumula información que recibe del componente **mul.paquete** y la pasa al componente de asignación.

6.4.3. Asignacion

Este componente asigna los recursos a los proyectos y reacciona a las desviaciones en los mismos. La información que recibe proviene de **mul.medida** y de **entorno**.

En la implementación actual, el paquete **entorno** le transmite la información de los recursos disponibles a lo largo del horizonte de planificación.

Entradas	Tipo	Dimensión	de:	Contenido
AsignadosPro	Discreto	2	**multiproyecto**	Recursos totales
Salidas	Tipo	Dimensión	a:	Contenido
MétricasAc	Discreto	9	**multiproyecto**	Acumulado del proyecto
EsfuerzoAc	Discreto	1	**multiproyecto**	Esfuerzo en medida y control acumulado

Cuadro 6.10: Entradas y salidas del módulo **proyecto**.

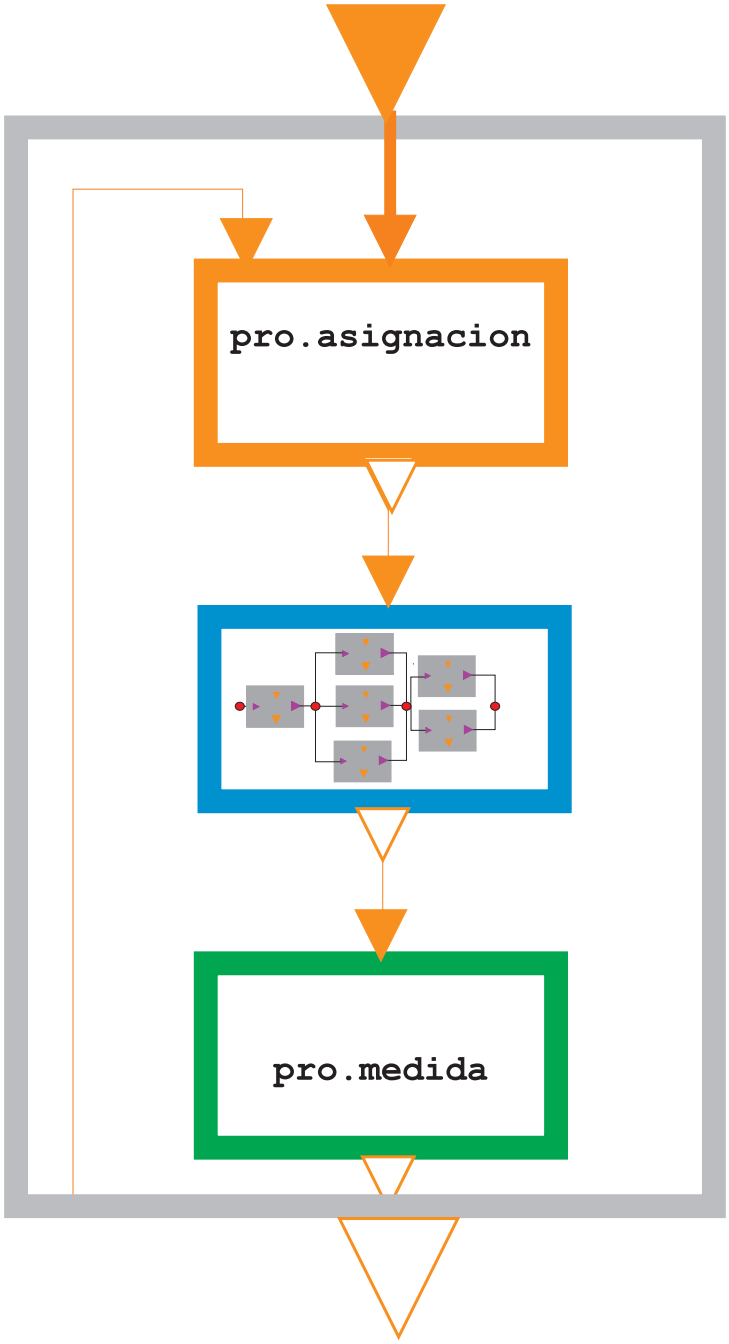


Figura 6.9: Módulo **proyecto**.

Con estos recursos disponibles se genera la planificación inicial de recursos por proyectos que **mul.actividad** envía a estos.

La generación de la asignación inicial se hace por medio de los procedimientos descritos en A.1, en particular generando una asignación factible con los hitos de los proyectos presentes resolviendo un problema de programación lineal. El algoritmo de mejora de la solución descrito en ese mismo apartado no se ha implementado por las dificultades que ofrece el lenguaje de simulación para ello. Sin embargo, la lógica del modelo permite incorporar los resultados de la aplicación del mismo. Lo mismo ocurre con la reacción a las desviaciones producidas que se puede implementar como una nueva llamada al algoritmo de asignación a partir de la detección de un retraso en la ejecución de los proyectos o con la ejecución del algoritmo de inserción descrito en el apartado 5.4.1.

6.5. Módulo de entorno

El módulo de **entorno** engloba los siguientes componentes:

- El personal (los recursos) y su dinamica
- Los impactos exógenos
- Las decisiones tácticas

El personal se trata como una variable real; un espacio de estados con tres posibilidades; operativo, en formación o de baja. La entrada y salida de personal en el sistema viene determinada por las decisiones externas de contratación fijadas en la planificación táctica, en el primer caso, y por una dinámica demográfica exponencial, en el segundo. El personal de baja temporal se modela de manera semejante a la baja definitiva pero con un flujo de retorno. Estos flujos demográficos se modelan a partir de sencillas constantes de tiempo. De manera similar se modela la posibilidad de promoción del grupo de programadores al grupo de analistas. El esquema básico de esta dinámica demográfica se recoge en la figura 6.11 en forma de diagrama de Forrester.

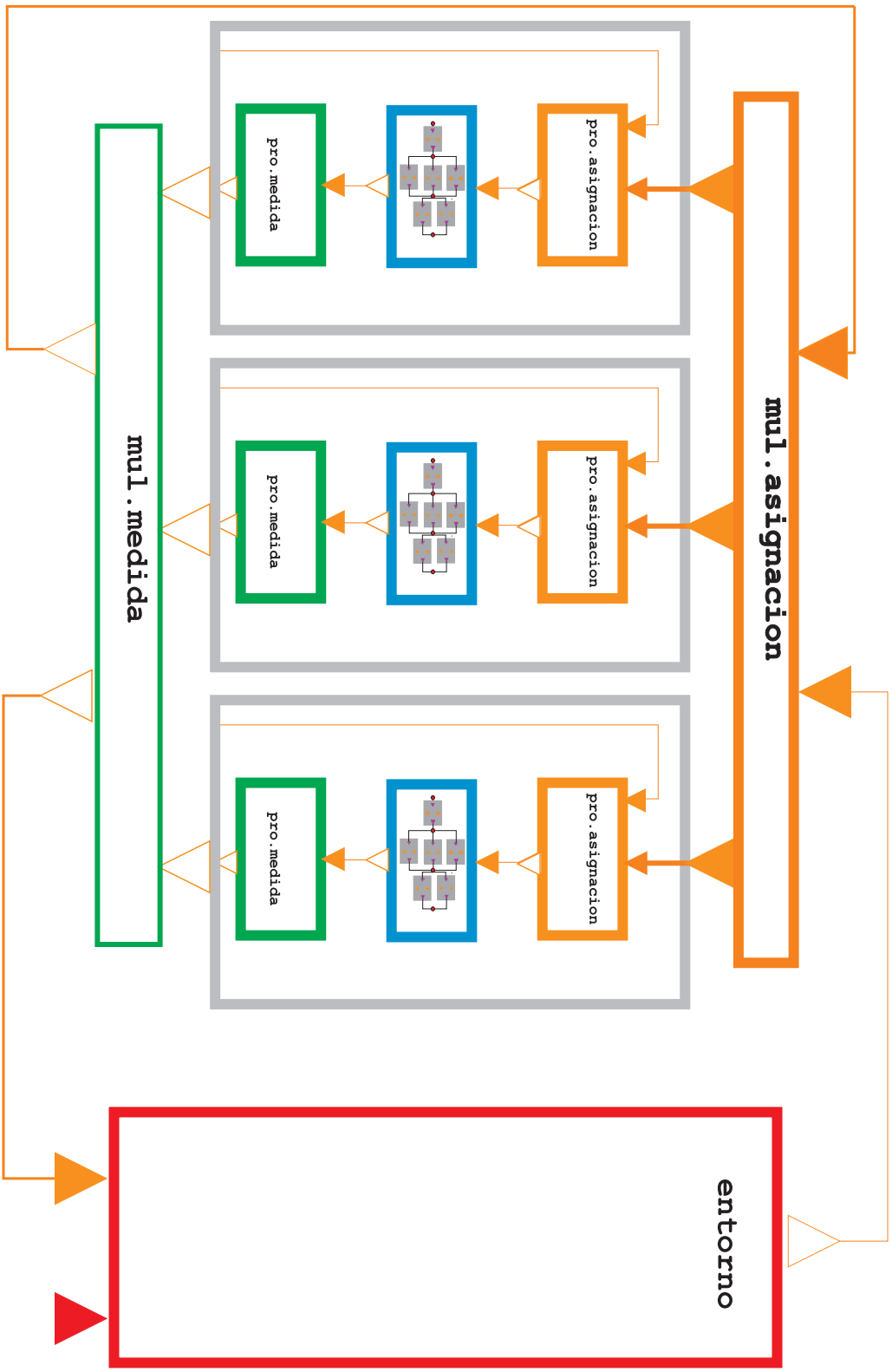


Figura 6.10: Relación multiproyecto-entorno.

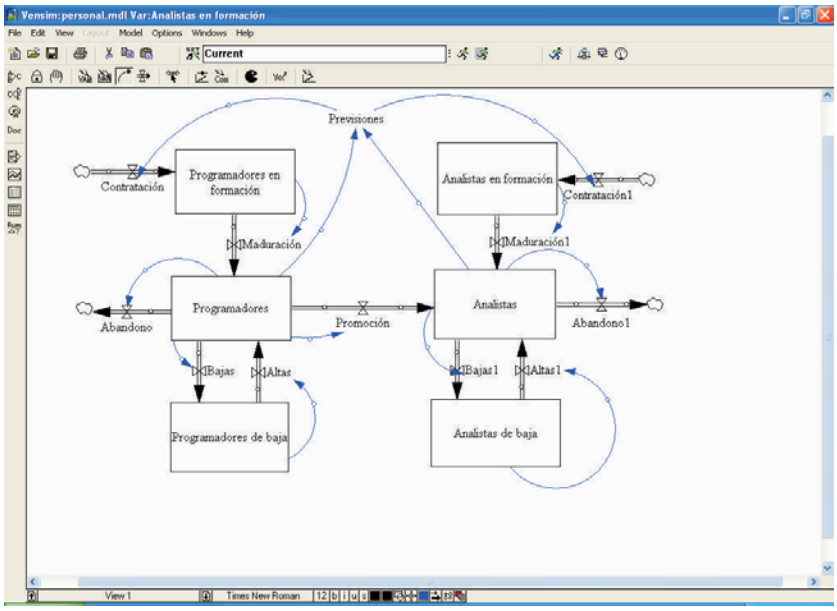


Figura 6.11: Diagrama de Forrester de la dinámica del personal

El personal operativo puede encontrarse asignado a proyectos, a otras tareas u ocioso. Dentro de la dedicación a otras tareas se puede recoger explícitamente el personal que se destina a tareas de infraestructura, gestión de configuración, mantenimiento dinámico de repositorios de componentes reutilizables, etc.

De esta forma se recoge el efecto de las diferentes estrategias de mejora de procesos sobre los recursos de la organización. Tanto estas decisiones como las de reasignación de personal entre proyectos se modelan como procesos exógenos.

Las decisiones de contratación se implementan a partir de la planificación táctica multiproyecto, por diferencias entre la desviación entre el personal disponible y el personal previsto por la planificación.

6.6. La operativa del modelo

6.6.1. Inicialización

El modelo se inicializa con los componentes siguientes:

- Los proyectos, con su descomposición en paquetes de tareas, la carga de trabajo inicial, las relaciones de precedencia y las fechas límite.
- Las asignaciones iniciales o planificación inicial que conlleva el número de recursos asignado a cada paquete de tareas en cada intervalo de tiempo.
- La previsión de incorporación del personal.
- Los eventos exógenamente decididos que deben determinarse a la inicialización aunque su aparición se producirá en el momento programado, siendo "desconocidos" por los decisores que actual de forma reactiva.
- Todos los parámetros de la simulación.

6.6.2. Simulación

El lenguaje de simulación seleccionado es un lenguaje declarativo y requiere que todo el modelo, sus ecuaciones y parámetros se mantengan inalterados en el tiempo de simulación. Esta circunstancia que permite aplicar métodos avanzados para las solución de sistemas algebraícos diferenciales (DAE) y posibilita la simulación híbrida, plantea problemas sin embargo para implementar procedimientos algorítmicos para la asignación recurrente de valores a las variables del modelo. Si bien es posible teóricamente expresar cualquier procedimiento algorítmico con una sintaxis declarativa, en la práctica presenta serias dificultades.

Esto hace que el modelo construido sea relativamente rígido y que la simulación esté predeterminada desde el principio, sin que en tiempo de ejecución se pueda alterar la estructura del modelo. Sin embargo, presenta como ventaja la posibilidad de implementar la simulación híbrida sin tener que manipular expresamente la integración numérica.

A efectos de la simulación continua, el paquete permite la especificación del sistema en forma de un sistema DAE, que se "aplana", se resuelve y se integra por métodos de integración numérica convencionales, siendo seleccionado el intervalo de integración dinámicamente por el propio simulador.

La componente de simulación discreta está gobernada por la secuencia de eventos que interrumpe la integración numérica en el momento inmediatamente anterior a un evento, ejecuta éste, reasigna a las variables de la simulación continua los valores que provienen de la ejecución del evento y retoma la integración.

Los eventos discretos son de tres tipos:

1. Eventos periódicos; se trata de los ciclos del reloj de medida y asignación.

2. Eventos no periódicos generados en la simulación; se trata de eventos provocados por el cumplimiento de alguna condición impuesta en la estructura del modelo como por ejemplo la finalización de un paquete, el cruce de algún umbral por el valor de una variable, etc.

3. Eventos no periódicos predeterminados; representan las decisiones que se toman de manera exógena así como determinadas circunstancias que se han programado para su empleo en los experimentos. Se describen en el apartado siguiente.

6.7. Eventos para la simulación de experimentos

Para probar el modelo se han diseñado una serie de experimentos con la finalidad de reproducir el comportamiento observado en el caso de estudio, al menos cualitativamente, y para verificar las soluciones propuestas por los mecanismos de reasignación de recursos y reconstrucción de secuencias.

Las características del paquete empleado obligan, como se ha dicho, a parametrizar las diferentes circunstancias en la inicialización de la simulación. Esto se ha programado en base a una serie de eventos representativos de diferentes circunstancias que afectan a los proyectos. La tabla 6.11 presenta esos eventos.

id.	Nivel	Aspecto simulado
evpaq.1	Paquete	Aparición de tareas no previstas
evpaq.2	Paquete	Detección de un error
evpaq.3	Paquete	Retirada de recursos

Cuadro 6.11: Eventos programados.

6.7.1. Aparición de tareas no previstas en un paquete

La aparición de tareas no previstas en un paquete es el modo elegido para representar un cambio en los requisitos, una de las circunstancias más habituales en los proyectos reales. Se trata de un caso muy estudiado en los modelos de simulación continua de proyectos.

Implementación

La implementación que se ha hecho se limita a reinicializar la variable de estado *carPrep* (cartera de tareas a preparar) de **paq.actividad** aumentándola instantáneamente en una cantidad constante. Se entiende que la aparición de nuevos requisitos supone una carga de trabajo no prevista que incorpora las modificaciones precisas sobre el trabajo ya realizado.

La aparición de un imprevisto de este tipo implica un cambio en el valor presupuestado que se controla desde **paq.medida** y en una revisión de la planificación. Supone también un cambio en la previsión de costes y plazos de finalización.

No se ha contemplado el modelado de este caso como la aparición de un paquete nuevo de trabajo, lo cual podría ser una alternativa. Pero entonces se trataría de una modificación dinámica de la estructura, algo que el modelo no permite.

Resultados

La figura 6.12 muestra el efecto en el trabajo previsto y el terminado de una circunstancia de este tipo.

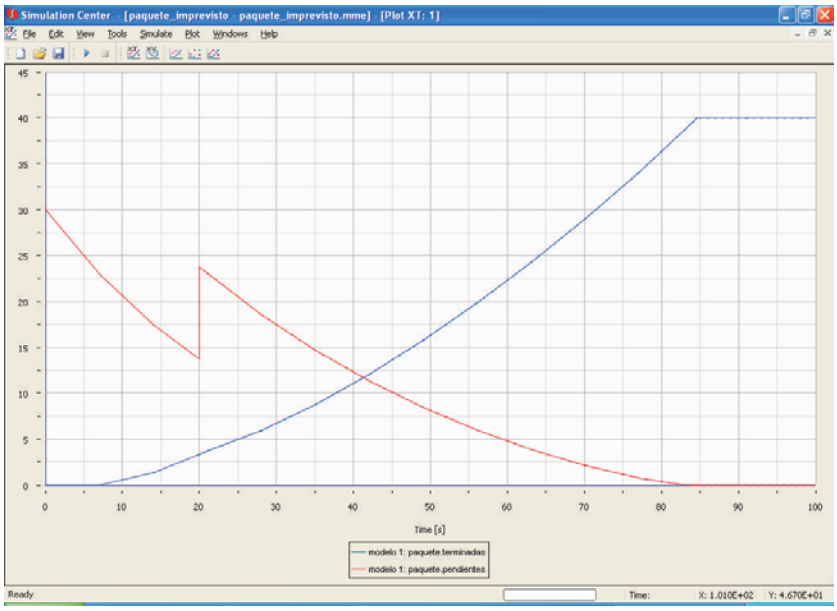


Figura 6.12: Aparición de un imprevisto.

6.7.2. Detección de un error en un paquete

La detección de un error en un paquete es otra de las circunstancias habituales en un proyecto. En los modelos tradicionales se representa como la aparición de trabajo extra (*rework*). Eventualmente se propaga ese trabajo extra "hacia atrás" generando trabajo inicial pendiente adicional.

Implementación

La implementación en este caso difiere del modelo tradicional. No se limita a reinicializar la variable de estado *carPrep*, sino que reduce instantáneamente el trabajo terminado de **paq.actividad** representando así el trabajo perdido. Al contrario que en el caso anterior en el que la aparición de nuevos requisitos supone una carga de trabajo no prevista, aquí la carga inicial de trabajo se actualiza dando por perdido el trabajo erróneo.

El componente **paq.medida** recoge este cambio como una desviación negativa de la planificación, puesto que el trabajo previsto inicialmente se mantiene. El valor presupuestado no cambia pero si lo hacen el alcanzado y la previsión de costes y plazos.

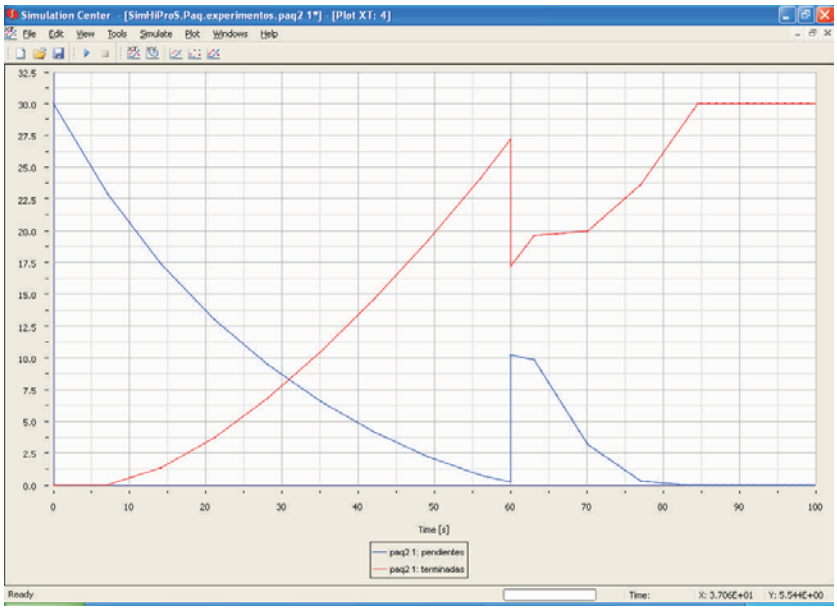


Figura 6.13: Error en un paquete.

Resultados

En la figura 6.13 se observa la diferencia entre este caso y el anterior. Esto se refleja en los indicadores de trabajo conseguido y plazo que presenta la figura 6.14.

6.7.3. Retirada de recursos

Esta circunstancia se produce cuando los recursos empleados en un paquete. Se trata de un evento que o bien puede representar una baja por enfermedad o bien se produce como consecuencia de una detención imprevista (a la espera de una decisión de los clientes, por ejemplo).

Implementación

La implementación de este evento es muy simple. Basta con reducir (o anular como en el ejemplo que se muestra más adelante) la dotación de recursos del paquete durante un cierto intervalo de tiempo. El reflejo de esta reducción es una detención en el progreso del trabajo terminado en

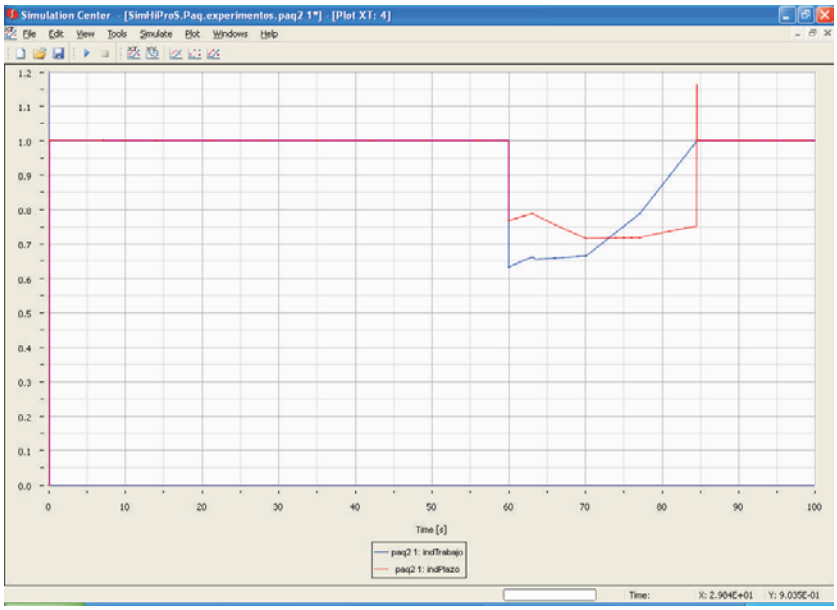


Figura 6.14: Efecto en plazo y valor de un error.

paq.actividad que se refleja en un incremento de la desviación en trabajo y plazo, así como un retraso en el coste.

Resultados

Como en los casos anteriores, la figura 6.15 presenta el efecto en las tareas pendientes y terminadas de este evento.

6.8. Un caso sencillo

Para finalizar la exposición del modelo se presenta un caso sencillo de aplicación con dos proyectos elementales cuyas características se recogen en la tabla 6.12.

El primer proyecto está formado por cuatro paquetes y tiene dos fechas límite: la entrega de una primera funcionalidad a la finalización del paquete **paq21**, y la entrega final. El segundo proyecto sólo tiene dos paquetes y una única fecha, la de la entrega final. Los recursos inicialmente disponibles son un total de 9 analistas y 9 programadores, constantes a lo largo de todo el periodo

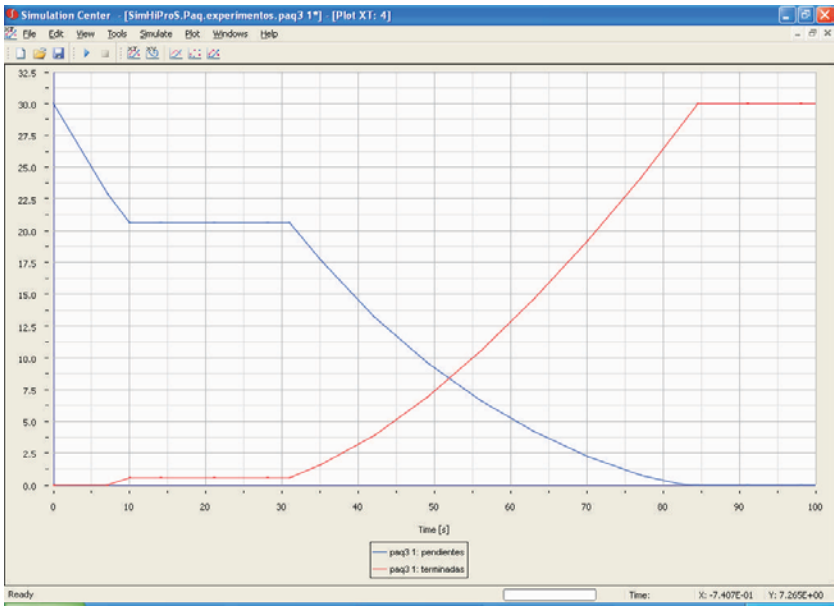


Figura 6.15: Detención por retirada de recursos.

considerado. Según el tamaño de los paquetes y los recursos empleados, la duración de cada uno de ellos es la que aparece en la tabla 6.13.

El ejemplo permite simular tres experimentos:

- El primero de ellos, necesario para la inicialización, simula el comportamiento "normal" de ambos proyectos; es decir, el que se empleará como base para la planificación inicial.

- El segundo experimento simula la aparición de un error en un paquete del primer proyecto, que da lugar a la reasignación de recursos dentro de ese mismo proyecto.

- El tercer experimento simula la aparición de un retraso en el primer proyecto para el que no existe solución factible con los recursos inicialmente asignados a ese proyecto y, por tanto, da lugar a un desplazamiento temporal de recursos del primer al segundo proyecto.

