Proyecto Fin de Carrera Septiembre 2013

MEMORIA Documento 1 Tomo I

Proceso de diseño y desarrollo de productos mediante técnica DFMA aplicando agentes inteligentes

Eduardo Barea Escobar
I.T. en Diseño Industrial

Tutor: Juan R. Lama Ruiz

Escuela Politécnica Superior, Universidad de Sevilla

Índice general

1. Objeto **1**

2. Alcance **2**

2.1. Estado de la técnica 2

2.2. Estudio de DFMA y sus aplicaciones 2

2.3. Estudio de modelos de software de agentes inteligentes 2

2.4. Desarrollo de herramienta soporte 3

2.5. Aplicación práctica de la herramienta para su validación 3

2.6. Conclusiones 3

3. Antecedentes **4**

3.1. Modelo de diseño 4

3.2. Software 4

3.3. Estandarización 5

3.4. Lenguaje de programación 5

3.5. Conclusión 5

4. Normas **6**

4.1. UNE 157001:2002 6

4.2. ISO 10303 [4,68] 6

4.2.1. ISO 10303-11 6

4.2.2. ISO 10303-21 6

4.2.3. ISO 10303-28 6

4.2.4. Protocolos de aplicación 7

4.3. Otras normativas 7

4.3.1. ISO 10646/Unicode 7

4.3.2. ISO 8601 7

5. Bibliografía **8**

5.1. Bibliografía 8

6. Definiciones y Abreviaturas **14**

7. Requisitos del proyecto **16**

7.1. Modelos de diseño 16

7.1.1. Ingeniería Concurrente 16

7.1.2. Design for X . . . 17
7.2. ISO 10303 - Ficheros STEP . . . 27
7.2.1. ISO 10303-11 EXPRESS . . . 28
7.2.2. ISO 10303-21 . . . 30
7.2.3. ISO 10303-28 . . . 33
7.3. Software CAD . . . 33

8. Análisis de Soluciones 35
8.1. Diseño Inteligente . . . 35
8.1.1. Inteligencia Artificial . . . 35
8.1.2. Sistemas Expertos . . . 36
8.2. Desarrollo de aplicaciones informáticas . . . 38
8.3. Programación multiagente . . . 39
8.4. Descripción del Software . . . 40
8.4.1. Programación en Python . . . 40
8.4.2. Software existente . . . 45

9. Caso de aplicación 50
9.1. Estructura general de la aplicación . . . 50
9.1.1. Estructura de una aplicación Python . . . 50
9.1.2. Estructura de agentes SPADE . . . 52
9.2. Estructura detallada . . . 55
9.2.1. Módulos importados . . . 56
9.2.2. Agente inicio . . . 56
9.2.3. Agente dfma . . . 62
9.2.4. Agente representacion . . . 79

10. Resultados Finales 82
10.1. Caso 1 - Caja con cuatro tornillos . . . 82
10.2. Caso 2 - Caja con cuatro pestañas . . . 88
10.3. Conclusiones de los resultados . . . 88
10.3.1. Conclusiones respecto a tiempos de ensamblado . . . 89
10.3.2. Conclusiones económicas . . . 89
10.3.3. Conclusiones finales - Índices de Ensamblabilidad . . . 90

11. Conclusiones del proyecto 91
11.1. Sistemas multiagentes . . . 91
11.2. Agentes inteligentes . . . 91
11.3. Software de análisis integrado en aplicaciones CAD . . . 91
11.4. Sistemas estandarizados . . . 92
11.5. Python como medio de desarrollo . . . 92
11.6. La metodología DFMA . . . 92
11.7. Posibles mejoras . . . 92

11.8. Viabilidad del proyecto 92

Índice de figuras

7.1. Esquema de diseño DFMA 19
7.2. DFMA - Evaluación de simetrías (imágenes sacadas de [20]) 22
7.3. DFMA - Tiempos de manipulación(imágenes sacadas de [20]) 23
7.4. DFMA - Tabla de los tiempos de inserción(imágenes sacadas de [20]) 26
7.5. Índice de eficiencia del diseño según DFMA 27
7.6. Ejemplo de programación en EXPRESS-G 28

8.1. Entorno gráfico DFMAPRO 46
8.2. Software DFA 47
8.3. Screenshot del XML-Translator 48
8.4. Ejemplo de uso del Step Analyzer de NIST 49

9.1. Relación entre agentes de la aplicación 52
9.2. Diagrama de flujo del agente inicio 57
9.3. Agente SPADE dfma 63
9.4. Interfaz gráfica ventana_critica 68
9.5. Número de manos necesarias 69
9.6. Una mano con sujeción 70
9.7. Necesidad de 2 manos y gran tamaño 71
9.8. Interfaz para introducir tamaño 72
9.9. Índice de eficiencia del diseño según DFMA 75
9.10. Interfaz para mostrar el valor del índice DFMA del producto analizado 76
9.11. Ejemplo visualizador 3D: view3dscene 80

10.1. Ventana de selección de fichero STEP-XML 82
10.2. Visualizador 3D y ventana con número de componentes 83
10.3. Listado de los nombres de cada componente 83
10.4. Criticidad de los componentes TAPA y TORNILLO 84
10.5. Simetría de forma TAPA1 84
10.6. Número de manos necesarias para su manipulación 85
10.7. Ventanas de introducción de tamaños de cada pieza 85
10.8. Ventanas de selección de formas de inserción 86
10.9. Ventanas de selección de formas de inserción de la TAPA1 y BASE1 86
10.10. Ventanas de selección de formas de inserción de la TAPA1 y BASE1 86
10.11. Ventanas de selección de formas de inserción de la TAPA1 y BASE1 87

10.12 Hoja de cálculo con toda la información sobre el producto 87

Índice de cuadros

10.1. Resultados diseño con tornillos 89
10.2. Resultados diseño con clips 89

1 Objeto

El objetivo de este proyecto fin de carrera es el desarrollo herramienta soporte al ingeniero de diseño, que facilite la evaluación de alternativas de diseño, para un mismo producto, respecto a un diseño enfocado en la fabricabilidad y el ensamblaje.

Ante la numerosa cantidad de formatos de archivo que pueden exportarse de cualquier software CAD, para la evaluación de los mismos, se ha utilizado el formato de archivo STEP por ser un estándar internacional para la representación e intercambio de información de productos industriales, normalizado según la ISO 10303.

Esta herramienta es un ejercicio de programación en un lenguaje script. Utilizado para conseguir un prototipo de programa que sea útil para el usuario, pero no sea necesario una aplicación final para distribución, si no una ayuda al diseñador.

También señalar que está enfocado a cualquier tipo de usuario que necesite una evaluación de alternativas de diseño y se ha utilizado un modelo de diseño basado en el diseño para la fabricabilidad y el ensamblaje (DFMA) por caracterizar el producto con parámetros evaluables y el interés ante el gran ahorro de costos que puede generar.

En conclusión se desarrollará una herramienta soporte en lenguaje script bajo un paradigma multiagente y apoyándose en herramientas clásicas como hojas de cálculo, que estudie alternativas de diseño generados en sistemas CAD con ficheros STEP y que evalué dichas alternativas según un modelo de diseño para la fabricabilidad y el ensamblaje, lo que conlleva a una reducción de costes antes de la fabricación del producto final.

2 Alcance

En consecuencia a lo anteriormente descrito, el objeto de diseño, el alcance de este proyecto está definido por la siguiente serie de objetivos:

2.1. Estado de la técnica

- Estudio del estado de la técnica en sistemas CAD
- Estudio de herramientas software relacionadas.
- Análisis de pros y contras de estos sistemas.
- Comprensión y utilización de normativas:
  - Estudio de la normativa ISO 10303.
    - Conocimientos de la normativa para ficheros STEP (ISO 10303-28).
    - Diferenciación entre ficheros STEP-EXPRESS y STEP-XML.
    - Características sobre los protocolos de aplicación (AP's).

2.2. Estudio de DFMA y sus aplicaciones

- Adquisición de conocimientos de modelos de diseño:
  - Conocimientos de modelos de evaluación de diseño.
  - Conclusiones sobre diseño para la fabricación.
  - Conclusiones sobre diseño para el ensamblaje.
  - Caracterización de parámetros para el diseño para la fabricación y el ensamblaje.

2.3. Estudio de modelos de software de agentes inteligentes

- Estudio de plataforma de agentes:
  - Inclusión de Sistema multiagentes.
  - Programación de Sistema Experto.
  - Evaluación de agentes inteligentes.

2.4. Desarrollo de herramienta soporte

- Estudio sobre entorno de programación para prototipado:
  - Generación de código y estructuras en lenguaje Python.
    - Búsqueda y aplicación de Módulos y Bibliotecas.
    - Desarrollo de interfaz gráfica.
    - Utilización de parsers para lectura de documentos.
    - Relación con herramientas clásicas (hojas de cálculo)

2.5. Aplicación práctica de la herramienta para su validación

- Estudio de un caso de diseño de producto
- Estudio del rediseño del producto
- Evaluación y conclusiones de los resultados

2.6. Conclusiones

En conclusión, con el desarrollo de este proyecto se pretende conseguir adquirir los conocimientos necesarios para la generación de un prototipo de una herramienta informática con un lenguaje script y apoyándose en herramientas clásicas como hojas de datos.

También se pretende conseguir el desarrollo de la misma bajo unas normativas y modelos de diseño que ya existen y están presentes en el proceso de generación de nuevos productos, en este caso DFMA.

Por ello, este proyecto es una prototipo de herramienta soporte para la mejora de la eficiencia de las primeras etapas de la creación y evaluación de nuevos productos enfocada a un uso particular de ingenieros de diseño.

3 Antecedentes

En este capítulo procederemos a describir los distintos antecedentes que han sido los motivos por los que se ha desarrollado el proyecto.

3.1. Modelo de diseño

En la actualidad el proceso de diseño ha cobrado una gran importancia en el ciclo de vida de un producto, puesto que es la etapa donde más recursos se necesitan. Hay numerosos modelos de diseño para la creación de nuevos productos. Entre ellos una serie de modelos denominados "design for x" o diseño para x, los cuales, en pocas palabras, buscan que en la fase preliminar del producto, este se adecúe con unos parámetros para conseguir una serie de características.

Entre estas estrategias de diseño se ha elegido el DFMA "Design for Manufacture and Assembly" [9](diseño para la fabricación y el ensamblaje). Por ser un método definido por el complemento de otros dos métodos DFA "Design For Assembly" (diseño para el ensamblaje) y DFM "Design For Manufacture (diseño para la fabricación).

En DFMA se optimiza la fabricación y el ensamblado para conseguir un ahorro de número de piezas, material, tiempo de ensamblado, y una serie de características que se detallarán más adelante; Estos generan los consecuentes beneficios tanto en ahorro de tiempo y como económicos.

3.2. Software

La aplicación de este tipo de procesos de diseño puede llegar a ser un tanto tediosa, por este motivo existen en el mercado algunas aplicaciones software para la gestión DFMA en procesos de desarrollo de producto.

Uno de los inconvenientes de este tipo de aplicaciones informáticas es su poca variedad, existen programas para optimización en DFA y existen otros programas para DFM, pero no existe un software para DFMA que además esté basado en análisis de archivos CAD [86].

Este problema significa que el software no está vinculado directamente a ningún software CAD, por lo tanto se tiene que implementar en otro punto de la cadena de creación de productos. Este inconveniente hace que el estudio DFMA se alargue en el tiempo de desarrollo y crea la necesidad de volver a rediseñar en fases más tardías del diseño.

Lo que se quiere desarrollar es una aplicación informática que evalúe de forma semi-automática las alternativas de diseño, desde ficheros CAD, en función del diseño para la fabricación y el ensamblaje, provocando ahorro de tiempo en el desarrollo de producto.

3.3. Estandarización

Otro de los aspectos más importantes de este proyecto es el uso de un modelo de archivo CAD, para que pueda ser utilizado sea cual sea el sistema de diseño. Para ello se ha utilizado un formato estandarizado de ficheros denominado STEP, el cual está normalizado por la ISO 10303 [4] que es un standard del modelo de datos para el intercambio de productos.

3.4. Lenguaje de programación

Cabe destacar que este proyecto ha sido desarrollado en su totalidad bajo programación en lenguaje Python por numerosos motivos.

Principalmente por ser un lenguaje script, característica esencial para crear una aplicación prototipo, que nos permita que sea útil para un usuario determinado. No se pretende crear una aplicación para su distribución, aunque el lenguaje tenga potencia suficiente para ello.

Otras características que nos proporciona Python [10,34] son la sencillez y claridad de su código, la cantidad de bibliotecas y módulos que pueden utilizarse para ahorrar escritura de código innecesaria y por que en la actualidad es un lenguaje de programación en auge.

Este lenguaje nos permite la inclusión de una plataforma multiagente, lo que posteriormente nos permitirá poder agregar nuevos módulos con más funciones de evaluación de alternativas.

3.5. Conclusión

En conclusión es imprescindible trasladar a la fases mas tempranas del proceso de desarrollo de productos ciertas consideraciones que antes no se llegaban a contemplar o que se consideraban en las etapas finales. Aplicando DFMA, que es uno de los modelos de diseño mas completos, a estas fases tempranas, bajo una automatización del proceso, conseguiremos un ahorro de tiempo y un sustancial ahorro económico.

4 Normas

Durante la extensión de este capítulo se abordará una enumeración de normativas que afectan al presente proyecto.

Uno de los objetivos principales de este proyecto es el desarrollo de una aplicación informática que pueda ser utilizada bajo cualquier software de diseño CAD, por ello durante la documentación de este proyecto se han estudiado diversas normativas, las cuales nos permiten cumplir estas expectativas bajo una estandarización internacional.

4.1. UNE 157001:2002

En primer lugar la ordenación del proyecto según la norma UNE 157001:2002 que especifica los "Criterios generales para la elaboración de proyectos". Esta normativa describe detalladamente cómo deben exponer cada uno de los capítulos que forman este proyecto.

4.2. ISO 10303 [4, 68]

Para la estandarización de los ficheros CAD, se ha indagado en la normativa ISO 10303, denominada "Automatización de sistemas industriales e integración – representación e intercambio de datos entre productos", la cual describe un mecanismo que sea capaz de describir la información de un producto a través de su ciclo de vida, independientemente del sistema donde haya sido desarrollado.

Esta ISO es comúnmente conocida como STEP "Standar for the Exchange of Product model data" (Estándar del modelo de datos para intercambio de productos) y se divide en varias partes, de las cuales se han destacado:

4.2.1. ISO 10303-11

Conocida como EXPRESS, es un lenguaje de programación.

4.2.2. ISO 10303-21

Es el fichero STEP que está generado por un esquema EXPRESS.

4.2.3. ISO 10303-28

Es la implementación de una estructura XML al esquema EXPRESS.

4.2.4. Protocolos de aplicación

Este tipo de ficheros STEP poseen una serie de "AP's" (Protocolos de Aplicación) que utilizan información de bajo nivel de recursos integrados para definir combinaciones y configuraciones para representar un modelo de datos en particular. Los AP's que se han utilizado son:

4.2.4.1. ISO 10303-203

Configuración de diseños controlados por 3D de partes mecánicas y ensamblajes.

4.2.4.2. ISO 10303-214

Núcleo de datos para diseño de procesos mecánicos en automoción. Este se ha estudiado por ser uno de los protocolos más utilizados por los sistemas CAD.

4.3. Otras normativas

La ISO 10303 a su vez contiene referencias a otras normativas, algunas de ellas son:

4.3.1. ISO 10646/Unicode

Define el "Conjunto de Caracteres Universal" que puede ser utilizado en las variables e instancias del código EXPRESS.

4.3.2. ISO 8601

Denominada como "Elementos de datos y formatos intercambiables - Intercambio de información - Representación de fechas y horas". Normaliza el formato de fecha y hora utilizado en los ficheros STEP.

5 Bibliografía

5.1. Bibliografía

[1] Design For Manufacture and Assembly. pages 219–246.

[2] DFMA Distributed Frameworks for Multimedia Applications. page x, 2005.

[3] Yoshinobu Adachi. Overview of Meta Model Definition in XML for EXPRESS Representation. (Xmi), 2001.

[4] H O W T O Adopt. The iso 10303 standards.

[5] Christoph Anthes and Jens Volkert. inVRs - A Framework for Building Interactive Networked Virtual Reality Systems. In Michael Gerndt and Dieter Kranzlmüller, editors, *High Performance Computing and Communications*, volume 4208 of *Lecture Notes in Computer Science*, pages 894–904. Springer Berlin / Heidelberg, 2006.

[6] E Appleton, J A Garside, and J A Garside. Assembly Automation Emerald Article : A team-based design for assembly methodology A team-based design for assembly methodology. 2011.

[7] Visual Basic. *XML Introducción al lenguaje.*

[8] I Bettles. Design for manufacture-assembly DFMA - the Boothroyd-Dewhurst approach. pages 316–321, 1992.

[9] Geoffrey Boothroyd. Product design for manufacture and assembly. *Computer-Aided Design*, 26(7):505–520, 1994.

[10] Jason R Briggs. Doma de Serpientes Para Niños. Aprendiendo a Programar con Python.

[11] R.V.E. Bryant, T.J. Laliberty, and L.J. Lapointe. DICE MO-a collaborative DFMA analysis tool. In *Proceedings of 3rd IEEE Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises*, pages 96–103. IEEE Comput. Soc. Press, 1994.

[12] Francisco J Ruiz-ruano Campaña. LYX : Con L de LaTEX. 2010.

[13] North Charleston. Recommended Practices for External References with References to the PDM Schema Usage Guide. pages 1–8, 2005.

[14] D E L R Í O Cidoncha, MÂª Gloria, and Martínez Palacios. INTERCAMBIO DE MODELOS SÓLIDOS ENTRE DISTINTOS SISTEMAS DE CAD MEDIANTE EL FORMATO NEUTRO STEP . (1).

[15] Tabla De Contenidos. Python + GTK + Glade.

[16] Lower Assembly Cost, Shorter Assembly Time, Increased Reliability, and Shorter Total Time-to market. Assembly Why Use DFMA ?

[17] Marithelma Costa and Ricardo Piglia. Entrevista a Ricardo Piglia. *Hispam{é}rica*, 15(44):39{\textendash}54, 1986.

[18] By Mark Curtis, Alan Lashbrook, Harland Simon, and Automation Systems. Boothroyd Dewhurst Design for Manufacture 8˜ Assembly.

[19] M Curtis, Jas Walia, and A Lashbrook. Boothroyd Dewhurst Design for Manufacture and Assembly. pages 4/1 —-4/9, 1992.

[20] Product Design. Design for Manufacturing and Assembly I : General Principles Assembly.

[21] Mechanical Desktop. TIPOS DE ARCHIVOS CON FORMATOS BÁSICOS EN UN PROGRAMA CAD.

[22] Cnicas D E Dise. Metodologia del Diseño. pages 1–31, 2012.

[23] Allen Downey, Jeffrey Elkner, and Chris Meyers. *How to think like a computer scientist: learning with Python*. Soho Books, Wellesley, Massachusetts, 2002.

[24] L Dykes and E Tittel. *XML for Dummies*. 2011.

[25] J A Garside E. Appleton. A team-based design for assembly methodology. *Assembly Automation*, 20(2):162–170, 2000.

[26] Dolores Mar\'ia Llidó Escriva, Isabel Gracia Luengo, and Pedro Garc\'ia Sevilla. Metodologia y Tecnologia de la Programacion. 2008.

[27] Harri Eskelinen, Tianhong Luo, and Xiaoan Chen. *DFMA analysis and aspects of applying Internet based collaborative design for axial eccentric oil-pump design*. Lappeenranta University of Technology, Lappeenranta, 2004.

[28] Carla Estorilio. Cost reduction of a diesel engine using the DFMA method. 4(December):95–104, 2006.

[29] Carla Estorilio and Marcelo César Simião. Cost reduction of a diesel engine using the DFMA method. *Product Management & Development*, 4(2):95{\textendash}103, 2006.

[30] Xenia Fiorentini and Sudarsan Rachuri. STEP-OAGIS Harmonization Joint Working Group PDM Subgroup Interim Report STEP-OAGIS Harmonization Joint Working Group PDM Subgroup Interim Report.

[31] Université De Franche-comté. Distributed Frameworks for Multimedia Applications Organisers. page 5.

[32] Pablo M Garc. Proyecto Physthones.

[33] Stephen Gordon and Stewart Moore. Improved CAD-CAE Integration with ISO STEP and XML Standards at Electric Boat.

[34] Isabel Gracia. Introducci ÂŽ on a la programaci ÂŽ on con Python. 2003.

[35] Metabolik Bio Hacklab. Breve tutorial de introducción a la programación con python + glade . Â¿ Porque python + gtk + glade ? Python Otras opciones. 2005.

[36] File Information. Package / File Information. pages 1–21.

[37] En Internet and L E D Samsung. Aprendiendo GtkBuilder y Python.

[38] Kyoung-yun Kim, David G Manley, and Hyungjeong Yang. Ontology-based assembly design and information sharing for collaborative product development. 38:1233–1250, 2006.

[39] Kyoung-Yun Kim, David G Manley, and Hyungjeong Yang. Ontology-based assembly design and information sharing for collaborative product development. *Computer-Aided Design*, 38(12):1233–1250, 2006.

[40] Kyoung-Yun Kim, Yan Wang, Obinna S Muogboh, and Bartholomew O Nnaji. Design formalism for collaborative assembly design. *Computer-Aided Design*, 36(9):849–871, 2004.

[41] X Y Kou and S T Tan. An XML Implementation for Data Exchange of Heterogeneous Object Models.

[42] R J Kuo and C Y Yang. Simulation optimization using particle swarm optimization algorithm with application to assembly line design. *Applied Soft Computing*, 11(1):605–613, 2011.

[43] Thomas J Laliberty. A Collaborative DFMA Analysis Tool. (8363):96–103, 1994.

[44] Justin J Y Lin. Research on collaborative concept design integrating the application of virtual reality and {DFMA}. In *Computer Supported Cooperative Work in Design, 2008. {CSCWD} 2008. 12th International Conference on*, page 727{\textendash}732, 2008.

[45] Robert R Lipman. STEP File Analyzer Users Guide NIST Interagency or Internal Reports. Technical Report {NIST} {IR} 7897, National Institute of Standards and Technology, Gaithersburg, {MD}, 2012.

[46] Robert R Lipman and Robert R Lipman. NISTIR 7897 STEP File Analyzer.

[47] Liu and Sheng. Modeling and Simulation for Packagin Assembly. *Chemistry & ...*, pages 7–14, 2011.

[48] David Loffredo. Fundamentals of STEP implementation. *STEP Tools, Inc*, 1999.

[49] Niels Lohse. *TOWARDS AN ONTOLOGY FRAMEWORK FOR THE INTEGRATED DESIGN OF MODULAR ASSEMBLY*. Number May. 2006.

[50] Niels Lohse, Hitendra Hirani, and Svetan Ratchev. Equipment ontology for modular reconfigurable assembly systems. *International Journal of Flexible Manufacturing Systems*, 17(4):301–314, 2005.

[51] Joshua Lubell. AN XML REPOSITORY ARCHITECTURE FOR STEP MODULES.

[52] Tianhong Luo, Wenjun Luo, and Zhaofeng Lu. DFMA for Internet-Based Collaborative Design. *2007 11th International Conference on Computer Supported Cooperative Work in Design*, pages 226–232, April 2007.

[53] Tianhong Luo, Wenjun Luo, and Zhaofeng Lu. DFMA for Internet-Based Collaborative Design. pages 226–232, 2007.

[54] Darrell L Mann. Integration and Application of TRIZ and DFMA. pages 1–10.

[55] Darrell L Mann. Integration and Application of TRIZ and DFMA.

[56] Robert E McGrath, Jason Kastner, and Jim Myers. Experiments in Data Format Interoperation Using Defuddle. *National Center for Supercomputing Applications, Urbana*, 2009.

[57] Robert E Mcgrath, Jason Kastner, Jim Myers, and Technologies Directorate. Experiments in Data Format Interoperation Using Defuddle. (September), 2009.

[58] Operaciones D E Mecanizado. Tesis doctoral. 1996.

[59] David G. Meeker. DFMA and Its Role in the Integrated Product Development Process.pdf. page 23, 1996.

[60] Inc NetLibrary and IPAS IFIP TC5 WG5.5 International Precision Assembly Seminar. *Precision assembly technologies for mini and micro products proceedings of the {IFIP} {TC5} {WG5.5} Third International Precision Assembly Seminar ({IPAS} '2006), 19-21 February 2006, Bad Hofgastein, Austria*. Springer, New York, 2006.

[61] Gustaf Neumann and Uwe Zdun. XOTcl - Tutorial. pages 0–66.

[62] J Ng and J O Muirhead. DFMA of a military test set: a case study. pages 375–380, 1994.

[63] Joseph Ng and James Muirhead. DFMA OF A MILITARY TEST SET. pages 0–5, 1910.

[64] Tomasz Nowak, Marcin Chromniak, Robert Sekula, and Lucas-lu Gao. Simplicity pays. pages 55–58.

[65] Russell S Peak, Ferst Drive, and Joshua Lubell. STEP , XML , and UML : Complementary Technologies 1. pages 1–20, 2004.

[66] João Pedro and Buiarskey Kovalchuk. PARTS DFMA APPLICATION ON THE DEVELOPMENT OF PARTS FOR INDUSTRY THE WHITE GOODS INDU STRY - A CASE STUDY. 2006, 2006.

[67] By David K Porter. Overview of Design for Manufacturing and Assembly Director of Engineering ,.

[68] Michael J Pratt. Introduction to ISO 10303 - the STEP Standard for Product Data Exchange. 1984.

[69] Diseño D E L Producto, E L Diseño, D E L Producto, L A Selección, and D E Productos Y Servicios. Diseño de Sistemas Productivos y Logisticos. 2005.

[70] J Rieffel and J Pollack. Evolving assembly plans for fully automated design and assembly. pages 165–170, 2005.

[71] John Rieffel and Jordan Pollack. Evolving Assembly Plans for Fully Automated Design and Assembly. (781):1–6, 2005.

[72] Salvador Capuz Rizo and Tomás Gómez Navarro. *Ecodiseño: Ingenieria Del Ciclo de Vida para el Desarrollo de Productos Sostenibles*. Ed. Univ. Polit{é}c. Valencia, 2002.

[73] Arkaitz Ruiz. Introducción a Python. 2006.

[74] Design Rules. Design for Assembly Time Estimation for Assembly.

[75] Douglas Schenck and Peter R Wilson. *Information modeling: the EXPRESS way*. Oxford University Press, New York, 1994.

[76] David Scherer. 1 Overview 2 Your first program 3 Running the program 4 Stopping the program 5 Save Your program 6 A ball in a box. pages 1–7, 2000.

