Escuela Técnica Superior de Ingeniería Informática

Análisis de Enfoques de Model Based Testing para Pruebas Funcionales orientados a Aplicaciones Web

María Elena Calderón Romero

Curso académico 2010-2011

Análisis de Enfoques de Model Based Testing para Pruebas Funcionales orientados a Aplicaciones Web

Autor

- María Elena Calderón Romero

Tutora

- María José Escalona

Profesores de la asignatura:

- Juan M. Cordero Valle
- David Benavides Cuevas

Miembros del tribunal

-
-
-

Fecha de lectura y defensa:

Calificación obtenida:

Resumen.

En los últimos años las aplicaciones web han ido incrementando en número y a la vez en complejidad debido a la incorporación de nueva tecnología. Esto ha repercutido en un aumento de complejidad de la fase de pruebas dentro del ciclo de vida del desarrollo de software, la cual nos permite asegurar la calidad del producto desarrollado. Esta fase representa un mayor costo y esfuerzo. Con otro tipo de aplicaciones no se le asignaba el tiempo ni esfuerzo necesario. Sin embargo, debido al impacto que puede tener una aplicación web mal probada durante la puesta en marcha de la aplicación, han surgido diversas investigaciones en técnicas para la simplificación de la fase de pruebas. Una de estas técnicas es model based testing, que mediante la representación del comportamiento esperado de la aplicación, genera automáticamente los casos de prueba, incluso permite la ejecución automática de los mismos y su evaluación. El presente trabajo presenta una revisión analítica de los enfoques en model based testing para aplicaciones web orientados a pruebas funcionales, identificando para ello los enfoques existentes dentro de este contexto y realizando un esquema de caracterización para el análisis de las principales características, herramientas y documentación disponible para la aplicación de los enfoques.

Palabras clave.

Model Based Testing, Pruebas Funcionales, Aplicaciones Web.

Índice

Índice de figuras

Índice de tablas.

1. Introducción

En los últimos años hemos experimentado un aumento significativo de las aplicaciones web, y con ello también, se ha visto aumentada la complejidad de las mismas con la aparición de nuevas tecnologías para la web. Todo ello ha significado que dentro del proceso de desarrollo de software se incremente el nivel de complejidad de la fase de pruebas para las aplicaciones web.

La fase de prueba dentro del ciclo de vida del desarrollo de software nos permite asegurar la calidad del producto desarrollado. A pesar de su importancia, a esta fase no se le asigna siempre los recursos y tiempo necesarios, y este problema no es exclusivo de las aplicaciones web. El principal factor que influencia el costo asociado a la prueba es el número total de casos de prueba. Por cada caso de prueba se debe asignar recursos relacionados con la planificación, diseño, y ejecución del mismo. Por consiguiente, la automatización de la generación de los casos de prueba es un mecanismo interesante para reducir el costo y esfuerzo de las pruebas.

Para solucionar este inconveniente, hace ya algunos años, han comenzado a surgir soluciones que permiten simplificar la fase de prueba con herramientas adecuadas para la automatización de la generación de pruebas, incluso algunas, permiten la ejecución de las mismas. Una de estas soluciones es Model Based Testing (MBT), una técnica que representa el comportamiento esperado de la aplicación bajo prueba (SUT) mediante un modelo, y que pretende que a través de este modelo se generen automáticamente los casos de pruebas, incluso plantea que es posible la ejecución y evaluación automática de los resultados si se contara con un oráculo. MBT aparece como una técnica capaz de controlar la calidad del software reduciendo los costos y esfuerzo relacionado al proceso de prueba, debido a que los casos de prueba son generados a partir de un modelo del comportamiento del SUT.

MBT permite la generación automática de casos de prueba usando los modelos extraídos de los artefactos del software [2], y esto trae beneficios para el proceso de prueba no sólo relacionados con un menor costo y esfuerzo, sino también con una mayor calidad, una mejor comunicación entre los desarrolladores y los probadores, una temprana exposición de las ambigüedades en la especificación y el diseño, capacidad de generar automáticamente pruebas útiles y no repetitivas, ejecución automática de las pruebas, y fácil actualización del banco de pruebas para cambios que se presenten en los requerimientos.