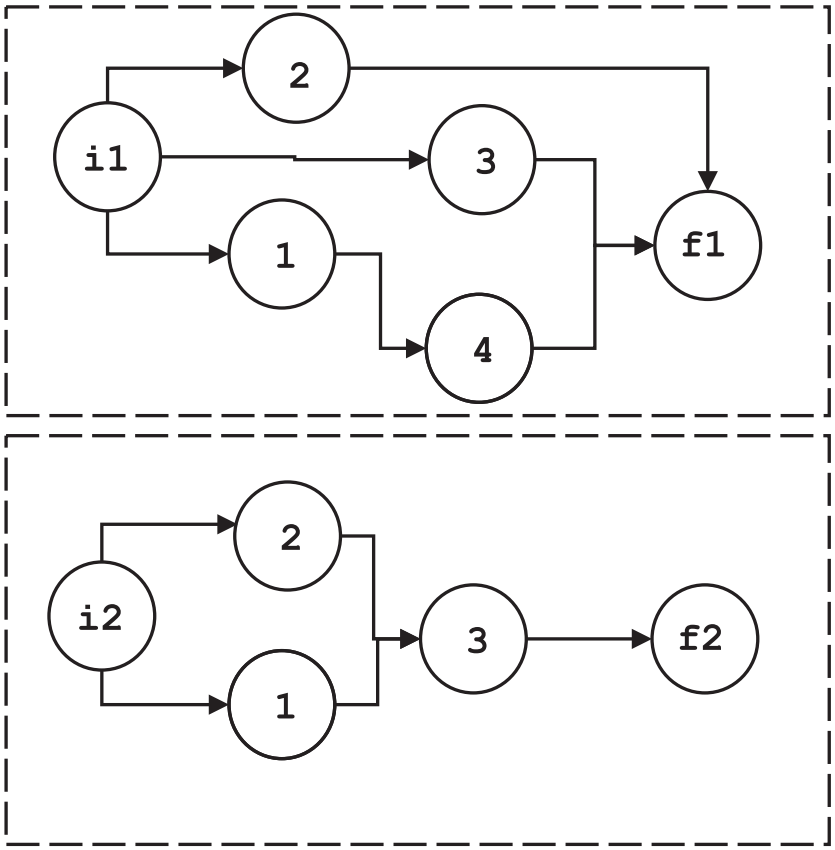


Figura 6.16: Proyectos del ejemplo en notación AEN.

Proyecto	Paquete	Carga de Trabajo	Predecesores	Fecha límite
pro1	paq11	20		
pro1	paq21	40		30
pro1	paq31	15	paq11	60
pro1	paq41	30	paq11	60
pro2	paq12	25		
pro2	paq22	35		
pro2	paq32	30	paq 12, paq22	60

Cuadro 6.12: Características de los proyectos del caso.

Recursos	1	2	3	4	5
10 tareas	24	13			
15 tareas		18	13		
20 tareas		23	16	13	
25 tareas		28	19	15	
30 tareas		33	22	17	
35 tareas		39	26	20	
40 tareas		44	29	22	18

Cuadro 6.13: Modos de ejecución. Duración según tamaño y recursos.

Unidad	paquetes	Ventanas factibles	Carga de trabajo
1	paq11,paq21	1	60
2	paq31, paq41	2	45
3	paq12, paq22, paq32	1-2	90

Cuadro 6.14: Agrupación de paquetes para la planificación táctica

6.8.1. Inicialización del modelo

Planificación táctica

Para la planificación táctica se definen dos ventanas de tiempo de 30 unidades de duración y se agrupan los paquetes en unidades de la forma que refleja la tabla 6.14.

En este caso la planificación táctica es un problema trivial que resulta en la asignación de 5 unidades de ambos tipos de recurso para el proyecto **pro1** y 4 unidades para el proyecto **pro2**.

Programación inicial por proyectos

la programación inicial de los proyectos se realiza según el método descrito en 6.3.3 obteniéndose la secuencia de asignación de recursos a los proyectos que presenta la tabla 6.15. Simulando esa asignación se obtienen los datos de la planificación que se introducirán en el modelo para analizar las desviaciones de producirse alguna disrupción.

La asignación ha dado lugar a una modificación de las precedencias en la red de los proyectos que ya no viene dada sólo por las precedencias derivadas

Evento	Paquete	Asignación
inicio	paq11	2
	paq21	3
	paq12	4
fin paq12	paq22	4
fin paq11	paq31	2
fin paq21	paq41	3
fin paq22	paq32	4

Cuadro 6.15: Asignación de Recursos a Paquetes

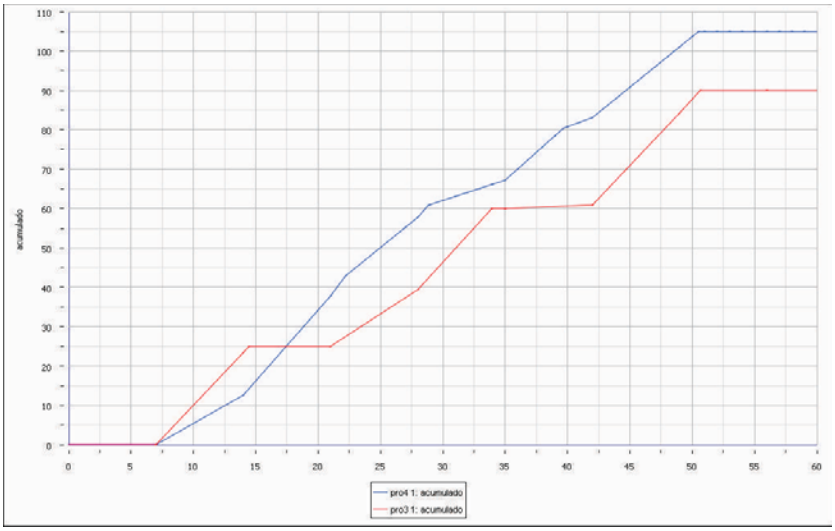


Figura 6.17: Valor conseguido de los proyectos del ejemplo.

de la lógica de los mismos sino también por las restricciones de los recursos. El resultado de la simulación inicial se recoge en las figuras 6.17 and 6.18.

6.8.2. Reparación de la programación

Reparación reasignando recursos dentro del mismo proyecto

Para simular el comportamiento del modelo ante una disrupción de la programación se ha introducido un evento que representa la aparición de un error en el paquete **paq21** que compromete el plazo de entrega parcial previsto (ver 6.7.2). Puede verse como el descubrimiento de un error en un

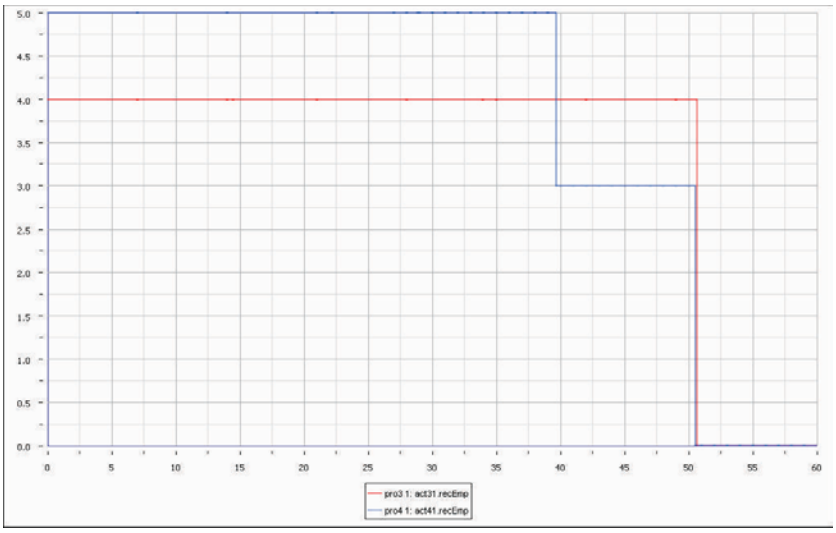


Figura 6.18: Recursos empleados en el ejemplo.

desarrollo que tiene por objeto la construcción de una funcionalidad anticipada del sistema en cuestión.

La respuesta de **pro.asignacion** al error detectado a través del componente **paq.medida** es recalcular la secuenciación de todo el proyecto. Para ello reasigna atribuyendo máxima prioridad a la restricción que representa el plazo de entrega de **paq21**. Ello se traduce en retirar temporalmente los recursos del otro paquete que se ejecuta en paralelo (**paq11**) hasta recuperar el retraso.

Una vez concluido en su plazo el paquete retrasado el otro recibe todos los recursos (no sólo los inicialmente asignados) ya que el resto de paquetes no pueden iniciarse hasta que éste último no concluya. Puesto que esta solución no incumple la restricción del plazo final del proyecto, el algoritmo de reconstrucción de la programación se detiene ahí. El procedimiento concreto se describe en F.8.

Reparación reasignando entre dos proyectos

Si lo que se produce es un evento que no puede corregirse con los recursos asignados al proyecto, entonces se hace necesario retirar recursos del otro. Este es el caso cuando, por ejemplo, la aparición de nuevos requisitos en el

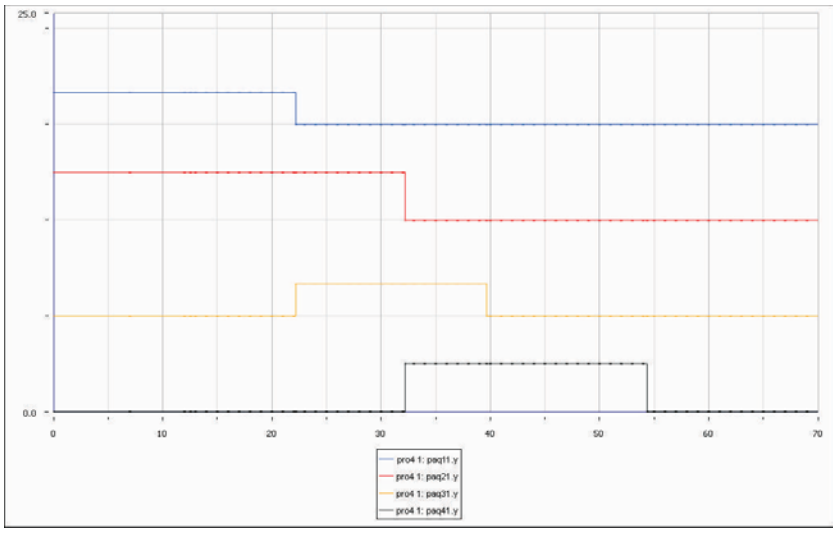


Figura 6.19: Asignación inicial a **pro1**.

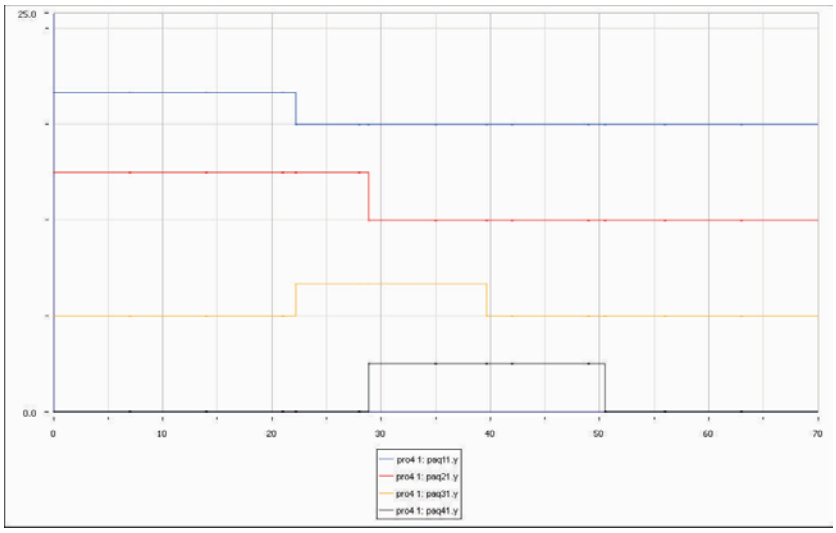


Figura 6.20: Recursos empleados sin variar la asignación.

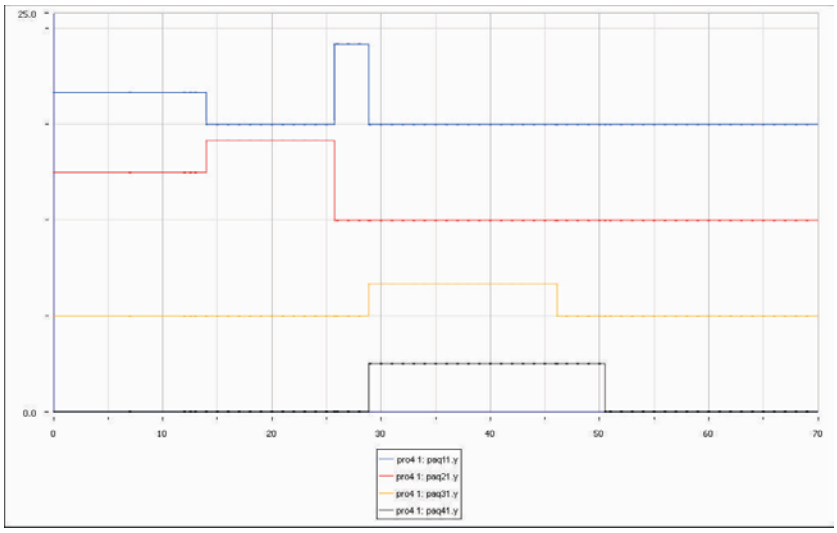


Figura 6.21: Reasignación de recursos a **pro1** tras detectar el error.

paquete **paq41** da lugar a una previsión de finalización que incumple los plazos previstos[8].

En este caso, el algoritmo de reasignación de recursos dentro del proyecto **pro1** no es capaz de generar una asignación factible, es decir compatible con la limitación de recursos y los plazos de entrega previstos[9]. En tal caso el módulo **pro1** emite un aviso con el que se indica que deberán aumentarse los recursos manualmente a costa de otros proyectos.

Esto es lo que se simula en el experimento en el que se resuelve el problema asignando temporalmente recursos del proyecto **pro2** al proyecto **pro1**. Así se recoge en la figura 6.22 donde, tras retirar temporalmente recursos de **pro2**, se incrementan los destinados a este proyecto para poder terminarlo también en plazo.

[8]Normalmente, en una situación real un incremento imprevisto de los requisitos suele implicar un cambio también en los plazos a través de una renegociación de los mismos con el cliente, pero es posible pensar en una situación en la que no sea así.

[9]El algoritmo implementado, SWO, no asegura alcanzar una solución factible para todo tipo de problemas, puesto que el número de iteraciones está limitado y, como se ha señalado, incluso la generación de una solución factible es un problema NP-duro. Sin embargo para la clase de problemas que aquí se analizan, es muy probable que se encuentre.

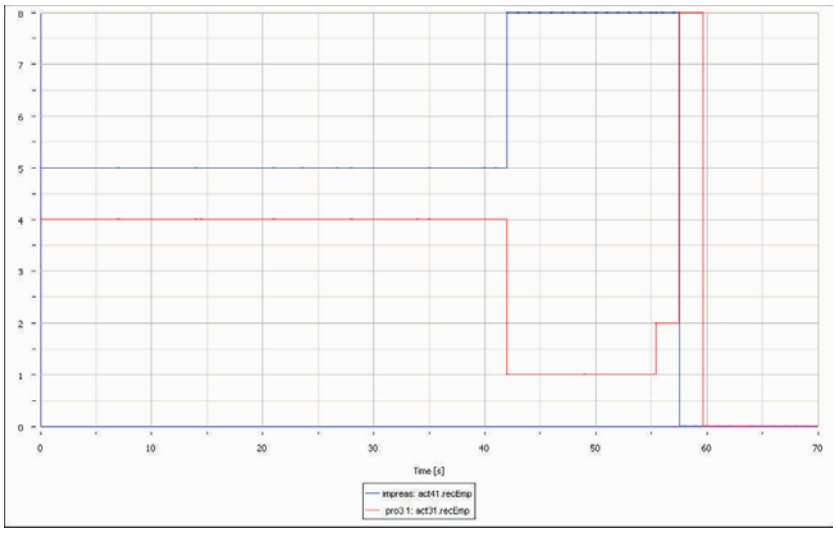


Figura 6.22: Reasignación de recursos entre proyectos.

6.9. La simulación de las actividades no constructivas

Uno de los objetivos de esta tesis es poder simular el efecto de las actividades no constructivas como una aproximación a la medida de la relación coste-efecto de las actividades dirigidas a la mejora de procesos. Para ello se ha realizado una implementación muy sencilla a nivel de proyecto (paquete **pro.actividad**) donde se acumula el esfuerzo de toma de datos que, a su vez, permite la asignación de recursos.

Evidentemente, ésta no es la única actividad de mejora posible, pero si es la más sencilla de definir al nivel de proyecto. Otras actividades a ese nivel, como el mantenimiento y actualización de la documentación del trabajo alcanzado o la gestión de configuración, pueden entenderse en términos de costes como un múltiplo de las anteriores pues su base es la información renovada periódicamente de la marcha de los proyectos. Quedarían aún fuera otras que deben definirse a nivel de organización, en el módulo **entorno** tales como el mantenimiento de repositorios de componentes reutilizables y otras tareas de soporte.

Volviendo a la implementación realizada, el funcionamiento es el siguiente:

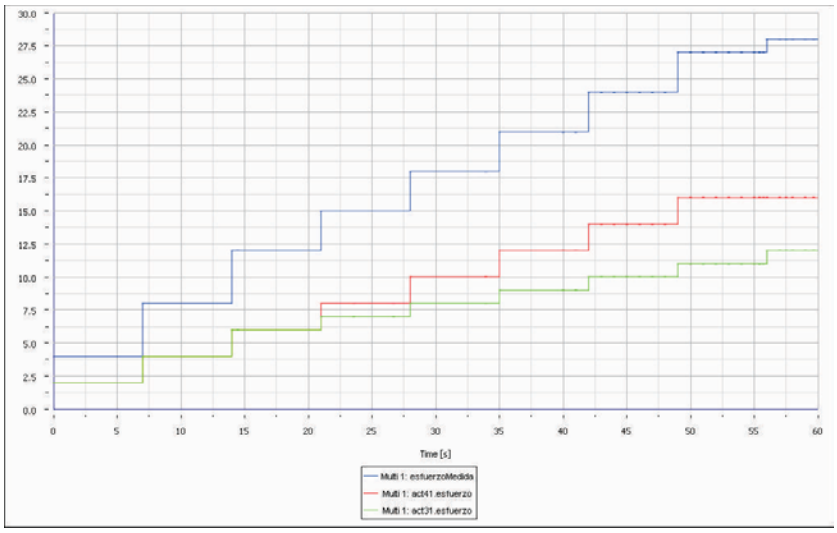


Figura 6.23: Esfuerzo de medida en el caso de ejemplo.

- Cada ciclo de medida ocasiona un coste en términos de recursos proporcional al número de paquetes que están activos en ese momento.
- A su vez,la explotación y sistematización de las métricas obtenidas así como la eventual reasignación de recursos ocasionan también un coste periódico incrementado en caso de reasignación.
- Se ha supuesto que el coste por paquete equivale a medio día de trabajo de un analista más otro tanto por proyecto y por reasignación, en caso de producirse.
- Si la reasignación se produce entre proyectos, ello ocasiona también costes a nivel táctico. Estos costes se atribuyen a los jefes de proyecto y a la dirección de producción.

Capítulo 7

Conclusiones

7.1. Introducción y contenidos

Esta tesis se propone dar respuesta a las siguientes preguntas:

1. ¿Cómo se modela, o se puede modelar, específicamente un entorno de gestión multiproyecto desde el punto de vista normativo más allá de agregar todos los proyectos en un único *metaproyecto*?

2. ¿Cómo se toman las decisiones de asignación de recursos a actividades y proyectos para cumplir plazos y minimizar costes?

3. ¿Cómo se tienen en cuenta los factores de riesgo e incertidumbre en estas propuestas normativas?

4. ¿Qué propuestas hay de modelos de simulación de los procesos software que tengan en cuenta el carácter a la vez continuo y discreto de los mismos?

5. ¿Cómo se pueden implementar las técnicas de los puntos anteriores en esos modelos?

6. ¿En qué medida la mejora de procesos puede reflejarse y evaluarse con esos modelos?

El principal resultado de esta tesis ha sido la construcción del modelo **SimHiProS**, un modelo híbrido para la simulación de la producción de *software* capaz de servir de soporte a la toma de decisiones en la gestión y de ayuda para comprender mejor la dinámica de interacción entre recursos, tareas, tiempo y calidad en este entorno.

El modelo, construido en el lenguaje *Modelica*, permite realizar experimentos con diferentes supuestos de perturbación de las previsiones iniciales y analizar las respuestas. Desarrollos posteriores podrán, como más adelantese indica, aumentar las posibilidades de explotación del modelo.

Este capítulo expone los resultados alcanzados, el grado de novedad e interés de los mismos, sus limitaciones y los trabajos posteriores que darán continuidad a lo conseguido.

7.2. Resultados alcanzados

El modelo construido permite simular una serie de aspectos relevantes de la producción de *software* a medida en un entorno multiproyecto, a saber:

- Representación de los aspectos discretos y continuos de la producción de *software*.
- Representación jerárquica del entorno multiproyecto empleando tres niveles, de abajo a arriba: *paquete*, *proyecto* y *multiproyecto*.
- Simulación de diferentes modelos de proceso *software*: en cascada, incremental, evolutivo, ...
- Obtención de métricas de proyecto según el modelo EVM (*earned value management*).

Por otra parte el modelo **SimHiProS** presenta las siguientes facetas innovadoras respecto a otros modelos de simulación de la producción de *software* preexistentes:

- Capacidad para incorporar modelos de asignación de recursos a nivel de proyecto y multiproyecto, habiéndose implementado algunas variantes sencillas.

- Capacidad para medir el esfuerzo en actividades "no constructivas" como una primera aproximación a la simulación del coste de implantación de la mejora de procesos *software*.

Aspectos discretos y continuos

El modelo **SimHiProS** combina la simulación continua y discreta en los aspectos siguientes:

- Aspectos continuos:
  - Evolución detallada del trabajo de construcción de *software* a través del componente **paq.actividad**.
  - Evolución general del entorno de la organización (dinámica general de los recursos) a través del componente **entorno**.
- : Aspectos discretos:
  - Asignación interna de recursos dentro de un paquete de trabajo (**paq.asignacion**).
  - Medida del trabajo, recursos y tiempo (**paq.medida**).
  - Dinámica de los entregables de un proyecto (**pro.actividad**).
  - Mantenimiento de métricas de proyecto (**pro.medida**) y globales (**mul.medida**).
  - Asignación de personal a paquetes de trabajo dentro de un proyecto (**pro.asignacion**) y global (**mul.asignacion**).

Jerarquía

El modelo se apoya en una propuesta de marco jerárquico para la caracterización del problema de la planificación y asignación de recursos a la producción de *software* que proviene del campo de la Ingeniería de Proyectos. Esta jerarquía se implementa a tres niveles:

Paquete. Es el nivel más bajo al que puede definir un conjunto de actividades medibles y controlables más allá del trabajador individual.

Proyecto. Es el nivel operacional, el conjunto de actividades orientadas a obtener un producto *software*.

Multiproyecto. Es el nivel táctico, representa el conjunto de todas las actividades que se realizan a la vez por la organización.

Modelos de proceso

La abstracción del *paquete* como elemento inferior de la jerarquía permite implementar cualquier modelo de proceso, desde el convencional en cascada a otros modelos más complejos como del desarrollo incremental, el evolutivo y otros. Ello se consigue básicamente de dos formas:

- Al nivel de paquete diferenciando diferentes tipos de trabajo, representados aquí por las categorías *analista* y *programador*. De esta forma, se puede representar un proceso incremental al nivel más bajo.

- A la escala de un proyecto, la geometría de la red de relaciones de precedencia entre paquetes permite configurar cualquier tipo de proceso.

Algoritmos de asignación

El modelo incorpora algoritmos de asignación de recursos a tres niveles:

Paquete. En la implementación actual el modelo asigna los recursos en función del trabajo pendiente en unidades nunca inferiores a 1 día-persona de acuerdo con el ciclo de medida del trabajo pendiente.

Proyecto. El modelo realiza una asignación inicial de recursos a los paquetes a partir del cual se obtiene una planificación inicial. En función de las desviaciones de esa planificación, reasigna el personal dentro de un proyecto mediante la exploración de reglas de prioridad.

Multiproyecto. En el nivel táctico, se asigna personal a proyectos en intervalos regulares de tiempo teniendo en cuenta el coste de los recursos extraordinarios y las limitaciones temporales.

Actividades no constructivas

El modelo simula el coste de las actividades de medida y planificación como una primera aproximación a la medida de las políticas de mejora de los procesos.

7.3. Los resultados de la tesis en relación con el *estado del arte*

El trabajo realizado se basa en resultados previos de dos campos de investigación con un alto grado de desarrollo, la Ingeniería de Proyectos, y dentro de ella los problemas de programación y secuenciación, y la Ingeniería del *Software*, en particular la simulación de procesos *software*. La principal aportación de la tesis es el empleo combinado de métodos y modelos que provienen de ambos campos de investigación.

Este empleo combinado se traduce en avances sobre los trabajos preexistentes en el terreno de la simulación de proyectos *software* en una serie de aspectos que se recogen en los apartados siguientes.

7.3.1. Modelos híbridos

La simulación híbrida es una tecnología de uso general en muchos dominios por lo que existen muchas soluciones. Pero hasta donde se ha podido verificar, en ninguno de los modelos anteriores se integran los aspectos discretos y continuos sin pérdida de la capacidad de representar los efectos de la realimentación como aquí se propone.

Únicamente en [CBK06] se presenta una aplicación del formalismo DEVS, *Discrete Event System Specification*, propuesto por Ziegler [ZKP00] que posibilita representar con naturalidad un sistema híbrido aplicado a la producción de *software*. Sin embargo, por lo menos hasta lo que se ha publicado, en este caso no se implementan sistemas complejos de asignación de recursos y el modelo se limita a emular el funcionamiento de los modelos continuos de la saga del modelo de Abdel Hamid y Madnick [AM91]. Puede decirse, por

tanto, que si bien ofrece una herramienta adecuada para implementar la simulación híbrida, no se ha empleado,que se tenga noticia, para otra cosa que para realizar una integración numérica por un método alternativo al basado en incrementos finitos constantes.