[77] Weiming Shen, Mauro Onori, José Barata, and Regina Frei. Evolvable Assembly Systems Basic Principles. In *Information Technology For Balanced Manufacturing Systems*, volume 220 of *{IFIP} International Federation for Information Processing*, pages 317–328. Springer Boston, 2006.

[78] Yo Sí. Python no Muerde.

[79] Andrew Sparrow. *Film and television distribution and the Internet: A legal guide for the media industry*. Gower Publishing Company, 2007.

[80] Richard Stamper. XML for STP data Astrogrid. (June):1–5, 2002.

[81] K Tanto and En Gtk. Primeros pasos con Glade y PyGTK. 2010.

[82] Yang Tao, Ji Xue-jun, and Li Chao-jian. Wrench assembly design in OpUs darwin SO (spray-off) automatic assembly machine. *2011 International Conference on Electric Information and Control Engineering*, pages 3799–3802, April 2011.

[83] Marcello Trolio. DFMA and Concurrent Engineering Application.

[84] The Srnallpeice Trust. and compatible computers and consists of a number of separate modules as shown in Figure 1. pages 316–321.

[85] I S O Ts. SPECIFICATION PROOF / ÉPREUVE Reference number. 2002, 2002.

[86] Assembly Tutorial and Dfma Software. Design for Manufacturing / Design for Assembly Tutorial for DFMA Software on CAE.

[87] Paco Villegas. Introducción a LYX. 2004.

[88] Chris Withers. Working with Excel files in Python. 2009.

6 Definiciones y Abreviaturas

ACC Agent Communication Channel — Canal de Comunicación entre Agentes.

AEM Assamblability Evaluation Method — Método de Evaluación de la Ensamblabilidad.

AMS Agent Management System — Sistema de Gestión de Agentes

AP Application Protocols — Protocolos de Aplicación.

ARRAY Vector. Zona de almacenamiento que contiene una serie de elementos del mismo tipo.

ASCII American Standard Code for Information Interchange — Código Estándar Estadounidense para el Intercambio de Información.

CAD Computer-Aided Design — Diseño Asistido por Ordenador.

CAE Computer-Aided Engineering — Ingeniería Asistida por Ordenador.

CAM Computer-Aided Manufacturing — Fabricación Asistida por Ordenador.

CM Coste de ensamblaje total.

DF Directory Facilitator — Facilitador de Directorios.

DFA Design For Assembly — Diseño para el Ensamblaje.

DFM Design for Manufacturing — Diseño para la Fabricación.

DFMA Design for Manufacturing and Assembly — Diseño para la Fabricación y el Ensamblaje

DOM Módulo de parseo de documentos XML.

ENTITY Entidad.

ETREE Módulo de parseo de documentos XML.

EXPRESS Lenguaje de programación del STEP, similar al PASCAL.

EXPRESS-G Forma de representación gráfica del lenguaje EXPRESS.

GLADE Herramienta de desarrollo de interfaces gráficas.

GUI Graphical User Interface — Interfaz gráfica de usuario.

IA Inteligencia Artificial

IDE Integrated Development Enviroment — Entorno de desarrollo integrado.

ISO International Organization for Standardization — Organización Internacional de Normalización.

NIST National Institute of Standards and Technology— Instituto Nacional de Estandarización y Tecnología.

NM Suma de piezas críticas. Número total de piezas críticas de un producto.

PARSER Analizador sintáctico.

PASCAL Lenguaje de programación basado en la programación estructurada.

PE Programación Estructurada.

PMI Product and Manufacturing Information — Información sobre el Producto y su Fabricación.

SCHEMA Esquema o bloque principal de programación EXPRESS.

SDAI Modelo de datos de ficheros STEP usando una base de datos compartida.

SE Sistema Experto.

SPADE Plataforma de sistemas multiagente.

SPE Stani's Python Editor — Editor para Python.

STEP Standar for the Exchange of Product model data — Estándar del modelo de datos para intercambio de productos.

STRING Cadena de caracteres.

TM Tiempo de ensamblaje total.

UNE Una Norma Española. Denominación de la normativa española.

XML eXtensible Markup Language — Lenguaje de Marcas eXtensible. Lenguaje de programación para estructuración estandarizada en el intercambio de información entre distintas plataformas.

7 Requisitos del proyecto

En este capítulo procederemos a describir los distintos requerimientos técnicos que precedían al proyecto. Para ello se ha dividido en tres secciones, la primera de ellas describe el modelo de diseño DFMA. Posteriormente se detallan los conocimientos necesarios para comprender las particularidades de la ISO 10303. Por último se justifica la implementación de este proyecto en plataformas CAD, también denominadas plataformas de diseño asistido por ordenador.

7.1. Modelos de diseño

La sociedad actual utiliza el desarrollo de productos para solucionar necesidades propias. Estas necesidades son caracterizadas, según contextos y objetivos. Para solventar estos problemas de diseño se han desarrollado diversas metodologías, puesto que los procesos que llevan hasta la solución final, necesitan guiarse bajo un modelo de diseño. Las actividades y tareas realizadas para la satisfacción de las necesidades son etapas de los modelos de diseño.

Algunos de estos métodos son enfocados a lograr la optimización en determinadas áreas. Este proyecto está enfocado bajo un modelo de diseño definido cómo "Design for X" y un tipo de ingeniería concurrente.

7.1.1. Ingeniería Concurrente

La Ingeniería Concurrente es una nueva forma de concebir la ingeniería de diseño y desarrollo de productos y servicios de forma global e integrada en donde concurren las siguientes perspectivas:

1. Desde el punto de vista del producto. Se toman en consideración tanto la gama que se fabrica y ofrece a la empresa como los requerimientos de las distintas etapas del ciclo de vida y los costes o recursos asociados.

2. Desde el punto de vista de los recursos humanos y las metodologías. Colaboran profesionales que actúan de forma colectiva en tareas de asesoramiento y de decisión (con presencia de las voces significativas) o de forma individual en tareas de impulsión y gestión (gestor de proyecto), tanto si pertenecen a la empresa como si son externos a ella (otras empresas, universidades o centros tecnológicos).

3. Desde el punto de vista de los recursos materiales. Concurren nuevas herramientas basadas en tecnologías de la información y la comunicación sobre una base de

datos y de conocimientos cada vez más integrada (modelización 3D, herramientas de simulación y cálculo, prototipos y útiles rápidos, comunicación interior, Internet).

Para designar este nuevo concepto, además del término Ingeniería Concurrente, en la literatura especializada aparecen otras denominaciones como Ingeniería Simultánea, Diseño Total o Diseño Integrado. Sin embargo, nos inclinamos por la primera denominación ya que, además de tener una buena aceptación, incide el hecho de la concurrencia de puntos de vista, de metodologías, de actores humanos y de herramientas de apoyo.

Tipos de Ingeniería Concurrente:

- Ingeniería concurrente orientada al producto (fabricación, costes, inversión, calidad, comercialización, apariencia): Está referida a la integración de todos aquellos aspectos que pueden tener una incidencia positiva en el producto, especialmente en sus funciones y en la relación entre prestaciones y coste.

- Ingeniería concurrente orientada al entorno (ergonomía, seguridad, medio-ambiente, fin de vida): Relacionada con los aspectos del entorno del producto que, a pesar de que con un diseño concurrente adecuado podrían mejorar o eliminarse, no hay incentivos suficientes para implementarlos pues, normalmente, sus efectos inciden fuera de la empresa y normalmente son soportados por los usuarios e indirectamente por la sociedad (consumos elevados, contaminaciones, fallos, falta de seguridad, problemática de fin de vida

7.1.2. Design for X

Son métodos enfocados a lograr la optimización del diseño en determinadas áreas como son: la fabricación, el ensamblaje, inspección y prueba, logística de materiales, almacenamiento y distribución, reciclado y disposición, impacto al medio ambiente, servicio, costo, ventas, uso y operación.

Los métodos y herramientas de DFX colectan y presentan hechos y datos con relación al diseño y sus procesos de producción, analizan todas las relaciones entre ellos, generan alternativas combinando fuerzas y reduciendo vulnerabilidades, y por último proporcionan recomendaciones de rediseño para la mejora.

7.1.2.1. DFA

Diseño para el ensamblaje (DFA). Es un método de diseño en el que se pretende simplificar la estructura del producto mediante una reducción en el número de piezas que lo componen. Este método se realiza con la ayuda de unos criterios contra los que se compara cada una de las piezas.

Es importante que en esta etapa el equipo de trabajo sea multidisciplinar, es decir, esté formado por personas de procedencia variada dentro de la fábrica, en concreto debe haber componentes del departamento de fabricación además de los habituales del departamento de diseño.

7.1.2.2. DFM

Diseño para fabricación (DFM) se desarrolla con una estimación temprana de los costes de fabricación de los componentes mediante una selección conjunta de los materiales y los procesos productivos. En esta etapa se trabaja con los componentes anteriormente definidos, y se pretende determinar los procesos productivos y materiales que harán mas económica la fabricación del producto.

7.1.2.3. DFMA

Design for Manufacture and Assembly (Diseño para la fabricación y el ensamblaje) es una técnica que permite analizar de forma sistemática cualquier diseño propuesto anteriormente. En pocas palabras es la aplicación de los principios de DFM y DFA, unidos para el análisis de diseño de un producto. A partir de este análisis se proponen posibles modificaciones del diseño existente que reducen el número de piezas y los tiempos de montaje, con ellos también se reducen el tiempo de desarrollo del producto y su coste.

Estas metodologías y herramientas fueron introducidas en la industria por el Dr. Boothroyd y el Dr. Dewhurst in 1983. De hecho, ellos son los propietarios de la marca registrada DFMA. La metodología DFMA consta de unos criterios de aplicación sistemática y una serie de principios o guías. Estos principios sirven como ayuda a la hora de tomar decisiones durante el diseño de un elemento concreto.

La técnica se basa en los principios siguientes:

- Diseño de un componente base (chasis).
- Diseño modular.
- Todas las operaciones de montaje deben hacerse en una dirección, a ser posible, verticalmente.
- Favorecer el uso de componentes multifuncionales.
- Eliminar los ajustes cuando sea posible.
- Proveer a los componentes de partes que los hagan autoposicionables.
- Proveer acceso directo a todos los submontajes.
- Minimizar los niveles de ensamblado.
- Facilitar la orientación de los componentes haciéndolos lo más simétricos posible.

El esquema general de la metodología de trabajo DFMA es el que se muestra en la figura 7.1

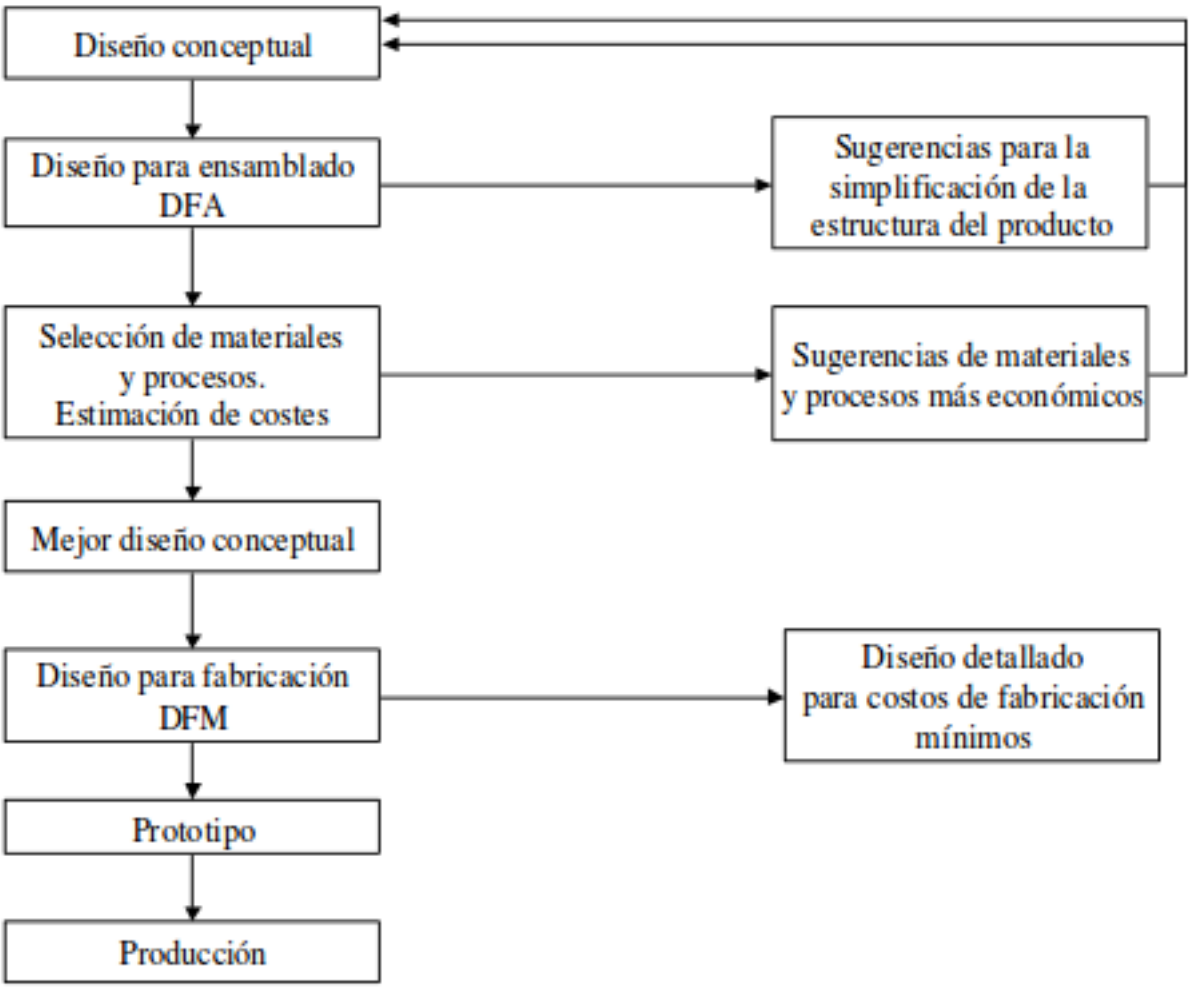


Figura 7.1: Esquema de diseño DFMA

Relación entre DFMA e Ingeniería Concurrente DFMA es fundamentalmente la aplicación de la Ingeniería Concurrente orientada al producto, a las primeras etapas del ciclo de vida y al ahorro de costes.

Conclusiones del uso de DFMA La aplicación de las técnicas DFMA en la etapa de diseño permite:

- Facilitar las operaciones de fabricación y montaje.
- Disminuir los costes de fabricación y montaje.
- Disminuir las inversiones y los costes de utillajes.
- Optimizar el uso de las herramientas y equipos de fabricación y montaje.
- Disminuir los costes de gestión.

- Aumentar la flexibilidad de la fabricación.
- Aumentar la configurabilidad de los productos.
- Disminuir el tiempo de introducción en el mercado.
- Disminuir los almacenajes intermedios, de expedición y la ocupación de espacios en general.
- Unificar componentes con la consiguiente disminución de costes de compra y menores referencias a gestionar.

Objetivos del DFMA respecto al montaje El montaje es una etapa que tiene un carácter integrador por excelencia en el que se detecta de forma inmediata una parte muy importante de los defectos de concepción de un producto y de fabricación de sus piezas. En el montaje confluyen un conjunto complejo de operaciones que hay que analizar cuidadosamente para su optimización y aplicación del DFMA. El montaje de un producto o de una máquina está relacionado tanto con la productividad y disminución de costes, como con la funcionalidad y la calidad, para ello Boothroyd y Dewhurst crearon una serie de análisis para la mejora de este proceso.

Uno de los detalles más importantes que podemos destacar en la evaluación del proceso de montaje son sus defectos, bajo los cuales se relaciona el tiempo que se tarda en montar cada pieza. Los defectos más frecuentes que afectan en las operaciones de montaje son:

- Defectos que inciden en las operaciones de manipulación:
  - Dificultad en el reconocimiento y referencia de piezas
  - Dificultad de prensión
  - Dimensiones o formas de difícil manipulación
  - Roturas en la manipulación y en la inserción
- Defectos que inciden en las operaciones de composición:
  - Errores dimensionales y de forma
  - Elementos deformados
  - Tolerancias excesivamente críticas
  - Falta de referenciación en la yuxtaposición de elementos
  - Falta de elementos de guía en las inserciones
- Defectos que inciden en las operaciones de unión:
  - Acceso difícil a los puntos de unión
  - Limitaciones en los movimientos para la unión
  - Incorrecto encaje de las piezas
  - Contaminación de superficies

Inclusión de metodología DFMA El método de Boothroyd Dewhurst permite cuantificar la eficiencia del diseño haciendo un análisis del producto. Esta eficiencia compara el tiempo de ensamblaje total de un producto real con un tiempo de ensamblaje de un producto ideal, determinado por los autores. La eficiencia del diseño puede ser utilizada para comparar diversos diseños y así evaluar sus eficiencias relativas. Hay que recalcar que esta interpretación de la eficiencia es evaluada sólo con carácter de ensamblaje manual.

Para el desarrollo de esta metodología hay que tener en cuenta dos consideraciones:

1. Hay que tener en cuenta que cualquier pieza puede ser candidata a ser eliminada o combinada con otras piezas del ensamblaje. El resto de las piezas, que no pueden ser eliminadas, se denominan piezas críticas.

2. Hay que reconocer el tiempo de manipulación y de inserción de los componentes que forman el producto. Estos tiempos vienen dados por una serie de características de aplicación de la pieza por parte del trabajador y los valores de tiempos vienen determinados por una serie de recetas heurísticas que finalmente nos devuelven el tiempo utilizado.

El procedimiento a seguir si se desea analizar un producto utilizando este método, consta de una serie de pasos los cuales se describen a continuación:

1. Obtener los detalles del diseño

   En este paso se adquieren conocimientos detallados sobre el producto analizado. Esta adquisición de conocimientos puede venir dada por planos, vistas 3D, el producto existente físicamente o un prototipo...

2. Diseño de ensamblaje

   Es necesario identificar cada componente del producto analizando la posibilidad de eliminar las partes o combinarlas con otras para simplificar el diseño. Es importante considerar los subensamblajes como piezas y analizarlos por separado. Este es el momento en el que se define el número de piezas criticas que tiene nuestro producto.

3. Re-Ensamblado

   Volver a ensamblar el producto prestando atención a los tiempos de ensamblaje de cada pieza, para así poder conocer el tiempo que se tarda en el ensamblaje (manual) del producto. Este método cuenta con que sólo se puede ensamblar una pieza al mismo tiempo, aunque en la realidad, a veces, el operario puede ensamblar piezas a la vez con cada mano, sin embargo si se eliminase ese tiempo de ensamblado, también habría que eliminarlo de nuestro producto ideal, por lo tanto se mantendría el valor de la eficiencia del diseño constante.

   DFMA se fundamenta en las siguientes tablas, y así se aplican en el sistema experto que se ha desarrollado:

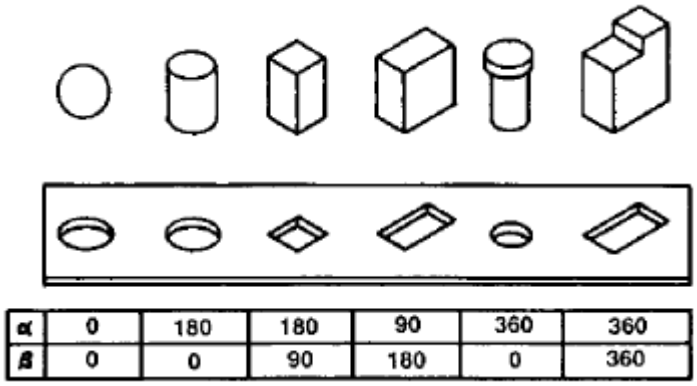


Figura 7.2: DFMA - Evaluación de simetrías (imágenes sacadas de [20])

MANUAL HANDLING — ESTIMATED TIMES (seconds)

Key.

ONE HAND

parts can be grasped and manipulated by one hand without the aid of grasping tools		parts are easy to grasp and manipulate: thickness > 2 mm, size > 15 mm	6 mm ≤ size ≤ 15 mm	size < 6 mm	thickness ≤ 2 mm, size > 6 mm	size ≤ 6 mm	parts present handling difficulties (1): thickness > 2 mm, size > 15 mm	6 mm ≤ size ≤ 15 mm	size < 6 mm	thickness ≤ 2 mm, size > 6 mm	size ≤ 6 mm
		0	1	2	3	4	5	6	7	8	9
$(\alpha+\beta) < 360°$	0	1.13	1.43	1.88	1.69	2.18	1.84	2.17	2.65	2.45	2.98
$360° \le (\alpha+\beta) < 540°$	1	1.5	1.8	2.25	2.06	2.55	2.25	2.57	3.06	3	3.38
$540° \le (\alpha+\beta) < 720°$	2	1.8	2.1	2.55	2.36	2.85	2.57	2.9	3.38	3.18	3.7
$(\alpha+\beta) = 720°$	3	1.95	2.25	2.7	2.51	3	2.73	3.06	3.55	3.34	4

ONE HAND with GRASPING AIDS

parts can be grasped and manipulated by one hand but only with the use of grasping tools				parts need tweezers for grasping and manipulation: parts can be manipulated without optical magnification, parts are easy to grasp and manipulate, thickness > 0.25 mm	thickness ≤ 0.25 mm	parts present handling difficulties (1), thickness > 0.25 mm	thickness ≤ 0.25 mm	parts require optical magnification for manipulation: parts are easy to grasp and manipulate, thickness > 0.25 mm	thickness ≤ 0.25 mm	parts present handling difficulties (1), thickness > 0.25 mm	thickness ≤ 0.25 mm	parts need standard tools other than tweezers	parts need special tools for grasping and manipulation
				0	1	2	3	4	5	6	7	8	9
	$\alpha \le 180°$	$0 \le \beta \le 180°$	4	3.6	6.85	4.35	7.6	5.6	8.35	6.35	8.6	7	7
		$\beta = 360°$	5	4	7.25	4.75	8	6	8.75	6.75	9	8	8
	$\alpha = 360°$	$0 \le \beta \le 180°$	6	4.8	8.05	5.55	8.8	6.8	9.55	7.55	9.8	8	9
		$\beta = 360°$	7	5.1	8.35	5.85	9.1	7.1	9.55	7.85	10.1	9	10

TWO HANDS for MANIPULATION

		parts present no additional handling difficulties: $\alpha \le 180°$, size > 15 mm	6 mm ≤ size ≤ 15 mm	size < 6 mm	$\alpha = 360°$, size > 6 mm	size ≤ 6 mm	parts present additional handling difficulties (e.g. sticky, delicate, slippery, etc.) (1): $\alpha \le 180°$, size > 15 mm	6 mm ≤ size ≤ 15 mm	size < 6 mm	$\alpha = 360°$, size > 6 mm	size ≤ 6 mm
		0	1	2	3	4	5	6	7	8	9
parts severely nest or tangle or are flexible but can be grasped and lifted by one hand (with the use of grasping tools if necessary) (2)	8	4.1	4.5	5.1	5.6	6.75	5	5.25	5.85	6.35	7

TWO HANDS required for LARGE SIZE

		parts can be handled by one person without mechanical assistance: parts do not severely nest or tangle and are not flexible: part weight < 10 lb, parts are easy to grasp and manipulate, $\alpha \le 180°$	$\alpha = 360°$	parts present other handling difficulties (1), $\alpha \le 180°$	$\alpha = 360°$	parts are heavy (> 10 lb): parts are easy to grasp and manipulate, $\alpha \le 180°$	$\alpha = 360°$	parts present other handling difficulties (1), $\alpha \le 180°$	$\alpha = 360°$	parts severely nest or tangle or are flexible (2)	two persons or mechanical assistance required for parts manipulation
		0	1	2	3	4	5	6	7	8	9
two hands, two persons or mechanical assistance required for grasping and transporting parts	9	2	3	2	3	3	4	4	5	7	9

Figura 7.3: DFMA - Tiempos de manipulación(imágenes sacadas de [20])

Gracias a la caracterización de la forma de cada pieza se puede determinar sus simetrías (valores alfa y beta). Estos valores serán clave en la selección del tiempo de manipulación de la pieza como veremos en otras tablas, en concreto en la tabla

de manipulación. Los valores de alfa y beta determinarán en la mayoría de los casos los valores de la primera cifra del código de manipulación.

Como se aprecia en la figura 7.3 para calcular el tiempo de manipulación existe un primer filtrado el cual es el número de manos utilizados en la manipulación de la pieza a ensamblar. Este primer filtro es:

- Una mano
- Una mano con utensilios de sujeción
- Dos manos
- Dos manos siendo la pieza de gran tamaño

Posteriormente como se aprecia en la tabla se siguen haciendo selecciones que definen con exactitud el código de manipulación y su equivalente, el tiempo de manipulación, en segundos. Se procede a hacer un esquema rápido para los valores superiores (segunda cifra del código de manipulación):

- Una mano:
  - Las piezas son fáciles de agarrar.
  - Las piezas presentan dificultades de agarre.
    - La pieza tiene un grosor mayor a 2mm.
    - La pieza tiene un grosor menor o igual a 2mm.
      - El tamaño es menor a 6mm.
      - El tamaño es mayor o igual a 6mm y menor o igual a 15mm.
      - El tamaño es mayor a 15mm.
- Una mano con utensilios de sujeción:
  - La pieza necesita de pinzas para su manipulación.
  - La pieza necesita herramientas comunes para su manipulación.
  - La pieza necesita de herramientas especiales para su manipulación.
    - Las piezas son fáciles de manipular.
    - Las piezas presentan dificultades en su manipulación.
      - El grosor de la pieza es menor o igual a 0.25mm.
      - El grosor de la pieza es mayor a 0.25mm.
- Dos manos:
  - Las piezas presentan dificultades adicionales en su manipulación.
  - Las piezas no presentan dificultades adicionales.
    - Simetría alfa menor o igual a 180º.
    - Simetría alfa igual a 360º.

- Dos manos de gran tamaño:
  - Puede ser manipulado por una persona sin asistencia mecánica.
  - Es necesario de asistencia mecánica para su manipulación.
    - Las piezas no se enredan entre si ni son flexibles.
    - Las piezas presentan enredos entre si y/o son flexibles.
  - La pieza pesa menos de 10 Lb (4.5Kg).
  - La pieza pesa más de 10 Lb (4.5Kg).
    - Simetría alfa menor o igual a 180º.
    - Simetría alfa igual a 360º.