El desarrollo de aplicaciones web, en esencia, no es diferente a cualquier otra aplicación de software, ya que incluso es factible utilizar las mismas metodologías de desarrollo de software en ambos casos. La diferencia entre una aplicación web y otra no web, son las tecnologías que se usan y el impacto que puede tener una aplicación mal probada durante la puesta en marcha de la aplicación. Si una aplicación web presenta fallos, tiene el potencial de lanzar un diluvio de quejas e insatisfacción casi inmediatamente, y lo peor, generar pérdidas inmediatas a los stakeholders.

Esto nos lleva a plantearnos qué deberíamos hacer para asegurarnos que la aplicación web que se va a poner en marcha funciona según los requerimientos definidos. Podríamos asegurarnos llevar a cabo una excelente fase de análisis, de diseño, o de implementación, pero siempre la fase de prueba termina siendo la fase previa a la puesta en marcha de la aplicación, con lo que resulta siendo la última oportunidad para detectar posibles fallos que puedan terminar, en ocasiones, en terribles catástrofes y penosas pérdidas. En el caso de las aplicaciones web, no sólo se trata de pérdidas económicas, sino también de pérdida de prestigio, de fiabilidad, de credibilidad por parte de nuestros usuarios o clientes.

El presente trabajo tiene como objetivo identificar los enfoques MBT existentes en la literatura que se orientan a las pruebas funcionales sobre aplicaciones web. Así mismo hace un análisis del contenido de los enfoques identificados para determinar las principales

características que presenta el enfoque, la herramienta que usa y la documentación disponible para la aplicación del mismo.

Para llevar a cabo dicho análisis, el trabajo presenta primero el estado del arte para comprender el marco teórico sobre el cual se desarrolla el trabajo. En la sección 3.1 se presenta el marco teórico para MBT que incluye detalle sobre la definición, el funcionamiento, los beneficios, los problemas y limitaciones, la taxonomía, las herramientas y los casos de éxito relacionados con el uso de MBT. En la sección 3.2 se presenta el marco teórico para las pruebas funcionales, en ella se explica la definición de las pruebas de software, la clasificación de las mismas, y la definición de las pruebas funcionales en base a la clasificación planteada. En la sección 3.3 se presenta el marco teórico para las aplicaciones web, definiendo lo que son las aplicaciones web y el porqué es difícil realizar las pruebas sobre este tipo de aplicaciones. Luego de presentar el marco teórico, en la sección 4.1 se realiza el proceso de búsqueda e identificación de enfoques MBT para aplicaciones web y pruebas funcionales. En la sección 4.2 se desarrolla un esquema de caracterización que permite crear un marco comparativo estándar para los enfoques MBT sobre los cuáles se desea realizar el análisis. Con el esquema de caracterización definido en la sección 4.2 y con los enfoques identificados para el estudio en la sección 4.1, se realiza una instancia del esquema de caracterización sobre cada uno de los enfoques seleccionados en la sección 4.3. Finalmente, en la sección 4.4 se realiza el análisis de los resultados obtenidos, y en la sección 5 se presentan las conclusiones obtenidas tras el análisis.

2. Trabajos Relacionados

Teniendo en cuenta el alcance de los enfoques MBT considerados en el presente trabajo, es decir, enfoques orientados a aplicaciones web y pruebas funcionales, este es el primer trabajo planteado. Esto se puede explicar debido a que existen unos pocos enfoques dentro de este alcance, por ejemplo, una completa revisión [23] de los enfoques existentes entre 1990 y 2006 identificaba sólo un enfoque orientado a aplicaciones web y pruebas funcionales. Sin embargo, para la realización del presente trabajo se han tenido en cuenta cinco trabajos anteriores que tenían un alcance más amplio y que han brindado las pautas para el análisis realizado. En los siguientes párrafos se describen brevemente los trabajos relacionados.