7.3.2. Modelo del entorno multiproyecto

En el trabajo realizado se ha considerado el problema de la gestión multiproyecto en dos marcos de referencia. El primero clasifica los entornos multiproyecto en términos de *variabilidad* y *dependencia*. El segundo diferencia entre los niveles de decisión *estratégico*, *táctico* y *operacional*. Esta es una aportación de la Ingeniería de Proyectos recogida en la tesis.

A pesar de que se reconoce generalmente la especificidad y la pertinencia del entorno multiproyecto para el tratamiento de la producción de *software*, las referencias concretas son bastantes limitadas. Esta tesis propone una arquitectura jerárquica en la que el entorno multiproyecto y la existencia de diversos niveles de decisión con objetivos diferentes se puede aplicar a una variedad muy amplia de casos, sin tener que recurrir a modelos "ad hoc´´. El carácter modular y la orientación a objeto del modelo lo permiten.

7.3.3. Empleo de modelos y métodos de secuenciación de tareas y programación de proyectos

Esta es precisamente una de las aportaciones singulares de la tesis que aquí se expone. Si bien el campo de la gestión de proyectos es un área de aplicación con solera en la metodología de la Dinámica de Sistemas, esta no ha sido hasta ahora capaz de implementar reglas de asignación que no sean elementales. Vease, a tal efecto, por ejemplo un trabajo reciente recogido en [JF05] en el que los algoritmos de asignación no pasan de métodos de generación de secuencias rudimentarios.

Sin embargo, en la práctica los responsables de los proyectos adoptan constantemente decisiones de asignación de recursos y esas decisiones, en muchos casos, se basan en estrategias implícitas de equilibrado de recursos,

asignación de prioridades y control de costes. Tradicionalmente la academia ha explorado el espacio de las soluciones, buscando optimizar resultados en modelos teóricos de difícil aplicación dada la imprecisión (*fuziness*) y variabilidad de los entornos reales. Es posible que la razón de la escasa aplicación de estas aportaciones esté en esos factores.

Pero como ha demostrado el estudio realizado existen propuestas tanto desde el mundo de la Investigación Operativa como desde la Inteligencia Artificial y la Ingenería de Producto que tienen una orientación estratégica antes que de optimización local. Pero quitando el modelo propuesto por Joslin [JP05] no hay referencias en la literatura de gestión de proyectos *software* que tengan en cuenta estas propuestas.

Recientemente han aparecido propuestas de aplicación de las técnicas de CSP (véase, por ejemplo [BdOBW08]) a la asignación de recursos a proyectos. Dentro de esta filosofía de aprovechar los resultados de otros campos de la ingeniería se incluye la presente tesis.

7.3.4. La mejora de procesos y la simulación

La aproximación al problema que se ha empleado es la de modelar explícitamente las tareas que la mejora impone. El modelado explícito de estas tareas tiene dos virtualidades:

- Proporciona mayor precisión en el cálculo del esfuerzo requerido que no es sólo el trabajo directo de análisis y construcción sino que incluye, además de la propia gestión, las tareas anexas a ésta que corresponden a lo que genéricamente se llama la sobrecarga de comunicación (*communication overhead*)

- Arroja un mapa efectivo para la mejora de los procesos al permitir contrastar el coste de la misma con el beneficio esperable

Son muchos los modelos que se hacen eco del efecto de la mejora de procesos y los trabajos "no constructivos"; desde su propia estructura, como es el caso del modelo de Ruiz Carreira [Car03], hasta la implementación explícita de la misma, como el modelo *IMMoS* [PR02] que combina a tres niveles, con

Dinámica de Sistemas, modelos de procesos y el empleo de modelos cuantitativos basados en la medida. El modelo que se presenta en esta tesis se corresponde más con el segundo tipo, pero a diferencia de él, consigue modularizar la descripción de los procesos y representar los procesos de obtención de métricas sin necesidad de un esfuerzo excesivo de readaptación a cada caso.

7.4. Novedad e interés de los resultados

Las razones que avalan la novedad y el interés de los resultados de la tesis aquí presentada se pueden agrupar en tres categorías que se describen en los apartados siguientes.

7.4.1. Continuidad e innovación

La Simulación de Sistemas Dinámicos tiene una tradición ya larga en la Universidad de Sevilla, donde esta metodología, introducida en su día por el profesor Aracil Santonja ha dado lugar a diferentes desarrollos y aplicaciones en muy diversas áreas de conocimiento [BL77, DM77, FM82, PN87, TB87].

La rama de aplicaciones a los proyectos *software*, inspirada en el trabajo seminal de Abdel-Hamid y Madnick [AM91] dentro de la universidad hispalense, arranca con la tesis doctoral de Isabel Ramos [Ram99] y sigue con la de Mercedes Ruiz Carreira [Car03].

Si bien algunos de los componentes de la tesis que aquí se propone ya han sido establecidos en dichos trabajos, como por ejemplo la reducción de los modelos y la simulación del efecto de los procesos de mejora, esta propuesta incorpora elementos novedosos tales como la metodología híbrida (los modelos anteriores son todos continuos), la consideración explícita de las tareas de medición y mejora como tareas "no constructivas" pero con contenido de trabajo, consumo de recursos e incidencia en el tiempo de ejecución o la consideración de modelos sofisticados de asignación de recursos.

7.4.2. Incorporando perspectivas relevantes para los problemas reales

Esta tesis nace del esfuerzo por comprender y mejorar el proceso de construcción de *software* en un caso real y representativo. Por ello se ha intentado incorporar al modelo construido aspectos que son relevantes para ese problema. En este sentido, la tesis realiza aportaciones desde varias perspectivas de este tipo.

A pesar de la recomendación de Abdel-Hamid [Abd93b] sobre la necesidad de tener en cuenta el entorno multiproyecto, la literatura revisada se concentra en proyectos singulares o, inculso, en fases dentro de estos. Sin embargo, en los problemas reales de muchas organizaciones dedicadas a la producción de *software* la "multitarea" y los recursos compartidos son la norma. Esta es una de las aportaciones de esta tesis que diferencia entre los niveles táctico y operacional pero a la vez los vincula en una jerarquía de decisiones.

Otra segunda perspectiva relevante que incorpora esta tesis para los problemas reales es el tratamiento de la incertidumbre, no como un margen de error en la estimación de las cargas de trabajo, que también, sino específicamente como un desconocimiento radical a "priori" del contenido de trabajo de un proyecto. Es el problema conocido en medios profesionales como "volatilidad de los requisitos". Eso da lugar a la necesidad de implementar políticas reactivas, sin perjuicio de que se elaboren planificaciones robustas para hacer frente a los proyectos. El límite a las planificaciones robustas lo impone la dificultad de caracterizar estadísticamente las disrupciones que se producen en este tipo de proyectos, razón por la cual tampoco la simulación de Monte Carlo ofrece una solución definitiva.

La tercera perspectiva relevante a efectos de los problemas reales es la consideración de las métricas y la mejora de los procesos como tareas con contenido en si mismas, algo que se encuentra detrás de las dificultades de las organizaciones de desarrollo, especialmente pequeñas y medianas, para adoptar los estándares existentes. Estos se ven, sobre todo, como una sobrecarga innecesaria ("burocracia"), por lo que es de interés analizar el *trade-off*

entre las consecuencias de la adopción de estas prácticas en términos de calidad y eficiencia, de una parte, y de coste, de otra.

7.5. Limitaciones y líneas de desarrollo futuro

Las líneas de desarrollo futuro para la línea de trabajo en que se inscribe esta tesis son básicamente las siguientes:

- La incorporación de otros desarrollos que son "estado del arte" en la simulación de proyectos al modelo básico aquí propuesto
- El empleo del modelo propuesto para el desarrollo de una herramienta de apoyo a la decisión en tiempo de ejecución
- La explotación del modelo para experimentos de aprendizaje automático en el marco de estrategias más sofisticadas

En cuanto a lo primero,caben destacar las siguientes extensiones posibles:

1. En el modelo no se ha incluido la representación de fenómenos de no-linealidades, presión de los plazos, influencia de los errores que son estado del arte en la simulación de proyectos. La razón para ello es que se buscaba aislar los efectos de las dinámicas de asignación de recursos. Metodológicamente, la incorporación de estas facetas no presenta mayor dificultad y de hecho bastaría añadir nuevos bloques.

2. A fin de refinar la capacidad de presentación de cualquier configuración de proceso, es posible construir paquetes que utilicen un número variable de tipos de recurso (en lugar de dos, cómo de ha considerado en el modelo en su versión primera). De hecho, la incorporación de trabajo "sólo de análisis" o de trabajo de mantenimiento de código del tipo *Y2K* se pude modelar con un sólo recurso. Eso supondría una leve modificación del módulo **paquete** actual.

3. La implementación de algoritmos más sofisticados de asignación tropieza con las limitaciones del lenguaje algorítmico de *Modelica*. Sin embargo, mediante la inserción en los módulos **asignacion** de las oportunas llamadas a funciones externas en C/C++ y FORTRAN es posible utilizar cualquier modelo exacto o metaheurístico avanzado codificado en esos lenguajes.

En cuanto al empleo en el modelo en una herramienta más conversacional de ayuda a la decisión del tipo "what if?" se trataría de desarrollar las *interfaces* necesarias para ello. El lenguaje *Modelica* nació con vocación de simulación de sistemas físicos y técnicos por lo que las plataformas de simulación que lo emplean están orientadas a este tipo de sistemas, lo mismo que las bibliotecas existentes. Algunas de esas plataformas proporcionan diversas funcionalidades entre las que cabe destacar:

- La interoperabilidad con otros entornos de simulación (Simulink, SciLab, ...).

- APIs para la construcción de modelos de manera gráfica a partir de bibliotecas de componentes.

- La más avanzada es la posibilidad de la *modificación dinámica* de la estructura de los modelos - que en éste caso podría significar la posibilidad de alterar la red de los proyectos, por ejemplo - en tiempo de ejecución en función de cambios en las variables de estado del sistema.

Las referencias de estas y otras plataformas y utilidades pueden encontrarse en página *web* del consorcio *Modelica*: `www.mdelica.org`.

Por último, existe una línea de trabajo sobre el aprendizaje y la inferencia de reglas a partir de la simulación [ARRRT01] que podría beneficiarse de la capacidad de este modelo de representar reglas de decisión más complejas que las tradicionalmente implementadas.

7.6. Trabajos originados

Los trabajos preliminares sobre los que se sustenta este proyecto de tesis han dado lugar a dos comunicaciones que han sido aceptadas en el CEP-MaW'08 Construction and Engineering Project Management International Workshop, a celebrar en Valladolid los días 2 y 3 de octubre de 2008.

Dichos trabajos llevan por título, respectivamente:

- Simulación de políticas de asignación de recursos en proyectos software: estado del arte.
- Una propuesta de aplicación de métodos heurísticos para la programación de proyectos a la producción de software a medida.

Una descripción del modelo será presentada en el taller ADIS'08, evento realizado en el marco de las XIII Jornadas de Ingeniería del Software y Bases de Datos (JISBD 2008- `http://www.sistedes.es/jisbd2008/`) a celebrar del 7 al 10 de octubre de 2008 en Gijón .

Apéndice A

Modelo formal del problema táctico

Este apéndice describe la formalización del problema de la planificación táctica de capacidad como un problema de programación lineal. El punto de partida es un horizonte temporal finito, T, dividido en intervalos - *buckets* - regulares de duración normalizada a la unidad (por ejemplo, una semana). La empresa tiene una cartera de proyectos definidos en términos de actividades agregadas. La empresa emplea recursos diferentes, disponiendo de Q_{jt} unidades del recurso en el intervalo de tiempo t. Cada actividad agregada i requiere un empleo del recurso R_j que viene dado por q_{ij}.

Las actividades agregadas pueden tener relaciones de precedencia entre si. De hecho, cada uno de los proyectos no es más que la subred resultante de esas relaciones de precedencia. Las actividades agregadas además se realizan en modo continuo, es decir, la duración de las mismas en términos de intervalos de tiempo depende de la cantidad de recursos de los que disponga. Debe disponer de todos los recursos que requiere para poder completarse. En general una actividad agregada dura varios periodos de tiempo.

A lo largo de cada periodo se completa una fracción del contenido en trabajo total de una actividad agregada. De un periodo a otro esa fracción puede cambiar. Sea x_{jkt} la fracción de la actividad agregada j que se realiza en el periodo t empleando el recurso k. Suponemos que esa fracción es igual

para todos los recursos lo que equivale a decir que la proporción de recursos requeridos de cada tipo es constante en una actividad, o lo que es lo mismo, cada actividad agregada implica una combinación fija de recursos. Esta visión de la actividad a un nivel agregado lo permite. De no ser así se puede descomponer una actividad agregada en varias. Eso nos permite referirnos en lo sucesivo a x_{jt} como la fracción de la actividad j que se realiza en el periodo t. La intensidad del trabajo puede ser variable de un periodo a otro representándose como un cambio en esa fracción.

Supongamos además que la máxima fracción de trabajo que puede hacerse de una actividad en un intervalo de tiempo es j es $\frac{1}{p_j}$. Eso equivale a decir que la duración mínima de cada actividad es p_j periodos. Una tarea está completa en el intervalo $[t_1, t_2]$ si la suma de todas las x_{jt} en ese intervalo es igual a la unidad.

Cada tarea tiene asignada una ventana de tiempo $[r_j, d_j]$. No puede empezarse antes de r_j ni terminarse después de d_j. Por tanto x_{jt} debe ser cero para $t < r_j$ y $t > d_j$. Además $d_j - r_j$ tiene que ser mayor o igual que p_j, el tiempo mínimo en que se pude hacer la tarea.

Las relaciones de precedencia las representaremos como la exigencia de que si la actividad i precede a la actividad j, en el momento τ para que $x_{j\tau}$ sea mayor que 0 es necesario que la suma de x_{it} desde cero hasta τ sea igual a 1.

Una solución está formada por una lista ordenada de $n \times T$ componentes, x_{jt}, con t de 1 a T y j de 1 a n que indica la fracción de la actividad j que se ejecuta en el periodo t. Para ser factible requiere:

- Respetar las relaciones de precedencia y de tiempo
- No ser mayor que $\frac{1}{p_j}$
- Que $\sum_{t=1}^{T} x_{jt} = 1$

Por último existe la posibilidad de emplear recursos extraordinarios o no regulares en cantidad U_{kt} para el recurso k en el intervalo t a un coste c_{kt}. Suponemos inicialmente que el empleo de recursos no regulares (horas extraordinarias, empleados eventuales, ...) no está limitado.

La función objetivo a considerar puede adoptar dos formas:

- *Problema restringido por el tiempo:* se trata de cumplir con un horizonte temporal para cada proyecto al menor coste de oportunidad que es el derivado del empleo de recursos no regulares.
- *Problema restringido por los recursos:* se trata de minimizar la duración de los proyectos sin emplear recursos irregulares.

A.1. El problema restringido por el tiempo

El problema restringido por el tiempo, sin tener en cuenta las relaciones de precedencia, puede representarse como:

$$\text{mín} \sum_{t=1}^{T} \sum_{k=1}^{K} c_{kt} U_{kt} \tag{A.1}$$

Sujeto a:

$$\sum_{t=1}^{T} x_{jt} = 1 \tag{A.2}$$

$$x_{jt} \leq \frac{1}{p_j} \tag{A.3}$$

$$U_{kt} \leq \sum_{j=1}^{n} q_{jk} x_{jt} - Q_{kt} \tag{A.4}$$

$$x_{jt}, U_{kt} \geq 0 \tag{A.5}$$

En el que se busca minimizar el coste de los recursos irregulares asegurando que se completan los proyectos respetando el límite a la duración mínima de las actividades pero sin tener en cuenta las relaciones de precedencia; es decir es una variante relajada del problema.

Para introducir las relaciones de precedencia, se define una ventana de tiempo VT_j para una actividad j como un intervalo $[S_j, C_j]$ tal que la actividad no puede comenzar antes de S_j ni concluir después de C_j. Un conjunto de ventanas de tiempo CVT es una lista de n elementos donde cada actividad

tiene asignada una ventana de tiempo. Un conjunto de ventanas de tiempo *factible* es un conjunto de ventanas de tiempo que cumple las relaciones de precedencia y la restricción A.3. Existe un problema A.1 para cada *CVT* factible.

Si ahora sustituimos $\frac{1}{p_j}$ en la restricción A.3 por $\frac{s_{jt}}{p_j}$ donde s_{jt} vale 1 si la actividad j se está ejecutando en el tiempo t y 0 de lo contrario, cada problema A.1 nos aparece como un problema de programación binaria que se puede descomponer en dos grupos de restricciones:

- *Restricciones que afectan a todos los proyectos*, las relacionadas con los recursos
- *Restricciones de cada proyecto*, las relacionadas con las relaciones de precedencia

Esto permite descomponer la matriz de las restricciones en dos bloques, uno primero que recoge las restricciones relativas a recursos y otro segundo relativo al conjunto CVT factible. Una solución cualquiera del problema (continuo) general es una cota inferior del problema restringido (mixto) en el tiempo. A su vez, una solución para un problema restringido para un CVT factible es una solución factible del problema general y puede servir, por tanto, de punto de partida para un heurístico de mejora. A partir de aquí los autores consultados proponen soluciones heurísticas que permiten obtener resultados de calidad en tiempos de computación razonables.

A.1.1. Generación de una primera solución factible

La generación de una primera solución factible en términos de CVT puede hacerse mediante un heurístico muy sencillo: iniciando la actividad agregada j lo antes posible y acabándola lo más tarde posible. Téngase en cuenta que en este nivel de definición del problema, las ventanas de tiempo vienen fijadas por los hitos externos del proyecto, por lo que tanto el inicio como el final de cada actividad viene dado por esos hitos.

$$S_j = r_j$$

$$C_j = \text{mín}(d_j, \text{mín}(r_k - 1; k \in S(j)))$$

A.1.2. Selección del modo de ejecución: recursos a emplear

El modo de ejecución (recursos a emplear) vendrá de la solución del problema de programación lineal continua A.1 con la restricción añadida:

$$x_{jt} = 0 \ \forall t < S_j \ , t > C_j \tag{A.6}$$

A.1.3. Mejora de la solución

Cualquier algoritmo de mejora debe basarse en reducir el consumo de recursos extraordinarios, o lo que es lo mismo, en modificar la restricción A.4 haciendo 0 alguno de los valores de x_{jt}. Lógicamente en el óptimo del problema inicial esa restricción no tiene holgura (pues no tiene sentido utilizar más recursos extraordinarios de los necesarios).

Para reducir el consumo de recursos extraordinarios el primer criterio es seleccionar aquellas actividades que los requieren, es decir, las actividades j, que cumplen que $U_{jt} > 0$ para algún valor de t. Las variaciones posibles son aumentar o dismuinuir los límites de la ventana $[S_j, C_j]$ en 1 unidad. Eso requiere, a su vez que las ventanas adyacentes admitan esa reducción o aumento. Si eso es así, se dispone de hasta cuatro "vecinos" (*neighbours*) para explorar.

El criterio de exploración se basa en las propiedades de la dualidad en programación lineal. Si se modifican los términos independientes de las restricciones y la solución dual sigue siendo óptima se produce un cambio en la función objetivo que es proporcional a la variación de las variables del dual. De este modo es posible implementar un método de exploración local *steepest descent* (descenso más rápido) seleccionando el mejor de los cuatro vecinos y siguiendo así hasta alcanzar una solución satisfactoria (recuérdese que se

dispone de una cota inferior que es la solución del problema sin considerar las ventanas de tiempo).

A.2. El problema restringido por los recursos: *forward resource loading.*

La variante restringida por los recursos se denomina problema de carga de recursos hacia adelante, *forward resource loading* (FRL). Aquí la función objetivo debe incorporar un término que penalice los retrasos. Si no es posible emplear recursos extraordinarios, basta con hacer 0 las variables U_{kt} que representan su consumo. Sin embargo la formulación se complica ya que para evaluar (y penalizar) los retrasos, se hace necesario determinar la fecha de finalización de cada proyecto y ello no puede hacerse sin incluir en la función objetivo alguna medida relacionada no ya con los tiempos de ejecución de cada actividad sino con la duración de los proyectos en su conjunto, lo que obliga a introducir explícitamente nuevas variables. Nótese que en el caso anterior esto no era necesario en la formulación del problema básico, bastaba con resolverlo en el subespacio de los CVT admisibles.

Una forma de simplificar el problema es suponer que la estructura de actividades agregadas de un proyecto particular es simplemente una cadena, es decir, una secuencia lineal. Esta simplificación al nivel que estamos considerando de planificación táctica de capacidad no supone una limitación grave al realismo de los modelos. En un proyecto de desarrollo *software* los hitos de entregas intermedias pueden servir para definir esas actividades agregadas.

Supónganse n proyectos formados por actividades agregadas *(b,j)* donde *(b,j)* es la actividad b-ésima del proyecto j. Los recursos vienen definidos como en el caso anterior como $R_1, R_2, \ldots, R_K$ disponiéndose de Q_{kt} unidades del recurso R_k en el intervalo de tiempo t. Cada proyecto debe empezar en la fecha r_j y concluir antes de la fecha d_j. Cada actividad agregada *(b,j)* requiere un empleo del recurso R_k que viene dado por q_{bjk}. Como en el caso anterior consideramos un empleo de recursos en proporciones constantes, de manera que p_{bjt} es la fracción de la actividad *(b,j)* que se realiza en el periodo

t.

Llamaremos un plan π para un proyecto j a un vector a^{π}_{bjt} con elementos binarios que contiene un 1 si la actividad *(b,j)* se puede realizar en el periodo t y un 0 de lo contrario. Sólo consideramos planes temporalmente factibles, es decir, planes en los que se cumplen las relaciones de precedencia entre actividades, las fechas finales y la duración mínima de las actividades. La generación de planes factibles es un problema relativamente sencillo puesto que hasta este punto los proyectos son independientes unos de otros. Un plan indica cuando se permite iniciar una actividad.

Llamaremos una *secuencia de carga* a un vector Y_{bjt} cuyos elementos son la fracción de la actividad b del proyecto j que se realiza en el tiempo t. Multiplicando cada elemento de ese vector por el correspondiente p_{bj} sabemos el consumo de recursos de cada actividad en cada momento.

El tiempo para completar un proyecto j, CT_j, es el último intervalo de valor 1 de un plan π para ese proyecto j. La tardanza de un proyecto δ es la diferencia entre la fecha señalada de finalización de un proyecto, d_j, y su tiempo real de finalización, CT_j. El retraso de un proyecto vale 0 si la tardanza es negativa y $CT_j - d_j$ si la tardanza es positiva. Esta variable es la que se incluye en la función objetivo. Nótese que como la tardanza negativa no se considera, se puede llegar a una solución de coste mínimo que puede mejorarse en términos de duración de otras actividades y proyectos.

La solución de este problema puede abordarse de diversas maneras a partir de la obtención de una primera solución admisible. Esto se consigue empleando un heurístico tradicional del problema RCPSP como los presentados en 2.4.3 o bien la fase I del modelo tradicional Simplex en dos fases para generar una solución inicial o demostrar la inexistencia de las mismas [Han01]. A partir de esta solución se pueden seguir métodos exactos o métodos heurísticos.

A.2.1. Solución exacta

Se trata de un algoritmo denominado *branch and price*[1] [BJN+98], [Han01], [SS03], [SW06] y [BD07]. Este método, utilizado para resolver problemas de asignación, combina el método de exploración dirigida a través de un árbol de soluciones con la inclusión progresiva de columnas, que corresponden a planes factibles, en función de un criterio derivado del precio sombra, es decir, evaluado a través de los costes reducidos.

A.2.2. Solución aproximada

El método exacto, a pesar de explotar la posibilidad de descomponer la matriz del problema agregando columnas progresivamente, cada una de las cuales es un plan factible, proporciona soluciones que no son factibles, al resultar planes que son combinaciones convexas de planes factibles (es decir, el proyecto debe hacerse en un 20 % de acuerdo con un plan y en un 80 % de acuerdo con otro, lo cual obviamente es imposible). Por ello [GS05] proponen un método heurístico de mejora que sólo evalúa soluciones factibles. Para ello se parte de un único plan factible (una columna) y se miran los precios sombra para ver que modificaciones deben hacerse a los proyectos, bien adelantando o demorando las fechas de inicio y/o finalización. Tras eliminar las modificaciones inviables eligen aquella que más mejora la solución. El heurístico prosigue resolviendo problemas restringidos de una sola columna por plan hasta que no se puede mejorar.

[1] Este nombre se deriva de la metodología de exploración dirigida *branch and bound* (ramificación y acotado)

Apéndice B

Modelo formal del problema operacional

Este apéndice presenta diversos modelos formales del problema operacional que, como se ha explicado, es el conocido normalmente en la literatura como problema RCPSP (*resource constrained project scheduling problem*).

B.1. El problema RCPSP determinista con un solo modo de ejecución

Comenzaremos por el problema más simple: el problema de la secuenciación de proyectos con recursos limitados, *resource constrained project scheduling problem*, RCPSP en la nomenclatura tradicional que ha sido y sigue siendo objeto de muchísima atención y trabajo.

El modelo que se presenta es una extensión del utilizado en la programación sin limitación de recursos con la notación *AEN* (actividades en nodos).

Variables de decisión:

- t_i tiempo de comienzo de la actividad i.
- $S(t)$ conjunto de actividades que se procesan durante el instante t.

Geometría de la red:

- N, conjunto de nodos de la red, corresponden a las actividades. Existen dos actividades ficticias, la *0* predecesora de todas las demás y la n que es la última.
- $A = \{(i, j)\}$ conjunto de arcos de la red que indican las relaciones de precedencia - en este caso i precede a j.

Datos:

- p_i duración de la actividad i. Se consideran $n + 1$ actividades, luego $\forall i \in N$ se tiene que $i = 0, \ldots, n$. Además, las actividades ficticias inicial y final cumplen que $p_0 = p_n = 0$
- r_{ik} consumo de recurso tipo k al realizar la tarea i.
- b_k cantidad disponible de recurso k.

Parámetros:

- K, número de recursos que intervienen en la realización del proyecto.
- T, horizonte temporal.