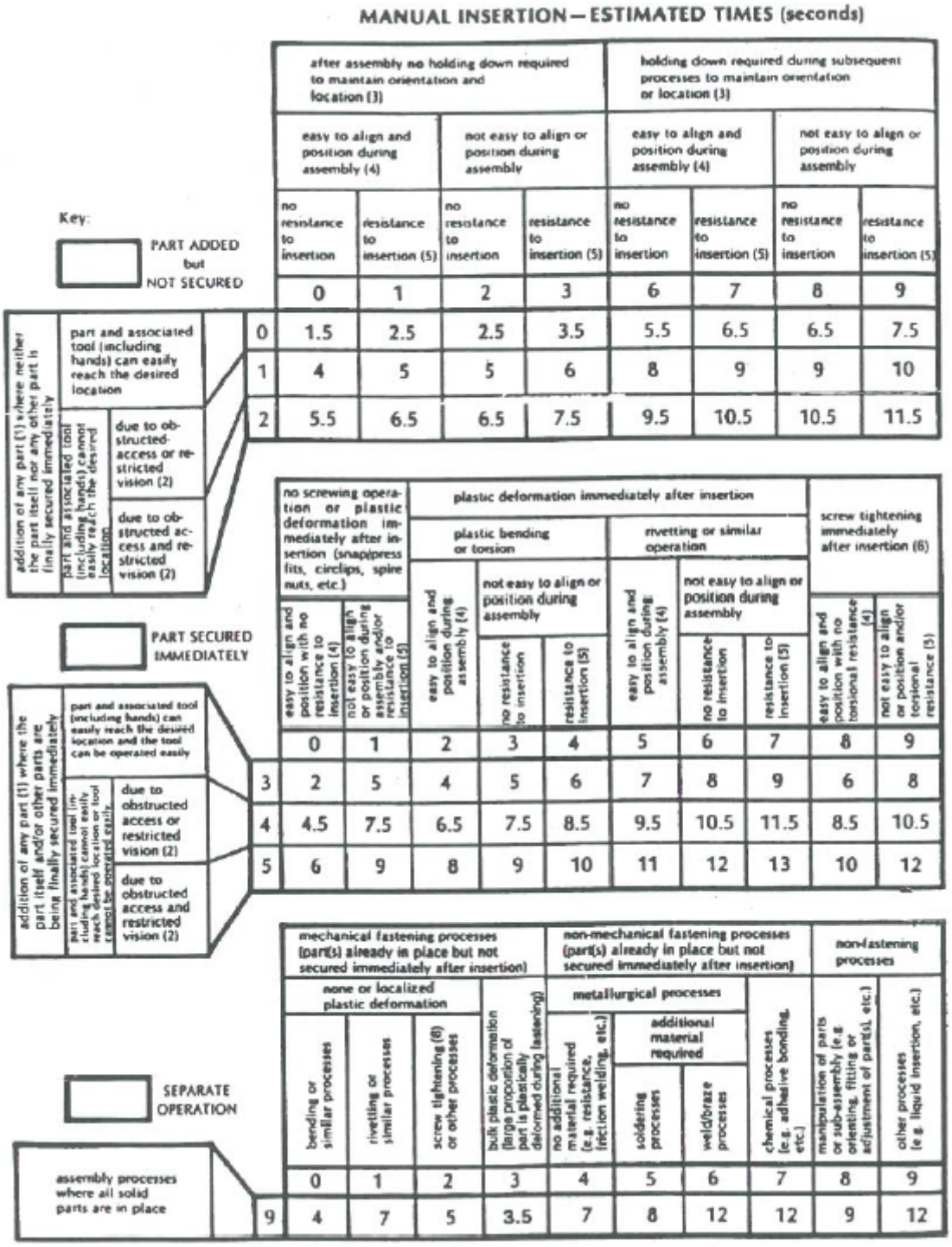
Para la selección del código de inserción con su respectivo valor utilizaremos la tabla de la figura 7.4 la cual funciona de la misma manera que la anterior.

Valores filas (primer dígito del código de inserción):

- La pieza se añade pero no se fija inmediatamente.
- La pieza se fija inmediatamente.
- La pieza forma parte de un proceso de fabricación distinto.

Valores columna (segundo dígito del código de inserción):

- Fácil localización.
  - Visión o acceso difícil.
  - Visión y acceso difícil.
    - No necesita sujeción después de la inserción.
    - Necesita sujeción después de la inserción.
  - Su alineación es fácil.
  - Su alineación es difícil.
    - Es necesario fuerza para aplicar la inserción porque la pieza ofrece resistencia.
    - No es necesaria la fuerza para aplicar la inserción puesto que la pieza no ofrece ningún tipo de resistencia.



MANUAL INSERTION — ESTIMATED TIMES (seconds)

Key:

PART ADDED but NOT SECURED

PART SECURED IMMEDIATELY

SEPARATE OPERATION

		after assembly no holding down required to maintain orientation and location (3)				holding down required during subsequent processes to maintain orientation or location (3)			
		easy to align and position during assembly (4)		not easy to align or position during assembly		easy to align and position during assembly (4)		not easy to align or position during assembly	
		no resistance to insertion	resistance to insertion (5)	no resistance to insertion	resistance to insertion (5)	no resistance to insertion	resistance to insertion (5)	no resistance to insertion	resistance to insertion (5)
		0	1	2	3	6	7	8	9
0	addition of any part (1) where neither the part itself nor any other part is finally secured immediately; part and associated tool (including hands) can easily reach the desired location	1.5	2.5	2.5	3.5	5.5	6.5	6.5	7.5
1	addition of any part (1) where neither the part itself nor any other part is finally secured immediately; part and associated tool (including hands) cannot easily reach the desired location; due to obstructed access or restricted vision (2)	4	5	5	6	8	9	9	10
2	addition of any part (1) where neither the part itself nor any other part is finally secured immediately; part and associated tool (including hands) cannot easily reach the desired location; due to obstructed access and restricted vision (2)	5.5	6.5	6.5	7.5	9.5	10.5	10.5	11.5

		no screwing operation or plastic deformation immediately after insertion (snap/press fits, circlips, spine nuts, etc.)		plastic deformation immediately after insertion						screw tightening immediately after insertion (6)	
				plastic bending or torsion			rivetting or similar operation				
				easy to align and position during assembly (4)	not easy to align or position during assembly		easy to align and position during assembly (4)	not easy to align or position during assembly			
		easy to align and position with no resistance to insertion (4)	not easy to align or position during assembly and/or resistance to insertion (5)		no resistance to insertion	resistance to insertion (5)		no resistance to insertion	resistance to insertion (5)	easy to align and position with no torsional resistance (4)	not easy to align or position and/or torsional resistance (5)
		0	1	2	3	4	5	6	7	8	9
3	addition of any part (1) where the part itself and/or other parts are being finally secured immediately; part and associated tool (including hands) can easily reach the desired location and the tool can be operated easily	2	5	4	5	6	7	8	9	6	8
4	addition of any part (1) where the part itself and/or other parts are being finally secured immediately; part and associated tool (including hands) cannot easily reach desired location or tool cannot be operated easily; due to obstructed access or restricted vision (2)	4.5	7.5	6.5	7.5	8.5	9.5	10.5	11.5	8.5	10.5
5	addition of any part (1) where the part itself and/or other parts are being finally secured immediately; part and associated tool (including hands) cannot easily reach desired location or tool cannot be operated easily; due to obstructed access and restricted vision (2)	6	9	8	9	10	11	12	13	10	12

		mechanical fastening processes (part(s) already in place but not secured immediately after insertion)				non-mechanical fastening processes (part(s) already in place but not secured immediately after insertion)				non-fastening processes	
		none or localized plastic deformation				metallurgical processes					
							additional material required				
		bending or similar processes	rivetting or similar processes	screw tightening (6) or other processes	bulk plastic deformation (large proportion of part is plastically deformed during fastening)	no additional material required (e.g. resistance, friction welding, etc.)	soldering processes	weld/braze processes	chemical processes (e.g. adhesive bonding, etc.)	manipulation of parts or sub-assembly (e.g. orienting, fitting or adjustment of part(s), etc.)	other processes (e.g. liquid insertion, etc.)
		0	1	2	3	4	5	6	7	8	9
9	assembly processes where all solid parts are in place	4	7	5	3.5	7	8	12	12	9	12

Figura 7.4: DFMA - Tabla de los tiempos de inserción(imágenes sacadas de [20])

4. Calcular el índice de eficiencia

   El último paso es calcular el índice de eficiencia del producto analizado. Este índice muestra de forma porcentual el valor de nuestro producto (a mayor valor del porcentaje, mejor diseño según el método de diseño para la fabricabilidad y el ensamblaje).

Esta método es secuencial por lo que para obtener este índice hay que haber realizado los pasos anteriores. Para el cálculo hay que utilizar la siguiente fórmula:

Siendo:

EM El índice de eficiencia del diseño según DFMA.

NM El número mínimo de piezas teóricas. Suma de piezas críticas.

TM Tiempo total empleado por el operario para el ensamblaje.

$$EM = \frac{3 * NM}{TM}$$

Figura 7.5: Índice de eficiencia del diseño según DFMA

7.2. ISO 10303 - Ficheros STEP

La ISO 10303 es un Standard internacional para la interpretación de ficheros de representación de productos e intercambio de datos. El objetivo de esta norma es tener un mecanismo capaz de detallar toda la información del ciclo de vida de un producto.

El tipo de fichero STEP es utilizado para el intercambio de ficheros entre sistemas CAD, CAM, CAE y otros CAx.

STEP está dividido en varias partes, agrupadas por:

1. ENTORNO
   - Partes 1x: Métodos descrptivos: EXPRESS, EXPRESS-X.
   - Partes 2x: Métodos de implementación: STEP-FIle, STEP-XML, SDAI.
   - Partes 3x: Metodología de pruebas y framework.
2. RECURSOS INTEGRADOS (IR)
   - Partes 4x y 5x: Recursos integrados genéricos.
   - Partes 1xx: Recursos integrados para aplicación.
   - Librería PLIB ISO 13584-20: Modelos lógicos de expresiones.
   - Partes 5xx: Estructura integrada de aplicación (AIC).
   - Partes 1xxx: Módulos de aplicación (AM).
3. CABECERAS
   - Partes 2xx: Protocolos de aplicación (AP)
   - Partes 3xx: Abstract Test Suites (ATS) para las AP's.
   - Partes 4xx: Módulos de implementación para las AP's.

7.2.1. ISO 10303-11 EXPRESS

La parte 11 de la ISO también es conocida como EXPRESS, es un lenguaje de programación STEP similar a PASCAL. Puede ser representado gráficamente (denominándose EXPRESS-G) o en lenguaje ASCII.

Un ejemplo de este lenguaje, en formato ASCII es el siguiente:

```
SCHEMA Family;
    ENTITY Person
        ABSTRACT SUPERTYPE OF (ONEOF (Male, Female));
            name: STRING;
            mother: OPTIONAL Female;
            father: OPTIONAL Male;
    END_ENTITY;
    ENTITY Female
        SUBTYPE OF (Person);
    END_ENTITY;
    ENTITY Male
        SUBTYPE of (Person);
    END_ENTITY;
END_SCHEMA;
```

Y el mismo ejemplo de estructura de programación pero en formato gráfico, que como anteriormente hemos descrito se denomina EXPRESS-G, tendría la forma que se muestra en la figura 7.6 .

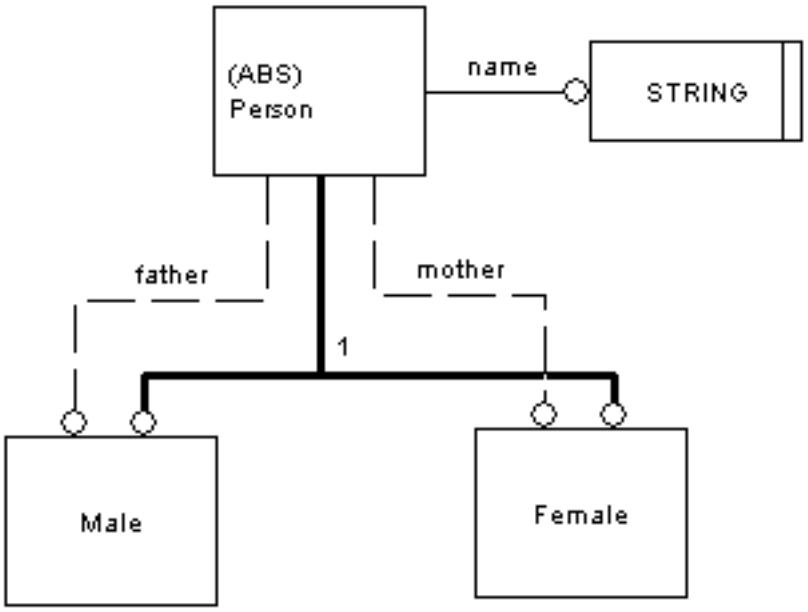


Figura 7.6: Ejemplo de programación en EXPRESS-G

Este modelo describe el esquema EXPRESS Familia. Contiene una entidad supertipo llamada Persona y dos subtipos llamadas Hombre y Mujer. El bloque Persona por si mismo es un abstracto y sólo puede tener un valor, el cual puede ser Hombre o Mujer. Además estos valores tienen unos atributos opcionales los cuales son Padre y Madre respectivamente. Todo esto podría leerse de la siguiente manera:

- Una Mujer es una Persona que puede ser Madre.
- Un Hombre es una Persona que puede ser Padre.

Tipos de datos

- Entity data type: es el tipo de datos más importante del lenguaje EXPRESS. Puede ser relacionado en forma jerárquica con supertipos y subtipos y también se relacionan con atributos.
- Enumeration data type: Son cadenas relacionadas entre si. Como por ejemplo de una enumeración rgb, el rojo, el verde y el azul. Pueden ser usados en otros esquemas.
- Defined data type: Detalla otros tipos de datos por ejemplo: positivo es un tipo de dato definido por un entero con valor > 0.
- Select data type: Define una alternativa o elección entre diferentes opciones. Normalmente se utiliza para la elección de distintas entidades. En el caso anterior la elección entre Hombre y Mujer.
- Simple data type: Hay varios tipos de datos simples, son los siguientes:
  - String: Es el tipo más utilizado. Son cadenas de cualquier longitud y cualquier tipo de carácter (incluido en la ISO 10646/Unicode).
  - Binary: Poco usado. Número de bits. Para algunos sistemas el límite es de 32 bits.
  - Boolean: Valores True o False
  - Logical: Mismos valores que Boolean añadiendo Uknown.
  - Integer: La denominación para el tipo de datos Enteros. En algunos sistemas su longitud máxima es de 32bits.
  - Real: Definición de los números reales.
  - Number: Es un tipo de dato que a su vez es un supertipo de Integer y Real.
- Aggregation data type: Pueden ser SET, BAG, LIST y ARRAY. Los dos primeros no tienen orden, al contrario que LIST y ARRAY que tienen un orden definido. SET no puede contener mas de un valor repetido al contrario que BAG. Lo que diferencia ARRAY del resto es que sólo puede añadir datos no definidos con anterioridad.

Cabe constatar que los tipos de datos pueden ser definidos dentro del esquema EXPRESS. Se usan para definir entidades, para especificar el tipo de atributos y agregar valores.

Pueden ser utilizadas en cadenas recursivas, pueden definirse un List dentro de un Array dentro de un Select.

Atributos

Los atributos de las entidades permiten añadir propiedades a las entidades. El nombre del atributo especifica el rol.

Hay tres tipos de atributos, los cuales pueden ser redeclarados en subtipos, son los siguientes:

- Explicit attributes: Son los que tienen los valores visibles directamente en el fichero STEP.
- Derived attributes: Obtienen sus valores a partir de una expresión, normalmente de otro atributo.
- Inverse attributes: No añaden información a la entidad pero nombran a otra entidad.

7.2.2. ISO 10303-21

El apartado 21 de la ISO, es un método de implementación del lenguaje EXPRESS. Es la forma más común de encontrar los ficheros STEP.

La ISO 10303-21 define la forma de representación del esquema EXPRESS formando una estructura de fácil comprensión definiendo una instancia por linea.

Este tipo de ficheros pueden ser definidos como .p21 o archivo físico STEP; También la extensión de fichero .stp o .step indican que son ficheros de protocolo STEP.

Un ejemplo sencillo sobre la implementación de la parte 21 de la normativa es la siguiente:

```
1  ISO-10303-21;
2  HEADER;
3  FILE_DESCRIPTION(
4  /* description */ ('A minimal AP214 example with a single part'),
5  /* implementation_level */ '2;1');
6  FILE_NAME(
7  /* name */ 'demo',
8  /* time_stamp */ '2003-12-27T11:57:53',
9  /* author */ ('Lothar Klein'),
10 /* organization */ ('LKSoft'),
11 /* preprocessor_version */ ' ',
12 /* originating_system */ 'IDA-STEP',
13 /* authorization */ ' ');
14 FILE_SCHEMA (('AUTOMOTIVE_DESIGN { 1 0 10303 214 2 1 1}'));
15 ENDSEC;
16 DATA;
17 #10=ORGANIZATION('O0001','LKSoft','company');
```

```
#11=PRODUCT_DEFINITION_CONTEXT('part definition',#12,'manufacturing');
#12=APPLICATION_CONTEXT('mechanical design');
#13=APPLICATION_PROTOCOL_DEFINITION('','automotive_design',2003,#12);
#14=PRODUCT_DEFINITION('0',$,#15,#11);
#15=PRODUCT_DEFINITION_FORMATION('1',$,#16);
#16=PRODUCT('A0001','Test Part 1','',(#18));
#17=PRODUCT_RELATED_PRODUCT_CATEGORY('part',$,(#16));
#18=PRODUCT_CONTEXT('',#12,'');
#19=APPLIED_ORGANIZATION_ASSIGNMENT(#10,#20,(#16));
#20=ORGANIZATION_ROLE('id owner');
ENDSEC;
END-ISO-10303-21;
```

Se puede analizar la estructura del código diferenciando dos grandes bloques, los cuales se definen como HEADER y DATA.

El bloque HEADER contiene a su vez una subestructura con tres campos principales y otros tres secundarios.

- HEADER
  - FILE_DESCRIPTION
    - description

      Aquí se describe la finalidad del archivo STEP
    - implementation_level

      Se define la versión: “1” para versiones anteriores al año 1994, “2” para la corrección de 1995, y “3” para la segunda edición de la ISO 10303-21. Normalmente el valor es “2:1”. Valores “2:2” son raros por utilizar ficheros externos. Valores “3:1” y “3:2” proceden de la ISO 10303-21:2001 y soportan las tres subestructuras del header secundarias.
  - FILE_NAME
    - name

      Nombre del fichero en el sistema de archivos.
    - time_stamp

      Indica la fecha cuando fue creado el documento en formato ISO 8601. Ejemplo: 2013-05-06T10:40:51.
    - author

      Nombre del autor del archivo.
    - organization

      Organización para la que trabaja el autor.
    - preprocessor_version

      Define el nombre del sistema y la versión que genera el fichero STEP.

- originating_system

  Primer sistema que generó el fichero STEP y su respectiva versión.

- authorization

  Nombre y/o dirección de la persona que autoriza el uso del archivo.

- FILE_SCHEMA

  Define el esquema EXPRESS que va a ser utilizado en la sección DATA. A partir de la segunda versión se pueden incluir más de un esquema EXPRESS.

Los siguientes tres grupos sólo son válidos en la segunda versión de la ISO.

- FILE_POPULATION

  Indica un grupo de entidades (ENTITY) que forman el documento.

  - governing_schema

    Es el esquema EXPRESS para el grupo de entidades indicados anteriormente.

  - determination_method

    Es el método utilizado que puede ser: SECTION_BOUNDARY, INCLUDE_ALL_COMPA o INCLUDE_REFERENCED.

  - governed_sections

    Indica las secciones en las cuales funcionan las entidades.

- SECTION_LANGUAGE

  Se indica el lenguaje por defecto.

- SECTION_CONTEXT

  En este apartado se puede incluir información complementaria para todas o para cada sección del fichero.

El bloque DATA contiene el código de un esquema EXPRESS específico, y se fundamenta en una serie de principios.

- DATA

  - Instance name: Cada instancia de la estructura viene definido por un nombre con el formato "#num", este número es un entero positivo. Este nombre sólo es válido para el documento actual, puesto que si el mismo contenido es exportado a otro sistema el nombre podría cambiar. Este nombre es utilizado como referencia desde otras instancias que pueden ser anteriores o posteriores al nombre.
  - Las instancias de una entidad vienen representadas con el nombre escrito en letras mayúsculas , seguido de los atributos ordenados dentro de un bloque de paréntesis. El formato es el siguiente: #num=ENTIDAD(atributos...)
  - Las instancias de entidades de datos más complejas de un fichero STEP pueden ser referenciadas a un mapeado externo.

7.2.3. ISO 10303-28

La parte 28 de la ISO 10303 también es conocida como STEP-XML, es un método de implementación XML para la representación de datos y esquemas STEP.

STEP-XML especifica el uso de programación XML para la representación de los esquemas en lenguaje EXPRESS (ISO 10303-11). Es un método alternativo al fichero STEP.

La norma ISO 10303-28 hace que los documentos STEP sean más sencillos de comprender, gracias a su estructuras de árbol definidas. Un ejemplo de este método de implementación de lenguaje EXPRESS, es el siguiente:

```
1  <?xml version="1.0"?>
2    <iso_10303_28_terse
3        xmlns="urn:oid:1.0.10303.238.1.0.1"
4        xmlns:exp="urn:oid:1.0.10303.28.2.1.1"
5        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
6        schema="integrated_cnc_schema">
7
8      <exp:header>
9        <exp:name>dataset</exp:name>
10       <exp:preprocessor_version>stepnc writer</
           exp:preprocessor_version>
11     </exp:header>
12
13       <!-- instance data -->
14     <Axis2_placement_3d id="id1" Name="" Location="id2" Axis="id3"
           Ref_direction="id4"/>
15     <Cartesian_point id="id2" Name="loc" Coordinates="3.5 -3.5
           -4.16875"/>
16     <Direction id="id3" Name="Z direction" Direction_ratios="0 0 1"/>
17     <Direction id="id4" Name="X direction" Direction_ratios="1 0 0"/>
18   </iso_10303_28_terse>
```

7.3. Software CAD

Los programas informáticos CAD (Computer-aided design) conocidos como programas de diseño asistido por ordenador, son herramientas para la asistencia de ingenieros, arquitectos y profesionales del diseño en sus respectivas labores. El CAD también es utilizado en el marco de procesos de administración del ciclo de vida de los productos, PLM (Product lifecycle management).

Estas herramientas se pueden dividir básicamente en programas de dibujo 2D y modelado en 3D. Las herramientas 2D se basan en entidades geométricas vectoriales como puntos, lineas, arcos y polígonos, operadas bajo una interfaz gráfica. Las aplicaciones de modelado tridimiensional añaden a esta lista de entidades las superficies y los sólidos.

El usuario puede asociar cada entidad una serie de propiedades como el color, estilo de linea, nombre... esto permite manejar la información de forma lógica. Además los programas

3D permiten renderizar el producto que básicamente es una previsualización fotorrealísitica del producto.

Proceso de diseño CAD

El proceso de diseño en CAD consiste en cuatro etapas:

1. Modelado geométrico.

   Se describe como forma matemática o analítica a un objeto físico, el diseñador construye su modelo geométrico emitiendo comandos que crean o perfeccionan líneas, superficies, cuerpos, dimensiones y texto; que dan a origen a una representación exacta y completa en dos o tres dimensiones.

2. Análisis y optimización del diseño.

   Después de haber determinado las propiedades geométricas, se somete a un análisis ingenieril donde podemos analizar las propiedades físicas del modelo (esfuerzos, deformaciones, deflexiones, vibraciones). Se disponen de sistemas con la capacidad de recrear con exactitud y rapidez esos datos.

3. Revisión y evaluación del diseño.

   En esta etapa importante se comprueba si existe alguna interferencia entre los diversos componentes, en útil para evitar problemas en el ensamble y el uso de la pieza. Para esto existen programas de animación o simulaciones dinámicas para el cálculo de sus tolerancias y ver que requerimientos son necesarios para su fabricación.

4. Documentación y dibujo (drafting).

   Por último, en esta etapa se realizan planos de detalle y de trabajo. Esto se puede producir en dibujos diferentes vistas de la pieza, manejando escalas en los dibujos y efectúa transformaciones para presentar diversas perspectivas de la pieza.

Actualmente el diseño asistido por ordenador es parte imprescindible en el proceso de desarrollo de productos.

8 Análisis de Soluciones

Durante el presente capítulo se describirá con detalle la solución del proyecto ante la necesidad creada de la evaluación de productos bajo el modelo de diseño para la fabricación y la ensamblabilidad integrado en un entorno CAD.

En la primera de las secciones que conforman el capítulo se define las características de la aplicación informática que se ha programado.

Posteriormente se analizan un listado de software existente para evaluación de DFMA detallando cada uno de sus puntos fuertes y de sus desventajas. En esta misma sección también se describe un software de traducción de ficheros STEP a ficheros XML, pues ha sido utilizada para testear el software.

Por último se especifica el lenguaje utilizado para la programación del software desarrollado, especificando sus características fundamentales. Dentro de este apartado han sido determinados los módulos y librerías utilizados más interesantes, como el método de interfaz gráfica llamado GLADE, los módulos de parseo de documentos XML como DOM, ETREE... por último una enumeración con su respectiva descripción de otros módulos utilizados en el desarrollo de la aplicación informática.

8.1. Diseño Inteligente

El Diseño Inteligente podría denominarse como una serie de tareas humanas que hace un ordenador u otra herramienta que el hombre ha tenido que configurar para resolver tal tarea. Para conseguir definir el Diseño Inteligente hay que tener claros una serie de conceptos. Entre ellas el concepto de Inteligencia Artificial y el de Sistema Experto.

8.1.1. Inteligencia Artificial

Según su definición se denomina Inteligencia Artificial a la capacidad de razonar de un agente no vivo.

Los sistemas de inteligencia artificial incluyen a las personas, los procedimientos, el hardware y software, los datos y los conocimientos necesarios para desarrollar sistemas, y máquinas de computación que presenten características de inteligencia. El objetivo del desarrollo de sistemas de IA contemporáneos no es el reemplazo completo de la toma de decisiones de los humanos, pero sí duplicarlas para ciertos tipos de problemas bien definidos.

La IA se divide en dos escuelas de pensamiento:

1. La inteligencia artificial convencional : Está basada en el análisis formal y estadístico del comportamiento humano ante diferentes problemas:

a) Razonamiento basado en casos: Ayuda a tomar decisiones mientras se resuelven ciertos problemas concretos y, aparte de que son muy importantes, requieren de un buen funcionamiento.

b) Sistemas expertos: Infieren una solución a través del conocimiento previo del contexto en que se aplica y ocupa de ciertas reglas o relaciones. Este es en el que está basado este proyecto y será descrito con mayor detenimiento más adelante.

c) Redes bayesianas: Propone soluciones mediante inferencia probabilística. Inteligencia artificial basada en comportamientos: esta inteligencia tiene autonomía y pueden auto-regularse y controlarse para mejorar. Smart process management: facilita la toma de decisiones complejas, proponiendo una solución a un determinado problema al igual que lo haría un especialista en la actividad.

2. La Inteligencia Computacional: Es la IA que implica desarrollo o aprendizaje interactivo y su aprendizaje se realiza basándose en datos empíricos.

A continuación se desarrolla el tipo de Inteligencia Artificial utilizado para la aplicación, que es el Sistema Experto, que como se a descrito anteriormente forma parte de la Inteligencia Artificial convencional.

8.1.2. Sistemas Expertos

Un Sistema Experto es, por definición, un conjunto de programas que, sobre una base de conocimientos, posee información de uno o más expertos en un área específica. Se puede entender como una rama de la inteligencia artificial, donde el poder de resolución de un problema en una aplicación informática viene del conocimiento de un experto en dominio específico.

Estos sistemas imitan las actividades de un humano para resolver problemas de distinta índole, en nuestro caso calcula el índice del diseño para la fabricabilidad y el ensamblaje, tal y como se denomina en el libro de Boothroyd Dewhurst, quien es en este caso quien a aportado el conocimiento experto.

Los sistemas expertos han de tener dos características principales:

1. Explicar la base del conocimiento:

   Los sistemas expertos se deben realizar siguiendo ciertas reglas o pasos comprensibles de manera que se pueda generar la explicación para cada una de estas reglas, que a la vez se basan en hechos.

2. Puede mostrar un comportamiento inteligente:

   Al examinar un grupo de datos, un SE puede proponer nuevas ideas o métodos para la solución del problema, o proporcionar asesoramiento en el trabajo previamente definido.

3. Puede obtener conclusiones de relaciones complejas:

   Evaluar relaciones complejas para llegar a conclusiones y solucionar problemas, por ejemplo: un SE propuesto trabajará con un sistema de fabricación flexible para determinar la mejor utilización de las herramientas, y otro sugerirá los mejores procedimientos de control de calidad.

4. Puede proporcionar conocimientos acumulados:

   Se puede usar para capturar conocimientos de humanos que de lo contrario podrían perderse. Ejemplo es el SE denominado DELTA (Diesel Electronic Locomotive Troble-shooting Aid), desarrollado para conservar el conocimiento de sobre reparaciones extremadamente técnicas.

5. Puede hacer frente a la incertidumbre:

   Una de las características más importantes de un SE es su capacidad para enfrentar conocimientos incompletos o inexactos en su totalidad. Para ello se utilizan métodos como el uso de las probabilidades, las estadísticas y las heurísticas.

6. Dispone de capacidades de diseño:

   Estos SE usan principios generales de diseño, comprensión de los procedimientos de fabricación y un grupo de reglas de diseño. Tal y como se desarrollan en este proyecto.

7. Utilidad de planificación:

   Puesto que pueden ayudar a este proceso mediante la sugerencia de factores que se deben considerar al tomar la decisión final, sobre la base de datos proporcionados por el programador.

Los sistemas expertos no intentan sustituir a los expertos humanos, sino que se desea ayudarlos a realizar con más rapidez y eficacia todas las tareas que realiza. Lo que diferencia a estos sistemas de un sistema tradicional de recuperación de información es que éstos últimos sólo son capaces de recuperar lo que existe explícitamente, mientras que un Sistema Experto debe ser capaz de generar información no explícita, razonando con los elementos que se le dan. Pero la capacidad de los SE en el ámbito de la recuperación de la información no se limita a la recuperación. Pueden utilizarse para ayudar al usuario, en selección de recursos de información, en filtrado de respuestas, etc. Un SE puede actuar como un intermediario inteligente que guía y apoya el trabajo del usuario final.

8.1.2.1. Desarrollo de Sistemas Expertos

El desarrollo de estos Sistemas Expertos es variado. El método seleccionado depende de los beneficios del sistema comparados con el costo, el control y la complejidad de cada alternativa. Es más fácil y menos costoso desarrollar un SE utilizando un paquete ya existente o un shell. Un shell (interprete) de SE es un grupo de paquetes y herramientas de software utilizados para diseñar, desarrollar, poner en operación y mantener SE con

una programación mínima.El usuario introduce los datos o parámetros apropiados y el SE proporciona el resultado para el problema o situación. Algunos de los shells más populares son Exsys de MultilLogic, Inc. Level 5, de Rule Machines Corporación y XpertRule, de Attar Software. Por tanto las opciones de desarrollo de este tipo de agentes es:

- Desarrollo en la propia empresa desde cero): Es el método más caro, pero la empresa tiene más control sobre las características y las entidades. Esta ha sido la elección por la que ha optado este proyecto ante el problema de la evaluación DFMA de productos.

- Desarrollo en la propia empresa con un interprete shell: Puede ser menos complejo y más fácil de mantener que desarrollarlo desde cero. Sin embargo, quizá sea necesario modificar el SE resultante para que sea idóneo para aplicaciones específicas.

- Compra paquetes/módulos ya existentes: Método menos costoso y más rápido. Aquel cuyo desarrollo corrió a cargo de una compañía de software o de asesoría.

8.2. Desarrollo de aplicaciones informáticas

En informática, una aplicación es un tipo de programa informático diseñado como herramienta para permitir a un usuario realizar uno o diversos tipos de trabajos. Los lenguajes de programación son la herramienta para desarrollar estas aplicaciones.

Las aplicaciones se utilizan para solventar procesos automáticos de ciertas tareas. Ciertas aplicaciones desarrolladas a medida suelen ofrecer una gran potencia ya que están exclusivamente diseñadas para resolver un problema específico, como es el caso de una herramienta para la evaluación de alternativas mediante la técnica DFMA, el cual es el motivo principal de este proyecto.

Hay diversos tipos o técnicas de programación. Estos tipos de programación se denominan paradigmas. Los más utilizados se proceden a explicar a continuación:

Programación estructurada (PE) La programación estructurada esta compuesta por un conjunto de técnicas que han ido evolucionando aumentando considerablemente la productividad del programa reduciendo el tiempo de depuración y mantenimiento del mismo.

Esta programación estructurada utiliza un número limitado de estructuras de control, reduciendo así considerablemente los errores.

Esta técnica incorpora:

- Diseño descendente (top-dow): el problema se descompone en etapas o estructuras jerárquicas.

- Recursos abstractos (simplicidad): consiste en descompones las acciones complejas en otras más simples capaces de ser resueltas con mayor facilidad.

- Estructuras básicas: existen tres tipos de estructuras básicas:

- Estructuras secuénciales: cada acción sigue a otra acción secuencialmente. La salida de una acción es la entrada de otra.
- Estructuras selectivas: en estas estructuras se evalúan las condiciones y en función del resultado de las mismas se realizan unas acciones u otras. Se utilizan expresiones lógicas.
- Estructuras repetitivas: son secuencias de instrucciones que se repiten un número determinado de veces.

Esto hace que los programas sean mas fáciles de entender, pues se reduce la complejidad de las pruebas y los programas queden mejor documentados internamente.

Lo más característico de este tipo de programación es que un programa esta estructurado si posee un único punto de entrada y sólo uno de salida. Cuando existen de "1 a n" caminos desde el principio hasta el fin del programa. Y por último, que todas las instrucciones son ejecutables sin que aparezcan bucles infinitos.

Programación modular En la programación modular se organizan varias secciones dividas de forma que interactúan a través de llamadas a procedimientos, que integran el programa en su totalidad.

En la programación modular, el programa principal coordina las llamadas a los módulos secundarios y pasa los datos necesarios en forma de parámetros.

A su vez cada modulo puede contener sus propios datos y llamar a otros módulos o funciones.

Programación orientada a objetos (POO) Se trata de una técnica que aumenta considerablemente la velocidad de desarrollo de los programas gracias a la reutilización de los objetos.

El elemento principal de este tipo de programación es el objeto. El objeto es un conjunto complejo de datos y programas que poseen estructura y forman parte de una organización. Se gana gran fluidez gracias al polimorfismo y la herencia, las cuales son unas de sus principales características.

Programación funcional Se caracteriza principalmente por permitir declarar y llamar a funciones dentro de otras funciones.

Programación lógica Se trata de una programación basada en el cálculo de predicados, en otras palabras, una teoría matemática que permite lograr que un ordenador basándose en hecho y reglas lógicas, pueda dar soluciones inteligentes.

8.3. Programación multiagente

Este tipo de programación se utiliza cuando tenemos que realizar varias acciones a la vez.

Se suele utilizar para controlar varios eventos que suceden al mismo tiempo como los accesos de usuarios y programas a un recurso de forma simultanea.

Aunque trata de una programación más lenta y laboriosa ha sido la utilizada para el desarrollo de la aplicación del proyecto por su gran funcionalidad.

Este tipo de programación al ser utilizada en el proyecto se describirá de forma detallada en el apartado correspondiente a SPADE, por ser la plataforma multiagente que se ha utilizado para el desarrollo del proyecto.

8.4. Descripción del Software

Cuando se analizó la problemática existente se optó por introducir un sistema de gestión para diseño DFMA que obtuviese directamente la mayor cantidad de información de los ficheros CAD para así poder ahorrar tiempo al usuario.

Para ello se han analizado las características y parámetros que describen un modelo DFMA. También se han estudiado la información que puede obtenerse bajo un fichero CAD del tipo ISO 10303-28, que como con anterioridad se han descrito son ficheros en lenguaje EXPRESS con una estructura XML. Y a partir de ello, se han realizado una serie de consideraciones.

El software debe ser adecuado y pertinente para ser incorporado al plan de trabajo de un equipo multidisciplinar y en un entorno de ingeniería concurrente. Por lo tanto debe tener un grado de standarización. Para solucionar este motivo se eligió el modelo de fichero CAD, EXPRESS-XML.

Una de estas consideraciones es la posibilidad de obtener parámetros muy útiles para la evaluación de DFMA, como es el número de piezas que forman el producto, parámetro fundamental en el diseño para el ensamblaje, pues como se describe en este modelo de diseño, a un menor numero de piezas para un producto con la misma funcionalidad goza de un diseño con mayor eficiencia.

Otra de las consideraciones es que muchos de los parámetros necesarios para la evaluación de DFMA no se encuentran en ningún tipo de fichero CAD. Algunos ejemplos de este tipo de datos son los referentes a datos de fabricación, por ejemplo: tiempo de fabricación, coste de mano de obra, etc. Estos datos han de ser introducidos manualmente. Para ello el software debe disponer de una interfaz gráfica para la introducción de determinados valores.

8.4.1. Programación en Python

Bajo estas lineas se describirán las aplicaciones y módulos que se han utilizado para la programación de la aplicación que motiva este proyecto. Se han clasificado por grupos según su función.

Como se definió en la introducción de esta memoria, el autor de esta memoria no tenía conocimientos sobre programación en lenguaje Python, pero este lenguaje fue elegido por ser un lenguaje a muy alto nivel, que permite expresar algoritmos casi de forma direc-

ta (denominado por algunos como "pseudocódigo ejecutable") es un lenguaje sencillo de aprender a utilizar y con mucho potencial.

8.4.1.1. Editores Python

Para programar en este lenguaje se han utilizado una interfaz de usuario llamada SPE "Stani's Python Editor" que es un interfaz sencillo que ayuda al usuario a ser mas eficiente en el desarrollo de código Python resaltando la sintaxis y mostrando ayuda contextual de las funciones y módulos.

SPE Este editor también crea de forma dinámica diagramas UML y otra serie de características muy útiles para programas de grandes cantidades de código y complejidad estructural.

Geany Durante el desarrollo del programa también se han utilizado otro tipo de editores como Geany que es otro IDE para Python muy útil y sencillo de utilizar.

Eclipse Eclipse también ha sido otra de las herramientas para crear el programa, con una interfaz más compleja y multiplataforma es una utilidad profesional para la creación de aplicaciones en cualquier lenguaje de programación, gracias a sus plugins

8.4.1.2. Paradigma Multiagente

Unas de las características fundamentales que caracterizan este proyecto es la implementación del paradigma multiagente. El proyecto comenzó como utilizando el paradigma de programación orientada a objetos, posteriormente se investigó sobre el paradigma multiagente y se incluyó en el proyecto por permitir la evolución futura de la aplicación. Un sistema multiagente es un sistema compuesto por múltiples agentes que interactúan entre ellos.

Para esto se ha utilizado una plataforma de sistemas multiagentes ya existente, la cual se procede a detallar.

SPADE SPADE, como anteriormente se ha mencionado, es una plataforma de sistemas multiagentes creada en Python en la Universidad Politécnica de Valencia en 2005. Esta plataforma nació como prueba de utilización de la mensajería instantánea como método de transporte de agentes inteligentes. Basada en protocolos standard como son XMPP/Jabber. Está basada en un entorno XML.

Una de sus características principales es que tiene un entorno gráfico web para administrar la plataforma. También permite su utilización multiplataforma. Y además posee una librería de agentes, una colección de clases, funciones y herramientas para crear nuevos agentes SPADE.Para su utilización es necesario hacer correr la plataforma principal con la aplicación *"runspade.py"*, esta debe estar activa durante toda la ejecución del programa. Esto activa el Canal de Comunicación entre Agentes (ACC), el Sistema de Gestión

de Agentes (AMS) y el Facilitador de Directorios (FD) que son agentes y entidades que permiten relacionar agentes para así crear el sistema multiagentes.

8.4.1.3. Entorno gráfico

El entorno gráfico es la forma de interacción que dispone el software para comunicarse con el usuario de forma gráfica. También se denominan GUI. Son un conjunto de imágenes y objetos gráficos para representar la información y las acciones disponibles en la interfaz. Su principal uso es el proporcionar un entorno visual sencillo, es la evolución de la linea de comandos.

Easygui es un módulo para Python que permite la implementación de un sistema de ventanas basados en Tkinter de la manera más sencilla que puede generarse. Permite que en unas pocas lineas se generen entornos gráficos de gran utilidad. Se implementan funciones, widgets, marcos... con un sólo comando. Easygui sólo ocupa un fichero con unos pocos kb's y da una serie de funciones de gran utilidad. El módulo se importa como cualquier otro módulo en Python, con la orden "import" o como sus creadores recomiendan "from easygui import *".

Se ha utilizado este módulo por su sencillez y por su compatibilidad con SPADE, ya que en su comienzo el proyecto iba a ser generado con Glade, este no es compatible con la forma de programación de los agentes de SPADE.

A continuación se enumeran algunas de las ventanas que se generan con este módulo:

- msgbox
- choicebox
- ccbox (Continue/Cancel)
- multenterbox (Multientrada de texto)
- choicebox
- multichoice
- passwordbox
- buttonbox with image

Glade Como anteriormente se ha descrito en el comienzo del proyecto se utilizó esta utilidad para la gestión del entorno gráfico del proyecto. Esta herramienta de desarrollo visual de interfaces nos permite crear las ventanas físicas que posteriormente utilizaremos en nuestro código fuente.

Este programa genera un fichero .xml con la información característica de la ventana y posteriormente deberemos hacer la llama al "gtkbuilder" para enlazar el fichero xml con la información al código fuente del programa.

Esto crea interfaces mediante GTK/GNOME, lo que significa que Glade es a GTK/GNOME como QtDesigner a QT ó similar a Easygui con TKinter.

El interfaz final es de mejor que la que se obtiene con Easygui pero la aplicación de este tipo de interfaces gráficas se abandonó cuando se comenzó a implementar el sistema SPADE al proyecto, pues no son compatibles, puesto que GTK necesita de una funcion de tiempo infinita (nunca acaba) para mostrar las ventanas de la aplicación y los agentes que componen nuestro software con SPADE no pueden continuar y el programa se detiene.

8.4.1.4. Parseadores

Los parseadores son herramientas para el análisis de ficheros. Estos nos permiten conocer si el fichero está bien formado o no. Los parseadores convierten un texto, en este caso un fichero XML, en una estructura de tipo árbol donde se ramifican los nodos y propiedades de estos.

En este proyecto se utilizan varios parseadores XML como son sax, dom y etree. Que nos permiten analizar la estructura del fichero STEP-XML para conocer si está bien formado y, si fuese el caso, buscar en su estructura la información necesaria para el desarrollo de la aplicación.

xml.sax SAX "Simple API for XML" es un analizador-parser para XML. Desarrollado originalmente para programación en Java, se ha convertido en la API estándar para XML tanto en Java como en otros lenguajes como Python. Es un parseador muy rápido porque lee secuencialmente de principio a fin, sin cargar todo el documento en memoria. SAX hace diversas tareas cómo detectar cuándo empieza y termina un elemento o un conjunto de caracteres, etc. En este proyecto ha sido utilizado para comprobar que el documento está bien formado. Por otra parte las desventajas que posee SAX es que no dispone de la estructura en árbol y realiza una lectura secuencial del documento por lo que una vez leído no se puede volver atrás, algo que DOM sí permite y por ello se ha utilizado conjuntamente en este proyecto.

xml.dom El "Document Object Model" también denominado DOM es una API multilenguaje que accede y modifica documentos XML. Muy útil por presentar una estructura de tipo árbol. Su punto más fuerte en comparación con SAX es, como anteriormente se ha descrito, que permite volver atrás tras la lectura del documento a analizar y que DOM permite estructurar el documento en un esquema de tipo árbol.

xml.etree Con su fácil forma de uso la API "ElementTree" (etree) es un parseador que permite buscar nodos e información dentro de estos. ElementTree trabaja de forma sencilla entre nodos padres e hijos, y muestra sus atributos y/o sus respectivos valores.

8.4.1.5. Hojas de cálculo

En la aplicación desarrollada se interviene con hojas de cálculo. Las hojas de cálculo son documentos digitales especialmente diseñados para la interactuación con un entorno de tablas. Uno de los pilares principales de la herramienta que se ha desarrollado es la sencillez y facilidad en el interfaz gráfico y la relación de este con el usuario, por ello se decidió que el uso de este tipo de documentos sería pieza fundamental del proyecto. En el caso en concreto que estamos estudiando han sido utilizadas para la exportación de los valores calculados y la enumeración de todos los parámetros necesarios.

La utilización de hojas de cálculo en lenguaje Python es sencilla, aunque han sido necesarios tres módulos distintos, cada uno se encarga de un propósito, tal y como se indican a continuación:

xlrd Este módulo nos permite importar una hoja de cálculo anteriormente creada (en el caso de estudio la plantilla) para su posterior modificación por orden de otros módulos.

xlwt El segundo módulo que se ha utilizado para trabajar hojas de cálculo es xlwt, este nos permite modificar el aspecto y los valores de las celdas que componen el documento.

xlutils Esta implementación permite guardar los datos modificados en documentos con formato de hoja de cálculo y así poder salvar nuestros datos.

8.4.1.6. Otros módulos

En este apartado se proceden a explicar el resto de los módulos utilizados de forma breve.

Sistema Funciones y módulos que interactúan con el sistema. Se encargan de funciones como cerrar la aplicación o de ejecutar un programa instalado en el sistema.

sys Este módulo nos da acceso a algunas de las variables que se usan en el intérprete (Consola) y a funciones que interactúan con este. Una de las formas de utilización mas comunes es la función:

```
sys.exit([arg])
```

está implementada para salir de la consola

time Gracias a este módulo, Python, a parte de devolver la hora, nos permite crear acciones y funciones relacionadas con el tiempo, como por ejemplo con el uso de:

```
time.sleep(secs)
```

que nos permite detener la aplicación un número de segundos concretos.

os El módulo provee distintas instrucciones para interactuar con el sistema operativo. Permite modificar ficheros y estructuras de carpetas y otras funciones como ejecutar una aplicación que abra un fichero:

```
os.open(file, flags[, mode])
```

Representación Uno de los agentes principales de la aplicación multiagente que se ha creado, es el agente de representación el cual nos permite visualizar en 3D el documento con el cual vamos a trabajar. Para este fin, al comienzo del desarrollo de la aplicación se probó a utilizar los módulos de pythonOCC, los cuales nos permiten representar ficheros STEP. Este propósito no fue conseguido por problemas de versiones en librerías, y por ello se procedió a utilizar la aplicación view3Dscene, que es una aplicación de representación para ficheros .wrl, que son un estándar de representación 3D, exportable desde cualquier software CAD.

pythonOCC Es un conjunto de módulos que nos permiten trabajar con documentos CAD, tanto su representación como actuar con ellos. La aplicación estaba preparada para actuar con estos módulos, pero por razones de librerías se omitió este procedimiento.

view3Dscene Aplicación de representación de ficheros 3D virtuales. Permite la navegación por el entorno tridimensional.

8.4.2. Software existente

Se han estudiado las posibles alternativas de software actuales. Y se han determinado los puntos fuertes y los contras que tiene cada uno de los programas. Aunque no existe gran variedad de software que analicen ficheros CAD, tras una profunda investigación se procede a analizar los siguientes:

8.4.2.1. Software DFMA

El siguiente listado de aplicaciones son herramientas para el análisis DFMA. Son útiles que nos facilitan la tarea de análisis DFMA. A continuación se procede a analizar los puntos fuertes y débiles de cada una.

DFMAPRO DFMAPRO es un software creado por SSA Technologies que nos provee un método de aplicación de los principios del diseño para la fabricación y el ensamblaje. El programa nos permite predecir el coste del proceso seleccionando métodos de ensamblado. También nos permite conocer cuales son las piezas innecesarias para realizar un diseño mas eficiente.

PROS: Combina DFA y DFM. Entorno sencillo y amigable

CONTRAS: No dispone de implementación CAD

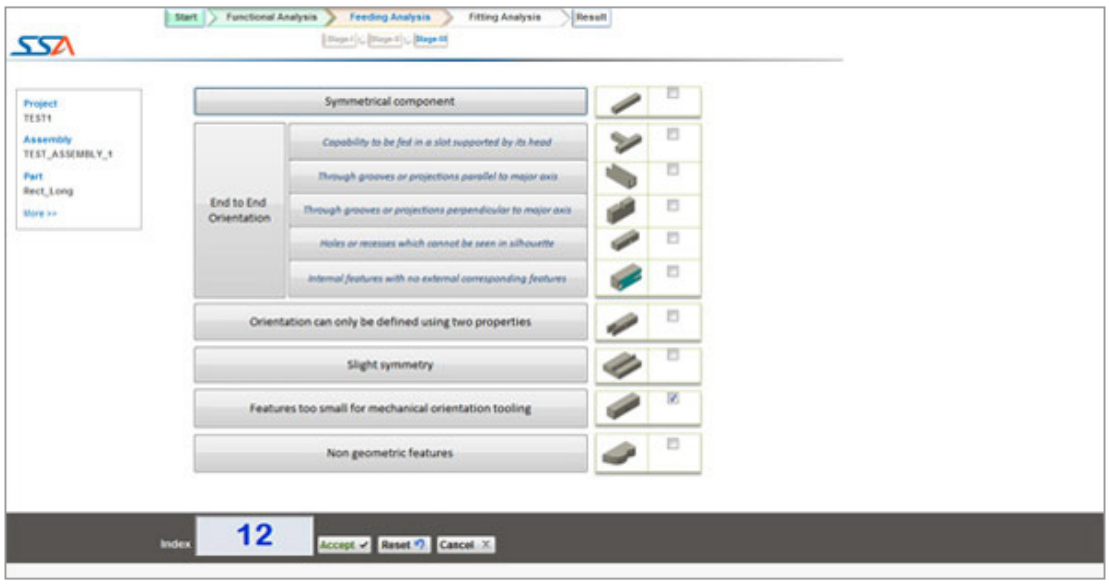


Figura 8.1: Entorno gráfico DFMAPRO

DFM - DFMA.com DFM es un software desarrollado por Boothroyd Dewhurst, Inc. con el cual podemos realizar tareas relacionadas con el diseño para la fabricación, permitiéndonos así optimizar el diseño y los costes del producto. Las tareas que nos permiten esta optimización son:

- Selección de materiales, procesos y operaciones.
- Estimación de costes de fabricación .
- Evaluación de costes unitarios.
- Comparativa entre materiales alternativos de cada pieza.

PROS: Puede importar alguna información de un fichero CAD tras una serie de pasos.

CONTRAS: No dispone de implementación CAD. No combina software DFM y DFA.

DFA - DFMA.com Es el programa de Bothroyd Dewhurst, Inc. que utiliza la metodología de análisis de ensamblabilidad. Simplifica la tarea de evaluar cada parte e identificar las piezas innecesarias para la simplificación del producto. También ayuda a la estimación de la dificultad de ensamblado del producto.

PROS: Entorno gráfico que ayuda a la rápida comprensión.

CONTRAS: No dispone de implementación CAD. No combina software DFM y DFA.

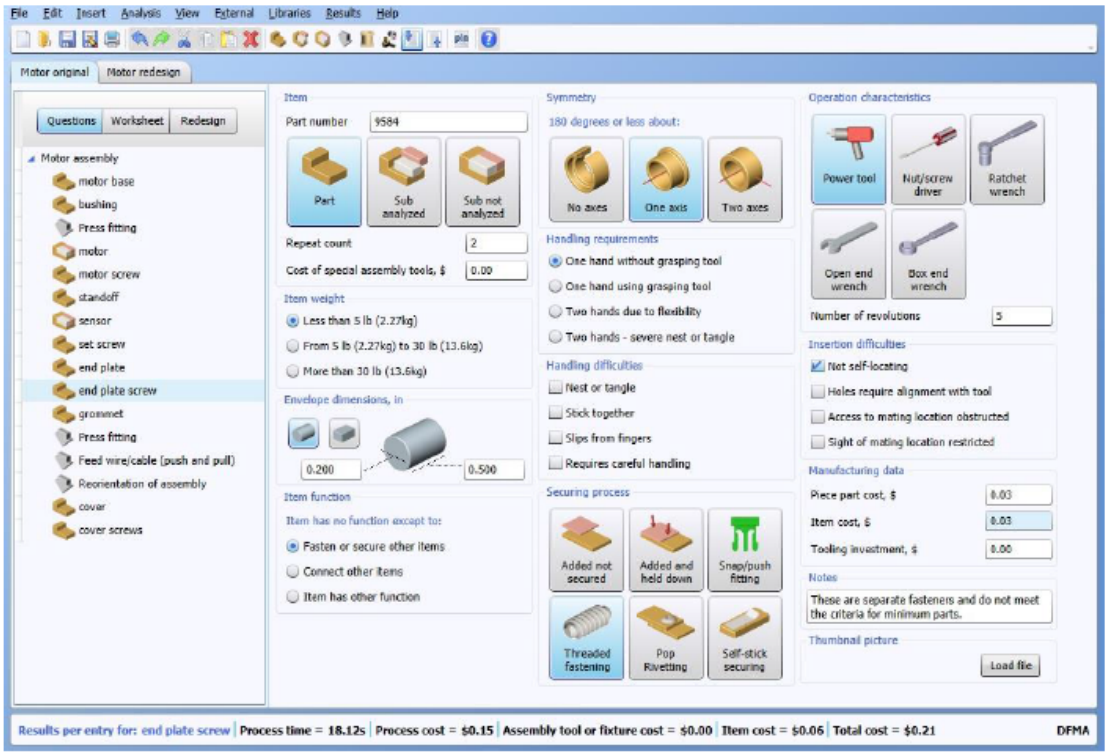


Figura 8.2: Software DFA

8.4.2.2. Otro software relacionado

En este apartado se procede a describir algunas aplicaciones que tienen relación con este proyecto. Se analizan dos aplicaciones de gran interés, un traductor de formato STEP a STEP-XML, el cual ha sido utilizado en los ejemplos evaluados por el proyecto; Un analizador de ficheros STEP, una aplicación que lee un fichero step y clasifica y denomina cada uno de los nodos que lo componen.