Con todo ello se puede escribir el modelo siguiente:

$$\text{mín } t_e \tag{B.1}$$

Sujeto a:

$$t_i - t_j \geq p_j \quad \forall (i, j) \in A \; \forall i \in N \tag{B.2}$$

$$\sum_{i \in S(t)} r_{ik} \leq b_k \quad t = 1, \ldots, T \quad k = 1, \ldots, K \tag{B.3}$$

$$t_i \geq 0 \tag{B.4}$$

Esta formulación del modelo es conceptual pero conlleva la dificultad para hacerlo operativo de la definición del conjunto de actividades que se realizan en el instante t, *S(t)*.

Con el objeto de resolver este problema, se plantea una variación sobre la formulación del modelo convirtiéndolo en un problema de *programación*

mixta [PWW69]. Para ello se define la variable *binaria* x_{it} que toma el valor de 1 si la actividad i se completa en el instante t (téngase en cuenta que el tiempo está discretizado) y 0 en caso contrario. El problema queda como:

$$\text{mín} \sum_{t=EFT_n}^{LFT_n} t \cdot x_{nt} \tag{B.5}$$

Sujeto a:

$$\sum_{t=EFT_i}^{LFT_i} x_{it} = 1 \quad i = 1, \ldots, n \tag{B.6}$$

$$\sum_{t=EFTi}^{LFT_i} x_{it} \leq \sum_{t=EFT_j}^{LFT_j} t \cdot x_{jt} - p_j \quad \forall (i.j) \in A \tag{B.7}$$

$$\sum_{i=1}^{n} \sum_{q=max\{t,EFT_i\}}^{min\{t+d_i-1,LFT_i\}} r_{ik} \cdot xiq \leq b_k \quad k = 1, \ldots, K \; t = 1, \ldots, T \tag{B.8}$$

$$x_{it} \in \{0, 1\} \; i = 1, \ldots, n \quad t = EFT_i, \ldots, LFT_i \tag{B.9}$$

Donde EFT_i y LFT_i son respectivamente la fecha más temprana, *earliest finish*, y más tardía, *latest finish*, de la actividad i-ésima.

Las restricciones respectivamente imponen, por el orden en el que se presentan, la necesidad de que todas las actividades se hayan completado, las relaciones de precedencia y el límite de consumo de recursos.

B.2. El problema RCPSP determinista con diferentes modos de ejecución - MRCPSP

El problema se basa en los mismos supuestos del caso anterior, pero también incluye la posibilidad de que el modo de realizar el procesamiento de la actividad haga variar la cantidad consumida de recurso.

La formulación presentada en [DH02] emplea una variable de decisión x_{imt} que cumple:

- $x_{imt} = 1$ si i comienza a realizarse en modo m en t
- $x_{imt} = 0$ de lo contrario

El modelo resultante es el siguiente:

$$\text{mín} \sum_{t=es_n}^{ls_n} t \cdot x_{nmt} \tag{B.10}$$

Sujeto a:

$$\sum_{m=1}^{M_i} \sum_{t=es_i}^{ls_i} x_{imt} = 1 \quad i = 1, \ldots, n \tag{B.11}$$

$$\sum_{m=1}^{M_i} \sum_{t=es_i}^{ls_i} (t + d_{im}) x_{imt} \leq \sum_{m=1}^{M_j} \sum_{t=es_j}^{ls_j} t \cdot x_{jmt} \quad \forall (i,j) \in A \tag{B.12}$$

$$\sum_{i=1}^{n} \sum_{m=1}^{M_i} r_{imk} \sum_{s=max\{t-d_{im}, es_i\}}^{min\{t-1, ls_i\}} x_{ims} \leq b_k \quad k = 1, \ldots, K \tag{B.13}$$

$$x_{imt} \in \{0,1\} \; i = 1, \ldots, n \quad m = 1, \cdots, M_i \quad t = es_i, \ldots, ls_i \tag{B.14}$$

En este caso, las fechas de referencia son es_i, el inicio más temprano, *earliest start*, y ls_i, el inicio más tardío, *latest start*. r_{imk} es el consumo del recurso k-ésimo por parte de la actividad i en el modo m (de entre los $1, \ldots, M_i$ posibles para esa actividad). La duración de las actividades depende del modo a través de d_{im}.

En el modelo aquí presentado, el recurso es *renovable* y las actividades no pueden suspenderse una vez iniciadas. A diferencia del problema táctico, el modo de ejecución seleccionado no varía en cada intervalo de tiempo.

B.3. El problema RCPSP con ventanas de tiempo

El problema RCPSP con ventanas de tiempo, también denominado TC-PSP - *time constrained project scheduling problem*- es una variante del problema RCPSP en el que las actividades solo pueden ejecutarse en un intervalo

de tiempo determinado, la *ventana de tiempo*, de manera análoga a como se ha descrito el problema de planificación táctica de capacidad restringido por el tiempo en A.1.

A su vez es un caso particular del problema *RCPSP/max* o también denominado como problema con relaciones de *precedencia generalizadas* (página 37). Cualquier problema con relaciones de precedencia generalizadas se puede modelar empleando un *retraso mínimo y/o máximo* entre las fechas de inicio de dos actividades que se encuentran ligadas por una de estas relaciones. Como en cualquier proyecto se incorporan dos actividades ficticias, una inicial y otra final, la imposición de una ventana de tiempo absoluta a cualquier actividad se puede resolver simplemente imponiendo un retraso mínimo y uno máximo (descontando la duración de la actividad implicada) respecto de la actividad ficticia inicial.

Para modelar el problema con ventanas de tiempo (ahora con una notación AEN, actividad en nodos) se definen:

- N, conjunto de nodos de la red, corresponden a las actividades. Existen dos actividades ficticias, la *0* predecesora de todas las demás y la$n$ que es la última.
- $[r_i, d_i]$ ventana de tiempo, fecha de inicio y plazo máximo (*release* y *deadline*), de la actividad i
- $A = \{(i, j)\}$ conjunto de arcos de la red que indican las relaciones de precedencia - en este caso i precede a j.
- p_i duración de la actividad i. Se consideran $n + 1$ actividades, luego $\forall i \in N$ se tiene que $i = 0, \ldots, n$. Además, las actividades ficticias inicial y final cumplen que $p_0 = p_n = 0$
- q_{ik} consumo de recurso tipo k por intervalo al realizar la tarea i.
- b_{kt} cantidad disponible de recurso k en el intervalo discreto t[1].

[1]Nótese, que al contrario que en los dos modelos antes expuestos, aquí la dotación de recursos no es constante con el tiempo, como por otra parte es previsible a partir de la solución del problema táctico que asigna grados de realización diferentes a cada uno de los intervalos discretos.

- K, número de recursos que intervienen en la realización del proyecto.

Además se definen dos variables binarias:

- x_{it}, vale 1 si la tarea i se realiza en el intervalo discreto t y 0 de lo contrario

- s_{it}, vale 1 si la tarea i se inicia en el intervalo discreto t y 0 de lo contrario

El espacio de las soluciones admisibles vendrá definido por:

$$\sum_{d_i-1}^{t=r_i} x_{it} = p_i \ \forall n \tag{B.15}$$

$$\sum_{n}^{0} x_{it} q_{kt} \leq b_{kt} \ \forall k, t \tag{B.16}$$

$$\sum_{d_i-1}^{t=r_i} s_{it} = 1 \ \forall n \tag{B.17}$$

$$x_{i0} = s_{i0} \ \forall i \tag{B.18}$$

$$x_{it} \leq x_{it-1} + s_{it} \ \forall i, t > 0 \tag{B.19}$$

$$s_j \geq s_i \ \forall (i,j) \in A \tag{B.20}$$

$$s_{it} = 0 \ \forall t \notin \{r_j, \ldots, d_j - p_j\} \tag{B.21}$$

$$x_{it} = 0 \ \forall t \notin \{r_j, \ldots, d_j - 1\} \tag{B.22}$$

La restricción B.15 requiere que todas las actividades se completen; la restricción B.16 que se respeten los límites de recursos; la B.17 que todas las actividades comiencen, y las demás que se respeten las reglas de precedencia y las ventanas de tiempo.

La función objetivo en este problema puede adoptar varias formas, de modo semejante a como ocurría con el problema de la planificación táctica. Con la formulación expuesta, minimizar la duración de todo el proyecto se

formula simplemente como:

$$\text{mín} \sum_{T}^{t=0} t \cdot s_n t \tag{B.23}$$

El problema de esta formulación es que el límite es la propia solución. El horizonte temporal se puede acotar, no obstante, con la suma de la duración de todas las actividades del proyecto,$\sum_{n-1}^{1} p_i$.

En el entorno de una programación operacional, otros objetivos podrían ser:

- minimizar el plazo de terminación de algunas, o todas, las tareas para asegurar la robustez de la programación ganando holguras respecto al plazo final
- equilibrar el empleo de recursos
- e incluso relajando las restricciones de recursos, minimizar el coste del empleo de recursos extraordinarios

B.4. El problema multimodo con ventanas de tiempo

El modelo que más se aproxima al caso en estudio es una combinación de los dos problemas anteriores, en el que además de las ventanas de tiempo, cabe la posibilidad de elegir entre diversos modos de ejecución. Para una planificación inicial, el modo de ejecución no varía una vez elegido (al contrario que en el problema táctico).

Para incorporar la multimodalidad, la variable clave es s_{imt} que vale 1 cuando la actividad i comienza en el intervalo t en mdo m. Como en el caso general MRCPSP, debe cumplirse:

$$\sum_{m=1}^{M_i} \sum_{t=r_i}^{d_i-1} s_{imt} = 1 \quad i = 1, \ldots, n \tag{B.24}$$

Esta restricción asegura que cada actividad se empieza dentro de su ventana de tiempo admisible y sólo en un modo.

En este caso, las duraciones, a su vez, dependen del modo por lo que p_i se convierte en:

$$\sum_{m=1}^{M_i} \sum_{t=r_i}^{d_i-1} s_{imt} \times p_{im}$$

El resto de las restricciones deben alterarse análogamente.

La gran cantidad de variables binarias convierte a este en un problema fuertemente combinatorio por lo que los modelos derivados de la programación lineal son menos eficientes. De ahí que se plantee su solución a través de un procedimiento de mantenimiento de ventanas de tiempo que propague las restricciones conforme se van secuenciando las tareas, como se plantea en F.1.

Apéndice C

Modelos alternativos del riesgo

En este apéndice se presentan las visiones alternativas al problema del *riesgo derivado de la complejidad* a las que se ha hecho referencia en el texto.

C.1. Redes estocásticas. El modelo GERT

El modelo original PERT ya incluía una aproximación muy elemental al problema de la variabilidad en la duración de las actividades caracterizándolas por tres estimaciones; "pesimista","optimista" y "normal". El GERT - *graphical evaluation reviewe technique* - es una (de entre muchas otras como el VERT, el GAPS, ...) extensión del modelo que tiene en cuenta la existencia de diferentes estados posibles en un sistema y de las interrelaciones entre las diversas variables de estado, es decir, la información que los diferentes componentes del sistema intercambian. Para ello emplea una representación denominada *red estocástica*.

Cuando las estructuras incluyen nodos AND e INCLUSIVE-OR, la solución analítica no es posible "a priori" y se requieren procedimientos con un elevado coste computacional. Cuando las redes son complejas, aunque todos los nodos sean EXCLUSIVE-OR, la reducción topológica también es excesivamente gravosa. Por ello se han desarrollado métodos basados en la simulación de redes estocásticas para el tratamiento de estos problemas. La mayor parte de las referencias recientes consultadas emplean estos modelos a

Referencia	Descripción
[Kur06]	Simulación de Monte Carlo con disyunciones
[Gav04]	GERT con lógica "fuzzy"
[GGGL03]	Disyunciones con el problema "knapsack"
[A.P03]	Transformación y simplificación de la red
[KN02]	Simulación de Monte Carlo
[Nag02]	GERT y algoritmos genéticos

Cuadro C.1: Algunas aplicaciones de los modelos GERT en el dominio de la gestión de proyectos.

efectos de representación y simulación como las que se recogen en el cuadro C.1, sin que se pretendan explotar sus posibilidades analíticas. Entre otros motivos porque otros formalismos, como el de la Redes de Petri que se expone a continuación, son más potentes tanto analítica como computacionalmente.

C.2. Las Redes de Petri

Las Redes de Petri fueron introducidas por C.A. Petri in 1962. Su éxito se debe básicamente a la simplicidad de su mecanismo básico, si bien, la representación de grandes sistemas es costosa. Básicamente es un modelo de representación de un sistema caracterizado por un conjunto de estados y sus regles de transición asociadas. Numerosos autores han extendido el modelo básico introduciendo diversos refinamientos.

Las redes de Petri, en particular en las versiones de alto nivel (HLPN) comentadas: temporizadas, coloreadas, estocásticas, ... son un formalismo de gran potencia para la descripción de un sistema. Como en el caso anterior una aplicación inmediata es la representación de los problemas. Pero posee también un potencial importante para el análisis y para la simulación.

C.2.1. Tratamiento analítico.

El principal tratamiento analítico es el *análisis del espacio de estados* en particular en términos de verificar la *alcanzabilidad* de determinadas regio-

nes. Sin embargo, para cualquier problema mínimamente relevante el espacio de estados alcanza dimensiones enormes. Para resolver el problema se han desarrollado herramientas muy potentes para reducir este espacio por agregación o condensación de subespacios, análisis de invariantes de lugar, etc. Los algoritmos de exploración del espacio de estados son semejantes a los algoritmos *branch and bound* ya comentados.

C.2.2. Simulación.

El formalismo de las redes de Petri y su implementación hace que estas sean *ejecutables* desde su especificación, de ahí que se empleen naturalmente para la simulación. El Departamento de Computación de la Universidad de Aarhus en Dinamarca ha desarrollado un paquete denominado *Design/CPN* que puede descargarse de la url `http://www.daimi.aau.dk/CPnets/`.

Las redes de Petri han venido a sustituir a todas las representaciones en forma de grafos mejorados previamente existentes y hay abundantes referencias en la literatura respecto a su empleo como herramientas para el tratamiento de problemas de programación de proyectos. Algunas de las consultadas son las que aparecen en el cuadro C.2.

C.3. Las *DSM*, una propuesta desde el dominio del desarrollo de producto

Las matrices de estructura de diseño, *design structure matrix*, *DSM*, son un formalismo para representar el grafo dirigido de un proyecto que se emplea en el dominio del *desarrollo de productos*. Se trata de una matriz cuadrada con elementos binarios con m filas y columnas y n elementos no nulos, representando m tareas y n arcos indicativos de relaciones de precedencia.

La distribución de la matriz es la siguiente: los procesos o actividades se disponen como encabezamiento de las columnas y etiquetas de las filas. Si existe un arco del nodo (actividad) i al nodo (actividad) j, el elemento (j, i) vale 1. Los elementos de la diagonal principal de la matriz no tienen ningún significado por lo que se dejan normalmente en blanco.

Referencia	Descripción
[HS08]	Desarrollo de productos
[HBH+06]	Aseguramiento de calidad basado en el valor
[KHY06]	Predicción y reparación de secuencias en entorno multiproyecto
[KWDK06]	Gestión de carteras multiproyecto basada en análisis de costes ABC
[KC06]	Desarrollo de productos
[dSP05]	Desarrollo de software basado en componentes
[ZnRF04]	Evaluación y control de riesgos plazo/coste
[RKC01]	MRCPSP combinado con algoritmo genético
[JKCR00]	Equilibrado de recursos
[Kum98]	Modelo básico de asignación de recursos
[EKR95]	Modelado de procesos work-flow

Cuadro C.2: Algunas aplicaciones de las Redes de Petri en el dominio de la gestión de proyectos.

La principal ventaja de este modo de representación sobre el grafo convencional es su compacidad y su capacidad de proporcionar una visión sistemática de las relaciones entre procesos o tareas más fácil de leer con independencia de su tamaño. El orden de presentación de las filas (y las columnas) es el de la secuencia de las actividades, por lo que los elementos de una fila cualquiera indican las tareas que deben haber terminado antes de iniciar la tarea que representa esa fila. De la misma forma, leyendo las columnas, cada fila indica las tareas que reciben información de la correspondiente a la columna en cuestión.

Los elementos no nulos (o *marcas*) que se encuentran por debajo de la diagonal principal indican información "hacia adelante". Los elementos que están por encima de la diagonal, por el contrario, representan procesos de *realimentación* según las cuales una tarea puede depender de sus sucesoras. Esas dependencias "hacia atrás" son la principal causa de iteraciones en proyectos complejos. En el caso de los proyectos de desarrollo *software* representan, por ejemplo, el clásico trabajo de corrección de errores en la codificación.

La arquitectura de la descomposición en tareas - WBS - de un proyecto puede, hasta cierto punto, definirse por la dirección del mismo. Evitar al máximo las iteraciones equivale a diseñar una estructura en la que el número de elementos situados por encima de la diagonal sea mínimo. En proyectos complejos es poco probable que se puedan anular dichos elementos, pero minimizarlos y desplazarlos lo más próximo posible a la diagonal permite llegar a una estructura de proyecto más fiable y predecible [Bro02]. El proceso para conseguirlo se llama *particionar* la matriz.

Desacoplar la matriz es el proceso de identificación de aquellos elementos que, si se eliminaran, convertirían a la matriz en triangular. Identificar estos elementos equivale a identificar los supuestos que dan lugar a las iteraciones en el proyecto, es decir, las tareas que están acopladas. En [YB03] se exponen procedimientos algorítmicos para particionar y desacoplar las matrices.

En la notación binaria, las DSM contienen un único atributo relativo a cada par de actividades, si existe o no una relación entre el resultado de una y el de la otra. Las DSM *numéricas* pueden contener muchos atributos,

lo cual enriquece la información sobre el proyecto. Así, por ejemplo, si la tarea B depende de información sobre la tarea A, pero esa información es predecible o tiene un impacto escaso el elemento puede ser eliminado. Con el fin de recoger esta mayor capacidad informativa, se proponen las matrices DSM numéricas en las que se incluyen nuevos atributos como:

Indicadores de Nivel: Reflejan el orden en el que las marcas "hacia atrás" pueden eliminarse en función de las posibilidades de estimar "a priori" la información que se genera en la actividad posterior.

Indicadores de Importancia: Una escala sencilla, entre 1 y 3, para indicar la importancia de la dependencia de la actividad sucesora en la actividad precedente.

Probabilidad de repetición: Una estimación de la probabilidad de que la actividad sucesora provoque una repetición de la antecesora; en este modelo, empleado entre otros en [Cho05], los elementos situados sobre la diagonal representan la probabilidad de una iteración hacia atrás mientras que los situados por debajo indican la probabilidad de una segunda iteración. Se han elaborado algoritmos de particionado que minimizan la duración esperada del proyecto.

Una de las más extendidas aplicaciones de este modelo es la simulación de Monte Carlo para obtener un perfil duración/coste de un proyecto de desarrollo de producto. La varianza en la duración y el coste son atribuibles en gran medida a las iteraciones. Dichas iteraciones pueden o no ocurrir dependiendo del comportamiento de una serie de variables que el modelo simula como variables estocásticas cuya probabilidad de ocurrencia depende de la existencia de determinados paquetes de información que activan un proceso de reelaboración (*rework*. Un proceso de reelaboración puede, a su vez, desencadenar otros sucesivos afectando así al proyecto en su totalidad.

Este modelo se ha utilizado en solitario o en combinación con otras técnicas de las ya reseñadas para elaborar secuenciaciones robustas modificando la arquitectura de la descomposición en tareas de los proyectos. Así ocurre en [CE05], [ZY04] y [MYB07].

C.4. La analogía cuántica

Una original alternativa al estudio del riesgo proviene de una *analogía* basada en la *mecánica cuántica* [FL07].

Se supone que la implementación de una actividad es análoga a un paquete de ondas que se propaga en dirección a un hito temporal. Cuando las actividades son coherentes, el hito se alcanza. Cuando una actividad se retrasa la probabilidad de la ocurrencia del hito en un determinado momento queda afectada, como ocurre con la probabilidad de un sistema cuántico de partículas cuando se producen fenómenos de interferencia.

Esta aproximación se basa en en la estructura del proyecto y de los hitos más que en la información detallada sobre cada una de las actividades. Las acciones externas o las decisiones de la dirección pueden afectar a la realización de los hitos y esto se representa por medio de dos parámetros internos, cada uno de los cuales toma un valor específico en cada uno de los hitos. La idea básica es que la actividad humana no puede definirse con precisión absoluta sino que más bien, cierto principio de incertidumbre gobierna las actividades y la productividad del trabajo. A partir de esto las tareas del proyecto e hitos son modelados mediante *funciones de onda* con su correspondiente *amplitud de probabilidad* ψ_i. La amplitud de probabilidad de un hito se representa como la superposición de las amplitudes de probabilidad de las actividades que *interfieren* en dicho hito.

La probabilidad del hito es $P = |\Psi|^2$. Si el hito no es *perturbado*, es decir, si las actividades no se retrasan, las amplitudes de probabilidad de las mismas interfieren en fase creando un máximo en el patrón de interferencia. La amplitud $\left(\sqrt{\kappa}\right)^{-1}$ mide la mayor importancia de las actividades de mayor duración.

En la proximidad de un hito, la superposición de los campos de difracción de las actividades individuales componen un patrón de difracción definido entre los mínimos más próximos. Para interpretar este patrón como una densidad de probabilidad debe ser normalizada a 1.

Si una tarea se retrasa en ΔD días, su fase ϕ cambia en $\Delta\phi = \kappa_0 \Delta D$, cambiando la contribución de la actividad a la probabilidad del hito. La

coherencia de la interferencia es parcialmente destruida y la probabilidad del hito desciende. El modelo recoge también el efecto de un desfase mayor para las tareas que dependen de otras tareas. Las actividades programadas para acabar inmediatamente antes de los hitos tienen un impacto mayor en la probabilidad de estos que otras tareas más alejadas. En cadenas de tareas mutuamente dependientes, la perturbación de la primera se propaga a las siguientes. Las tareas más alejadas contribuyen menos que las más próximas, etc.

Analizando la programación del proyecto, el modelo calcula dos parámetros, la incertidumbre de hito, *milestone date uncertainty*- MDU, y el periodo de recuperación, PR. MDU depende de la distribución de las duraciones de las tareas y las holguras, decrece para tareas cortas. El PR, como indica su nombre, no es más que el plazo necesario para corregir los posibles retrasos que puedan producirse antes del hito. Ambos parámetros *dependen sólo* de la estructura de descomposición en tareas y proporcionan la tolerancia, en términos de unidades de tiempo, de la duración de las actividades así como la capacidad de recuperación del proyecto. Si se dispone de información adicional sobre el "clima" del proyecto, se pueden escalar dichos parámetros para reflejar situaciones de mayor o menor riesgo.

Al contrario que en el caso del camino crítico en el que la demora en una de las actividades situadas sobre ese camino cambia la probabilidad de los hitos del 100 % al 0 %, este modelo refleja que cualquier variación en cualquier tarea afectará en alguna medida al proyecto. Matemáticamente eso se debe al desplzamiento de fase de la función de onda de las actividades, pero en la práctica lo que refleja es que cualquier retraso en alguna tarea afecta a los hitos al reducir la flexibilidad de la programación.Presenta, pues, una representación continua puesto que los hitos son eventos con una distribución continua en el tiempo y no eventos puntuales. Este modelo está implementado en una aplicación comercial denominada $Quantum^{TM}$ de la que puede obtenerse información y algún ejemplo en `www.ibico-cor.com`.

Apéndice D

Diagramas de bloques de los modelos

Figura D.1: Conectores.

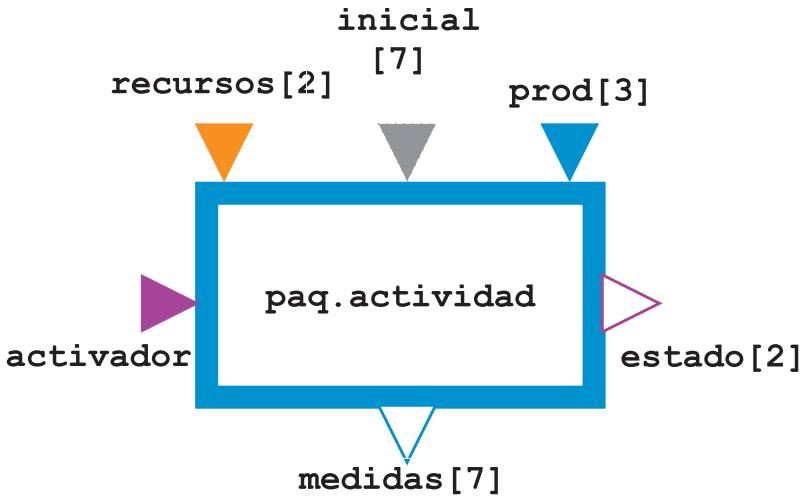


Figura D.2: Componente **paq.actividad**.

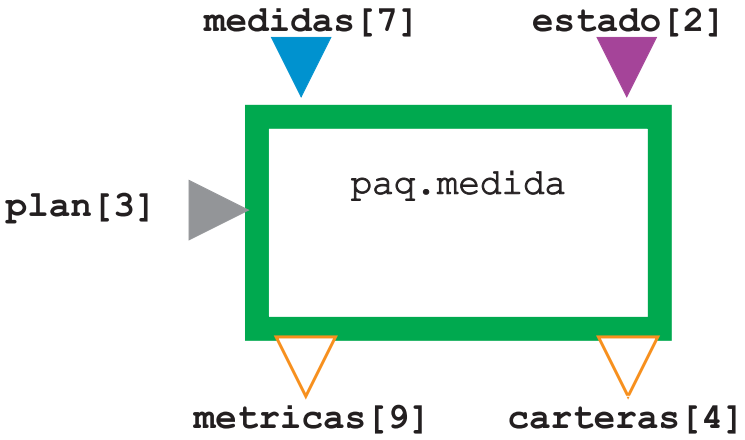


Figura D.3: Componente **paq.medida**.