Traductor EXPRESS-XML Esta aplicación ha sido de gran utilidad para el desarrollo de este proyecto, pues sin esta no se podría haber desarrollado la aplicación para el análisis de ficheros STEP-XML.

En la actualidad ningún software CAD exporta a ficheros STEP-XML directamente, el motivo es porque los ficheros STEP-XML son de mayor peso que los STEP-EXPRESS. En pequeños ficheros no es una diferencia notable, pero si se trabajan con grandes proyectos como por ejemplo una aeronave el peso del fichero se multiplica. Esto provoca un problema de capacidad, por lo que con seguridad el fichero STEP-XML se implementará en un corto período de tiempo, por lo que se ha desarrollado este proyecto en base a este tipo de archivos.

La aplicación es muy sencilla, sólo hay que indicar el fichero STEP-EXPRESS (ISO 10303-21) a traducir a STEP-XML (ISO 10303-28) y en unos segundos nos genera un

fichero ".xml" que podemos utilizar en el software desarrollado para este proyecto. En el programa se puede elegir el tipo de AP, en el caso analizado se ha utilizado el AP209e2.

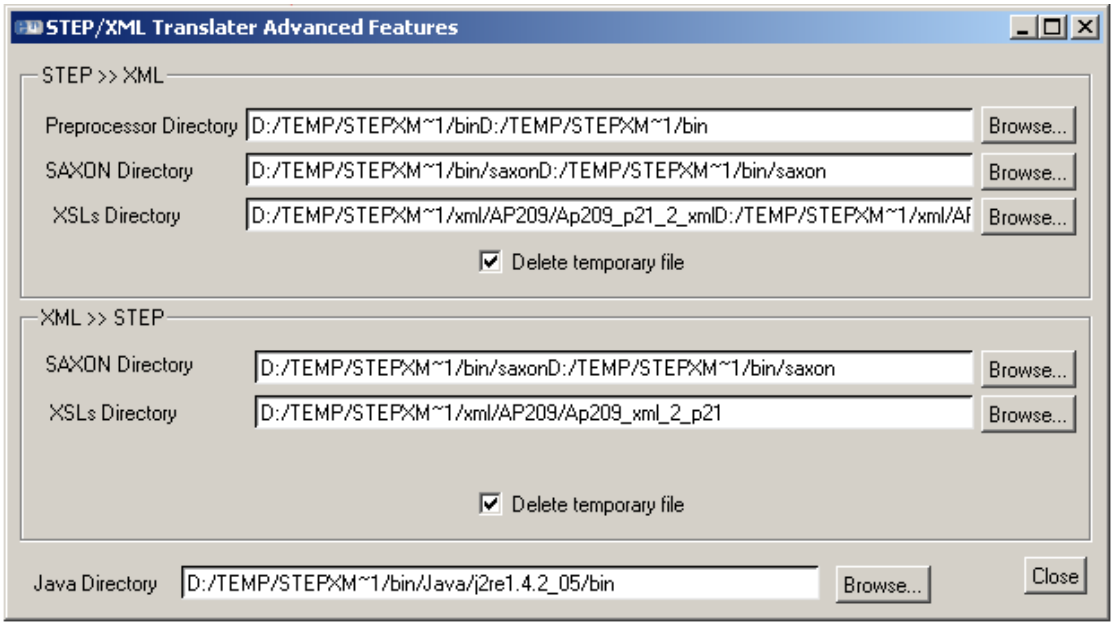
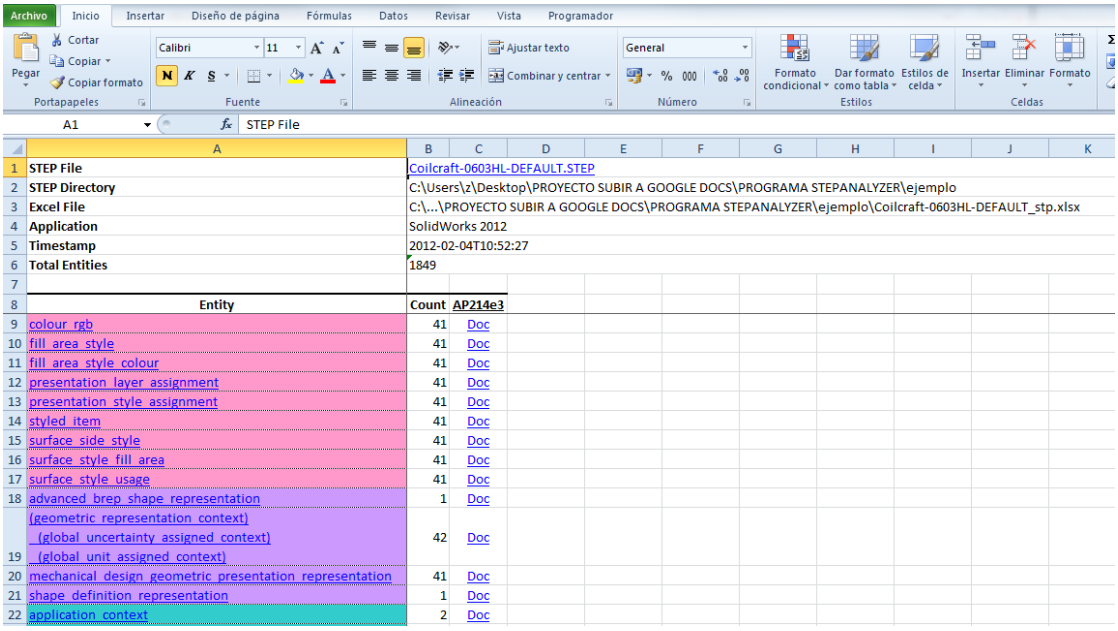


Figura 8.3: Screenshot del XML-Translator

Step Analyzer Esta herramienta desarrollada por el NIST (Instituto Nacional de Estandarización y Tecnología de Estados Unidos) genera una hoja de cálculo con información sobre un fichero STEP. También muestra las propiedades de validación del documento, incluyendo PMI (Información sobre la fabricación del Producto.

Tras ejecutar el programa y seleccionar el fichero a analizar, el software genera una serie de pestañas con información sobre cada uno de los campos. También permite entrar en cada campo con un enlace en la hoja principal, que actúa como índice.



	A	B	C
1	**STEP File**	Coilcraft-0603HL-DEFAULT.STEP	
2	**STEP Directory**	C:\Users\z\Desktop\PROYECTO SUBIR A GOOGLE DOCS\PROGRAMA STEPANALYZER\ejemplo	
3	**Excel File**	C:\...\PROYECTO SUBIR A GOOGLE DOCS\PROGRAMA STEPANALYZER\ejemplo\Coilcraft-0603HL-DEFAULT_stp.xlsx	
4	**Application**	SolidWorks 2012	
5	**Timestamp**	2012-02-04T10:52:27	
6	**Total Entities**	1849	
7			
8	**Entity**	**Count**	**AP214e3**
9	colour_rgb	41	Doc
10	fill_area_style	41	Doc
11	fill_area_style_colour	41	Doc
12	presentation_layer_assignment	41	Doc
13	presentation_style_assignment	41	Doc
14	styled_item	41	Doc
15	surface_side_style	41	Doc
16	surface_style_fill_area	41	Doc
17	surface_style_usage	41	Doc
18	advanced_brep_shape_representation	1	Doc
19	(geometric_representation_context) (global_uncertainty_assigned_context) (global_unit_assigned_context)	42	Doc
20	mechanical_design_geometric_presentation_representation	41	Doc
21	shape_definition_representation	1	Doc
22	application_context	2	Doc

Figura 8.4: Ejemplo de uso del Step Analyzer de NIST

9 Caso de aplicación

En este capitulo se desarrollará la forma en la que hemos aplicado todos los requisitos del proyecto, fundamentándose en el análisis de soluciones que se han aportado anteriores capítulos de este proyecto.

Se comenzará con una visión general de la aplicación para, posteriormente, indagar en aspectos más concretos de la herramienta. Por ello el primer subcapítulo describirá de forma general el sistema multiagente que engloba la aplicación y entre otras descripciones la estructura general de los agentes SPADE. Para, posteriormente, proceder a desarrollar cada uno de los agentes y sus respectivas funciones, las cuales generan los comportamientos del software que se ha desarrollado.

Se intentará describir en forma de diagramas cada una de las partes que conforman este capítulo, para conseguir una mejor comprensión. Esta serie de diagramas irán acompañado de una explicación desarrollada y si fuese necesario del código fuente.

9.1. Estructura general de la aplicación

Introduciendo este apartado se describirán las estructuras principales utilizadas en el desarrollo de la aplicación. Se comenzará por dar una visión general de cómo es una aplicación en lenguaje Python y posteriormente la estructura de agentes de la plataforma multiagente SPADE.

9.1.1. Estructura de una aplicación Python

La estructura del código fuente varía según el lenguaje utilizado, en este caso en concreto, y por motivos descritos con anterioridad, se ha utilizado lenguaje Python. Python es un lenguaje interpretado lo que se distingue de otros lenguajes como C/C++ esto aunque hace que el programa sea mas lento al tener que interpretarse nos permite que no sea necesario una estructura demasiado estricta.

Aún así, este lenguaje se caracteriza por presentar una estructura determinada, de mayor o menor complejidad, de una forma similar a la que se muestra a continuación:

```
1 #!/usr/bin/python
2 # -*- coding: utf-8 -*-
3 #
4 #  Comentarios
5 #
6
```

```
7  import module
8  from module import *
9
10 class nombre_clase(parametros):
11         """
12         información pydoc
13         """
14         def funcion(self, parametros):
15                 valor = 1  #definicion de valores de variables
16                 if valor == 1:
17                         print ("funcion para mostrar por pantalla
                           ")
18                 else:
19                         print ("ejemplo de condicional")
20
21 if __name__ == "__main__":
22
23     a = nombre_clase()
```

Se comienza determinando el lenguaje de programación con la instancia de la linea 1. En la linea 2 se identifica el tipo de coding que se va a utilizar, en nuestro caso por utilizar tildes se determina UTF-8.

Los comentarios son agregados comenzando la linea con #. También pueden estar al final de la linea como se muestra en la linea 15.

Una de las características fundamentales de Python es que se pueden agregar módulos, que nos permite ahorrar mucho tiempo de programación. Para adquirir estos módulos, además de tenerlos preinstalados en el ordenador se pueden agregar con la orden "import" de la manera que se muestra en las lineas 7 y 8.

En la linea 10 se define una clase. Una clase es un molde del que luego se pueden crear múltiples objetos, con similares características. Es pieza fundamental en la programación orientada a objetos. Una clase define los atributos y métodos comunes a los objetos de ese tipo, pero luego, cada objeto tendrá sus propios valores y compartirán las mismas funciones.

De la linea 11 a la 13 existe un tipo de "comentario" especial que permite describir las clases y las funciones que se crean, gracias a la ayuda de pydoc. Este crea una estructura de clases y funciones con sus respectivos comentarios aclaratorios, muy útiles si se ha de continuar programando un código que comenzó otro programador.

Como se acaba de explicar dentro de las clases hay funciones, las cuales son definidas como se muestra en la linea 14. Dentro de una función se definen valores para las variables (linea 15) o se crean estructuras condicionales o reiterativas... (lineas 16 a 19).

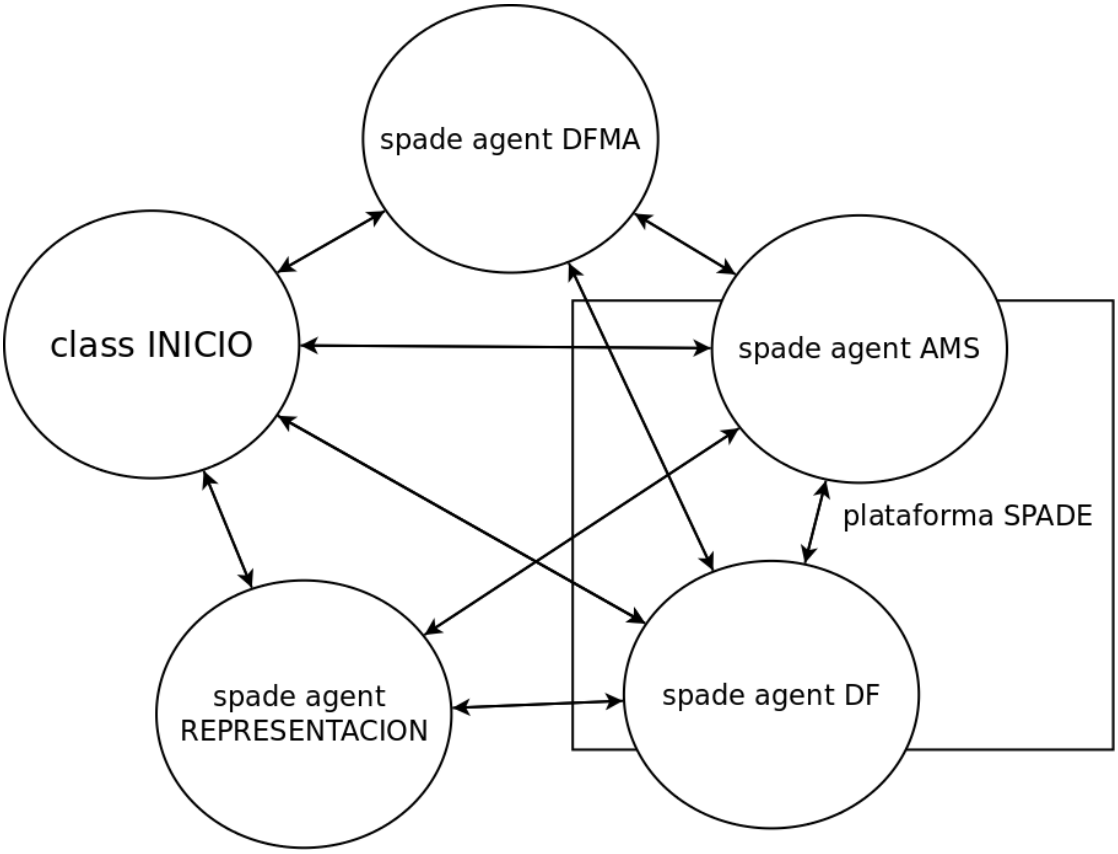


Figura 9.1: Relación entre agentes de la aplicación

Por último destacar la forma de llamar a la función o clase principal para comenzar el programa, como muestra en la linea 21, siendo __main__ el "módulo" principal del programa y por ello la forma de ejecutarse.

9.1.2. Estructura de agentes SPADE

Como con anterioridad se ha descrito, el software desarrollado se ha programado siguiendo un paradigma multiagente. Para ello se ha utilizado una plataforma multiagente denominada SPADE, la cual también se ha descrito con anterioridad. Esta plataforma está formada por agentes y medios para la comunicación entre agentes.

Tal y como se muestra en la figura 9.1 hay cinco agentes, cuatro de ellos son agentes con estructura SPADE, el otro es un módulo que inicia los agentes spade.

El módulo Inicio, aunque no tiene la estructura de un agente spade, es parte fundamental del software pues arranca el resto de los módulos.

Inicio El agente inicio, es el encargado de iniciar el resto de los agentes en el caso de que se den las condiciones definidas en este. Estas condiciones se explicarán más adelante, bajo el subapartado "Estructura Detallada".

Tan sólo destacar que este módulo tiene una estructura general, cómo si fuera programación orientada a objetos, pero puede llegar a considerarse un agente por interactuar con estos.

Los agentes que conforman la base de la plataforma multiagente SPADE tienen una particularidad, hay que crear la plataforma con la orden:

```
runspade.py
```

la cual creará el sistema de plataforma SPADE, incluyendo la interfaz gráfica web, en el host que hayamos configurado previamente. En el caso de estudio se ha utilizado *localhost* por lo tanto si se abre el navegador y se escribe

```
http://127.0.0.1:8008
```

podremos entrar al servicio web que nos permite administrar la plataforma SPADE.

Los agentes propios de dicha plataforma son:

DF El "Directory Facilitator" de SPADE funciona como unas páginas amarillas para los agentes. Es donde están publicados los agentes. Para publicar los agentes hay que definir su "Service Description" que es la información que describe a cada agente. Toda esta información es compartida registrado el servicio como muestra el siguiente código:

```
sd = spade.DF.ServiceDescription()
sd.setName("test")
sd.setType("testservice")
dad = spade.DF.DfAgentDescription()
dad.addService(sd)
sd = spade.DF.ServiceDescription()
sd.setName("MYSERVICE")
sd.setType("MYTYPE")
dad.addService(sd)
dad.setAID(self.myAgent.getAID())
res = self.myAgent.registerService(dad)
print "Service Registered:",str(res)
```

Aunque en el código fuente de nuestra aplicación no se ha registrado finalmente el código, uno de los posibles TODO podría ser este.

AMS Agent Management System - El Sistema de Gestión de Agentes, es el encargado de organizar, identificar y buscar agentes en la plataforma SPADE. Siempre está presente en todas las plataformas SPADE y se inicializa de forma automática al iniciar la plataforma. El AMS permite buscar agentes registrados en la plataforma y modificar estos agentes.

Agente.SPADE El quedan por describir dos agentes, programados con una estructura de agentes SPADE, estos agentes se han nombrado como "representacion" y "dfma". En este apartado sólo procederemos a ver una estructura general de este tipo de agentes SPADE y posteriormente, en su respectivo apartado, se detallarán las características y funciones de cada agente.

Los agentes SPADE tienen una estructura tal y como se muestra a continuación:

```
1  import spade
2  import time
3
4  class MyAgent(spade.Agent.Agent):
5          class MyBehav(spade.Behaviour.Behaviour):
6                  def onStart(self):
7                          print "Starting behaviour . . ."
8                          self.counter = 0
9
10                 def _process(self):
11                         print "Counter:", self.counter
12                         self.counter = self.counter + 1
13                         time.sleep(1)
14
15         def _setup(self):
16                 print "MyAgent starting . . ."
17                 b = self.MyBehav()
18                 self.addBehaviour(b, None)
19
20 if __name__ == "__main__":
21         a = MyAgent("agent@myhost.myprovider.com", "secret")
22         a.start()
```

Lo primero que se debe hacer para trabajar con SPADE es importar el módulo como se muestra en la línea 1 del código fuente anterior. En este caso también se ha importado el módulo tiempo, es utilizado para detener la aplicación durante un segundo, utilizando la función tal y como se muestra en la línea 13.

Es destacable la llamada que se hace para iniciar el agente (lineas 21 y 22) donde hay que introducir como parámetros la configuración utilizada para arrancar *runspade.py*. Tanto el nombre del agente seguido de @ como el host configurado y posteriormente una coma y la contraseña configurada (por defecto *"secret"*). En la línea 22 simplemente se inicia el agente con la variable a la que hemos atribuido todos los atributos de la linea 21 seguido de *.start()*.

La class MyAgent debe ser seguido del parámetro *(spade.Agent.Agent)* así SPADE reconoce esta clase como propia e interactúa con ella. Dentro de la clase que denomina al agente, se muestran clases con *MyBehav* que son las tareas dentro de los agentes. La

clase spade.Behaviour.Behaviour es la predeterminada para todos los agentes SPADE, pero existen otras como:

- spade.Behaviour.PeriodicBehaviour
- spade.Behaviour.OneShotBehaviour
- spade.Behaviour.TimeOutBehaviour
- spade.Behaviour.EventBehaviour
- ...

Para conocer cuándo utilizar el resto de las clases Behaviour por favor leer la documentación referente descrita en la bibliografía.

En la línea 6 nos encontramos con el método *onStart()* el cual es similar al método *_setup()* de la clase principal agente. Estos métodos son, de sus respectivas clases, lo que primero arranca de cada clase, digamos que es la función de inicialización. Dentro del método *_setup(),* en nuestro ejemplo, se define el nombre de la clase Behaviour, en el caso explicado *MyBehav* y en la línea 18 se hace la llamada a la tarea con la expresión:

```
self.addBehaviour(b, None)
```

En la línea 10 nos encontramos con el método *_process()* el cual es el núcleo de la clase programada, es el cuerpo de la clase Behaviour. Existen otros métodos como:

- _onTick()
- onEnd()
- onStart()
- timeOut()
- ...

Para conocer los comportamientos de estos métodos, por favor leer documentación referente indicada en la bibliografía.

9.2. Estructura detallada

A continuación se describe con detalle cada uno de los agentes que forman la aplicación programada. Se procederá a describir en cada apartado las funciones y valores de las variables y comportamientos de estos cuando se ejecuta el programa.

9.2.1. Módulos importados

Como ya se ha explicado, una de las características principales del lenguaje de programación Python es la posibilidad de importar módulos y ahorrar tiempo en escribir lineas de programación.

```
1  import spade
2  from easygui import *
3  import sys
4  import os
5  import time
6  from xml.sax.handler import ContentHandler
7  import xml.sax
8  from xml.dom import minidom
9  from xml.etree import ElementTree as ET
10 from xlrd import open_workbook
11 import xlwt
12 from xlutils.copy import copy
13 try:
14         from occmodel import Tools
15         from occmodelviewer import viewer
16 except:
17         print "No se pueden cargar módulos occmodel"
```

los módulos que se han importado para el desarrollo del código son los mostrado en el listado del programa. Todos los módulos utilizados han sido explicados en capítulos anteriores, por lo que no se desarrollarán en este subapartado, tan sólo se enumerarán y se describirá su uso en cada de uno de los agentes si es necesario.

Por último, reseñar la lineas 13 a la 17, las cuales intentan cargar el módulo occmodel de pythonocc, pero al tener problemas con las librerías se añade el mensaje de excepción escrito por consola “No se pueden cargar módulos occmodel”.

9.2.2. Agente inicio

A continuación se procede a describir con detalle la estructura de la clase inicio. Se comenzará por presentar un diagrama con la estructura del agente, tal y como se muestra en la figura 9.2.

Este módulo lo primero que hace es inicializar una ventana de inicio con información sobre el título del proyecto y el programador. Al pulsar Aceptar en la primera ventana, se procede a seleccionar el fichero .xml que vamos a analizar, esto se hace con una ventana de selección de archivos a la que previamente se le ha aplicado un filtro para ficheros .stp, .step y .xml. Inmediatamente después de seleccionar el fichero, se parsea por medio de xml.sax

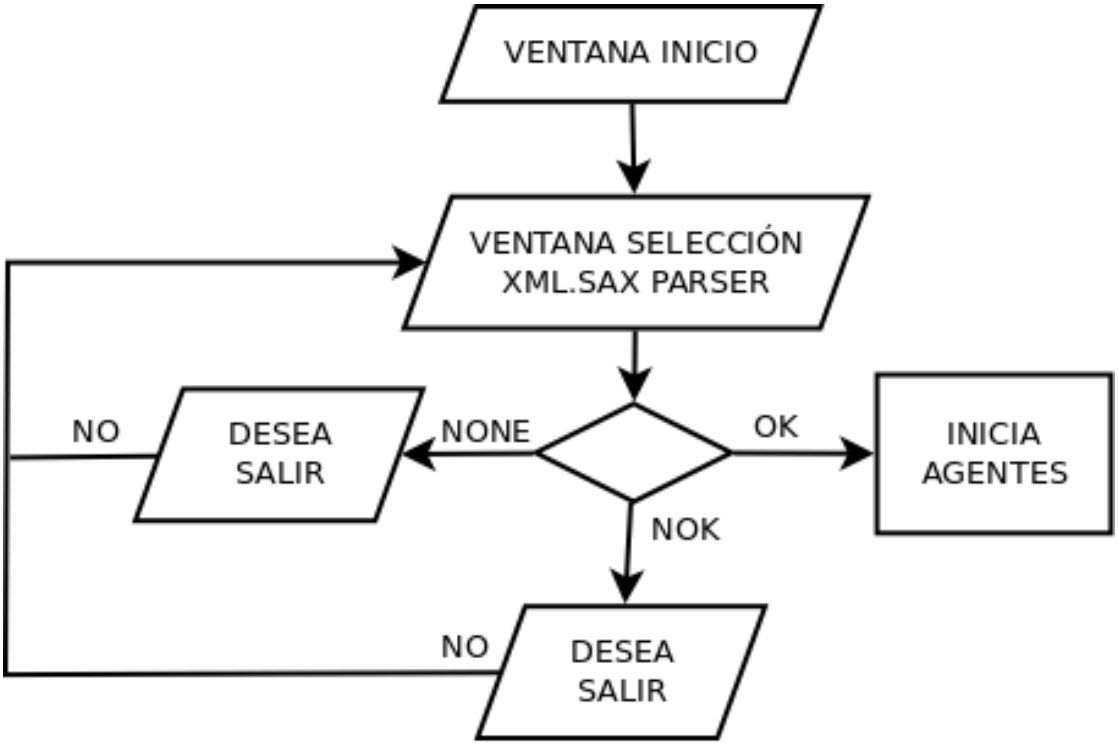


Figura 9.2: Diagrama de flujo del agente inicio

para comprobar que es un fichero compatible y está bien formado, lo que nos brinda tres posibilidades:

OK El documento seleccionado está bien estructurado como fichero STEP-XML y por consiguiente inicia el resto de los agentes.

NOK El documento no está formado con una buena estructura y por lo tanto el programa emite un mensaje sobre si se desea salir. Si se pulsa NO vuelve a la ventana de selección de fichero y reitera el proceso. Si se elige SI el programa se cierra.

NONE No se ha seleccionado ningún documento y se ha pulsado ACEPTAR. La aplicación muestra el mensaje sobre si se desea salir y las opciones son las mismas que si el fichero fuese NOK.

A continuación se muestra el código y se detallan elementos característicos del código fuente:

```
class inicio:
    """
    Este class inicia el programa, muestra la ventana de inicio,
    muestra la ventana de selección de archivo, comprueba el
        documento es
```

```
5       compatible con el formato XML y si es así inicia los agentes
            dfma y
6       representación.
7       """
8       def __init__(self):
9           self.ventana_inicio()
10
11      class ventana_inicio:
12          def __init__(self):
13              print "Comienza ventana inicio . . ."
14              self.ventana_show()
15
16          def ventana_show(self):
17              print "Se va a cargar la ventana inicio"
18              msg = """
19  Software para la interpretación de ficheros XML-STEP
20
21    Bajo paradigma multiagente para el análisis del
22        Diseño para la Fabricación y Ensamblaje.
23            Desarrollado por Eduardo Barea
24        Escuela Politécnica Superior de Sevilla."""
25              image = "logo.gif"
26              choices = ["Aceptar"]
27              title = "DFMAsoft para XML-STEP"
28              reply = buttonbox( msg, image = image, choices =
                    choices, title=title)
29              self.ventana_end()
30
31          def ventana_end(self):
32              print "fin ventana inicio."
33              inicio.ventana_seleccion()
34
35      class ventana_seleccion:
36          def __init__(self):
37              print "Comienza ventana selección . . ."
38              self.ventana_show()
39
40          def ventana_show(self):
41              print "Se va a cargar la ventana de selección"
42              msg = "Por favor seleccione el fichero CAD en formato
                    XML-STEP"
43              title = "DFMAsoft - Seleccion de fichero XML-STEP"
```

```
44              filetypes = [["*.step", "*.stp", "*.xml", "XML-STEP"]
                    ]
45              global fichero
46              fichero = fileopenbox(msg=msg, title=title, default='
                    *', filetypes=filetypes)
47              print fichero
48              self.ventana_end()
49
50          def ventana_end(self):
51              """
52              aquí se parsea el documento para comprobar si es un
                    fichero xml
53              compatible
54              """
55              print "fin de la ventana de selección"
56              print "Parseando documento . . ."
57              xml_analizador = xml.sax.make_parser()
58              xml_analizador.setContentHandler(ContentHandler())
59              try:
60                  xml_analizador.parse(fichero)
61                  print "El fichero XML %s está bien formado." %
                        fichero
62              except Exception, err:
63                  if fichero == None:
64                      msg = "No se ha seleccionado ningún fichero.
                            ¿Desea salir de la aplicación?"
65                      title = "¿Desea salir?"
66                      error = ynbox(msg=msg, title=title)
67                      if error == 1:
68                          #os.system('clear')
69                          print "deteniendo agentes"
70                          a = dfma("dfma@127.0.0.1", "secret")
71                          b = representacion("representacion@127
                                .0.0.1", "secret")
72                          a.stop()
73                          b.stop()
74                          sys.exit(0)
75
76                      else:
77                          self.ventana_show()
78                  else:
```

```
79                      msg = "Error El fichero " + str(fichero) + "
                            no tiene"\
80                      " formato XML-STEP. Por favor elija otro
                            fichero"
81                      title = "Error de selección"
82                      error = msgbox(msg=msg, title=title,
                            ok_button="Aceptar", image=None, root=None
                            )
83                      if error == "Aceptar":
84                          self.ventana_show()
85
86              a = dfma("dfma@127.0.0.1", "secret")
87              #a.setDebugToScreen()
88              a.start()
89
90              b = representacion("representacion@127.0.0.1", "
                    secret")
91              print "inicia representación"
92              #b.setDebugToScreen()
93              b.start()
94
95              alive = True
96              while alive:
97                  try:
98                      time.sleep(1)
99                      # Si detecto que el agente se ha parado,
                            salimos del bucle infinito
100                     if not a.isAlive():
101                         alive = False
102                 except KeyboardInterrupt:
103                             alive=False
104                             a.stop()
105                             b.stop()
106             sys.exit(0)
```

Como se observa en el diagrama, el la clase inicio comienza mostrando una ventana inicial de presentación, lanzada con un buttonbox. Este buttonbox es una función del módulo *easygui*. Para lanzar esta ventana, tal y como se muestra en el código adjunto hay que llamar a la función:

```
buttonbox(parametros)
```

los parámetros a los que se refiere la función son:

- Title: Es el texto que se muestra en el título de la ventana.
- Msg: Mensaje de texto que muestra el cuerpo principal del entorno gráfico.
- Image: Imagen que se muestra bajo el texto principal.
- Choices: Botones que se muestran en el diálogo (En este caso sólo el botón de Aceptar)

Todos estos parámetros han sido definidos con las lineas anteriores a la orden.

Esta plataforma de entorno gráfica muestra una ventana con información inicial sobre el proyecto y datos del programador también muestra una imagen en forma de logotipo del programa y por último un botón de Aceptar.

Al pulsar el botón Aceptar se cierra la ventana inicial y se inicia la clase ventana_selección() como se muestra en la linea 33 del código adjunto.

Posteriormente se llama a la función que muestra la ventana de selección, esta función se llama ventana_show(), dentro de la clase ventana_seleccion(). Aquí se utiliza otra función del módulo *easygui* que nos permite crear una ventana de selección de ficheros:

```
fileopenbox(parametros)
```

Esta función devuelve la ruta con un fichero y su extensión correspondiente del fichero seleccionado.

Los parámetros de esta función son prácticamente los mismos que para la función buttonbox (msg, title) pero se añaden dos parámetros característicos de esta función:

- filetypes: Aquí se debe introducir en forma de lista los tipo de ficheros que el programa debe filtrar para su selección.
- default: Aquí se muestra la ruta del fichero predeterminado. Si no se desea marcar ninguna en concreto se pone "*".

Cabe destacar la declaración de la variable fichero como *global* esto significa que es una variable pública para todo el código y no sólo de la clase en la que se define. Esto se creo para poder interactuar con el nombre del fichero y su ruta en otros métodos del código.

Después de esto, se cierra la ventana lo que hace que se llame a la función ventana_end(), la que se encarga primero de preparar el parseador xml, como se muestra en la linea 57 y 58, con *xml.sax*. Para parsear intentar parsear el documento (lineas 59 y 60) y si da algún error realizar lo que se encuentra entre la linea 63 y 84. Esto se realiza con la orden *try: except:* la cual intenta hacer algo y si no puede, en vez de mostrar un mensaje de error realiza lo que se defina dentro de *except*.

En concreto dentro del except se ha programado un condicional que determina lo que se hace si no se ha elegido ningún fichero (*if fichero == None:)* mostrando una ventana de salida y, como en casos anteriores, permitir salir de la aplicación o volver al paso anterior. Para salir de la aplicación se crea utiliza otra función de easygui:

```
ynbox(msg, title)
```

La cual devuelve 1 o 0 si se ha elegido Sí o No, respectivamente. Si se elige que desea salir, se detienen los agentes por si se han activado con anterioridad y se procede a la salida del sistema con la orden del módulo sys:

```
sys.exit(0)
```

La cual cierra la consola con un mensaje predeterminado.

Si el fichero ha sido seleccionado en vez de mostrar una excepción de error, se muestra una ventana con el mensaje "Error el fichero (*fichero seleccionado*) no tiene formato XML-STEP. Por favor elija otro" y vuelve a permitir elegir otro fichero desde la ventana de selección.

Desde la linea 86 hasta la 106 se muestra el código de la actuación de la aplicación en caso de que no hayamos elegido en ningún momento la salida del programa. Entre estas lineas de código, se muestran las llamadas a los agentes SPADE dfma y representación, tal y como se describió con anterioridad que había que realizarlas.

La el atributo *.setDebugToScreen()* permite mostrar en pantalla el debug de los agentes SPADE, útil en el momento de desarrollar dichos agentes para tener conocimientos de sus actuaciones (líneas 87 y 92). Posteriormente entre la línea 95 y 101 se determina que el agente se mantenga "vivo". Y el resto de código es la excepción que permite cerrar los agentes cuando se ordena por teclado tal fin.

9.2.3. Agente dfma

Dfma es el agente mas largo del programa, en total aproximadamente unas mil cien lineas de código. En este agente se pone en práctica *Sistemas Expertos,* donde, fundamentándose en el método de Boothroyd y Dewhurst se consigue desarrollar un agente inteligente que permite el cálculo del índice de diseño para la fabricación y ensamblaje.

El agente de forma general puede describirse como se muestra en la figura 9.3, aunque posteriormente se analizarán algunas de sus partes de forma mas detallada. Tal y como se muestra en la figura, el agente tiene un formato secuencial, con reiteraciones en varios de sus métodos.

En este subapartado no se transcribirá el código fuente por su excesiva longitud pero se describirán los puntos más importantes de dicho código.

El código mantiene la estructura de los agentes SPADE con sus respectivas clases y llamadas.

Comenzamos mencionando que en el método *_setup(self):* de la clase índice_dfma se ponen a cero la variable_atributo y se le da el valor del número de columnas a la matriz de datos donde se irán guardando los valores para cada pieza.

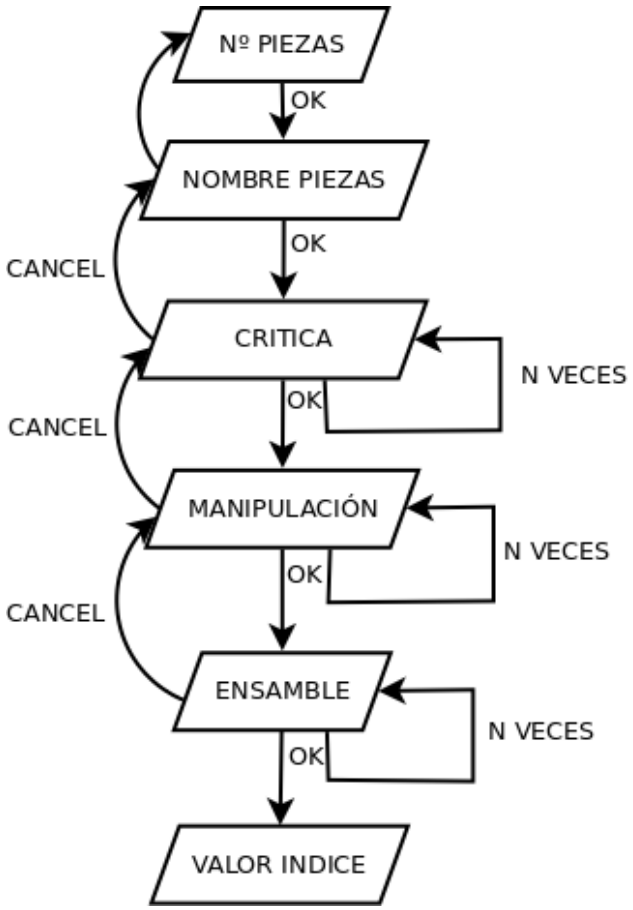


Figura 9.3: Agente SPADE dfma

9.2.3.1. Cálculo de numero de piezas

Lo primero que hace el agente es llamar al método *ventana_numero(self)* el cual utiliza la función *partes(self)* para calcular el número de partes que consta nuestro producto. Todo esto se hace de forma automática analizando el fichero STEP-XML.

```
1   #———FUNCIONES UTILIZADAS———#
2   def partes(self):
3       """
4       esta funcion es la encargada de buscar cada parte que compone el
5       producto y enumerarla
6       """
7       print ("listado de cada parte 'pieza' del producto")
8       global vari_nodo
9       vari_nodo = "Config_control_design:Next_assembly_usage_occurrence"  #LAST MODIFICATION
10      #vari_nodo = "Config_control_design:Product_category"  #buscar partes
11      #vari_nodo = "Config_control_design:Product"  #buscar productos
12      self.buscar()
13      global lista
14      lista = range(len(valor_id))
15      global numero_filas
16      numero_filas = len(lista)
17      global matriz
18      """
19      ahora se crea una matriz con N lineas según el numero de piezas
20      y a cada pieza se le da una id igual al numero de linea (empieza
21      por 0)
22      """
23      matriz = [None] * numero_filas
24      for i in range(numero_filas):
25          matriz[i] = [None] * numero_columnas
26      for i in range(len(lista)):
27          matriz[i][0] = i
```

Lo más destacable de este método es la forma de búsqueda, en la linea 9 se le da un valor a la variable vari_nodo = *"Config_control_design:Product_category"* este texto es

el que dentro del fichero STEP-XML muestra el nodo con el nombre de los componentes que forman el producto.

El método *partes*, utiliza la función *buscar* para parsear el documento en busca del nodo con ese nombre.

```
1  def buscar(self):
2              """
3              esta función es la que busca en el documento los
                  valores
4              de vari_nodo. Se utilizan dos parseadores por la
                  facilidad
5              de busqueda de uno entre lineas y otro entre nodos
6              """
7              global archivo_leer
8              archivo_leer = minidom.parse(fichero)
9              global etree
10             etree = ET.parse(fichero)
11             global root
12             root = etree.getroot()
13             print ("Busca los nodos y atributos previamente
                  definidos")
14             print ("Búsqueda de etiqueta: " + vari_nodo)
15             print "....................."
16             global valor_id
17             valor_id = []
18             for x in range(len(archivo_leer.documentElement.
                  childNodes)):
19                 try:
20                     if "id" in archivo_leer.getElementsByTagName(
                          vari_nodo)[x].attributes.keys():
21                         valor_id.append(archivo_leer.
                              getElementsByTagName(vari_nodo)[x].
                              attributes.get("id").value)
22                 except:
23                     break
24             print "lista de valores encontrados: "
25             print(valor_id)
26             print "__________"
```

De vuelta a la función *partes(),* adquiere el valor del número de filas (que es el mismo valor que el número de piezas) contando el numero de valores de *valor_id* se han encontrado.

La función *buscar* utiliza dos parseadores distintos *etree* y *xml.dom* porque uno es más útil para la búsqueda de valores entre los nodos y otro es más sencillo para la búsqueda

de valores entre los atributos de dichos nodos. Para la búsqueda se crea un bucle con *for* y se busca entre los nodos del árbol XML, dados por el atributo *.getElementsByTagName* del módulo *xml.dom,* el texto "id" del nodo con el texto de la variable *vari_nodo ("Config_control_design:Product_category"),* puesto que estos nodos tienen una id asignada con un número. Esta id se añade a una lista llamada *valor_id,* la cual crece en un valor cada vez que encuentra el valor buscado.

Lo que se hace a partir de la línea 23 es crear la matriz. Para ello primero se crea una linea de *None's* por el número de columnas que definimos en *_setup()* (en el caso de esta aplicación son 14 columnas) y posteriormente multiplicamos esa lista de 14 *None's* por el número de filas o lo que es lo mismo, el número de piezas encontradas.

Posteriormente se añade el valor de la *matriz [i][0]* que es el primer valor, un valor que empieza desde 0 y va hasta i-1.

9.2.3.2. Obtención de los nombres de cada pieza

Para obtener los nombres de cada pieza, la aplicación utiliza la función *ventana_nombre()* la cual tiene un comportamiento prácticamente idéntico a la de búsqueda del número de piezas, pero interviene una nueva función llamada *obtener()* que tiene la siguiente estructura:

```
1  def obtener(self):
2              """
3              realiza iteraciones por cada numero de pieza  para
                  conseguir
4              N numero de valores en la busqueda
5              """
6              for y in range(len(valor_id)):
7                  print "XXXXXXXX ITERACION  XXXXXXXXXXX"
8                  for child in root.findall(".//*[@id='" + valor_id
                      [y] + "']"):
9                      print "BUSQUEDA de ID=" + valor_id[y]
10                     print "BUSQUEDA DE ATRIBUTO/NODO=" +
                          vari_atributo
11                     #print "PADRE: " + str(root.findall(".*/"))
12                     print "NODO: " + str(child)
13                     #print "HIJO: " + str(root.findall(".*/"))
14                     #print "TAG: " + str(child.tag)
15                     #print "ATRIBUTOS: " + str(child.attrib)
16                     try:
17                         valor_atributo.append(str(child.find(
                              vari_atributo).text))
```

```
18                          print "VALOR ATRIBUTO: " + valor_atributo
                                [y]
19                      except:
20                          print "No tiene característica " +
                                vari_atributo
```

Lo que hace este código es iterar el mismo procedimiento el número de veces igual al que piezas hay. El procedimiento es el uso de *etree* para buscar entre todas las ramas de la estructura XML el valor del TAG, que es el "atributo" del nodo que buscamos con la id que obtuvimos en la función buscar.

Posteriormente lo añade al final de la lista *valor_ atributo* para cuando vuelva a la función ventana_nombre() aplicar el valor correspondiente a la matriz utilizando un bucle for al igual que se utilizó para el valor matriz[i][0], siento esta vez matriz[i][1] donde se coloca el valor del nombre.

9.2.3.3. Selección de piezas críticas

La función para asignar el valor de pieza crítica es de gran sencillez. Se utiliza una función denominada *choicebox* del módulo *easygui* la cual nos genera una ventana con varios valores seleccionables, como se muestra en la figura 9.4. Los valores a elegir han sido determinados por el método DFMA y determina si la pieza analizada es una pieza crítica o no, o lo que es lo mismo si podría ser eliminada o simplificada. Los valores referidos son:

- La pieza tiene movimiento respecto a otras piezas.
- La pieza tiene que ser de un material distinto al resto de las piezas.
- La pieza debe ser separada del resto para ensamblar otras piezas.
- Ninguna de las anteriores.

En la función se le asigna un valor a la matriz[i][2], este valor es 0 si se elige la cuarta opción ("Ninguna de las anteriores") lo que significa que esa pieza no es una pieza crítica. Para el resto de valores, la matriz[i][2] es igual a 1, o lo que es lo mismo es una pieza crítica y por lo tanto no podemos deshacernos de ella.

9.2.3.4. Índice de Manipulación

Bajo estas lineas se explicará cómo el método ventana_manipulación() es capaz de calcular el código de manipulación y su correspondiente valor de tiempo en segundos. Este apartado es donde se muestra de forma más desarrollada el Sistema Experto del agente dfma, es el entramado que convierte al agente en un agente inteligente.

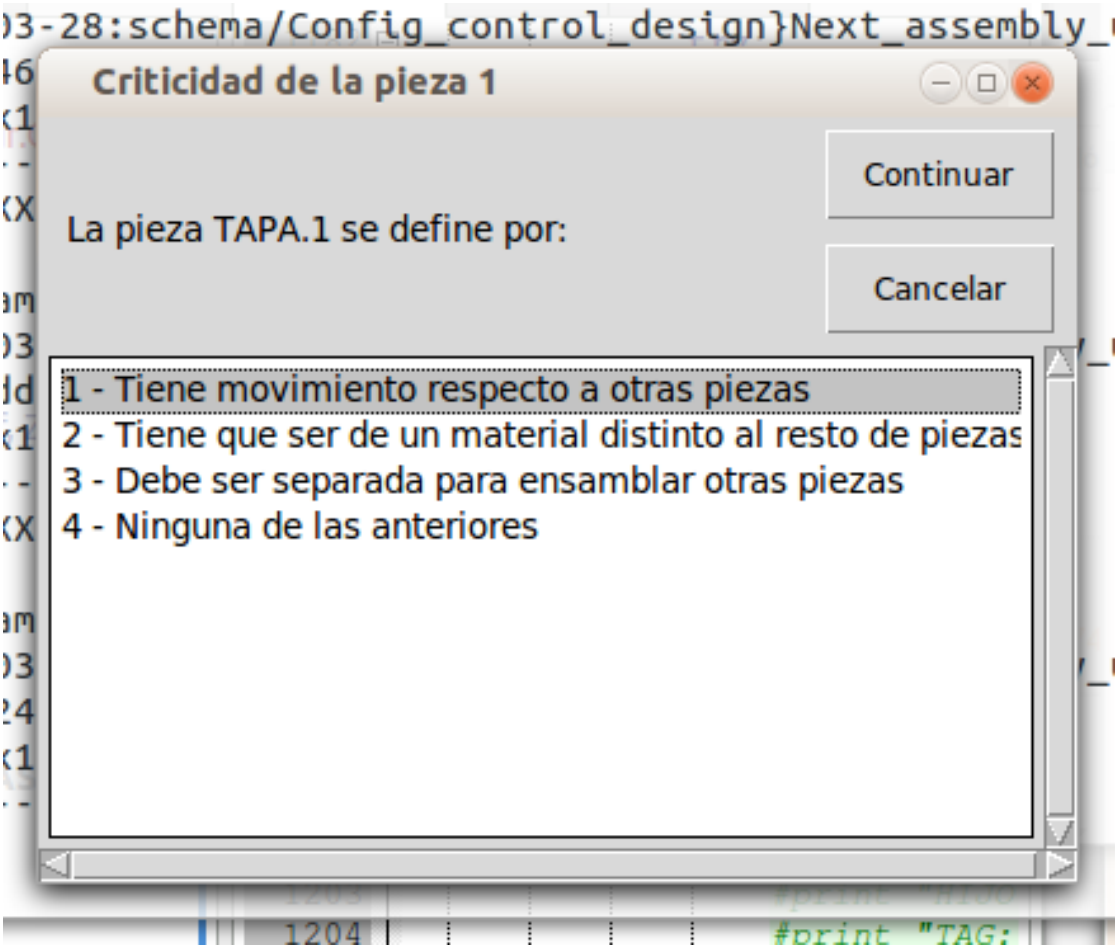


Figura 9.4: Interfaz gráfica ventana_critica

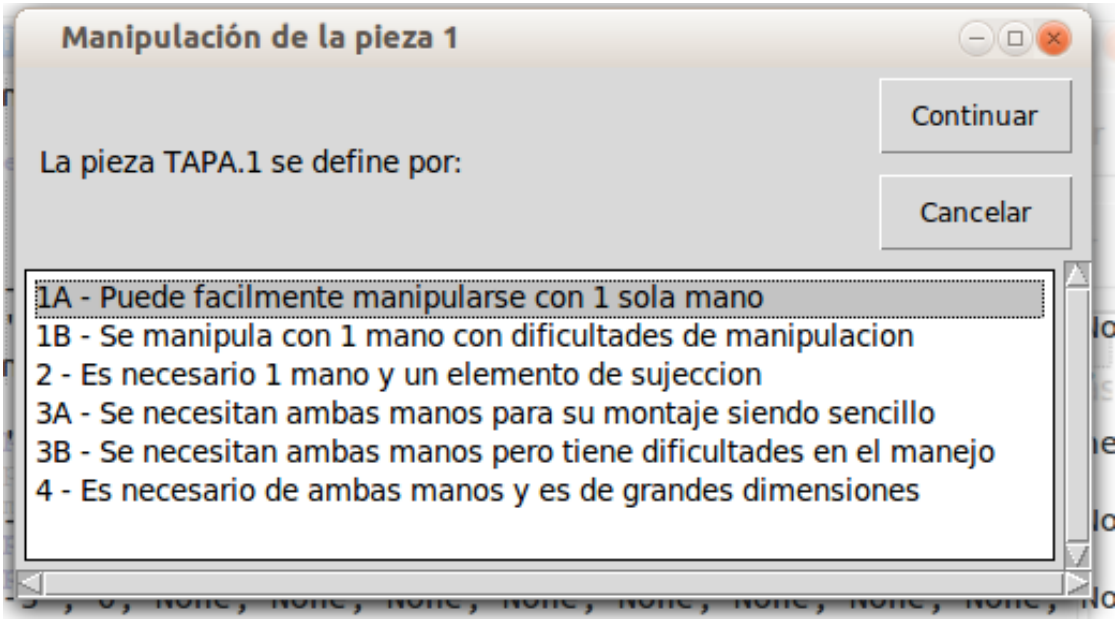


Figura 9.5: Número de manos necesarias

Valores de la simetría Recordemos que para conseguir el código de manipulación uno de los primeros pasos era seleccionar el tipo de simetría que presentaba la pieza (figura 7.2). Para este conocimiento el agente interactúa con el usuario el cual selecciona con un *buttonbox* la figura que más se ajusta a la forma de la pieza analizada.

Según la selección y siempre fundamentándose en la base del método DFMA, se aportan los valores matriz[i][3] y matriz[i][4], los cuales son los valores alfa y beta respectivamente.

Número de manos necesarias Al igual que para la selección de pieza crítica, para seleccionar el valor de las manos necesarias se utiliza una función *choicebox* importando el módulo *easygui*.

La figura 9.5 muestra los criterios de selección necesarios, tal y como se vio en el apartado de la metodología DFMA respecto al índice de manejabilidad. En los casos 1A, 1B, 3A y 3B, aparte de definir el tamaño de la pieza, no es necesario más cuestiones sobre manipulación para el cálculo del índice de manipulación. En cambio, si se selecciona la opción 2 ó 4, la aplicación muestra otra ventana. La figura 9.6 muestra la ventana que aparece después de seleccionar "Es necesario una mano con un elemento de sujeción".

Esta ventana ofrece las opciones correspondientes a la manipulación con una mano y un elemento de sujeción.

Ocurre lo mismo si elegimos la opción 4 del cuadro donde nos preguntan el número de manos necesarias "Es necesario ambas manos y es de grandes dimensiones", al clickar este valor, después de introducir los valores del tamaño de la pieza, nos permite elegir entre los valores que muestra la figura 9.7

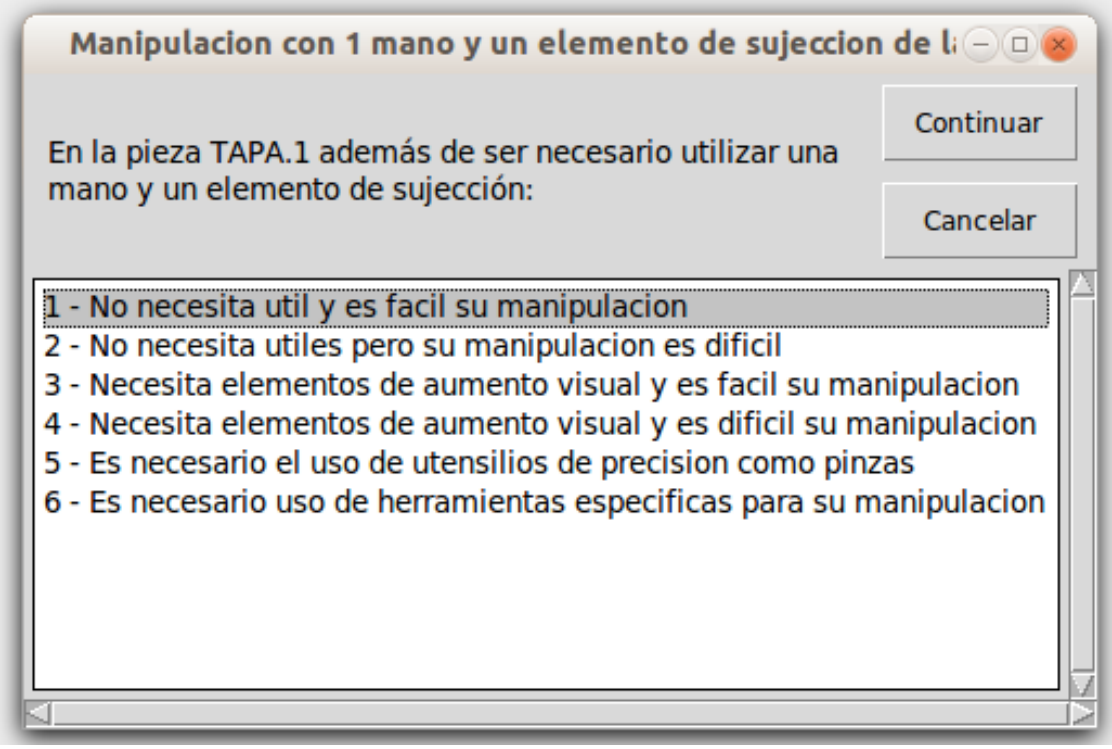


Figura 9.6: Una mano con sujeción

Entre las opciones que se muestran, se determinan pesos de la pieza y la posibilidad de necesitar ayuda de una segunda persona o incluso ayuda mecánica.