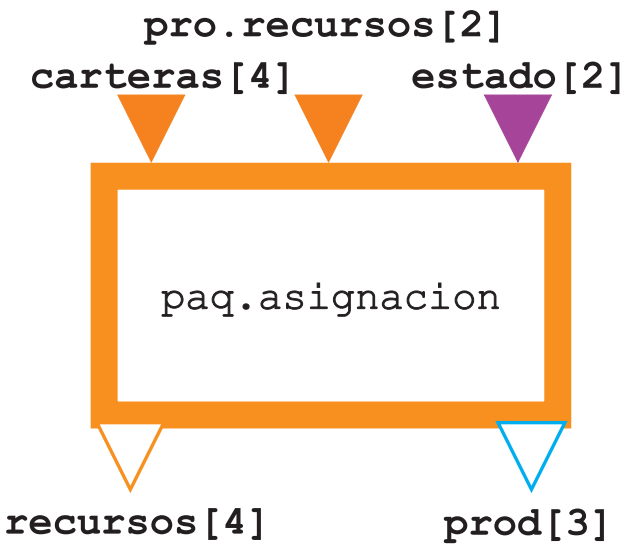


Figura D.4: Componente **paq.asignacion**.

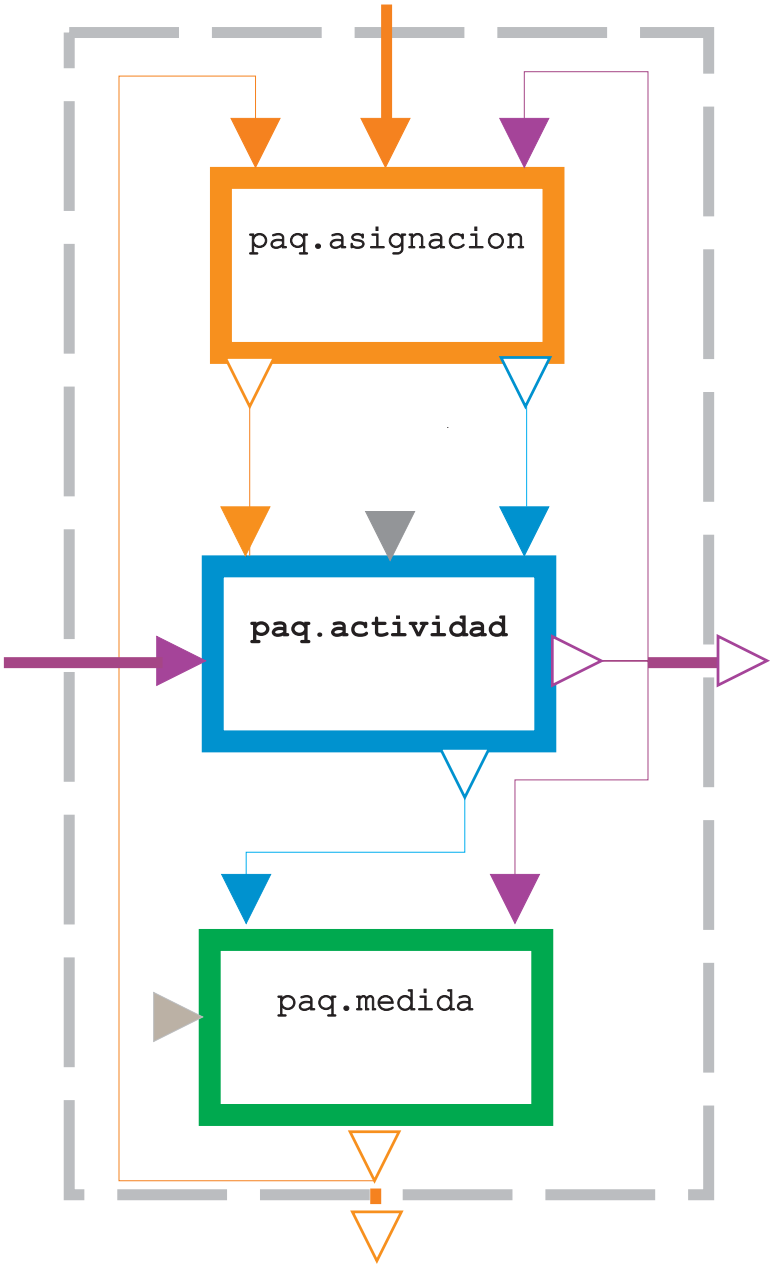


Figura D.5: Módulo **paquete**.

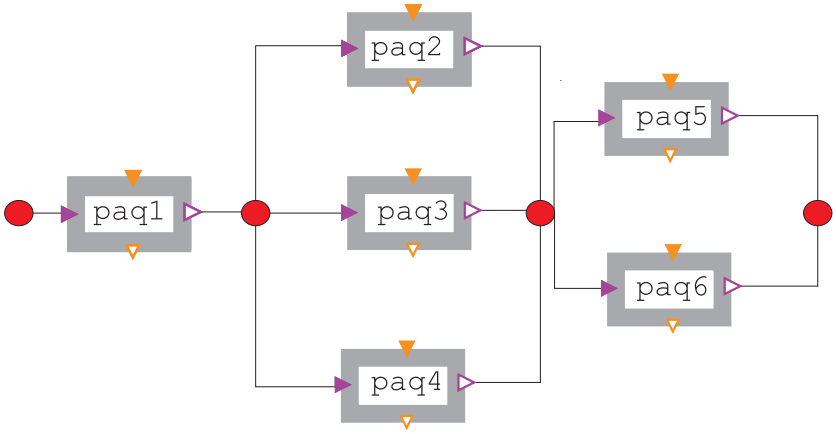


Figura D.6: Componente **pro.actividad**.

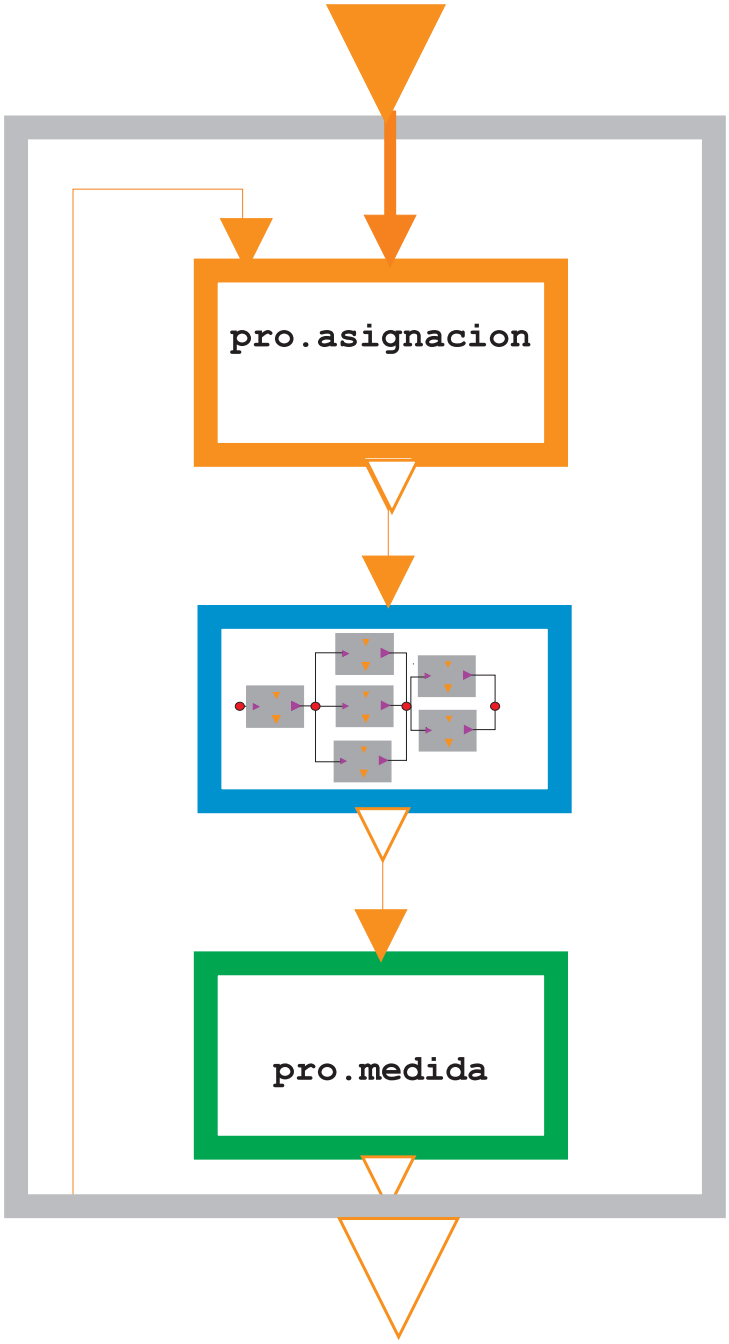


Figura D.7: Módulo **proyecto**.

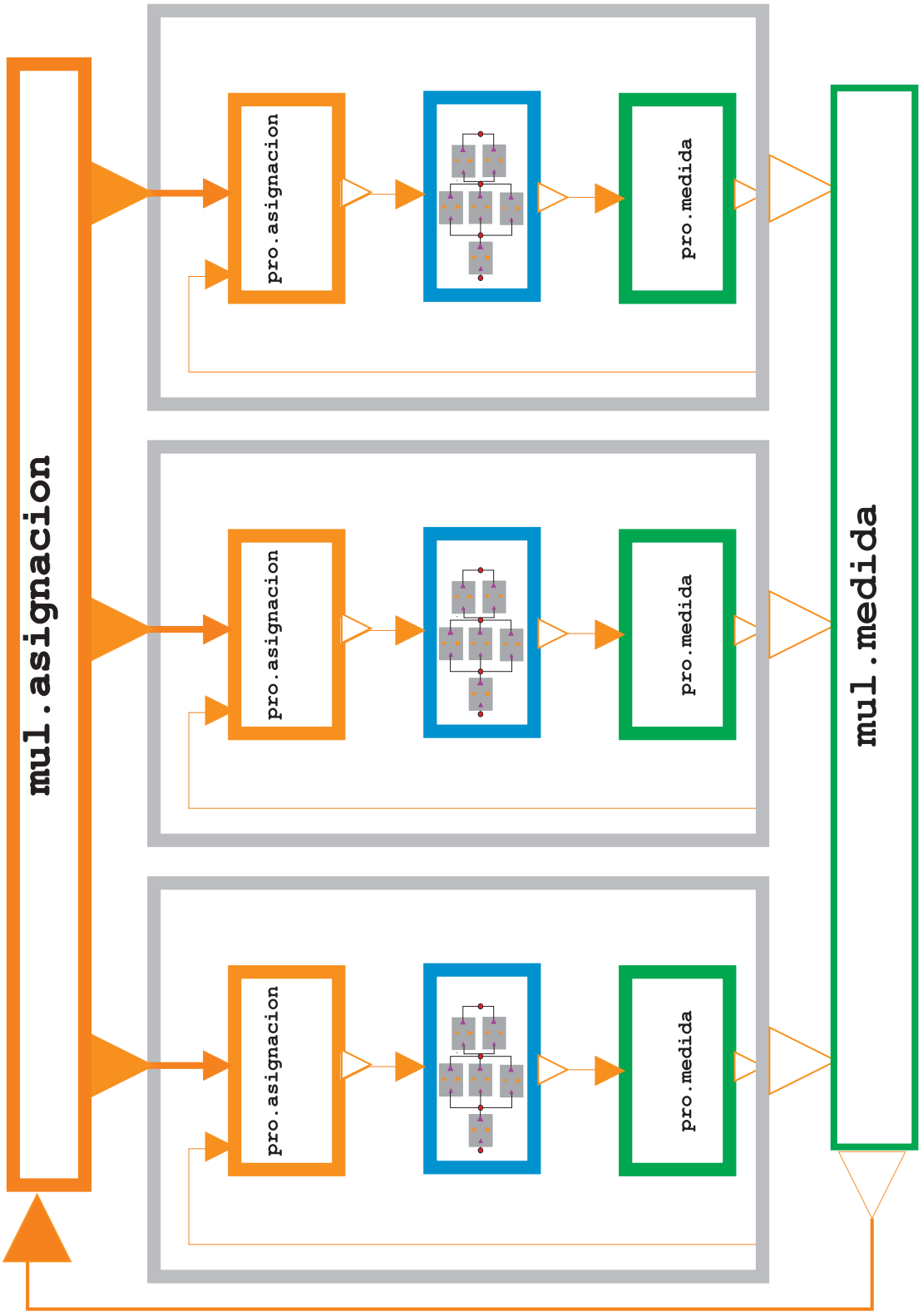


Figura D.8: Módulo **multiproyecto**.

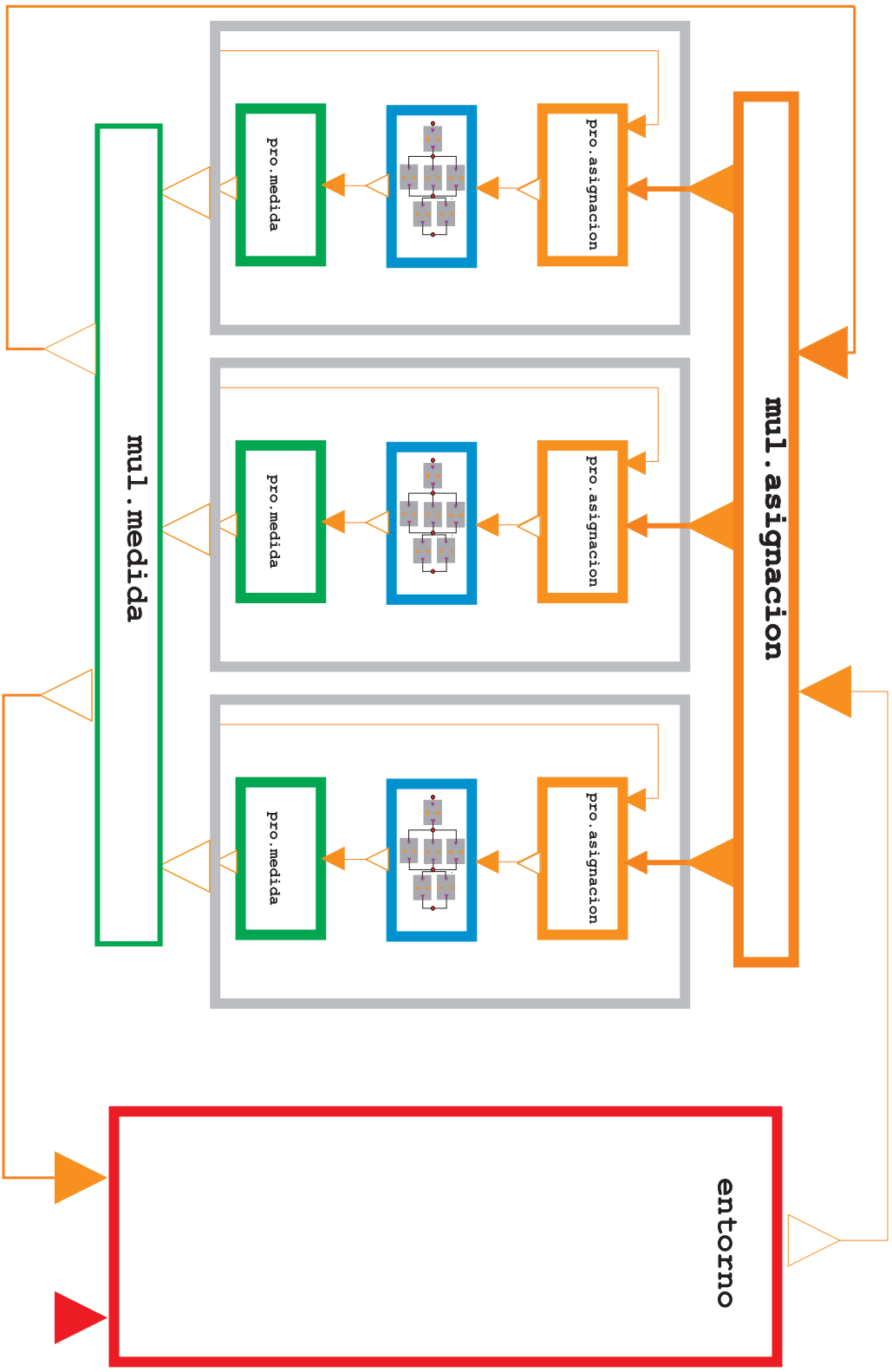


Figura D.9: Relación multiproyecto-entorno.

Apéndice E

Ajuste del modelo

Con el fin de ajustar los parámetros del modelo se ha procedido, en primer lugar, a aplicar diversos modelos de predicción del esfuerzo en horas totales basados en una estimación del tamaño a partir de los Puntos Función de Albrecht determinados de acuerdo con lo propuesto en la metodología METRICA 3, estándar que es el que se emplea por la empresa objeto del estudio.

Puesto que la información de que se dispone sobre los proyectos del caso de estudio es información de coste, es decir, de esfuerzo, ha sido preciso en primer lugar elaborar una estimación de tamaño de una muestra de proyectos para luego ajustar ese resultado a un modelo de previsión.

E.1. Estimación del tamaño

Para ello se ha procedido de la forma siguiente:

- Selección de una muestra de proyectos de la base de datos.
- Análisis de los registros y reparación de aquellos que fueran contradictorios.
- Entrevistas a los Jefes de Proyecto para completar la información.
- Determinación "post mortem" del tamaño en Puntos Función.

E.1.1. Selección de la muestra

El tamaño de la base de datos, como se ha indicado en el capítulo 4 es muy importante. Por ello se optó en su momento por seleccionar una muestra para el ajuste de parámetros. Para la selección de la muestra se han seguido tres criterios:

Fiablilidad de la información registrada. La fiabilidad de los registros correspondientes a proyectos anteriores al año 2000 resulta muy baja dado que el sistema se implantó en 1999 y durante casi un año estuvo en pruebas. Por otra parte, el personal necesitó un cierto tiempo para habituarse a cargar la información periódicamente.

Proyectos terminados. En la fecha de realización del trabajo de campo, muchos proyectos estaban "abiertos", es decir, o no habían sido recepcionados por los clientes o estaban en fase de integración con otros sistemas de estos. Como ya se ha indicado, esta fase supone una carga de trabajo importante.

Importe económico. Aunque el número de proyectos es muy alto, aquellos que superan un cierto umbral económico son relativamente pocos, siendo en cambio su contribución a la facturación de la empresa muy importante. De hecho, los proyectos por importe superior a los 15.000 euros suponen un 90 % de la facturación total.

Con estos criterios, se obtuvo una muestra de 35 proyectos, todos ellos cerrados en el año 2005 y por un importe superior a la cifra citada. Estos proyectos eran, a pesar de todo, muy variados especialmente desde el punto de vista de la tecnología (lenguajes, sistemas, ...) lo que obligó a una posterior reducción de la muestra como más adelante se indica.

E.1.2. Validación de la información

La validación de la información se realizó a partir de la detección de contradicciones o vacíos y, posteriormente, entrevistando a los directores de

los proyectos. Para ello se han tratado los diferentes problemas aparecidos de la forma que se indica a continuación.

Problemas en los registros de tiempo

La tabla $T \times P$ presentaba solapes de fechas y espacios en blanco que hubo que corregir. En algunos casos se trataba de errores obvios (inclusión de festivos, error en los dígitos del año, ...) que se pudieron corregir de forma más o menos automática. En otros casos hubo que recurrir a los Jefes de los proyectos respectivos para corregir manualmente la información.

Contradicciones entre fechas

En ocasiones las fechas recogidas en la tabla $T \times P$ y las fechas de la tabla **proyecto** resultaban contradictorias: trabajo registrado antes del inicio formal del proyecto, después de la entrega, etc. En este caso la casuística era muy diversa, desde errores simples hasta diferencias debidas a problemas formales de la contratación (caso de trabajos previos) o simplemente a los trabajos posteriores a la entrega a los que ya se ha hecho referencia.

Contradicciones en la asignación de personal

En este caso, salvo algunos errores, lo que se producía era que había personas que habían estado trabajando en más de un proyecto durante el periodo registrado; en particular cuando obviando el procedimiento habitual llevaban demasiado tiempo sin registrar el trabajo realizado y "actualizaban en bloque" sus registros.

E.1.3. Estimación de Puntos Función

Para la estimación de los Puntos Función se construyó una hoja de cálculo que se ha ido completando con el análisis de los diferentes módulos funcionales de los proyectos a partir de la documentación de Casos de Uso. De la explotación de estas hojas de cálculo se obtuvo el valor de los Puntos Función No Ajustados (PFNA). Junto con dicha hoja de cálculo se cumplimentó para

cada proyecto un listado de los atributos a partir de cuya suma (SVA) se determina el Factor de Ajuste (FA) y, con éste, los Puntos Función Ajustados (PFA). La tabla E.1 recoge los valores obtenidos.

Proyecto	PFNA	SVA	FA	PFA
205	1334	32	0,97	1.293,98
195	572	25	0,9	514,8
240	1923	42	1,07	2.057,61
216	338	17	0,82	277,16
235	713	22	0,87	620,31
231	622	23	0,88	547,36
232	456	19	0,84	383,04
369	141	16	0,81	114,21
297	200	6	0,71	142
294	422	20	0,85	358,7
291	170	18	0,83	141,1
272	473	13	0,78	368,94
260	186	20	0,85	158,1
253	105	13	0,78	81,9
305	406	25	0,9	365,4
402	320	33	0,98	313,6
338	207	19	0,84	173,88
269	576	41	1,06	610,56
204	3648	3	0,68	2.480,64
237	282	32	0,97	273,54
283	428	19	0,84	359,52
266	1432	3	0,68	973,76
222	104	19	0,84	87,36
215	93	17	0,82	76,26

Cuadro E.1: Puntos Función de la muestra

E.2. Ajuste a un modelo de predicción

Para relacionar tamaño y esfuerzo se ha intentado construir un modelo de predicción mediante la regresión de los datos de esfuerzo sobre los de tamaño. El criterio para seleccionar entre las diferentes ecuaciones de regresión ha sido

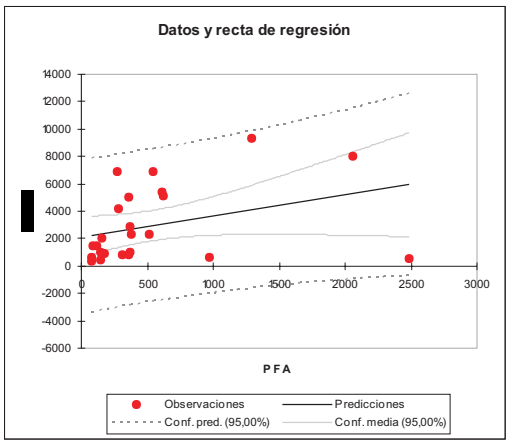


Figura E.1: Regresión Lineal Simple.

el del MMRE, PRED(0,25) [DF00].

E.2.1. Regresión lineal simple

Se han evaluado las siguientes alternativas:

Regresión lineal simple al conjunto de los datos

En primer lugar se ha aplicado un modelo simple de regresión lineal al conjunto de datos disponibles. De dicha aplicación se deduce que el modelo lineal no explica más que de manera muy poco eficiente los resultados obtenidos.

Regresión lineal simple limitada a tecnología Oracle Forms

A continuación se ha utilizado el mismo modelo de estimación pero sobre un conjunto de proyectos basados en una misma tecnología, Oracle Forms. En este caso la correlación encontrada es mucho más alta.

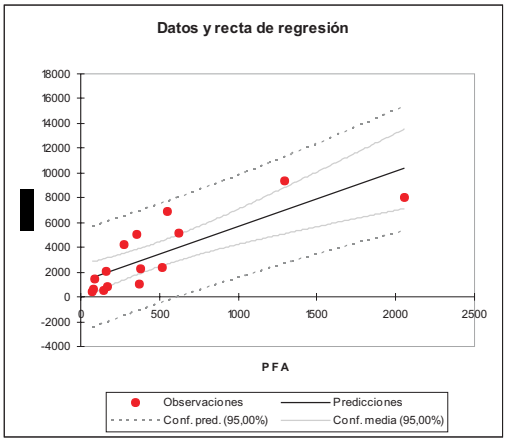


Figura E.2: RLS limitada a Oracle Forms.

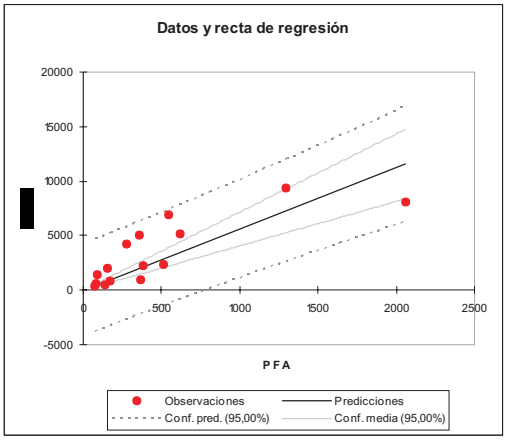


Figura E.3: RLS limitada forzando la ordenada en el origen.

R.L.S. limitada a tecnología Oracle Forms forzando la ordenada en el origen

Una variante sobre el procedimiento anterior consiste en forzar a la recta de regresión a pasar por el origen de coordinadas. El resultado se recoge en la figura E.3.

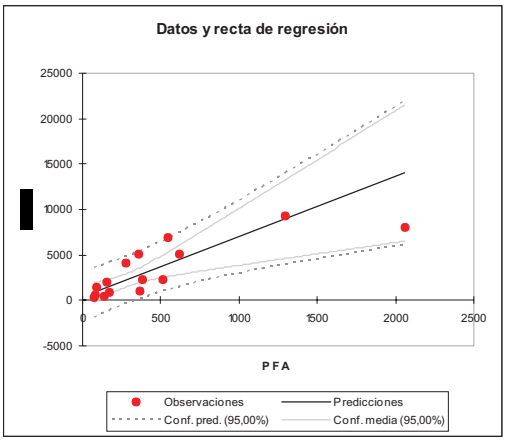


Figura E.4: RLS limitada forzando la ordenada en el origen y eliminando "outliers".

Regresión lineal simple limitada a tecnología y forzada al origen con eliminación de *outliers*

La capacidad predictiva así alcanzada es escasa con el modelo anterior por la dispersión de los puntos observados en torno a la recta de regresión. Para intentar paliar este problema se anulan las observaciones más descentradas.

E.2.2. Modelos de regresión no lineales

La aplicación de un modelo logarítmico o cuadrático apenas mejora la capacidad predictiva si nos atenemos a los indicadores citados, como indican las figuras E.5 y E.6.