Tamaño de la pieza Ante la necesidad de evaluar tamaños de la pieza para poder obtener los valores del índice de manipulación, tal y como Boothroyd y Dewhurst desarrollan en su método de evaluación, era necesario conocer las dimensiones de las piezas analizadas.

Para ello cuando se comenzó a desarrollar la aplicación se intentó encontrar los valores en el código del programa, y aunque posible para una máquina, era de gran dificultad de programación el calculo de las dimensiones del producto desde el propio fichero STEP-XML, por lo que se optó por la introducción de los datos mediante inserción manual. Para ello se ha preparado, gracias al tantas veces mencionado módulo *easygui,* una interfaz de introducción de valores, una función denominada *multenterbox()* que como el resto de las funciones de easygui hay que introducir una serie de parámetros como, los que anteriormente hemos visto, title y msg, pero en este caso hay que definir dos nuevos parámetros que son:

- fieldNames: Una lista de los nombres de los textbox que se tienen que rellenar.
- fieldValues: Una lista de los valores preestablecidos para los campos.

Todo esto se describe en una función nomenclada *ventana_tamanos().* Que físicamente es tal y como se muestra en la figura 9.8

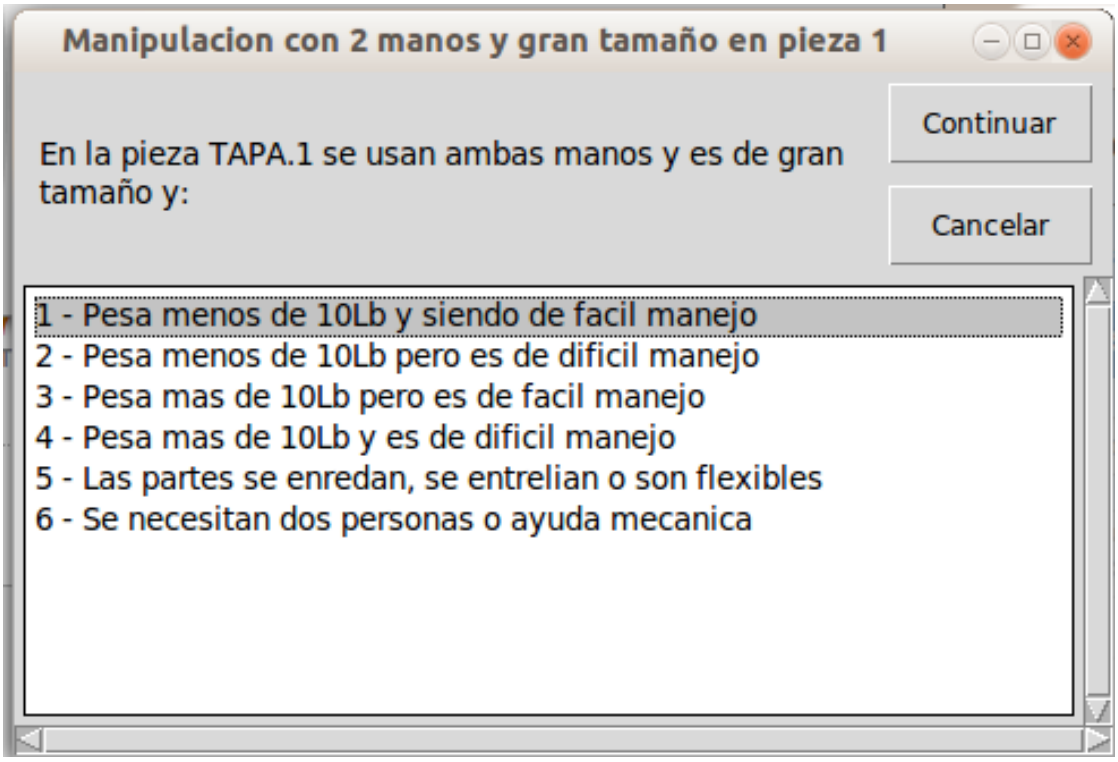


Figura 9.7: Necesidad de 2 manos y gran tamaño

Después de introducir todos estos datos el programa es capaz de calcular por medio de una compleja estructura de condicionales, los valores que del índice de manejabilidad que tiene la pieza. Esta tarea se reitera el número de veces igual al número de piezas que hay.

9.2.3.5. Índice de Inserción

Al igual que el índice de manipulación, el índice de inserción es un valor numérico de dos cifras obtenidas por deducción entre unas sentencias posibles. Este valor se refleja como un tiempo de inserción en segundos.

```
def ventana_ensamble(self, Z):

    msg ="\nRespecto a otra pieza a ensamblar, la pieza " +
        matriz[Z][1] + ":\n"
    title = "Insercion de piezas"
    choices3 = ["1 - Son procesos completamente distintos"\
    , "2 - Se insertan de forma inmediata"\
    , "3 - Se ensamblan mediante un proceso"]
    choice3 = choicebox(msg, title, choices3)
```

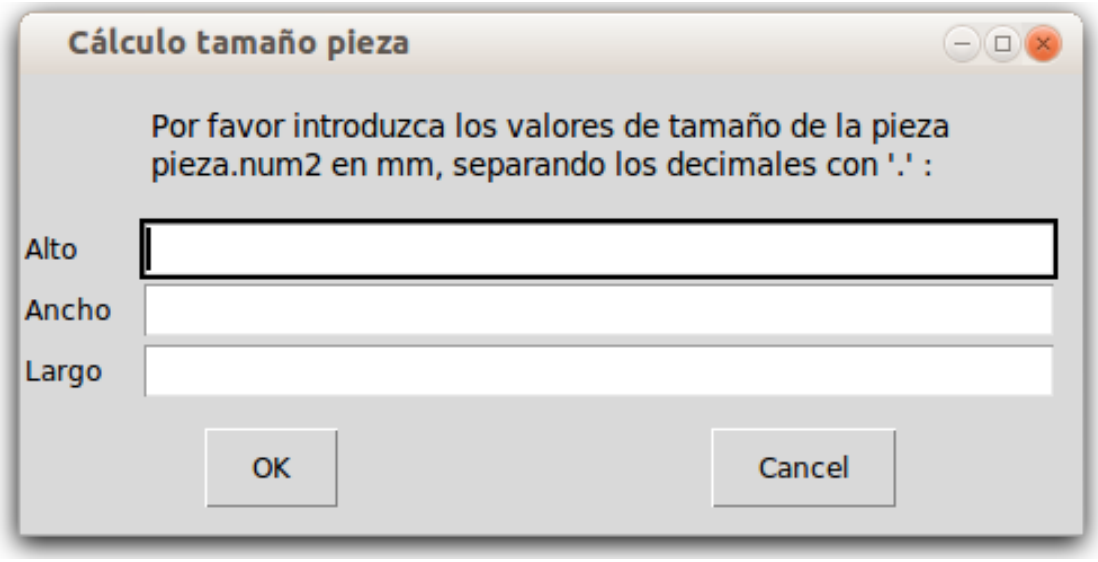


Figura 9.8: Interfaz para introducir tamaño

Primero se llama a la función *ventana_ ensamble(self, Z)* siendo Z el valor id de la pieza que se está analizando. Comienza con una selección *choice3* según el valor de esta selección finalmente se obtendrán unos valores u otros como índice.

A continuación se muestra el código si la se elige la primera opción del *choice3.* Es interesante cómo se define la matriz[i][10] según la elección, puesto que este valor es la primera cifra del índice de inserción, y de aquí pueden obtenerse los valores 0, 1 ó 2:

```
1     if choice3[0] == "1":
2         msg ="\n Seleccione la facilidad de inserción:\n"
3         title = "Inserción Inmediata - Inserción"
4         choices4 = ["1 - Fácil localización"\
5         , "2 - Visión o acceso dificil"\
6         , "3 - Visión y acceso dificil"\
7         ]
8         choice4 = choicebox(msg, title , choices4)
9         matriz[Z][10] = int(choice4[0]) - 1
```

Después de este código y aún en la selección *"choice3 == 1"* se muestran una serie de preguntas, de la misma manera que se hizo en el índice de manipulación, para concretar la segunda cifra del índice de ensamblabilidad, estas cifras vienen dadas en las variables entre las lineas 15 y 30 del siguiente código fuente:

```
1     msg ="\n ¿Después del ensamblado necesita de sujección para
          mantener la posción y la orientación?\n"
2     title = "Procesos distintos"
3     choices5 = ["1 - No, y es de fácil alineación"\
4     , "2 - No, pero su alineación es compleja"\
```

```
5       , "3 - Sí, pero es de fácil alineación"\
6       , "4 - Sí y su alineación es dificil"\
7       ]
8       choice5 = choicebox(msg, title , choices5)
9
10          msg = "¿Es necesario el uso de fuerza porque la pieza
               ofrece resistencia?"
11          title = "Resistencia de la Pieza"
12          error = ynbox(msg=msg, title=title)
13          if error == 0:
14              if choice5[0] == 1:
15                  matriz[Z][11] = 0
16              elif choice5[0] == 2:
17                  matriz[Z][11] = 2
18              elif choice5[0] == 3:
19                  matriz[Z][11] = 6
20              else:
21                  matriz[Z][11] = 8
22          else:
23              if choice5[0] == 1:
24                  matriz[Z][11] = 1
25              elif choice5[0] == 2:
26                  matriz[Z][11] = 3
27              elif choice5[0] == 3:
28                  matriz[Z][11] = 7
29              else:
30                  matriz[Z][11] = 9
```

Al igual que pasa con la selección *"choice3 == 2"*, donde podemos encontrar que se le da el valor a la matriz[i][10], el cual es el valor de la primera cifra del índice de inserción, en la línea 10, fruto de la suma del valor elegido en la selección sumándole la cifra "2". También se muestra cómo se da el valor a la matriz[i][11] en la última línea del siguiente código, otorgado por las dos primeras cifras de la selección menos la cifra "1".

```
1
2       elif choice3[0] == "2":
3           msg ="\n Seleccione la facilidad de inserción:\n"
4           title = "Inserción Inmediata - Inserción"
5           choices4 = ["1 - Fácil localización"\
6           , "2 - Visión o acceso dificil"\
7           , "3 - Visión y acceso dificil"\
8           ]
9           choice4 = choicebox(msg, title , choices4)
```

```
10          matriz[Z][10] = int(choice4[0]) + 2
11          if choice4 == None:
12              Z = 0
13              self.ventana_manipulacion(Z)
14          else:
15              msg ="\n Seleccione la facilidad de alineación:\n"
16              title = "Inserción Inmediata - Alineación"
17              choices5 = ["01 - Apriete con fácil alineación"\
18              , "02 - Apriete con alineación dificil"\
19              , "03 - Inserción por doblado fácil"\
20              , "04 - Inserción por doblado dificil"\
21              , "05 - Inserción por doblado dicil con resistencia"\
22              , "06 - Remachado fácil"\
23              , "07 - Remachado dificil"\
24              , "08 - Remachado dificil con resistencia"\
25              , "09 - Roscado con fácil alineación"\
26              , "10 - Roscado de dificil alineación"\
27              ]
28              choice5 = choicebox(msg, title, choices5)
29              if choice5 == None:
30                  Z = 0
31                  self.ventana_manipulacion(Z)
32              else:
33                  matriz[Z][11] = int(choice5[0:2]) - 1
```

La misma situación sucede cuando el *choice3* es igual a 3. Se aplican valores de la primera y segunda cifra del índice de inserción en las lineas 21 y 23 respectivamente:

```
1
2       elif choice3[0] == "3":
3           msg ="\n Elija uno de estos procesos:\n"
4           title = "Operación distinta"
5           choices4 = ["01 - Doblado"\
6           , "02 - Remachado"\
7           , "03 - Roscado"\
8           , "04 - Grandes deformaciones"\
9           , "05 - Unión por fricción"\
10          , "06 - Soldadura eléctrica"\
11          , "07 - Soldadura por arco"\
12          , "08 - Proceso químico"\
13          , "09 - Orientación/Ajuste de piezas sin sujección"\
14          , "10 - Otros procesos (Ej: Inserción de liquidos...)"\
15          ]
```

$$EM = \frac{3 * NM}{TM}$$

Figura 9.9: Índice de eficiencia del diseño según DFMA

```
16          choice4 = choicebox(msg, title , choices4)
17          if choice4 == None:
18              Z = 0
19              self.ventana_manipulacion(Z)
20          else:
21              matriz[Z][10] = 9
22              print choice4[0:2]
23              matriz[Z][11] = int(choice4[0:2]) - 1
```

Posteriormente se unen ambas cifras en una cadena para obtener el índice de inserción, y este valor lo toma la matriz[i][12].

```
matriz[Z][12] = str(matriz[Z][10]) + str(matriz[Z][11])
```

El tiempo de inserción asociado al código de inserción, es definido con una sentencia para cada caso posible de códigos que existan en las tablas DFMA. Estos casos son aplicados de la siguiente manera:

```
1           if matriz[Z][12] == "00":
2               matriz[Z][13] = "1.5"
3           elif matriz[Z][12] == "01":
4               matriz[Z][13] = "2.5"
5           elif matriz[Z][12] == "02":
6               matriz[Z][13] = "2.5"
7           elif matriz[Z][12] == "03":
8               matriz[Z][13] = "3.5"
9           elif matriz[Z][12] == "06":
10              matriz[Z][13] = "5.5"
11          elif matriz[Z][12] == "07":
12              matriz[Z][13] = "6.5"
13          [...]
```

9.2.3.6. Mostrar el valor Índice DFMA

En este subapartado se muestra la forma para calcular el valor del índice según la fórmula del método DFMA:

Para ello, como se aprecia en el código a continuación, hay que obtener el tiempo de ensamblado total el cual se obtiene cuando se suman todos los valores de tiempos de manipulación de todas las piezas con los tiempos de inserción de todas las piezas, esto se consigue con un bucle *for:*

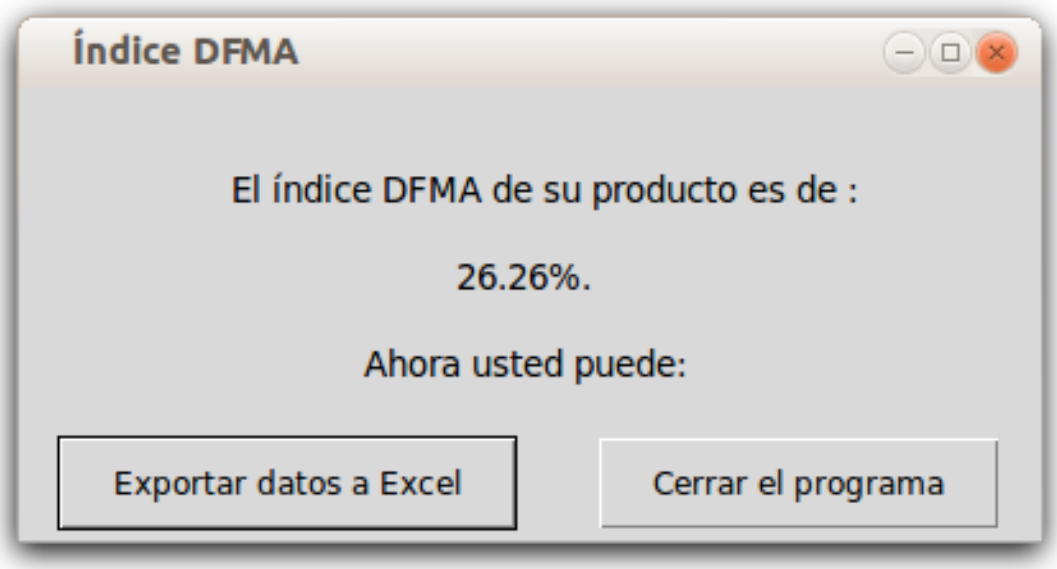


Figura 9.10: Interfaz para mostrar el valor del índice DFMA del producto analizado

```
for i in range(numero_filas):
        tiempo_total = float(matriz[i][8]) + float(matriz
            [i][13]) + tiempo_total
        minimas_partes = matriz[i][2] + minimas_partes
```

Después se aplica la fórmula de la siguiente manera:

```
valor_indice = (3 * minimas_partes / tiempo_total) * 100
```

Y finalmente se muestra por pantalla con un *buttonbox*. El cual además de mostrarnos el índice nos permite exportar los datos a una hoja de cálculo y cerrar la aplicación.

9.2.3.7. Exportar datos a una hoja de cálculo

Como mencionamos en otro capítulo para exportar los valores de la matriz, los cuales hemos ido rellenando conforme se iba avanzando en el Sistema Experto de cálculo de DFMA, son necesario tres módulos para trabajar con hojas de cálculo en Python. Los módulos son los importados con el siguiente código fuente:

```
from xlrd import open_workbook
import xlwt
from xlutils.copy import copy
```

Como estos módulos ya se han explicado en capítulos anteriores, se procederá a analizar la función encargada de abrir, modificar y guardar la hoja de datos. Esta función se llama *excel()* y lo que hace es abrir el fichero de "plantilla.xls" el cual es la plantilla donde se pegarán los datos de la matriz[i][j]. Después de abrir el documento se copia y se pega en un fichero nuevo. Entre la linea 7 y 10 se crean los estilos que serán utilizados a la hora de

rellenar el documento. En la línea 6 se escribe el nombre y la ruta del fichero analizado, en la celda (2, 3). A partir de la línea 11 se crea un bucle que coge cada pieza (cada fila de la matriz) y va escribiendo los valores en las celdas [i] + 6 (Para encajar debajo del cabecero azul de la plantilla) y los valores se van escribiendo en cada casilla de forma ordenada.

```
1  def excel(self):
2
3      rb = open_workbook('plantilla.xls',formatting_info=True)
4      wb = copy(rb)
5      ws = wb.get_sheet(0)
6      ws.write(2,3,fichero)#nombre fichero
7      style0 = xlwt.easyxf('font: colour blue, bold on')
8      style1 = xlwt.easyxf('alignment: horizontal center')
9      style2 = xlwt.easyxf('font: colour black, bold on')
10     for i in range(numero_filas):
11         ws.write(i + 6,0,int(matriz[i][0]) + 1,style1)#id
12         ws.write(i + 6,1,str(matriz[i][1]),style1) #nombre
13         ws.write(i + 6,2,str(matriz[i][2]),style1)#critica
14         ws.write(i + 6,3,str(matriz[i][9]) + str(matriz[i][6]),
               style1)#cod manip
15         ws.write(i + 6,4,float(matriz[i][8]),style1)#tiempo man
16         ws.write(i + 6,5,str(matriz[i][12]),style1)#cod inser
17         ws.write(i + 6,6,matriz[i][13],style1)#tiempo inserc
18         ws.write(i + 6,7,float(matriz[i][8]) + float(matriz[i
               ][13]),style1)#tiempo unit pieza
19         ws.write(i + 6,8,0.007 *(float(matriz[i][8]) + float(
               matriz[i][13])),style1)
20     ws.write(i + 8,2,"NM = " + str(minimas_partes), style0)#suma
           criticas
21     ws.write(i + 8,7,"TM = " + str(tiempo_total), style0)#suma
           tiempos
22     ws.write(i + 8,8,"CM=" + str(0.007 * tiempo_total), style0)#
           suma criticas
23     ws.write(i + 10,7,"Indice DFMA = " + valor_indice_r + " %",
           style2)
24
25     msg = "Guardar Tabla de estudio DFMA del producto"
26     title = "Exportar datos excel"
27     default = "DFMAsoft"
28     filetypes = [["*.xls", "*.xlsx", "*.odf", "*.ots", "Hojas de
           datos"] ]
29     excel = filesavebox(msg, title, default, filetypes)
```

```
30      if filesavebox == None:
31          self.valor_indice()
32      elif filesavebox == "plantilla.xls":
33          msgbox("Se ha elegido otro nombre para el fichero. No
               puede sustituir la plantilla")
34          archivo_salida = "DFMAnoPLANTILLA.xls"
35      else:
36          archivo_salida = excel
37      wb.save(excel)
```

Los valores a los que hacemos referencia son los valores de las columnas de la matriz. Cada columna se rellena de la siguiente forma:

1. ID
2. Nombre
3. Crítica
4. Código de manipulación
5. Tiempo de manipulación
6. Código de inserción
7. Tiempo de inserción
8. Tiempo unitario por pieza (Suma de tiempo de manipulación y tiempo de inserción)
9. Coste unitario por pieza (Multiplica 0.007€ * tiempo en segundos) 0.007€/segundo viene dado de una estimación de 25€/h del tiempo del trabajador ensamblando productos.

Y finalmente se escribe unas celdas mas abajo la suma total de:

- NM: Piezas críticas
- TM: Tiempo total
- CM: Coste total
- y el Índice DFMA

*Todos los tiempos indicados son en segundos y el coste en euros.

Posteriormente se utiliza el módulo *easygui* para utilizar la función *filesavebox* que es la que permite elegir dónde queremos guardar los ficheros. Y por último un msgbox indica que los datos han sido exportados a la hoja de cálculo y que se procede a cerrar el programa (y por lo tanto, cerrar los agentes implicados)

9.2.4. Agente representacion

El agente representación es un agente de la aplicación desarrollada que utiliza estructura SPADE. Gracias a la estructura, que se definió con anterioridad en el capítulo correspondiente, este agente se ayuda del conjunto de la plataforma SPADE y crea una red multiagente.

Tal y como se describió al comienzo del presente capítulo, en la descripción general de los agentes, este agente presta relación directa con el resto de los agentes de la aplicación, exceptuando el agente *dfma*.

Este agente, es iniciado desde el agente *inicio*, el cual envía a la plataforma SPADE la señal para iniciarlo.

La peculiaridad de este agente reside en su código, tal y como se muestra a continuación:

```
1  class representacion(spade.Agent.Agent):
2
3      class representador(spade.Behaviour.OneShotBehaviour):
4          def onStart(self):
5              print "start agente representación"
6
7          def _process(self):
8              print "Representando documento . . ."
9  ##      NO CONSIGO INSTALAR OCCMODEL, GEOTOOLS, GLTOOLS
10 ##      try:
11 ##              objects = Tools.readSTEP(fichero)
12 ##              viewer(objects)
13 ##      except:
14 ##              print "agente de representación no funciona"
15             nombrearchivo = fichero.split(".")
16             del nombrearchivo[-1]
17             nombrearchivo.append("wrl")
18             ficherowrl = ".".join(nombrearchivo)
19             print ficherowrl
20             orden = "view3dscene " + ficherowrl
21             try:
22                 os.system(orden)
23             except:
24                 print "no se encuentra fichero 3D wrl"
25
26         def onEnd(self):
27             print "kill agente representación"
28
29     def _setup(self):
30         self.addBehaviour(self.representador())
```

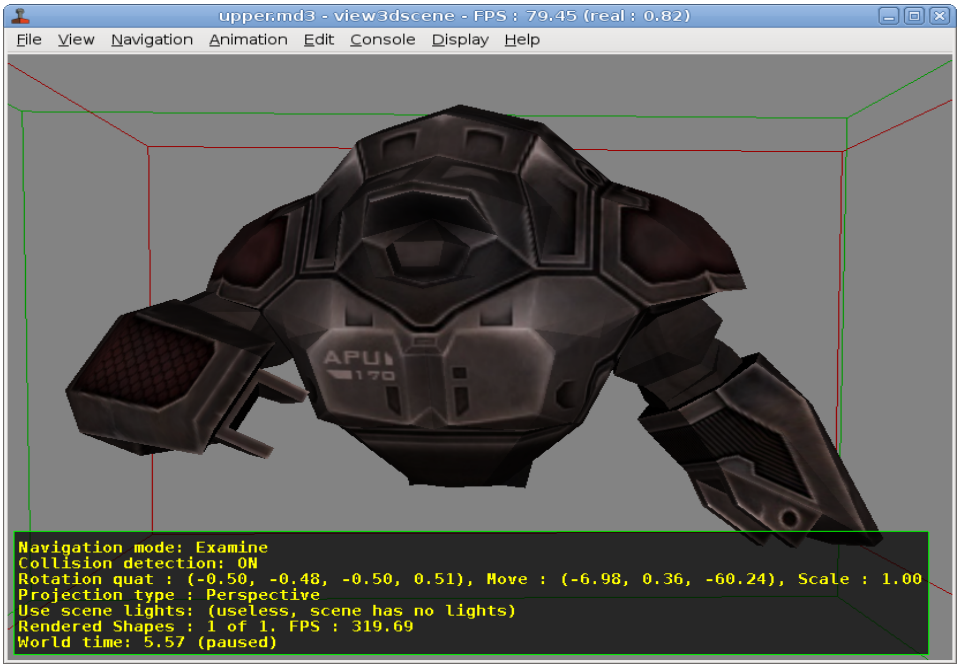


Figura 9.11: Ejemplo visualizador 3D: view3dscene

Lo más característico de esta parte del código es la forma de utilizar el módulo pythonOCC. Cómo el módulo OCC no ha podido ser instalado por incompatibilidad de librerías, la instancia ha sido convertida a comentario. No se ha querido eliminar por si el proyecto consigue hacer compatibles dichas librerías.

Las funciones utilizadas para la representación en PythonOCC son las que aparecen en la línea 11 y 12 *Tools.readSTEP(fichero)* y *viewer(objects)*. De esta forma el módulo lee el fichero STEP y lo muestra en una ventana propia.

Por los problemas de compatibilidad, como ya se ha mencionado, hubo que buscar una alternativa para la representación 3D del producto.

La alternativa que se llevo a cabo manda al sistema la orden de abrir una aplicación (ajena a la de este proyecto) y mostrar el fichero .wrl de la pieza. Esta aplicación alternativa se llama *view3dscene*, la cual es compatible con todas las plataformas y es un visor 3D para ficheros *.wrl* que son un standard de representación de archivos 3D, exportable desde cualquier aplicación CAD.

Para aplicar la solución alternativa se carga el módulo *sys* el cual permite interactuar con el sistema operativo, y se le ordena con la función *os.system(orden)* que ejecute la orden que se determina en la línea 20 (*orden = "view3dscene" + ficherowrl)*.

Entre las líneas 16 y 18 del código se crea una lista con las partes del nombre del fichero y su ruta separadas por "." y se elimina de esta lista el último elemento, posteriormente se añade el tipo de documento *.wrl* vuelve a unir dicha lista y esta cadena es la que se da a *ficherowrl*. Esto se hace para encontrar el fichero *.wrl* del fichero *.step.xml* analizado por la aplicación.