E.2.3. Modelos de regresión múltiple

Alternativamente a los modelos propuestos, cabe la aplicación de modelos de estimación basadas en la regresión múltiple. Ello implica utilizar varios estimadores, en lugar de uno sólo.

Basados en los análisis realizados por los directores de proyecto y tomando como referencia el modelo que proponen El Emam *et al.* del Laboratorio de Ingeniería Empírica del *software* de Canadá [EB97], se ha estimado por

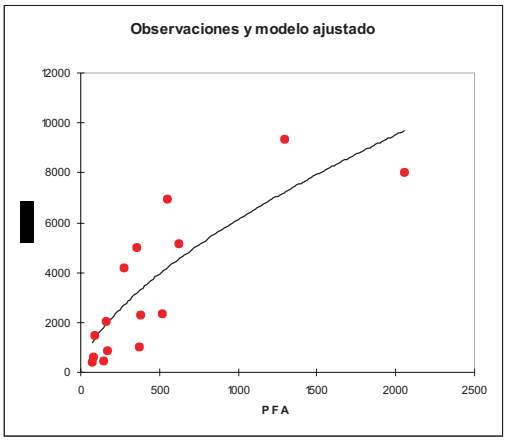


Figura E.5: Regresión logarítmica.

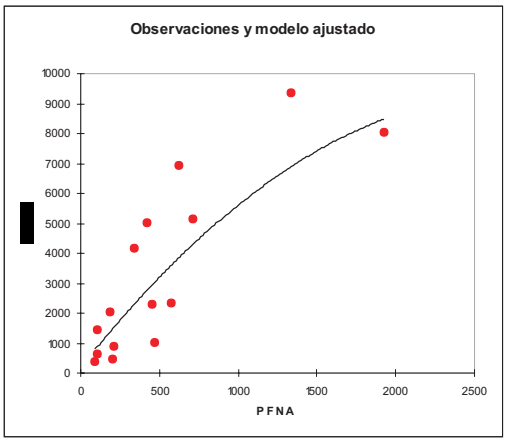


Figura E.6: Regresión cuadrática.

modelo	R^2	MMRE	PRED(0,25)
RLS	0.117	1.81	0.13
RLS Oracle Forms (OF)	0.655	0.91	0.2
RLS-OF forzando origen		0.43	0.33
RLS-OF sin "outliers"		0.64	0.13
R. logarítmica (RLog)	0.724	0.81	0.33
R. cuadrática	0.676	0.64	0.13
RLog múltiple	0.811	0.39	0.29
RLog múltiple sin extremos	0.948	0.14	0.82

Cuadro E.2: Resultados de los diferentes modelos empleados.

regresión lineal logarítmica múltiple con dos estimadores, los Puntos Función no Ajustados (PFNA) y el factor de complejidad técnica estimado por los directores.

Este modelo de estimación arroja mejores resultados que los anteriores en términos de R^2, aunque los indicadores de la bondad de la predicción aún son insuficientes. Sin embargo, si se omiten los proyectos extremos, en términos de PFNA, se obtiene un resultado apreciablemente mejor como muestran el cuadro E.2 y la figura E.7.

La figura E.8 representa los residuos normalizados de la regresión logarítmica múltiple definitiva. La ecuación resultante es:

$$Y = 0,420 + 1,066 \times X_1 + 1,188 \times X_2$$

Donde X_1 es el logaritmo del tamaño en PFNA, X_2 es el de un factor de dificultad estimado entre 1 y 5 e Y es el logaritmo de la carga total en horas.

El resultado es, pues, un modelo exponencial con un factor amplificador relacionado con la *complejidad del proceso lógico interno de la aplicación* tal y como se define en el método de Albrecht:

$$e = K_1 K_{complejidad} t^{1,066}$$

Donde e y t son, respectivamente, el esfuerzo en horas y el tamaño en PFNA (ver E.9).

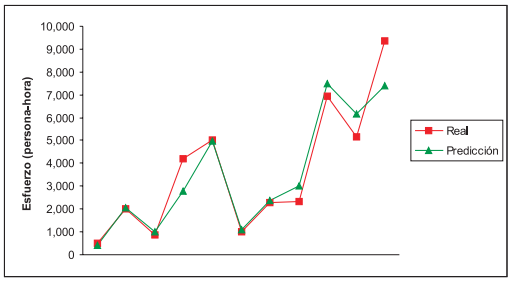


Figura E.7: Valores reales y predichos por la regresión logarítmica múltiple.

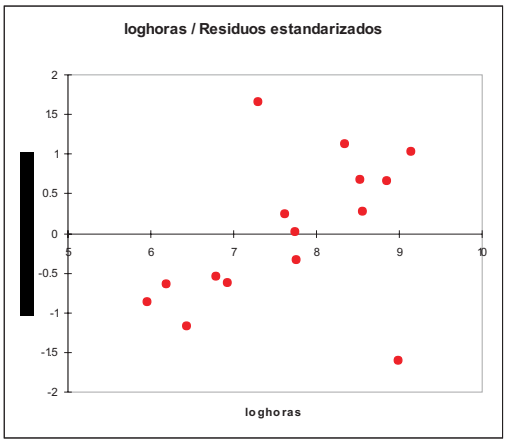


Figura E.8: Residuos estandarizados de la regresión logarítmica múltiple.

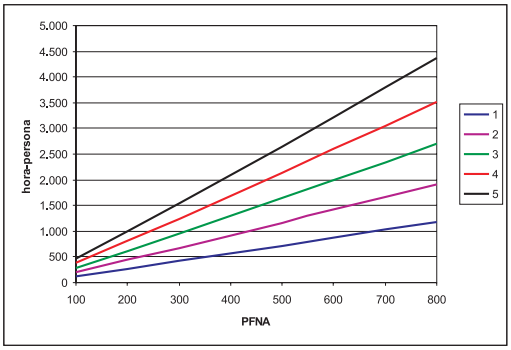


Figura E.9: Esfuerzo estimado frente a tamaño para diferentes grados de complejidad

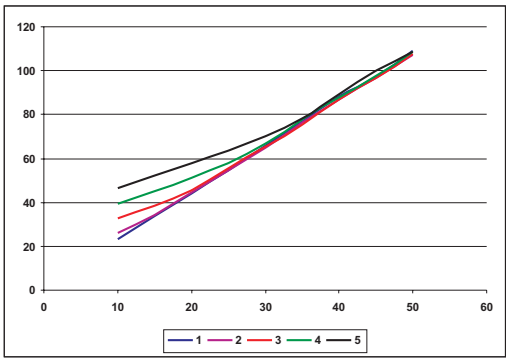


Figura E.10: Esfuerzo frente a tamaño en un conjunto de paquetes simulados.

E.3. Estimación de parámetros para el modelo

La estimación de parámetros para el modelo se ha realizado a partir de la simulación de diferentes tamaños de paquete, medidos en unidades ficticias (tareas), con diferentes dotaciones de recursos en la proporción habitual de los proyectos de la muestra, que viene ser de un analista por cada dos programadores.

El gráfico E.10 muestra el resultado de la simulación para paquetes entre 10 y 50 tareas y de 1 a 5 analistas. Ajustando a una los resultados de la simulación de los paquetes se obtiene un resultado con una correlación muy alta ($R^2 > 0,99$) pero con un exponente ligeramente superior al obtenido al ajustar la muestra (entre 1,13 y 1,21 frente a 1,066). Las discrepancias detectadas pueden deberse a diversas causas:

- El que el exponente resultante sea ligeramente mayor que el obtenido del ajuste de los datos reales se puede interpretar teniendo en cuenta que la ejecución de un proyecto completo es parcialmente lineal, en la medida en que está formado por diversos paquetes por lo que el esfuerzo al final es la suma algebraica del esfuerzo parcial en varios paquetes, éste sí exponencial[1]

[1]En un proyecto de tamaño t descompuesto en dos paquetes de tamaños t_1 y t_2 tales que $t = t_1 + t_2$ se cumple que $t^\alpha \geq t_1^\alpha + t_2^\alpha$ para todo $\alpha > 1$.

- Un aumento del término independiente con el tamaño del equipo recuerda en cierta medida a la "Ley de Brooks" en la medida en que un aumento del tamaño del equipo de desarrollo puede dar lugar a una productividad menor sin necesidad de modelar explícitamente las sobrecargas de comunicación o los efectos aprendizaje, probablemente por la forma en que se ha implementado el mecanismo de asignación interna dentro de los paquetes.

- Por último, todos estos resultados deben relativizarse en la medida en que el modelo de simulación tiene un comportamiento básicamente discreto, mientras que los ajustes se están haciendo empleando funciones continuas.

Teniendo en cuenta las consideraciones anteriores, se han ajustado los datos a partir de los promedios de las diferentes simulaciones de la forma siguiente:

- Se determina la *productividad aparente* de los datos simulados y de los estimados para la muestra.

- Se igualan dichos valores.

- Se determinan, a partir de ellos, los valores de la productividad en las fases de los paquetes de analistas y programadores, manteniendo una proporción constante entre ambos tipos de recursos.

Con estos datos se establece la proporción entre Punto Función No Ajustados, PFNA, y la "unidad" de tamaño empleada en el modelo, la *tarea*, en función del *factor de complejidad*, resultando los valores que aparecen en el cuadro E.3.

Complejidad	PFNA/tarea
1	1,51
2	2,44
3	3,44
4	4,48
5	5,57

Cuadro E.3: Relación entre PFNA y tareas, en función de la complejidad

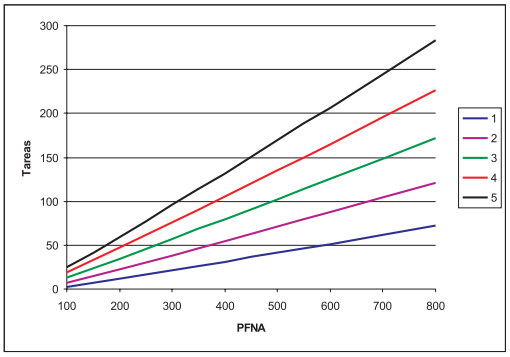


Figura E.11: Tareas frente a Puntos Función para diferentes grados de complejidad

Apéndice F

Algoritmos implementados

F.1. Ventanas de tiempo

Un punto de partida para una solución heurística de los problemas vistos anteriormente es limitar el espacio de exploración garantizando la factibilidad temporal de las secuencias generadas y, a partir de ahí, buscar estrategias de asignación de recursos que cumplan las restricciones que estos imponen. Una técnica para hacerlo se basa en los métodos de la Inteligencia Artificial para tratar los problemas CSP (*constaint satisficing problems*).

El mantenimiento de ventanas de tiempo parte de la definición de un proyecto como una n-tupla de Actividades $\{A_1, \ldots, A_n\}$ *ordenadas topológicamente*. Cada actividad, A_i, tiene un *tiempo de inicio*, $start_{A_i}$, y *un tiempo de finalización*, $finish_{A_i}$.

Las restricciones pueden ser de dos tipos:

Restricciones unarias . Fijan límites superiores e inferiores a $start_{A_i}$ y $finish_{A_i}$. Son, respectivamente:

- es_{A_i}, fecha más temprana de inicio.
- ls_{A_i}, fecha más tardía de inicio.
- ef_{A_i}, fecha más temprana de finalización.
- lf_{A_i}, fecha más tardía de finalización (*deadline*).

Restricciones binarias . Fijan relaciones entre dos Actividades, A_i y A_j. Esas relaciones pueden ser:

- *Start-Start* (SS).
- *Finish-Finish* (FF).
- *Start-Finish* (SF).
- *Finish-Start* (FS).

La relación binaria relevante está definida por un *intervalo*, lag_{A_i,A_j}, de manera que la fecha relevante de A_j tiene que ser mayor o igual que la de A_i más lag_{A_i,A_j}.

A partir de lo anterior se define una *ventana de tiempo* de una actividad A_i como una 5-tupla de valores de tiempo:

$$(hes_{A_i}, ses_{A_i}, start_{A_i}, sls_{A_i}, hls_{A_i})$$

cuyos elementos definen respectivamente:

- $start_{A_i}$, fecha de inicio; que será la fecha de inicio de la actividad si está secuenciada o *default-time* por defecto
- $[hes_{A_i}, hls_{A_i}]$, ventana fuerte de tiempo; que representa el intervalo de valores de t en los que A_i tiene una fecha de inicio factible; no varía en la secuenciación
- $[ses_{A_i}, sls_{A_i}]$, ventana débil de tiempo;, contiene los valores de t para los que hay secuencias parciales compatibles de los sucesores de A_i si esta comienza en t; es, pues,un intervalo acotado por la ventana dura.

F.2. Mantenimiento de ventanas

El mantenimiento de las ventanas de tiempo consiste en construir, deconstruir y volver a construir las ventanas aplicando un algoritmo de propagación de restricciones. En el caso que nos ocupa es relativamente sencillo puesto

que las restricciones que interesan, aparte de las relaciones de precedencia, son las fechas límite bien del proyecto bien de paquetes parciales, que se incorporan directamente como límite superior de la ventana "fuerte".

Los límites débiles son simplemente para cada actividad los extremos clásicos del algoritmo del CPM (*critical path method* recalculados cada vez que se secuencia una actividad. Como una red de proyectos-tiempo es un problema sub-determinado, la fijación de una fecha determinada de inicio para una actividad permite determinar las fechas más temprana y tardía de comienzo y final de todas las demás por medio de un algoritmo muy sencillo de doble pasada a través de la obtención del camino crítico desde el nodo origen al nodo del que parte la actividad objeto de la secuenciación y desde éste al nodo final.

Una forma eficiente de hacerlo es utilizar una variante del algoritmo de Floyd-Warshall [AMO93] para calcular, en este caso, la separación entre los inicios de las actividades como un problema de distancia máxima[1]. El resultado es una matriz que se emplea para actualizar las ventanas de tiempo en la secuenciación y desecuenciación. Para ello se construye dicha matriz *camino* de dimensión $n \times n$ (siendo n el número de actividades).

En el caso que nos interesa la distancia temporal entre actividades depende del modo elegido de ejecución (recursos empleados por las mismas), lo cual daría lugar a tantas matrices como combinaciones posibles de modos. Para evitar esto se recurre a tomar una matriz *camino* inicial calculada con las duraciones máximas de cada actividad y se actualiza cada vez que se secuencia una de ellas determinando su modo.

[1]El algoritmo citado está pensado para calcular la distancia mínima.

```
1: for i, j = 1, ... n do
2:    if i es predecesora inmediata de j then
3:       c(i, j) ← d_i
4:    else
5:       c(i, j) ← 0
6:    end if
7:    for k = 1, ... n − 1 do
8:       c(i, j) = máx(c(i, j), c(i, k) + c(k, j))
9:    end for
10:   return camino(1 : n, 1 : n)
```

F.3. Construcción de secuencias

Una secuencia o solución $S = [S_0, S_1, \ldots, S_n]$ es un vector de $n+1$ fechas s_i de inicio o de final f_i de las *n+1* actividades del proyecto, incluyendo las actividades ficticias inicial, 0, y final, n. Se denomina longitud o duración de la secuencia, *T(S)*, al plazo de compleción del proyecto si se sigue esa secuencia, es decir, a la fecha de compleción de la actividad final. Una *secuencia factible* es aquella que respeta las restricciones de tiempo, precedencia y recursos. Dada una secuencia S, se denominan actividades *activas* en un momento t al conjunto *A(t)* de actividades que están ejecutándose en ese momento de seguirse esa secuencia.

Para la construcción de secuencias se utilizan los denominados *esquemas de generación se secuencias* [Bal03]. Los esquemas de generación de secuencias se agrupan básicamente en dos categorías, *serie* y *paralelo*. Las primeras parten de una secuencia vacía y van iterando *incorporando actividades*. Las segundas, partiendo también de una secuencia vacía realizan una *iteración temporal*.

El modelo aquí elegido corresponde a un esquema en serie, y se realiza en tantas etapas como actividades hay que secuenciar. En cada etapa de secuenciación se diferencian dos conjuntos; S, actividades ya secuenciadas yE, conjunto de actividades elegibles, es decir actividades que aun no han sido secuenciadas y cuyas predecesoras ya lo han sido. Igualmente se determina en cada momento t la disponibilidad de recursos, es decir, la cantidad de recursos que no está siendo usada por las actividades activas en ese momento. Por último se determinan F, el vector de los tiempos de finalización de las actividades comprendidas en S y M, vector de modos de las mismas.

El algoritmo SGS (*serial generation scheme*) procede como sigue:

1: $S = \{\}, F = \{\}, M = \{\}$ {Inicialización}
2: **for** $i = 1 : n - 1$ **do**
3: `DeterminarElegibles` {devuelve E, actividades elegibles}
4: `SeleccionarPrioridad` {devuelve $j \in E$, actividad con mayor prioridad}
5: `FijarInicio` {devuelve esp_j fecha más temprana de inicio, $modo_j$ y d_j duración}
6: $s_j \leftarrow \text{máx}\, es_j$
7: `ActualizarRecursos` {devuelve D actualizado}
8: $S \leftarrow S \cup j$ {Incluye la actividad en las ya secuenciadas}
9: **end for**
10: $S \leftarrow S \cup n$ {Incluye la última actividad (ficticia) en las ya secuenciadas}
11: $s_n \leftarrow \text{máx}(s_i)$
12: $T \leftarrow s_n$
13: **return** $s(1 : n), F, M$

F.4. Determinación de las prioridades iniciales

Para determinar las prioridades iniciales utilizando la regla RWK (*remaining workload* basta aplicar el algoritmo de Floyd-Warshall descrito antes tomando como peso de los arcos la carga de trabajo de cada actividad.

F.5. Determinación de las actividades elegibles

Para determinar las actividades elegibles basta verificar las ya secuenciadas:

```
1: E = {}
2: for i = 1 : n − 1 do
3:    if i ∈ S then
4:       for j = i + 1 : n − 1 do
5:          if i ∈ Pred(j) then
6:             E ← E ∪ j
7:          end if
8:       end for
9:    end if
10: end for
11: return E
```

Donde $Pred(j)$ es un vector con las actividades predecesoras de la actividad j.

F.6. Selección del modo y fecha de inicio

La regla para asignar el modo de ejecución es la EFFT (*earliest feasible finsh time*) [LTB06] que asigna a la actividad a secuenciar el máximo de recursos disponibles (indicado por D).

1: $modo \leftarrow modo(D, j)$
2: $inicio \leftarrow inicio(D, j)$
3: $duracion \leftarrow duracion(j, modo)$
4: $es_j \leftarrow inicio$
5: $d_j \leftarrow duracion$
6: $modo_j \leftarrow modo$
7: $M \leftarrow M \cup modo_j$
8: $F \leftarrow F \cup es_j + d_j$
9: **return** es_j, d_j

La selección de modo y, por tanto, la duración de la actividad dependen de D y de la propia actividad. La fecha más temprana de inicio tiene que ser compatible con D. Los procedimientos por los que se determinan son triviales.

F.7. Control del uso de recursos

D es una matriz con dos columnas que registra en la primera los recursos empleados a partir del intervalo indicado en la segunda. Se van agregando filas cada vez que se inicia o concluye una actividad, reduciéndose luego en el caso de que coincidan más de un evento en el mismo intervalo de tiempo. Para ello se emplea la matriz S de actividades secuenciadas, y los vectores $[s_i, i \in S]$ y F de las fechas de inicio y final de las mismas.

1: **for all** $i \in S$ **do**
2: $recursos \leftarrow recursos\,(modo_i)$
3: **if** $NoExiste\,[D(s_i)]$ **then**
4: $D \leftarrow D \cup [s_i, 0]$
5: **end if**
6: **if** $NoExiste\,[D(f_i)]$ **then**
7: $D \leftarrow D \cup [f_i, 0]$
8: **end if**
9: **for** $t = s_i : t_i$ **do**
10: $D\,(t, 2) \leftarrow D\,(t, 2) + recursos$
11: **end for**
12: **end for**
13: **return** D

F.8. Algoritmo SWO *squeaky wheel optimization*

El algoritmo SWO se emplea tanto para buscar un óptimo inicial como para reconstruir una secuencia en la que se ha producido una disrupción. El problema real que se intenta representar es el de la detección de un retraso excesivo sobre la programación, a consecuencia de un imprevisto, un error en un paquete o la asignación de personal a otro proyecto. En tal caso lo que se persigue es reparar una secuencia reasignando recursos dentro de un mismo proyecto si es posible. De lo contrario, detectar la imposibilidad de hacerlo para emitir una petición de recursos adicionales al nivel táctico.

La solución propuesta recuerda a la metodología de *agentes* en el sentido de que se simula una solución circunscrita a un proyecto, cuyo asignador actúa como un agente autónomo. Para ello, y teniendo en cuenta que la respuesta clásica de un jefe de proyecto a una eventualidad de este tipo es la de reasignar personal de unas tareas a otras para atender a la más crítica, se ha implementado un algoritmo del tipo SWO (*squeaky wheel optimizaton*)[JC99].

La filosofía del algoritmo es, de manera análoga al agente humano, modificar las prioridades que dieron lugar a la primera asignación para el incremento sucesivo de recursos para la actividad más prioritaria.

El punto de partida es un paquete que se retrasa lo que se traduce en fijar una prioridad mayor para ese paquete que la mayor de todas las prioridades calculadas con la regla general. Se reprograma todo el proyecto priorizando dicha actividad. El resto se programan fijando como prioridad el límite fuerte hls_i más bajo sin excepción.

El algoritmo de reparación se desenvuelve en los siguientes pasos:

1: $contador \leftarrow 1$
2: $DuracionOptima \leftarrow \infty$
3: `FijaCotaInferiorHorizonte`
4: $(P_1, P_2, \ldots, P_n)$ = prioridades iniciales
5: **for** $contador = 1$ to max_iter **do**
6: $factible =$ **true**
7: **for** $i = 1$ to n **do**
8: $A_i \leftarrow$ actividad no programada con P_i mayor
9: **if** `Secuenciar`(A_i)falla **then**
10: $factible =$**false**
11: **end if**
12: **if** $factible$ **then**
13: $DuracionOptima \leftarrow Duracion$ `ReducirHorizonte`
14: **else if** 10 iteraciones sin encontrar una solución factible **then** `IncrementarHorizonte`
15: **end if**
16: **end for**
17: **end for**

El método `Secuenciar`, a su vez se limita a asignar una fecha inicial al paquete. Al hacerlo, actualiza las ventanas de tiempo y el perfil de ocupación de los recursos. Si encuentra una fecha de inicio dentro de su ventana débil de tiempo compatible con los recursos disponibles se secuencia en ese momento y devuelve un mensaje VERDADERO. De lo contrario, se secuencia al límite inferior de su ventana y se devuelve FALSO. De este modo se mantiene una secuenciación que es factible en términos de tiempo con las duraciones actuales pero no necesariamente en términos de recursos.

Si el problema estriba en un paquete retrasado, y éste sigue sin ser factible, se mantiene su prioridad. Si se consigue secuenciarlo en forma factible pero otros han dejado de serlo en términos de recursos, se incrementa la prioridad de estos en una cantidad constante y se modifican aleatoria-

mente los demás en una pequeña cantidad.

Bibliografía

[AA99] K. Ananthanarayanan and P. J. Abhilash. Resource constrained scheduling techniques for multiple projects - a review. *Journal of the Institution of Engineers (India): Civil Engineering Division*, 80(1):33–36, 1999.

[Abd93a] Tarek K. AbdelHamid. Modeling the dynamics of software reuse: An integrating system dynamics perspective. In *Proceedings of the Sixth Workshop on Institutionalizing Software Reuse*, 1993.

[Abd93b] Tarek K. AbdelHamid. A multiproject perspective of single-project dynamics. *Journal of Systems and Software*, 22(3):151–165, sep 1993.

[ABG+05] Jay April, Marco Better, Fred Glover, James Kelly, and Manuel Laguna. Enhancing business process management with simulation optimization. *Proceedings of the 2006 Winter Simulation Conference*, page 642, 2005.

[ABH05a] B. C. D. Anda, H. C. Benestad, and S. E. Hove. A multiple-case study of effort estimation based on use case points. In *ISESE'2005 (Fourth International Symposium on Empirical Software Engineering)*, pages 407–416. IEEE Computer Society, Noosa, Australia, November 17-18, 2005.

[ABH+05b] O. Armbrust, T. Berlage, T. Hanne, P. Lang, Jürgen Münch, H.Ñeu, S.Ñickel, I. Rus, A. Sarishvili, S. van Stockum, and A. Wirsen. *Simulation-based software process modeling and evaluation*, volume 3. 2005.

[ABZK02] A. V. Arutyunov, V.Ñ. Burkov, A. Yu Zalozhnev, and D. Yu Karamzin. A problem of optimal distribution of resources over a setof independent operations. *Translated from Automatika i Telemekhanika*, (5):108–119, 2002.

[ACC+04] Lerina Aversano, Gerardo Canfora, Giovanni Capasso, Giuseppe A. Di Lucca, and Corrado A. Visaggio. Introducing quality system in small and medium enterprises: An experience report. *Springer-Verlag Berlin Heidelberg*, 2004.

[ACLP04] G. Antoniol, A. Cimitile, G. A. Di Lucca, and M. Di Penta. Assessing staffing needs for a software maintenance project through queuing simulation. *IEEE Transactions on Software Engineering*, 30(1):43–58, 2004.

[ADH+01] Klaus-Dieter Althoff, Björn Decker, Susanne Hartkopf, Andreas Jedlitschka, Markus Nick, and Jörg Rech. Experience Management: The Fraunhofer IESE Experience Factory. Technical Report IESE-Report No. 035.01/E, 2001.

[AFT02] Jarmo J. Ahonen, Marko Forsell, and Sanna-Kaisa Taskinen. A modest but practical software process modeling technique for software process improvement. *Software Process Improvement and Practice*, 7:33–44, 2002.

[AG08] Amir Azaron and S. M. T. Fatemi Ghomi. Lower bound for the mean project completion time in dynamic PERT networks. *European Journal of Operational Research*, 186(1):120–127, 4/1 2008.

[AH01] Rainer Austen and Tracy Hall. An analysis of some core studies of software process improvement. *Software Process: Improvement and Practice*, 6(4):169–187, 2001.

[AHT07] Monique Aubry, Brian Hobbs, and Denis Thuillier. A new framework for understanding organisational project management through the PMO. *International Journal of Project Management*, 25(4):328–336, 5 2007.