Lo que genera una ventana donde podemos visualizar el fichero y permite moverlo, ampliarlo e interactuar con él en el espacio, como se muestra en la figura 9.11.

Para concluir se explica el código *except* que muestra una excepción por pantalla cuando no se encuentra el *fichero.wrl* y el final del agente representación cuando se cierra este.

10 Resultados Finales

Durante el transcurso de este capítulo se procederá a desarrollar la aplicación mediante un método práctico. Se evaluará un diseño y se comparará el resultado a un rediseño del mismo.

Posteriormente se compararán los resultados conseguidos y se reflejarán las conclusiones finales.

10.1. Caso 1 - Caja con cuatro tornillos

Se inicia la aplicación y seleccionamos el proyecto Caja cuatro tornillos tal y como se muestra en la figura 10.1.

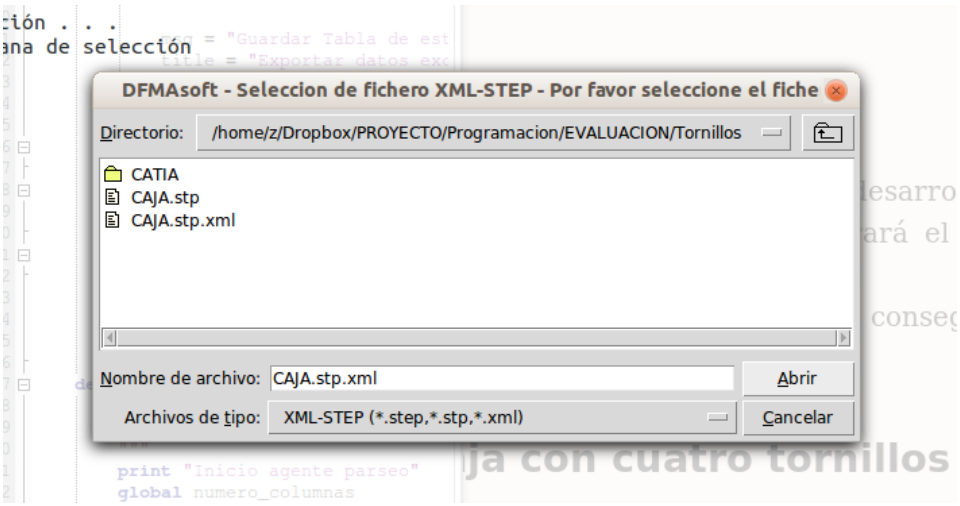


Figura 10.1: Ventana de selección de fichero STEP-XML

Cuando se acepta el programa comienza a parsear el documento hasta confirmar que el fichero está bien formado y que es un fichero STEP-XML. En ese momento la aplicación hace las llamadas para activar la plataforma SPADE y arrancar el agente *representación* y el agente *dfma*. Estos agentes se muestran como en la figura 10.2

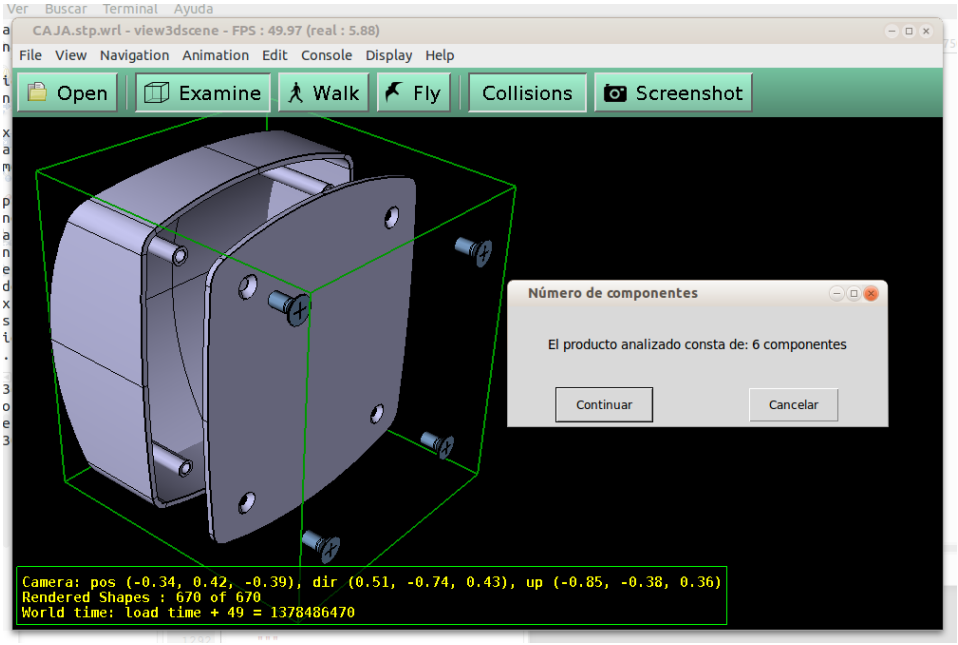


Figura 10.2: Visualizador 3D y ventana con número de componentes

Como se aprecia en la representación 3D, y tal y como muestra la ventana, el número de componentes del sistema analizado es de 6. A continuación, en la siguiente ventana se muestra de forma automática el nombre de cada pieza del producto analizado (figura 10.3)

Figura 10.3: Listado de los nombres de cada componente

Después de esta ventana, se muestra una ventana de selección con distintas sentencias que evalúan la criticidad de cada parte (todas las respuestas hacen que la pieza sea una

pieza crítica excepto la última). En la figura 10.4 se muestran los valores seleccionados para las pieza TAPA1 y TORNILLO4.

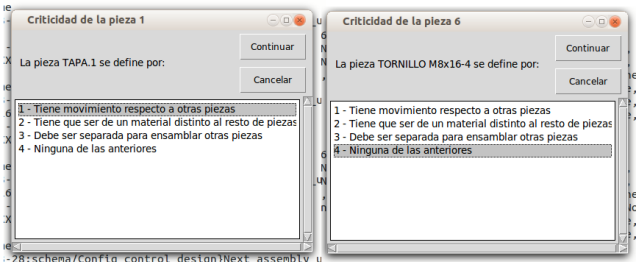


Figura 10.4: Criticidad de los componentes TAPA y TORNILLO

A continuación comienza la selección para calcular el código e manipulación, el que nos da como resultado el tiempo de manipulación. Para ello hay que seleccionar los valores correspondientes en cada una de las ventanas. Los valores marcados en la imagen, son los que han sido seleccionados para cada pieza. (La pieza TAPA y la pieza BASE son iguales respecto al índice de fabricación y ensamblaje, lo único que varía es el tamaño. Por otra parte los tornillos son iguales entre si, por eso sólo se han realizado capturas de pantallas de uno de ellos).

Figura 10.5: Simetría de forma TAPA1

Se comienza con la selección de los valores de la simetría de forma. Para la TAPA1 y BASE1 la selección es la forma D lo que equivale a un valor alfa e 90º y un valor beta de

180º. Para el valor elegido de los tornillos es la forma E, o su equivalente en ángulo alfa de 360º y beta de 0º. La figura 10.5 muestra la ventana de selección correspondiente.

A continuación se estudia el número de manos que se utilizan para manejar cada pieza, la selecciones elegidas son las que aparecen en la figura 10.6.

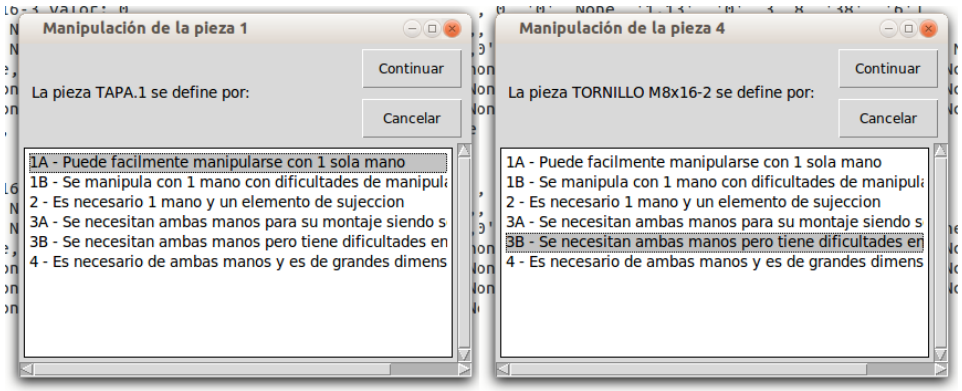


Figura 10.6: Número de manos necesarias para su manipulación

Después de esta selección, se abre la ventana para indicar los tamaños de cada pieza. (fig 10.7). Recordamos que era un *multenterbox* del módulo *easygui* el cual nos permitía introducir varios valores en los campos correspondientes y después no devolvía dichos valores en forma de lista.

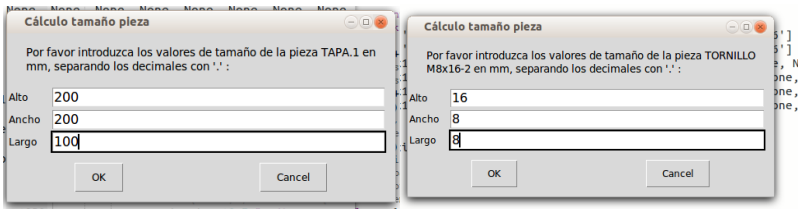


Figura 10.7: Ventanas de introducción de tamaños de cada pieza

Así finalizábamos las ventanas referentes al cálculo del código de manipulación y comenzamos con el cálculo del módulo de inserción. La primera ventana para este índice es la que se muestra en la figura 10.8 donde debemos seleccionar el proceso de inserción.

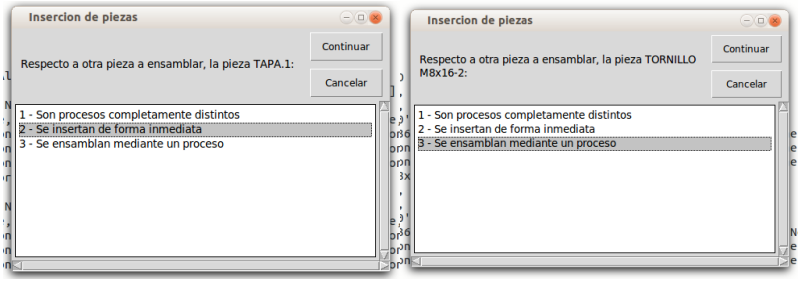


Figura 10.8: Ventanas de selección de formas de inserción

A continuación la selección de TAPA1 y de BASE1 (“*2 - Se insertan de forma inmediata”)* nos muestra el siguiente diálogo (fig 10.9) donde se elige “*fácil localización”,* y posteriormente se elige “*Roscado con fácil alineación*”.

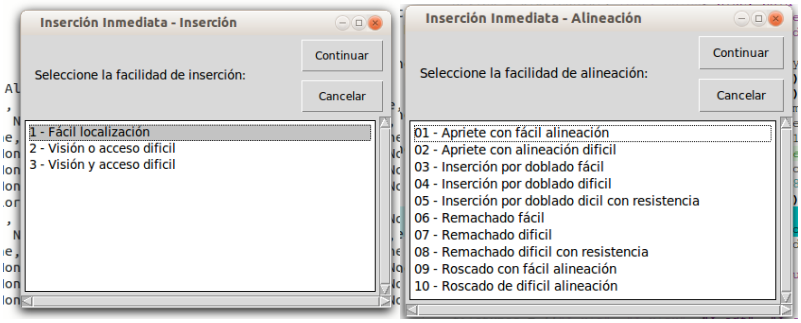


Figura 10.9: Ventanas de selección de formas de inserción de la TAPA1 y BASE1

El mismo caso para los tornillos después de seleccionar el valor “*3 - Se ensamblan mediante un proceso”,* genera la siguiente combinación de ventanas:

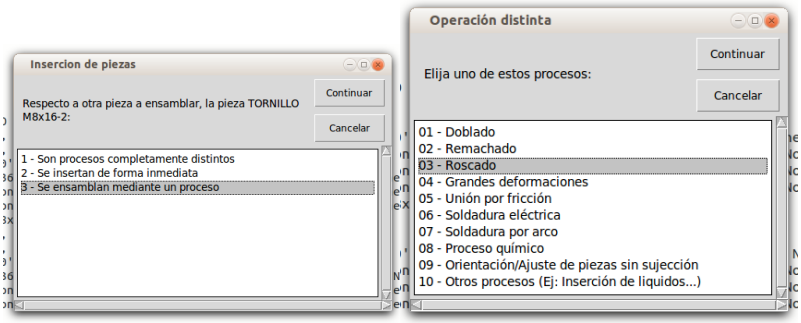


Figura 10.10: Ventanas de selección de formas de inserción de la TAPA1 y BASE1

Con las selecciones que se muestran en las respectivas figuras, el programa calcula los códigos de inserción correspondientes para cada pieza y por lo tanto dispone de todo lo

necesario para calcular el índice DFMA del producto. Por ello la siguiente ventana es la ventana donde aparece el valor del índice DFMA (fig 10.10)

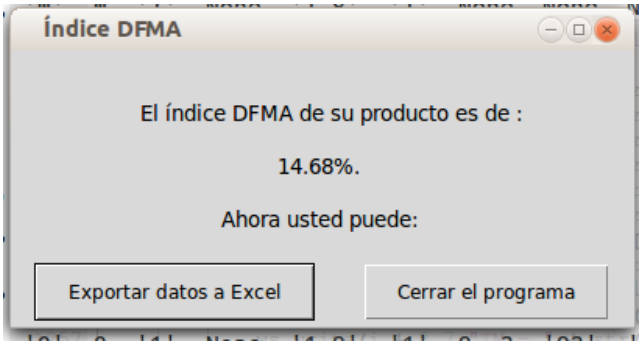
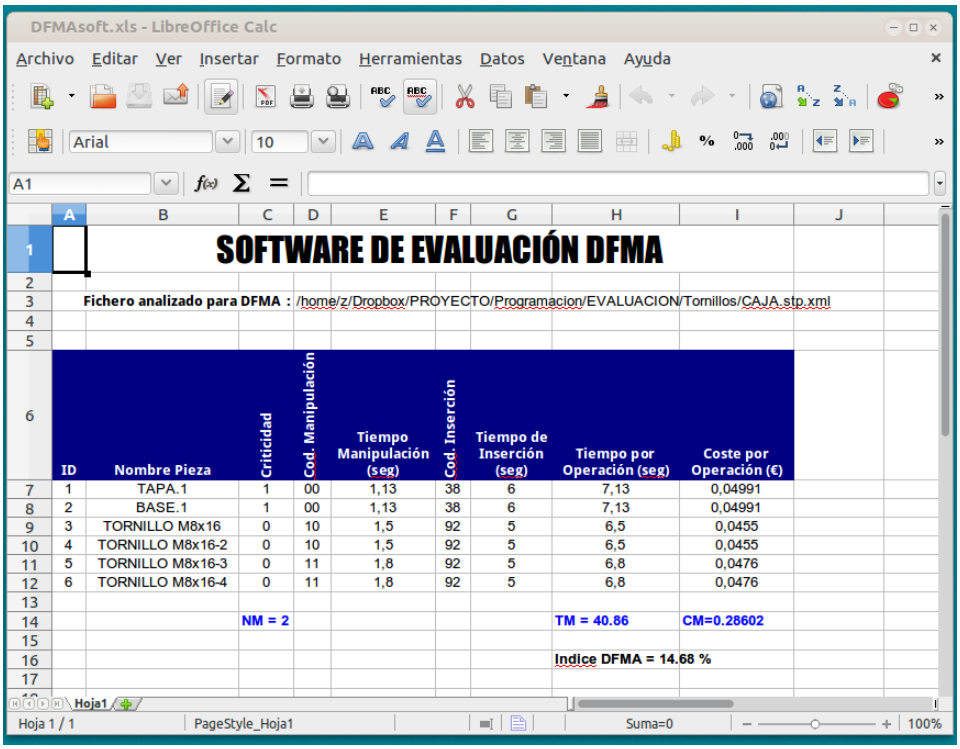


Figura 10.11: Ventanas de selección de formas de inserción de la TAPA1 y BASE1

Como se puede apreciar por el valor del porcentaje obtenido, el diseño de la caja, respecto a su fabricabilidad y ensamblabilidad, deja mucho que desear. De todas formas podemos analizar de una forma más detallada los datos obtenidos en la hoja de cálculo que exporta el programa:



ID	Nombre Pieza	Criticidad	Cod. Manipulación	Tiempo Manipulación (seg)	Cod. Inserción	Tiempo de Inserción (seg)	Tiempo por Operación (seg)	Coste por Operación (€)
1	TAPA.1	1	00	1,13	38	6	7,13	0,04991
2	BASE.1	1	00	1,13	38	6	7,13	0,04991
3	TORNILLO M8x16	0	10	1,5	92	5	6,5	0,0455
4	TORNILLO M8x16-2	0	10	1,5	92	5	6,5	0,0455
5	TORNILLO M8x16-3	0	11	1,8	92	5	6,8	0,0476
6	TORNILLO M8x16-4	0	11	1,8	92	5	6,8	0,0476

Figura 10.12: Hoja de cálculo con toda la información sobre el producto

10.2. Caso 2 - Caja con cuatro pestañas

En el segundo caso se evaluará de igual manera el rediseño del producto. Para no extendernos demasiado se opta por no mostrar las capturas de pantalla de todo el proceso, en su lugar se describirá el proceso de selecciones de forma escrita de la manera más detallada posible.

En este caso, la ventana de número de pieza dice que sólo existen dos partes en el producto. Estas partes son nombradas en la siguiente ventana como BASE1 y TAPA1, pues son pequeñas modificaciones del caso anterior.

A la hora de elegir la criticidad de ambas piezas, ambas piezas son piezas críticas porque ambas son de la elección "*han de ser separadas para ensamblar otras piezas".* Al igual que en el caso anterior.

A la hora de la selección del tipo de simetría, en ambos casos se ha elegido la forma "*D"* que significa que tiene simetría alfa de 90º y simetría beta de 180º.

En la selección del número de manos que son necesarias para la manipulación de la pieza, para ambas piezas se han seleccionado el mismo valor "*Pueden manipularse con una sola mano".*

Posteriormente se han introducido los valores aproximados de medida:

- BASE1: 200,200,100mm
- TAPA1: 200,200, 5mm

Con todos los valores introducidos hasta ahora para ambas piezas, se consigue el valor del índice de manipulación, el programa lo calcula de forma automático gracias al Sistema Experto preprogramado. Con este índice ya se conoce el tiempo de manipulación empleado de cada pieza.

Ahora se procede a calcular el índice de inserción. En la primera ventana donde hay que elegir el tipo de proceso, se elige "*Son procesos completamente distintos".*

Después de esta ventana se elige el la facilidad de inserción, en este caso "*Visión o acceso difícil*", cuando nos pregunta si después del ensamblado necesita de algún útil para mantener su posición, seleccionamos "*No pero su alineación es compleja".* Y por último cuando nos pregunta si es necesario aplicar fuerza porque la pieza ofrece resistencia indicamos *"Sí".*

Este proceso de selección de posibilidades es idéntico para las dos piezas que forman el producto analizado.

Después de los cálculos internos de la aplicación, obtenemos el valor del índice DFMA de nuestro producto, el cual es de "*26.95 %.*"

10.3. Conclusiones de los resultados

Tras un proceso de rediseño y utilizando como guía la metodología de diseño para la fabricabilidad y el ensamblaje, desarrollado por Boothroyd y Dewhurst, se ha conseguido mejorar nuestro producto de una manera sustanciosa.

ID	Nombre Pieza	Criticidad	Cod. Manipulación	Tiempo Manipulación (seg)	Cod. Inserción	Tiempo de Inserción (seg)	Tiempo por Operación (seg)	Coste por Operación (€)
1	TAPA.1	1	00	1,13	38	6	7,13	0,04991
2	BASE.1	1	00	1,13	38	6	7,13	0,04991
3	TORNILLO M8x16	0	10	1,5	92	5	6,5	0,0455
4	TORNILLO M8x16-2	0	10	1,5	92	5	6,5	0,0455
5	TORNILLO M8x16-3	0	11	1,8	92	5	6,8	0,0476
6	TORNILLO M8x16-4	0	11	1,8	92	5	6,8	0,0476
		NM = 2					TM = 40.86	CM=0.28602
							Indice DFMA = 14.68 %	

Cuadro 10.1: Resultados diseño con tornillos

ID	Nombre Pieza	Criticidad	Cod. Manipulación	Tiempo Manipulación (seg)	Cod. Inserción	Tiempo de Inserción (seg)	Tiempo por Operación (seg)	Coste por Operación (€)
1	TAPA.1	1	00	1,13	19	10	11,13	0,07791
2	BASE.1	1	00	1,13	19	10	11,13	0,07791
		NM = 2					TM = 22.26	CM=0.15582
							Indice DFMA = 26.95 %	

Cuadro 10.2: Resultados diseño con clips

La eliminación de los tornillos y la aplicación de pestañas tiene sus ventajas e inconvenientes.

10.3.1. Conclusiones respecto a tiempos de ensamblado

Las relaciones de tiempos entre el diseño anterior y el rediseño, podrían determinarse por:

- Por un lado el tiempo de inserción en el diseño con pestañas es de "*11,13 segundos*" muy superior al tiempo de operación de la misma pieza en el diseño con tornillos "*7,13 segundos*".
- Por otro lado, se eliminan todas las piezas no críticas, y al tener menos piezas la suma total de tiempos es mucho menor, en el rediseño se tarda prácticamente la mitad en ensamblar la pieza. *"40,86 segundos"* en comparación a "*22,26 segundos*".

10.3.2. Conclusiones económicas

Una sencilla comparativa económica, determinada por la relación del tiempo de operación y el sueldo del operario. Esta consiste en que aunque las operaciones del rediseño son más

caras, hay menos operaciones y el coste total es menor: "*0,28 €*" en el diseño previo y *"0,15 €"* en el rediseño.

Para una conclusión más exhaustiva de los aspectos económicos se hace referencia al capítulo 5 - Mediciones y Presupuestos

10.3.3. Conclusiones finales - Índices de Ensamblabilidad

La conclusión a la que se llega con todos estos detalles analizados es obvia. Es mejor el rediseño, con clips, porque se tarda menos en fabricar, tiene menos piezas por lo tanto es mas barato fabricarlo tanto materialmente como en tiempo por operación. Y los resultados de sus índices DFMA lo demuestran "*14,68 %*" el diseño con tornillos y "26,95 %" la alternativa con clips.

11 Conclusiones del proyecto

En este capítulo se procede a la discusión y conclusión de los resultados obtenidos. Se evaluará la satisfacción con los deseos del proyecto, y de los medios estudiados para cumplir los objetivos.

También se hará un comentario sobre la viabilidad del proyecto en cuestión, después de haber estudiado la aplicación práctica.

Posteriormente se definirán posibles mejoras o futuros proyectos relacionados. Para finalizar este capítulo con el estudio de viabilidad de los objetivos marcados por el proyecto.

11.1. Sistemas multiagentes

Comenzaremos estudiando la inclusión de los sistemas multiagentes en el proyecto. Como ya se explicó con anterioridad, la utilización de este tipo de sistemas y plataformas multiagentes, permitían que el proyecto estuviese vivo, y pudiese seguir creciendo con la inclusión de nuevos agentes. La utilización de los agentes en la aplicación desarrollada ha sido como forma de ejemplo de las posibilidades que se pueden desarrollar bajo un software multiagente.

11.2. Agentes inteligentes

También es de gran importancia en este proyecto la inclusión de agentes inteligentes. Programando un Sistema Experto, como unos de los agentes de la plataforma, creamos una inteligencia artificial que ayuda al usuario y facilita el uso de la metodología DFMA en el diseño de productos. El uso de agentes inteligentes de este proyecto ha sido en instancias concretas, pero su uso puede desarrollarse con sistemas BDI, donde cada agente tiene unos deseos que otros agentes pueden satisfacer, y redes inteligentes de mayor complejidad.

11.3. Software de análisis integrado en aplicaciones CAD

Otro de los temas tratados en este proyecto ha sido la inclusión de software de análisis de diseño en las primeras etapas del diseño asistido por ordenador. Lo que mejora los plazos de los procesos de diseño, y se obtienen mejores resultados de tiempos y económicos. La inclusión de este tipo de sistemas evaluadores integrados en sistemas CAD es una necesidad que no está satisfecha actualmente, por lo tanto es una gran oportunidad para conseguir un nicho de mercado de gran interés.

11.4. Sistemas estandarizados

La utilización de formatos estandarizados para el desarrollo de productos bajo plataformas CAD es esencial. En la actualidad la posibilidad de diseño desde distintas partes del mundo es un hecho por lo que el uso de este tipo de sistemas es completamente necesario. La incompatibilidad entre distintos software CAD debe desaparecer, mientras esto ocurre el uso de formatos estándar como el formato STEP, STEP-XML... y la fiabilidad de estar determinado bajo una normativa ISO, es una de las opciones que se contemplan con mejores perspectivas.

Otro de los sistemas estandarizados utilizados en este proyecto es el de ficheros .wrl, utilizados para representación de escenas tridimensionales en lenguaje VRML. También muy útiles por su sencillez y facilidad a la hora de exportarse de un sistema a otro.

11.5. Python como medio de desarrollo

La evaluación de este lenguaje y de su potencial es muy positiva. Su sencillez estructural, su comodidad ante la gran cantidad de módulos existentes que nos ahorran tiempo de desarrollo y su sintaxis clara son motivos suficientes para la elección de este lenguaje de programación como medio para el desarrollo de herramientas para la evaluación de alternativas de diseño.

11.6. La metodología DFMA

La herramienta de análisis de productos utilizada en esta aplicación. Descrito y detallado en capítulos anteriores, este método de diseño es pieza fundamental para el ahorro económico y para la disminución de tiempos de trabajo. Un análisis a tempranas etapas del proceso de diseño de productos, hace que esta potente herramienta sea útil y gracias a herramientas que simplifiquen su aplicación es pieza indispensable en el diseño de productos.

11.7. Posibles mejoras

Este proyecto puede seguir desarrollándose hasta actuar en todos los pasos del diseño de productos. Una de las mejoras principales que se pueden hacer a este proyecto con tanto potencial, es la inclusión de nuevos módulos para análisis de soluciones, módulos como el de ensamblaje robotizado o el cálculo de sistemas eléctricos, que también tienen sus parámetros para evaluación.

11.8. Viabilidad del proyecto

La viabilidad del proyecto era materia del principal análisis de esta conclusión. Cumpliendo todos los objetivos marcados por los objetivos del proyecto y con los resultados obtenidos

en la aplicación práctica, los ahorros tanto de tiempo de ensamblado (aunque sólo se hayan estudiado ensamblajes manuales) como los ahorros económicos, queda desmostrado que este proyecto es 100 % viable y que tiene mucho potencial para seguir desarrollándose y continuar por distintas ramas del diseño de productos.