[AIG03] S. Anavi-Isakow and B. Golany. Managing multi-project environments through constant work-in-process. *International Journal of Project Management*, 21(1):9–18, 1 2003.

[AJ04] Silvia T. Acuña and Natalia Juristo. Assigning people to roles in software projects. *Software-Practice and Experience*, 2004.

[Akp00] E. O. P. Akpan. Priority rules in project scheduling: A case for random activity selection. *Production Planning and Control*, 11(2):165–170, 2000.

[ALM+05] Haldun Aytug, Mark A. Lawley, Kenneth McKay, Shantha Mohan, and Reha Uzsoy. Executing production schedules in the face of uncertainties: A review and some future directions. *European Journal of Operational Research*, 161(1):86–110, 2/16 2005.

[Alt01] Steven Alter. Are the fundamental concepts of information systems mostly about work systems? *Communications of the Association for Information Systems*, 5(11), 2001.

[AM91] Tarek K. AbdelHamid and S. E. Madnick. *Software Project Dynamics An Integrated Approach*. Prentice Hall, Englewood Cliffs NJ, 1991.

[AM01] J. Alcaraz and C. Maroto. A robust genetic algorithm for resource allocation in project scheduling. *Annals of Operations Research*, 102(1-4):83–109, 2001.

[AM02] M. A. Ammar and Y. A. Mohieldin. Resource constrained project scheduling using simulation. *Construction Management and Economics*, 20(4):323–330, 2002.

[AM07] I. Allison and Y. Merali. Software process improvement as emergent change: A structurational analysis. *Information and Software Technology*, 49(6):668–681, 6 2007.

[AMNS95] Paul S. Adler, Avi Mandelbaum, Viên Nguyen, and Elizabeth Schwerer. From project to process management: An empirically-based framework for analyzing product development time. *Management Science*, 41(3):458, 1995.

[AMO93] Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network Flows : theory, algorithms, and applications*. Prentice Hall, Englewood Cliffs, N.J, 1993.

[AMR03] Christian Artigues, Philippe Michelon, and Stéphane Reusser. Insertion techniques for static and dynamic resource-constrained project scheduling. *European Journal of Operational Research*, 149(2):249–267, 9/1 2003.

[A.P03] Shibanov A.P. Finding the distribution density of the time taken to fulfill the gert network on the basis of equivalent simplifying transformations. *Automation and Remote Control*, 64:279–287(9), 2003.

[AP05] N. Angkasaputra and D. Pfahl. Towards an agile development method of software process simulation. Technical report, 2005.

[APH04] G. Antoniol, M. Di Penta, and M. Harman. A robust search-based approach to project management in the presence of abandonment, rework, error and uncertainty. In *Proceedings - 10th International Symposium on Software Metrics, METRICS 2004*, pages 172–183, 14 September 2004 through 16 September 2004 2004.

[APR07] Ahmed AlEmran, Dietmar Pfahl, and Gnther Ruhe. Dynarep: A discrete event simulation model for re-planning of software releases. In *ICSP*, pages 246–258, 2007.

[AR00] Christian Artigues and François Roubellat. A polynomial activity insertion algorithm in a multi-resource schedule with cumulative constraints and multiple modes. *European Journal of Operational Research*, 127(2):297–316, 12/1 2000.

[ARRRR02] Jesús S. Aguilar-Ruiz, José C. Riquelme, Daniel Rodríguez, and Isabel Ramos. Generation of management rules through system dynamics and evolutionary computation. *Springer-Verlag Berlin Heidelberg*, pages 615–628, 2002.

[ARRRT01] Jesús S. Aguilar-Ruiz, Isabel Ramos, José C. Riquelme, and Miguel Toro. An evolutionary approach to estimating software development projects. *Information and Software Technology*, 43(14):875–882, 12/15 2001.

[ASA06] Babak Abbasi, Shahram Shadrokh, and Jamal Arkat. Bi-objective resource-constrained project scheduling with robustness and makespan criteria. *Applied Mathematics and Computation*, 180(1):146–152, 9/1 2006.

[ASD99] R. Ash and D. E. Smith-Daniels. The effects of learning, forgetting, and relearning on decision rule performance in multiproject scheduling. *Decision Sciences*, 30(1):47–79, 1999.

[Ass08] Modelica Association. Modelica, 2008.

[AT93] Javier Aracil and Miguel Toro. *Métodos cualitativos en dinámica de sistemas*. Sevilla. Universidad de Sevilla. Secretariado de Publicaciones , 11/1993 1993.

[AT08] Osman Alp and Tarkan Tan. Tactical capacity management under capacity flexibility. *IIE Transactions*, 40(3):221, 2008.

[Aue04] M. Auer. Increasing the accuracy and reliability of analogy-based cost estimation with extensive project feature dimension weighting. In *Proceedings. 2004 International Symposium on Empirical Software Engineering, 2004. ISESE '04.*, pages 147–155, 2004.

[Bal03] Francisco Ballestín. *Nuevos métodos de resolución del problema de secuenciación de proyectos con recursos limitados*. PhD thesis, Universidad de Valencia, 2003.

[Bal07] Francisco Ballestín. When it is worthwhile to work with the stochastic RCPSP? *Journal of Scheduling*, 10(3):153–166, 6 2007.

[BD04] Oddur Benediktsson and Darren Dalcher. New insights into effort estimation for incremental software development projects. *Project Management Journal*, 35(2):5–12, 2004.

[BD07] Jeroen Beliën and Erik Demeulemeester. On the trade-off between staff-decomposed and activity-decomposed column generation for a staff scheduling problem. *Annals of Operations Research*, 155(1):143–166, 11/22 2007.

[BDM+99] Peter Brucker, Andreas Drexl, Rolf Möhring, Klaus Neumann, and Erwin Pesch. Resource-constrained project scheduling: Notation, classification, models, and methods. *European Journal of Operational Research*, 112(1):3–41, 1/1 1999.

[BdOBW08] Milton Barreto, Marcio de O. Barros, and Claudia M. L. Werner. Staffing a software project: A constraint satisfaction and optimization-based approach. *Computers & Operations Research*, In Press, Corrected Proof:2153, 2008.

[BDR02] C. Briand, E. Despontin, and François Roubellat. Scheduling with time lags and preferences: a heuristic. In *8th Workshop on Project Management and Scheduling.*, Valencia., 2002.

[Bel72] Richard E. Bellman. *Dynamic Programming*. Princeton University Press, 6 edition, 1972.

[Ben03] O. Benediktsson. Effort estimation in incremental software development, 2003.

[BF04] V. Bernier and Y. Frein. Local scheduling problems submitted to global fifo processing constraints. *International Journal of Production Research*, 42(8):1483–1503, 04/15 2004.

[BHR03] Sarah Beecham, Tracy Hall, and Austen Rainer. Software process improvement problems in twelve software companies: An

empirical analysis. *Empirical Software Engineering*, 8:7–42, 2003.

[BJ07] Barry Boehm and Apurva Jain. Developing a process framework using principles of value-based software engineering. *Software Process: Improvement and Practice*, 12(5):377–385, 2007.

[BJN+98] C. Barnhart, E. L. Johnson, G. L. Nemhauser, M. W. P. Savelsbergh, and P. H. Vance. Branch-and-price: Column generation for solving huge integer programs. *Operations Research*, 46(3):316–329, 1998.

[BkN02] Ulrike Becker-kornstaedt and Holger Neu. Lecture notes in computer science 1 learning and understanding a software process through simulation of its underlying model, 2002.

[BKS04] Akhilesh Bajaj, Sunder Kekre, and Kannan Srinivasan. Managing NPD: Cost and schedule performance in design and manufacturing. *Management Science*, 50(4):527–536, April 1 2004.

[BL77] José María Bueno Lidón. *Dinámica de sistemas y planificación regional : planteamiento, desarrollo y explotación de un modelo demográfico regional*. PhD thesis, 1977.

[BL03] K. Bouleimen and H. Lecocq. A new efficient simulated annealing algorithm for the resource-constrained project scheduling problem and its multiple mode version. *European Journal of Operational Research*, 149(2):268–281, 9/1 2003.

[BMM07] Antonia Bertolino, Eda Marchetti, and Raffaela Mirandola. Performance measures for supporting project manager decisions. *Software Process: Improvement and Practice*, 12(2):141–164, 2007.

[BN99] VladimirÑ. Burkov and Dmitri A. Novikov. Models and methods of multiprojects management. *Systems Science*, 256(2):5–14, 1999.

[Boc96] F. F. Boctor. Resource-constrained project scheduling by simulated annealing. *International Journal of Production Research*, 34(8):2335–2351, 1996.

[Boe98] R. De Boer. *Resource-constrained multi-project management*. PhD thesis, University of Twente, Enschede, The Netherlands., 1998.

[Boe01] B. Boehm. Top 10 list [software development]. *Computer*, 34(1):135–137, 2001.

[Bro02] Tyson R. Browning. Process integration using the design structure matrix. *Systems Engineering*, 5(3):180, 2002.

[BS99] R. De Boer and J. M. J. Schutten. *Multi-project rough-cut capacity planning*, pages 631–644. Begell House, Wallingford, , 1999.

[BS04] M. Barut and V. Sridharan. Design and evaluation of a dynamic capacity apportionment procedure. *European Journal of Operational Research*, 155(1):112–133, 5/16 2004.

[BWT02] Marcio Oliveira Barros, Claudia Maria Lima Werner, and Guilherme Horta Travassos. A system dynamics metamodel for software process modeling. *Software Process: Improvement and Practice*, 7(3-4):161–172, 2002.

[BY06] Tyson R. Browning and Ali A. Yassine. Resource-constrained multi-project scheduling: Priority rule performance revisited. Technical report, Texas Chritian University, 2006.

[BZ06] Z. A. Banaszak and M. B. Zaremba. Project-driven planning and scheduling support for virtual manufacturing. *Journal of Intelligent Manufacturing*, 17(6):641–651, 2006.

[Car03] Mercedes Ruiz Carreira. *Modelado y simulación para la mejora de los procesos software*. PhD thesis, Universidad de Sevilla, 2003.

[CBK06] KeungSik Choi, DooHwan Bae, and TagGon Kim. An approach to a hybrid software process simulation using the devs formalism. *Software Process: Improvement and Practice*, 11(4):373–383, 2006.

[CC00] Carl K. Chang and Mark Christensen. A net practice for software project management. *IEEE Software*, page 80, 2000.

[CDM+02] Kevin M. Calhoun, Richard F. Deckro, James T. Moore, James W. Chrissis, and John C. Van Hove. Planning and re-planning in project and production scheduling. *Omega*, 30(3):155–170, 6 2002.

[CdOBT07] H. R. Costa, Marcio de O. Barros, and Guilherme H. Travassos. Evaluating software project portfolio risks. *Journal of Systems and Software*, 80(1):16–31, 1 2007.

[CDSC05] Rodrigo Cern, Juan C. Dueñas, Enrique Serrano, and Rafael Capilla. A meta-model for requirements engineering in system family context for software process improvement using CMMI. In Frank Bomarius and Seija Komi-Sirviø, editors, *PROFES; Product Focused Software Process Improvement, 6th International Conference, PROFES 2005, Oulu, Finland, June 13-15, 2005, Proceedings; Lecture Notes in Computer Science*, volume 3547, pages 173–188. Springer, 2005.

[CE05] Soo-Haeng Cho and Steven D. Eppinger. A simulation-based process model for managing complex design projects. *IEEE Transactions on Engineering Management*, 52(3):316, 2005.

[CF02] Reidar Conradi and Alfonso Fuggetta. Improving software process improvement. *IEEE Software*, page 2, 2002.

[CFR04] L. Caccetta, L. R. Foulds, and V. G. Rumchev. A positive linear discrete-time model of capacity planning and its controllability properties. *Mathematical and Computer Modelling*, 40(1-2):217–226, 7 2004.

[CGC06] Yu Chen, Gerard C. Gannod, and J. S. Collofello. A software product line process simulator. *Software Process: Improvement and Practice*, 11(4):385–409, 2006.

[CGnSGR06] Juan J. Cuadrado-Gallego, Miguel Ángel Sicilia, Miguel Garre, and Daniel Rodríguez. An empirical study of process-related attributes in segmented software cost-estimation relationships. *Journal of Systems and Software*, 79(3):353–361, 3 2006.

[CGR07] Giuseppe Confessore, Stefano Giordani, and Silvia Rismondo. A market-based multi-agent system model for decentralized multi-project scheduling. *Annals of Operations Research*, 150(1):115–135, 03/08 2007.

[CH08] Haadi Chtourou and Mohamed Haouari. A two-stage-priority-rule-based algorithm for robust resource-constrained project scheduling. *Computers & Industrial Engineering*, In Press, Corrected Proof:154, 2008.

[Che05] Z. Chen. Finding the right data for software cost modeling. *Software, IEEE*, 22(6):38–46, 2005.

[Che07] Shih-Pin Chen. Analysis of critical paths in a project network with fuzzy activity times. *European Journal of Operational Research*, 183(1):442–459, 11/16 2007.

[Cho02] Shih-Chien Chou. Proactnet: Modeling processes through activity networks. *International Journal of Software Engineering and Knowledge Engineering*, 12(5):545–580, 2002.

[Cho05] Soo-Haeng Cho. A simulation-based process model for managing complex design projects. *IEEE Transactions on Engineering Management*, 52(3):316–328, 2005.

[Chr99] Alan M. Christie. Simulation in support of CMM-based process improvement. *Journal of Systems and Software*, 46(2-3):107–112, 4/15 1999.

[Cio05] D. F. Cioffi. Completing projects according to plans: an earned-value improvement index. *The Journal of the Operational Research Society*, 57(3):290–295, 05/25 2005.

[Cio06] Denis F. Cioffi. Designing project management: A scientific notation and an improved formalism for earned value calculations. *International Journal of Project Management*, 24(2):136–144, 2 2006.

[CIZ+04] Bocheng Chen, WaiHung Ip, Yuebo Zhou, Bing Liang, and H. L. Yu. The design of a lean crm software process model. In Jian Chen, editor, *ICEB; The Fourth International Conference on Electronic Business - Shaping Business Strategy in a Networked World*, pages 335–339. Academic Publishers/World Publishing Corporation, 2004.

[CJBM04] T. R. Collins, E. J. Montes Jr., M. G. Beruvides, and T. C. Maku. A performance model for multiple and simultaneous projects. Technical report, ADepartment of Industrial Engineering, Texas Tech University, 2004.

[CLT02] Zhi-Long Chen, Shanling Li, and Devanath Tirupati. A scenario-based stochastic programming approach for technology and capacity planning. *Computers & Operations Research*, 29(7):781–806, 6 2002.

[CM01] John Callahan and Brian Moretton. Reducing software product development time. *International Journal of Project Management*, 19:59–70, 2001.

[CMPP03] M. Corso, A. Martini, L. Pellegrini, and E. Paolucci. Technological and organizational tools for knowledge management: In search of configurations. *Small Business Economics*, 21(4):397–408, 2003.

[CPB06] Leonardo Chwif, Ray J. Paul, and Marcos Ribeiro Pereira Barretto. Discrete event simulation model reduction: A causal approach. *Simulation Modelling Practice and Theory*, 14(7):930–944, 10 2006.

[CR06] A. Ceselli and G. Righini. A branch-and-price algorithm for the multilevel generalized assignment problem. *Operations Research*, 54(6):1172–1184, 2006.

[Cue03] Gonzalo Cuevas. *Gestión del proceso software*. Centro de Estudios Ramón Areces, Madrid, 2003.

[Day00] Julian Day. Software development as organizational conversation: analogy as a systems intervention. *Systems Research and Behavioral Science*, 17(4):349–358, 2000.

[DB05] O. Dalcher and D. Benediktsson. Estimating size in incremental software development projects. *IEE proceedings. Software*, 152(6):253, 2005.

[DBT05] Darren Dalcher, Oddur Benediktsson, and Helgi Thorbergsson. Development life cycle management: A multiproject experiment. In *ECBS*, pages 289–296. IEEE Computer Society, 2005.

[DDP08] Florian Deissenboeck, Florian Deissenboeck, and Markus Pizka. Probabilistic analysis of process economics. *Softw. Process*, 13(1):5–17, 2008.

[DDW92] Burton V. Dean, David R. Denzler, and James J. Watkins. Multiproject staff scheduling with variable resource constraints. *IRER Transactions or Reenginering Management*, 39(1):59, 1992.

[DER98] B. Dodin, A. A. Elimam, and E. Rolland. Tabu search in audit scheduling. *European Journal of Operational Research*, 106(2-3):373–392, 1998.

[DF00] José Javier Dolado and Luis Fernández. *Medición para la gestión en la Ingeniería del software*. Ra-Ma, Madrid, 2000.

[DH02] Erik L. Demeulemeester and Willy Herroelen. *Project scheduling*, volume 49. Kluwer Academic Publishers, Boston, 2002.

[DI01] Paolo Donzelli and Giuseppe Iazeolla. Hybrid simulation modelling of the software process. *Journal of Systems and Software*, 59(3):227–235, 12/15 2001.

[DM77] José Antonio Domínguez Machuca. *Dinámica de sistemas, aplicación en macroeconomía : realización y experimentación de un modelo del sistema financiero español*. PhD thesis, 1977.

[DR03] Dennis R. Goldenson,Joe Jarzombek and Terry Rout. Measurement and analysis in capability maturity model integration models and software process improvement. *CrossTalk - The Journal of Defense Software Engineering*, jul 2003.

[DRLV06] D. Debels, B. De Reyck, R. Leus, and M. Vanhoucke. A hybrid scatter search/electromagnetism meta-heuristic for project scheduling. *European Journal of Operational Research*, 169(2):638–653, 2006.

[dSP05] Leandro Dias da Silva and Angelo Perkusich. Composition of software artifacts modelled using Colored Petri nets. *Science of Computer Programming*, 56(1-2):171–189, 4 2005.

[DV06] D. Debels and M. Vanhoucke. The electromagnetism meta-heuristic applied to the resource-constrained project scheduling problem. Technical report, 2006.

[dVDHL05] Stijn Van de Vonder, Erik Demeulemeester, Willy Herroelen, and Roel Leus. The use of buffers in project management: The trade-off between stability and makespan. *International Journal of Production Economics*, 97(2):227–240, 8/18 2005.

[EB97] Khaled El Emam and Lionel C. Briand. Costs and benefits of software process improvement. Technical Report IESE-Report No. 047.97/E, 1997.

[EB00] Khaled El Emam and Andreas Birk. Validating the ISO/IEC 15504 measures of software development process capability. *Journal of Systems and Software*, 51(2):119–149, 4/15 2000.

[Ebe89] R. L. Eberlein. Simplification and understanding of. models. *System Dynamics Review*, 5(1):59–68, 1989.

[EKR95] Clarence Ellis, Karim Keddara, and Grzegorz Rozenberg. Dynamic change within workflow systems, 1995.

[Elm70] Salah Eldin Elmaghraby. *Some network models in management science*, volume 29. Springer-Verlag, Berlin etc., 1970.

[Elm05] Salah E. Elmaghraby. On the fallacy of averages in project risk management. *European Journal of Operational Research*, 165(2):307–313, 9/1 2005.

[ETT74] F. E. Emery, Einar Thorsrud, and Eric L. Trist. *Form and content in industrial democracy : some experiences from Norway and other European countries*. Tavistock Institute, London, 1974.

[FBS+01] Peter Freeman, Donald J. Bagert, Hossein Saiedian, Mary Shaw, Robert Dupuis, and J. Barrie Thompson. Software Engineering Body of Knowledge (SWEBOK). In *ICSE*, pages 693–696, 2001.

[FCMA08] H. Fei, C. Chu, N. Meskens, and A. Artiba. Solving surgical cases assignment problem by a branch-and-price approach. *International Journal of Production Economics*, 112(1):96–108, 3 2008.

[FL07] Ilya M. Fishman and Raymond E. Levitt. The virtual design team and quantum tm: Comparison of project organization models. Technical report, 2007.

[FM82] Emilio Freire Macías. *Análisis cualitativo y de bifurcaciones en sistemas dinámicos*. PhD thesis, 1982.

[FMK01] S. Fujii, H. Morita, and T. Kanawa. Resource constrained planning of multiple projects with separable activities. *JSME International Journal, Series C: Mechanical Systems, Machine Elements and Manufacturing*, 44(1):261–266, 2001.

[Fre01] Michael L. Fredley. *A de-composition approach for the multi-modal, resource-constraint, multi-project scheduling problem with generalized precedences and expediting resources.* PhD thesis, Air University, 2001.

[Fri04] Peter Fritzson. *Principles of Object-Oriented Modeling and Simulation with Modelica 2.1*, volume 4470. Wiley-Interscience, 2004.

[Gar01] Roland Gareis. The project-oriented society: A new creation to ensure international competitiveness. In *IPMA International Symposium and NORDNET 2001*, 2001.

[Gav04] M. H. K. Gavareshki. New fuzzy gert method for research projects scheduling. In *Proceedings. 2004 IEEE International Engineering Management Conference, 2004.*, volume 2, pages 820–824 Vol.2, 2004.

[GC03] Alain Guinet and Sondes Chaabane. Operating theatre planning. *International Journal of Production Economics*, 85(1):69–81, 7/11 2003.

[gdf02] GOTHA groupe de flexibilité. Flexilibilité et robustesse en ordonnancement. *Bulletin de la ROADEF*, 8:10–12, 2002.

[GGGL03] Dimitri Golenko-Ginzburg, Aharon Gonik, and Zohar Laslo. Resource constrained scheduling simulation model for alternative stochastic network projects. *Mathematics and Computers in Simulation*, 63(2):105–117, 6/10 2003.

[GMR08] J. F. Gonçalves, J. J. M. Mendes, and M. G. C. Resende. A genetic algorithm for the resource constrained multi-project scheduling problem. *European Journal of Operational Research*, In Press, Corrected Proof, 2008.

[Gol97] Eliyahu M. Goldratt. *Critical chain.* North River Press, Great Barrington, MA, 1997.

[GS05] Noud Gademann and Marco Schutten. Linear-programming-based heuristics for project capacity planning. *IIE Transactions*, 37(2):153–165, 02 2005.

[Han01] E. W. Hans. *Resource loading by branch-and-price techniques*. PhD thesis, University of Tvendte, Enschede,The Netherlands, 2001.

[Har98] S. Hartmann. A competitive genetic algorithm for resource-constrained project scheduling. *Naval Research Logistics*, 45(7):733–750, 1998.

[HBH+06] L. Huang, Barry Boehm, H. Hu, J. Ge, J. Lu, and C. Qian. Applying the value/Petri process to ERP software development in China. In *28th International Conference on Software Engineering 2006, ICSE '06*, volume 2006, pages 502–511, 20 May 2006 through 28 May 2006 2006.

[HD98] S. Hartmann and A. Drexl. Project scheduling with multiple modes: A comparison of exact algorithms. *Networks*, 32(4):283–297, 1998.

[Her05] Willy Herroelen. Project scheduling-theory and practice. *Production and Operations Management*, 14(4):413, Winter 2005.

[HFC+01] Dan X. Houston, Susan Ferreira, James S. Collofello, Douglas C. Montgomery, Gerald T. Mackulak, and Dan L. Shunk. Behavioral characterization: finding and using the influential factors in software process simulation models. *The Journal of Systems and Software*, 59(3):259–270, dec 2001.

[HGdVZ02a] E. W. Hans, A. J. R. M. Gademann, S. L. Van de Velde, and W. H. M. Zijm. Resource loading by branch-and-price techniques: models and algorithms. Technical report, University of Tvendte, Enschede,The Netherlands, 2002.

[HGdVZ02b] E. W. Hans, A. J. R. M. Gademann, S. L. Van de Velde, and W. H. M. Zijm. Rough-cut capacity planning by branch-and-

price techniques. Technical report, University of Tvendte, Enschede,The Netherlands, 2002.

[HH00] Peter Henderson and Yvonne Howard. Simulating a process strategy for large scale software development using systems dynamics. *Software Process: Improvement and Practice*, 5(2-3):121–131, 2000.

[HHLW07] E. W. Hans, Willy Herroelen, R. Leus, and G. Wullink. A hierarchical approach to multi-project planning under uncertainty. *Omega*, 35(5):563–577, 10 2007.

[HK05] C. C Hsu and D. S. Kim. A new heuristic for the multi-mode resource investment problem. *Journal of the Operational Research Society*, 56(4):406–413, 2005.

[HL89] Steven T. Hackman and Robert C. Leachman. Aggregate model of project-oriented production. *IEEE Transactions on Systems, Man and Cybernetics*, 19(2):220–231, 1989.

[HL04a] Willy Herroelen and Roel Leus. The construction of stable project baseline schedules. *European Journal of Operational Research*, 156(3):550–565, 8/1 2004.

[HL04b] Willy Herroelen and Roel Leus. Robust and reactive project scheduling: a review and classification of procedures. *International Journal of Production Research*, 42(8):1599–1620, 04/15 2004.

[HLD02] Willy Herroelen, Roel Leus, and Erik Demeulemeester. Critical chain project scheduling: Do not oversimplify. *Project Management Journal*, 33(4):48, 12 2002.

[HMC01] Dan X. Houston, Gerald T. Mackulak, and James S. Collofello. Stochastic simulation of risk factor potential effects for software development risk management. *Journal of Systems and Software*, 59(3):247–257, 12/15 2001.

[HRD98] Willy Herroelen, B. De Reyck, and E. Demeulemeester. Resource-constrained project scheduling: A survey of recent developments. *Computers and Operations Research*, 25(4):279–302, 1998.

[HS96] S. Hartmann and A. Sprecher. A note on "hierarchical models for multi-project planning and scheduling". *European Journal of Operational Research*, 94(2):377–383, 1996.

[HS08] Seungchul Ha and Hyo-Won Suh. A timed colored Petri nets modeling for dynamic workflow in product development process. *Computers in Industry*, 59(2-3):193–209, 3 2008.

[Hum02] Watts S. Humphrey. Three process perspectives: Organizations, teams, and people. *Annals of Software Engineering*, 14(1-4):39–72, Diciembre 2002.

[Ins04] Project Management Institute. *Guía de los fundamentos de la Dirección de proyectos [Project Management Institute]*. Project Management Institute, Newtown Square, Pa., 2004.

[JC99] David E. Joslin and David P. Clements. Squeaky wheel optimization. *Journal of AI Researh*, 10:353, 1999.

[JDSR08] B. Jarboui, N. Damak, P. Siarry, and A. Rebai. A combinatorial particle swarm optimization for solving multi-mode resource-constrained project scheduling problems. *Applied Mathematics and Computation*, 195(1):299–308, 1/15 2008.

[JF05] Nitin R. Joglekar and David$\tilde{\text{N}}$. Ford. Product development resource allocation with foresight. *European Journal of Operational Research*, 160(1):72–87, 1/1 2005.

[JKCR00] V. A. Jeetendra, O. V. Krishnaiah Chetty, and J. Prashanth Reddy. Petri nets for project management and resource levelling. *The International Journal of Advanced Manufacturing Technology*, 16(7):516–520, 06/21 2000.

[JMR+01] J. Josefowska, M. Mika, R. Rozycki, G. Waligora, and J. Weglarz. Simulated annealing for multi-mode resource-constrained project scheduling. *Annals of Operations Research*, 102(1-4):137–155, 2001.

[JP05] David Joslin and William Poole. Agent-based simulation for software project planning. In *Winter Simulation Conference*, pages 1059–1066, 2005.

[KABK08] Corbin G. Koepke, Andrew P. Armacost, Cynthia Barnhart, and Stephan E. Kolitz. An integer programming approach to support the us air force's air mobility network. *Computers and Operations Research*, 35(6):1771–1788, 2008.

[KC06] S. Kumanan and O. V. Krishnaiah Chetty. Estimating product development cycle time using Petri nets. *The International Journal of Advanced Manufacturing Technology*, 28(1):215–220, 02/13 2006.

[KG07] Konstantinos G. Kouskouras and Andreas C. Georgiou. A discrete event simulation model in the case of managing a software project. *European Journal of Operational Research*, 181(1):374–389, 8/16 2007.

[KH06] Rainer Kolisch and Sönjnke Hartmann. Experimental investigation of heuristics for resource-constrained project scheduling: An update. *European Journal of Operational Research*, 174(1):23–37, 10/1 2006.

[KHY06] H. P Kao, B. Hsieh, and Y. Yeh. A petri-net based approach for scheduling and rescheduling resource-constrained multiple projects. *Journal of the Chinese Institute of Industrial Engineers*, 23(6):468–477, 2006.

[KJ01] Ernesto Kofman and Sergio Junco. Quantized-state systems: a DEVS approach for continuous system simulation. *Trans. Soc. Comput. Simul. Int.*, 18(3):123–132, 2001.

[KJR06] S. Kumanan, G. Jegan Jose, and K. Raja. Multi-project scheduling using an heuristic and a genetic algorithm. *The International Journal of Advanced Manufacturing Technology*, 31(3):360–366, 11/15 2006.

[KLRW01] G. Kahen, M. M. Lehman, J. F. Ramil, and P. Wernick. System dynamics modelling of software evolution processes for policy investigation: Approach and example. *Journal of Systems and Software*, 59(3):271–281, 12/15 2001.

[KMR99] Marc I. Kellner, Raymond J. Madachy, and David M. Raffo. Software process simulation modeling: Why? what? how? *Journal of Systems and Software*, 46(2-3):91–105, 4/15 1999.

[KN02] Kenzo Kurihara and Nobuyuki Nishiuchi. Efficient Monte Carlo simulation method of GERT-type network for project management. *Computers & Industrial Engineering*, 42(2-4):521–531, 4/11 2002.

[Kol96] Rainer Kolisch. Serial and parallel resource-constrained project scheduling methods revisited: Theory and computation. *European Journal of Operational Research*, 90(2):320–333, 1996.

[Kol00] Rainer Kolisch. *Make-to-order assembly management*. Springer, New York, 2000.

[KP01] Rainer Kolisch and R. Padman. An integrated survey of deterministic project scheduling. *Omega*, 29(3):249–272, 6 2001.

[Kum98] A. Kumar. Use of Petri nets for resource allocation in projects. *IEEE Transactions on Engineering Management*, 45(1):49–56, 1998.

[Kur06] K. Kurihara. Branching probabilities planning of stochastic network for project duration planning. In *IEEE Conference on Emerging Technologies and Factory Automation, 2006. ETFA '06.* , pages 1333–1339, 2006.

[KWDK06] Hsing-Pei Kao, Brian Wang, James Dong, and Kuo-Cheng Ku. An event-driven approach with makespan/cost tradeoff analysis for project portfolio scheduling. *Computers in Industry*, 57(5):379–397, 06 2006.

[KYY+05] K. Kim, Y. Yun, J. Yoon, M. Gen, and G. Yamazaki. Hybrid genetic algorithm with adaptive abilities for resource-constrained multiple project scheduling. *Computers in Industry*, 56(2):143–160, 2005.

[Lar60] Juan Larrañeta. *Programación y control de proyectos unitarios.* Universidad de Sevilla, E.T.S. de Ingenieros Industriales, Sevilla, 1992 1960.

[Lar87] Juan Larrañeta. *Programación lineal y grafos*, volume 1. Universidad de Sevilla, Sevilla, 1987.

[LCM04] R. Liao, Q. X Chen, and N. Mao. Genetic algorithm for resource-constrained project scheduling. *Jisuanji Jicheng Zhizao Xitong/Computer Integrated Manufacturing Systems, CIMS*, 10(7), 2004.

[LCWB08] Jun Lin, Kah Hin Chai, Yoke San Wong, and Aarnout C. Brombacher. A dynamic model for managing overlapped iterative product development. *European Journal of Operational Research*, 185(1):378–392, 2/16 2008.

[LJF04] Xiaoxia Lin, Stacy L. Janak, and Christodoulos A. Floudas. A new robust optimization approach for scheduling under uncertainty: I. bounded uncertainty. *Computers & Chemical Engineering*, 28(6-7):1069–1085, 6/15 2004.

[LK02] Christoph H. Loch and Stylianos Kavadias. Dynamic portfolio selection of npd programs using marginal returns. *Management Science*, 48(10):1227–1241, October 1 2002.

[LKR02] M. M. Lehman, G. Kahen, and J. F. Ramil. Behavioural modelling of long-lived evolution processes– some issues and an

example. *Journal of Software Maintenance and Evolution: Research and Practice*, 14:335–351, 2002.

[LM04] Bengee Lee and James Miller. Multi-project management in software engineering using simulation modelling. *Software Quality Journal*, 12:59–82, 2004.

[LN94] E. V. Levner and A. S. Nemirovsky. A network flow algorithm for just-in-time project scheduling. *European Journal of Operational Research*, 79(2):167–175, 12/8 1994.

[L.O98] L.Ozdamar. On scheduling project activities with variable expenditure rates. *IIE Transactions (Institute of Industrial Engineers)*, 30(8):695–704, 1998.

[LO08] Luong Duc Long and Ario Ohsato. Fuzzy critical chain method for project scheduling under resource constraints and uncertainty. *International Journal of Project Management*, In Press, Corrected Proof, 2008.

[LOL88] Juan Larrañeta, Luis Onieva, and Sebastián Lozano. *Métodos modernos de gestión de la producción*. Alianza, Madrid, 1988.

[LTB06] Antonio Lova, Pilar Tormos, and Federico Barber. Multi-mode resource constrained project scheduling: Scheduling schemes, priority rules and mode selection rules. *Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial*, 30:69–86, 2006.

[LTN06] H. C. Lau, R. Thangarajoo, and K. M. Ng. A hybrid MIP/heuristic model for experience based driver assignment. In *8th IEEE International Conference of Tools with Artificial Intelligence*, pages 407–415, 2006.

[LWT01] S. X Liu, M-G Wang, and J. F Tang. Optimization algorithms for solving resource-constrained project scheduling problem: A review. *Kongzhi yu Juece/Control and Decision*, 16(SUPPL.):647–651, 2001.

[LY01] Hareton K.Ñ. Leung and Terence C. F. Yuen. A process framework for small projects. *Software Process: Improvement and Practice*, 6:67–83, 2001.

[MBCDH06] Harvey Maylor, Tim Brady, Terry Cooke-Davies, and Damian Hodgson. From projectification to programmification. *International Journal of Project Management*, 24(8):663–674, 11 2006.

[ME78] Joseph J. Moder and Salah Eldin Elmaghraby. *Handbook of operations research*. Van Nostrand Reinhold, New York etc., 1978.

[Mer06] Jack R. Meredith. *Project management : a managerial approach*. Wiley, New York ; Chichester, 2006.

[MGR08] J. J. M. Mendes, J. F. Gonçalves, and M. G. C. Resende. A random key based genetic algorithm for the resource constrained project scheduling problem. *Computers & Operations Research*, In Press, Corrected Proof:2153, 2008.

[MHSZ04] T. McBride, B. Henderson-Sellers, and D. Zowghi. Project management capability levels: an empirical study. In B. Henderson-Sellers, editor, *11th Asia-Pacific Software Engineering Conference*, pages 56–63, 2004.

[MRGT08] María Moreno, Isabel Ramos, Francisco J. García, and Miguel Toro. An association rule mining method for estimating the impact of project management policies on software quality, development time and effort. *Expert Systems with Applications*, 34(1):522–529, 1 2008.

[MS00] Rolf H. Mohring and Frederik Stork. Linear preselective policies for stochastic project scheduling. *Mathematical Methods of Operations Research*, 52(3):501, 2000.

[MSSU03] Rolf H. Mohring, Andreas S. Schulz, Frederik Stork, and Marc Uetz. Solving project scheduling problems by minimum cut computations. *Management Science*, 49(3):330–350, March 1 2003.

[MT00] Ray Madachy and Denton Tarbet. Case studies in software process modeling with system dynamics. *Software Process: Improvement and Practice*, 5:133–146, 2000.

[MV06] Jürgen Münch and Matias Vierimaa, editors. *Product-Focused Software Process Improvement, 7th International Conference, PROFES 2006, Amsterdam, The Netherlands, June 12-14, 2006, Proceedings*, volume 4034 of *Lecture Notes in Computer Science*. Springer, 2006.

[MWW08] Marek Mika, Grzegorz Waligóra, and Jan Woglarz. Tabu search for multi-mode resource-constrained project scheduling with schedule-dependent setup times. *European Journal of Operational Research*, 187(3):1238–1250, 6/16 2008.

[MYB07] Christoph Meier, Ali A. Yassine, and Tyson R. Browning. Design process sequencing with competent genetic algorithms. *Transactions of the ASME*, 129:566, 2007.

[Nag02] M.Ñagai. Project duration planning method based on the combination use of genetic algorithm and Monte Carlo simulation. In *2002 IEEE International Conference on Systems, Man and Cybernetics*, volume 5, page 6 pp. vol.5, 2002.

[NC03] Rita Nienaber and Elsabe Cloete. A software agent framework for the support of software project management. *Proceedings of SAICSIT*, pages 16–23, 2003.

[NM01] José Niño-Mora. *Encyclopedia of optimization*, volume V, chapter Stochastic scheduling, pages 367–372. Kluwer Academic Publishers, 2001.

[NS98a] David N. and John D. Sterman. Modeling dynamic development processes. *System Dynamics Review*, 14(1):31–68, 1998.

[NS98b] K.Ñeumann and C. Schwindt. A capacitated hierarchical approach to make-to-order production. *Journal Europeen des Systemes Automatises*, 32(4):397–413, 1998.

[NZBS07] M.Ñonaka, L. Zhu, M. A. Babar, and M. Staples. Project cost overrun simulation in software product line development. In *8th International Conference on Product-Focused Software Process Improvement, PROFES 2007*, volume 4589 LNCS, pages 330–344, 2007.

[OA01] L. Ozdamar and Ebru Alanya. Uncertainty modelling in software development projects (with case study). *Annals of Operations Research*, 102(1-4):157–178, 2001.

[Pad02a] F. Padberg. Using process simulation to compare scheduling strategies for software projects. In *Ninth Asia-Pacific Software Engineering Conference, 2002.*, pages 581–590, 2002.

[Pad02b] Frank Padberg. A discrete simulation model for assessing software project scheduling policies. *Software Process: Improvement and Practice*, 7(3-4):127–139, 2002.

[Pad03] F. Padberg. A software process scheduling simulator. In *Proceedings. 25th International Conference onSoftware Engineering, 2003.* , pages 816–817, 2003.

[Pad04a] F. Padberg. Computing optimal scheduling policies for software projects. In *11th Asia-Pacific Software Engineering Conference, 2004.* , pages 300–308, 2004.

[Pad04b] F. Padberg. Linking software process modeling with markov decision theory. In *Proceedings of the 28th Annual International Computer Software and Applications Conference, 2004. COMPSAC 2004.*, volume 2, pages 152–155 vol.2, 2004.

[Pad05] F. Padberg. On the potential of process simulation in software project schedule optimization. In *29th Annual International Computer Software and Applications Conference, 2005. COMPSAC 2005.* , volume 2, pages 127–130 Vol. 1, 2005.

[PAER07] Dietmar Pfahl, Ahmed Al-Emran, and Günther Ruhe. A system dynamics simulation model for analyzing the stability of softwa-

re release plans. *Software Process: Improvement and Practice*, 12(5):475–490, 2007.

[PAM04] Mireille Palpant, Christian Artigues, and Philippe Michelon. Lssper: Solving the resource-constrained project scheduling problem with large neighbourhood search. *Annals of Operations Research*, 131(1):237–257, 10/08 2004.

[Pau95] Mark C. Paulk. The evolation of the sei's capacity maturity model for software. *Software Process: Improvement and Practice*, 1(1):3–15, 1995.

[PHC08] Nai-Hsin Pan, Po-Wen Hsaio, and Kuei-Yen Chen. A study of project scheduling optimization using tabu search algorithm. *Engineering Applications of Artificial Intelligence*, In Press, Corrected Proof, 2008.

[PL00a] D. Pfahl and K. Lebsanft. Using simulation to analyse the impact of software requirement volatility on project performance. *Information and Software Technology*, 42(14):1001–1008, 11/15 2000.

[PL00b] Dietmar Pfahl and Karl Lebsanft. Knowledge acquisition and process guidance for building system dynamics simulation models : An experience report from software industry. *International Journal of Software Engineering and Knowledge Engineering*, 10(4):487–510, 2000.

[PMB99] Antony Powell, Keith Mander, and Duncan Brown. Strategies for lifecycle concurrency and iteration;a system dynamics approach. *Journal of Systems and Software*, 46(2-3):151–161, 4/15 1999.

[PN87] Enrique PonceÑúñez. *Técnicas de análisis cualitativo en sistemas dinámicos : métodos numéricos y aplicaciones en ingeniería*. PhD thesis, 1987.

[PR02] Dietmar Pfahl and Günther Ruhe. IMMoS: a methodology for integrated measurement, modelling and simulation. *Software Process: Improvement and Practice*, 7(3-4):189–210, 2002.

[PWW69] A. Alan B. Pritsker, Lawrence J. Watters, and Philip M. Wolfe. Multiproject scheduling with limited resources: a zero-one programming approach. *Management Science*, 16(1):93–108, 09 1969.

[Ram99] Isabel Ramos. *Un Nuevo Enfoque en la Gestion de Proyectos-de Desarrollo de Software*. PhD thesis, Universidad de Sevilla, 1999.

[RCL99] Ioana Rus, James Collofello, and Peter Lakey. Software process simulation for reliability management. *The Journal of Systems and Software*, 46(2–3):173–182, apr 1999.

[RGCL+05] B. De Reyck, Y. Grushka-Cockayne, M. Lockett, S. R. Calderini, M. Moura, and A. Sloper. The impact of project portfolio management on information technology projects. *International Journal of Project Management*, 23(7):524–537, 2005.

[RHB03] I. Rus, M. Halling, and S. Biffl. Supporting decision-making in software engineering with process simulation and empirical studies. *International Journal of Software Engineering and Knowledge Engineering*, 13(5):531–545, 2003.

[Ric01] Ita Richardson. Software process matrix: A small company SPI model. *Software Process: Improvement and Practice*, 6:157–165, 2001.

[RKC01] J. Prashant Reddy, S. Kumanan, and O. V. Krishnaiah Chetty. Application of Petri nets and a genetic algorithm to multi-mode multi-resource constrained project scheduling. *The International Journal of Advanced Manufacturing Technology*, 17(4):305–314, 01/05 2001.

[RKP+99] D. Raffo, T. Kaltio, D. Partridge, K. Phalp, and Juan Fernández-Ramil. Empirical studies applied to software process models. *Empirical Software Engineering*, 4(4):353–369, 1999.

[RM00] David Raffo and Robert H. Martin. A model of the software development process using both continous and discrete models. *Software Process Improvement and Practice*, 5:147–157, 2000.

[Rou03] T. P. Rout. ISO/IEC 15504 - evolution to an international standard. *Software Process Improvement and Practice*, 8(1):27–40, 2003.

[RRK08] M. Ranjbar, B. De Reyck, and F. Kianfar. A hybrid scatter search for the discrete time/resource trade-off problem in project scheduling. *European Journal of Operational Research*, In Press, Corrected Proof, 2008.

[RRT01] Mercedes Ruiz, Isabel Ramos, and Miguel Toro. A simplified model of software project dynamics. *Journal of Systems and Software*, 59(3):299–309, 12/15 2001.

[RRT02] M. Ruiz, I. Ramos, and Miguel Toro. A dynamic integrated framework for software process improvement. *Software Quality Journal*, 10(2):181–194, 2002.

[RRT+03] Mercedes Ruiz, Isabel Ramos, Miguel Toro, Nuno David, Jaime Simão Sichman, and Helder Coelho. An integrated framework for simulation-based software process improvement; towards an emergence-driven software process for agent-based simulation. *Software Process: Improvement and Practice*, 9(2):81–93, mar 12 2004; 2003.

[RVM99] David M. Raffo, Joseph V. Vandeville, and Robert H. Martin. Software process simulation to achieve higher CMM levels. *The Journal of Systems and Software*, 46(2–3):163–172, apr 1999.

[San00] U. Z. Sanal. A decision support system for fuzzy scheduling of software projects. In *2000 IEEE AUTOTESTCON Proceedings*, pages 263–272, 2000.

[SB05] Thomas J. Schriber and Daniel T. Brunner. Inside discrete-event simulation software: How it works and why it matters. *Proceedings of the 2005 Winter Simulation Conference*, page 167, 2005.

[Sca01] Walt Scacchi. *Process Models in Software Engineering.* Encyclopedia of Software Engineering. John Wiley and Sons, New York, 2001.

[SCFR06] Neil Smith, Andrea Capiluppi, and Juan Fernández-Ramil. Agent-based simulation of open source evolution. *Software Process: Improvement and Practice*, 11(4):423–434, 2006.

[SdOC03] Miguel A. Serrano, Carlos Montes de Oca, and Karina Cedillo. An experience on using the team software process for implementing the capability maturity model for software in a small organization. In *QSIC*, page 327. IEEE Computer Society, 2003.

[SG01] Friedrich Stallinger and Paul Grünbacher. System dynamics modelling and simulation of collaborative requirements engineering. *Journal of Systems and Software*, 59(3):311–321, 12/15 2001.

[Sho06] Y. Y Shou. Composite random sampling algorithm for scheduling concurrent projects. *Zhejiang Daxue Xuebao (Gongxue Ban)/Journal of Zhejiang University (Engineering Science)*, 40(2):344–347, 2006.

[SM07a] B. Skoaud and B. Marcineczyk. Ant colony optimization in project management. *Computer Assisted Mechanics and Engineering Sciences*, 14(4):745–752, 2007.

[SM07b] Susan A. Slotnick and Thomas E. Morton. Order acceptance with weighted tardiness. *Computers & Operations Research*, 34(10):3029–3042, 10 2007.

[SS03] C. S. Sung and S. H. Song. Branch-and-price algorithm for a combined problem of virtual path establishment and traffic

packet routing in a layered communication network. *Journal of the Operational Research Society*, 54(1):72–82, 2003.

[Sta00] Friedrich Stallinger. Software process simulation to support ISO/IEC 15504 based software process improvement. *Software Process: Improvement and Practice*, 5(2-3):197–209, 2000.

[SV93] M. Grazia Speranza and Carlo Vercellis. Hierarchical models for multi-project planning and scheduling. *European Journal of Operational Research*, 64(2):312–325, 1/22 1993.

[SW06] E. F. Silva and R. K. Wood. Solving a class of stochastic mixed-integer programs with branch and price. *Mathematical Programming*, 108(2-3):395–418, 2006.

[SWR07] Siri-On Setamanit, Wayne Wakeland, and David M. Raffo. Using simulation to evaluate global software development task allocation strategies. *Software Process: Improvement and Practice*, 12(5):491–503, 2007.

[Tav99a] L. V. Tavares. *Advanced models for project management*. Kluwer Academic Publishers, Boston ; London, 1999.

[Tav99b] L. Valadares Tavares. *Advanced models for project management*, volume 16. Kluwer Academic Publishers, Boston, MA, 1999.

[Tav02] L. V. Tavares. A review of the contribution of operational research to project management. *European Journal of Operational Research*, 136(1):1–18, 1/1 2002.

[TB87] Miguel Toro Bonilla. *Análisis cualitativo y caos en dinámica de sistemas*. PhD thesis, 1987.

[TC95] John D. Tvedt and James S. Collofello. Evaluating the effectiveness of process improvements on software development cycle time via system dynamics modeling. In *COMPSAC*, pages 318–325, 1995.

[TC06] Lin-Yu Tseng and Shih-Chieh Chen. A hybrid metaheuristic for the resource-constrained project scheduling problem. *European Journal of Operational Research*, 175(2):707–721, 12/1 2006.

[TFC98] L. Valadares Tavares, J. A. Antunes Ferreira, and J. Silva Coelho. On the optimal management of project risk. *European Journal of Operational Research*, 107(2):451–469, 6/1 1998.

[TGR05] K. Taaffe, J. Geunes, and H. Edwin Romeijn. Capacity acquisition and stochastic customer demand assignment in a network of facilities. Technical report, Department of Industrial and Systems Engineering,University of Florida, 2005.

[THP93] Walter F. Tichy, Nico Habermann, and Lutz Prechelt. Summary of the Dagstuhl workshop on future directions in software engineering: February 17-21, 1992, Schloss Dagstuhl. *SIGSOFT Softw.Eng.Notes*, 18(1):35–48, 1993.

[TL03] P. Tormos and A. Lova. An efficient multi-pass heuristic for project scheduling with constrained resources. *International Journal of Production Research*, 41(5):1071–1086, 2003.

[TWE04] Gregório Baggio Tramontina, Jacques Wainer, and Clarence Ellis. Applying scheduling techniques to minimize the number of late jobs in workflow systems. *ACM Symposium on Applied Computing*, page 1396, 2004.

[Uet01] Marc Uetz. *Algorithms for Deterministic and Stochastic Scheduling*. Cuvillier Verlag, Göttingen, 2001.

[VBQ05] Vicente Valls, Francisco Ballestín, and Sacramento Quintanilla. Justification and RCPSP: A technique that pays. *European Journal of Operational Research*, 165(2):375–386, 9/1 2005.

[VBQ08] V. Valls, F. Ballestín, and S. Quintanilla. A hybrid genetic algorithm for the resource-constrained project scheduling problem. *European Journal of Operational Research*, 185(2):495–508, 2008.

[VV07] M Vanhoucke and S Vandevoorde. A simulation and evaluation of earned value metrics to forecast the project duration. *Journal of the Operational Research Society*, 58:1361–1374, 2007.

[Wal08] Grzegorz Waligóra. Discrete and continuous project scheduling with discounted cash flows: a tabu search approach. *Computers & Operations Research*, 35(7):2141–2153, 7 2008.

[WH02] Paul Wernick and Tracy Hall. Simulating global software evolution processes by combining simple models: An initial study. *Software Process Improvement and Practice*, 7:113–126, 2002.

[WKR04] Linda Wallace, Mark Keil, and Arun Rai. Understanding software project risk: a cluster analysis. *Information & Management*, 42(1):115–125, 12 2004.

[WL77] Jerome D. Wiest and Ferdinand K. Levy. *A management guide to PERT CPM : with GERT PDMD CPM and other networks*. Prentice-Hall, Englewood Cliffs, N.J, 1977.

[XMNU08] Ningxiong Xu, Sally A. McKee, Linda K. Nozick, and Ruke Ufomata. Augmenting priority rule heuristics with justification and rollout to solve the resource-constrained project scheduling problem. *Computers & Operations Research*, In Press, Corrected Proof:2153, 2008.

[XMQ+04] Zhao XP, Li MS, Wang Q, Chan K, and Leung H. An agent-based self-adaptive software process model. *Journal of Software*, 15(3):348–359, 2004.

[XOZ+07] Junchao Xiao, Leon J. Osterweil, Lei Zhang, Alexander Wise, and Qing Wang. Applying little-JIL to describe Process-Agent knowledge and support project planning in SoftPM. *Software Process: Improvement and Practice*, 12(5):437–448, 2007.

[YB03] Ali A. Yassine and Dan Braha. Complex concurrent engineering and the design structure matrix method. *Concurrent Engineering*, 11(3):165, 2003.

[YS93] Kum-Khiong Yang and Chee-Chuong Sum. A comparison of resource allocation and activity scheduling rules in a dynamic multi-project environment. *Journal of Operations Management*, 11(2):207–218, 6 1993.

[YS97] Kum-Khiong Yang and Chee-Chuong Sum. An evaluation of due date, resource allocation, project release, and activity scheduling rules in a multiproject environment. *European Journal of Operational Research*, 103(1):139–154, 11/16 1997.

[ZKP00] B. P. Zeigler, T. G. Kim, and H. Praehofer. *Theory and practice of modeling and simulation: Integrating Discrete Event and Continuous Complex Dynamic Systems.* Academic Press, New York, 2000.

[ZL04] Hong Zhang and Heng Li. Simulation-based optimization for dynamic resource allocation. *Automation in Construction*, 13(3):409–420, 5 2004.

[ZnRF04] Ascensión Zafra, Miguel Ángel Ridao, and Eduardo Fernández. An algorithm for optimal scheduling and risk assessment of projects. Technical report, Departamento de Ingenieria de Sistemas y Automática and Universidad de Sevilla. Grupo de Investigación Automática y Robótica Industrial, 2004.

[ZY04] M. Zhuang and A. A. Yassine. Task scheduling of parallel development projects using genetic algorithms. In *Proceedings of the ASME Design Engineering Technical Conference*, volume 1, pages 215–223, 2004.