El estudio presentado en [34] (2007), tiene el propósito principal de realizar una revisión sistemática de enfoques MBT para proveer de una base a nuevas actividades de investigación y resumir las evidencias relacionadas con MBT. Por otro lado, define un protocolo de revisión para guiar la ejecución de la revisión de la literatura. Responde a la pregunta sobre qué enfoques MBT han sido descritos en la literatura técnica y cuáles son sus principales características. Es elaborado en diciembre del 2006 y publicado en agosto de 2007, y realiza una completa investigación basada en un esquema de caracterización de los enfoques existentes desde 1990 hasta 2006, con el objetivo de identificar los enfoques que aplican UML. De la búsqueda que realiza se obtienen 406 trabajos, los cuales son agrupados según sean funcionales o estructurales. De esta forma tras analizar los enfoques logra identificar 119 enfoques para pruebas funcionales, 83 enfoques para pruebas estructurales, y el resto como no relacionado con la descripción de un enfoque MBT. En el Appendix B – Model-based Testing Approaches Classification del trabajo, identifica el dominio del software de aplicación MBT para los enfoques estudiados y si son orientados para pruebas estructurales y funcionales, identificando de esta forma sólo un trabajo [23] relacionado con aplicaciones web y pruebas funcionales.

El estudio presentado en [38] (2007), tiene como objetivo caracterizar los enfoques MBT a partir de la literatura técnica disponible al público que sea basada en lo que es relevante para MBT. Responde a la pregunta de cuáles son los enfoques MBT publicados y cuáles son sus principales características.

En el estudio presentado en [39] (2008), se presenta los resultados iniciales con miras a la caracterización y la construcción de un conjunto de conocimientos sobre los enfoques MBT. Su objetivo es que estos resultados sirvan de base para proponer una infraestructura de apoyo a la selección de un enfoque MBT para un determinado proyecto de software.

En el estudio presentado en [42] (2008), se describe la planeación, ejecución y el análisis de los resultados iniciales de un estudio realizado con los investigadores y profesionales relacionados con MBT. El estudio tiene dos propósitos: por un lado observar los atributos que podrían adecuarse a la caracterización de enfoques MBT, y por otro lado, definir lo que es relevante para la selección de un enfoque MBT para proyectos de software. Detalla una lista inicial de los atributos con su importancia identificada. Trata que los resultados obtenidos consoliden una base de conocimiento para apoyar a los ingenieros de software usando enfoques MBT en proyectos de software.

En el estudio presentado en [45] (2010), se presenta una revisión sistemática de las herramientas de apoyo MBT más importantes que se basan en modelos basados en estados. Provee a los profesionales en MBT las pautas en la selección de la herramienta más apropiada según sus necesidades. Responde a las preguntas como cuáles son las herramientas MBT comerciales y de investigación, cuáles son las capacidades de esas herramientas en términos de la cobertura de la prueba, nivel de automatización, y construcción de la prueba.

3. Estado del Arte

3.1. Model Based Testing

El principal factor que influencia el costo asociado a la prueba es el número total de casos de prueba. Por cada caso de prueba se debe asignar recursos relacionados con la planificación, diseño, y ejecución del mismo. Por consiguiente, la automatización de la generación de los casos de prueba sería un mecanismo interesante para reducir el costo y esfuerzo de las pruebas.

MBT aparece como un enfoque capaz de controlar la calidad del software reduciendo los costos y esfuerzo relacionado al proceso de prueba, debido a que los casos de prueba son generados a partir de los artefactos (en MBT a partir de los artefactos se construye el modelo, y de este modelo se derivan los casos de prueba) construidos a lo largo del proceso de desarrollo de software.

Si bien existen otros métodos para la prueba de software, existe una clara diferencia entre estos y MBT. En otros métodos, aunque se utilice un instrumento automatizado para la prueba, el ingeniero de pruebas debe primero conceptualizar cada caso de prueba, a continuación, documentarlo para cada entrada manual y luego ejecutar un programa de pruebas. Por otro lado, en MBT, el ingeniero de pruebas desarrolla un modelo, el cual soporta la generación y evaluación automática de los casos de prueba. Podemos concluir que en MBT el objetivo es que el ingeniero de pruebas se dedique sólo a describir el comportamiento que se espera de la aplicación y que la generación de los casos de prueba abstractos y posteriormente la transformación en ejecutables se pueda hacer de forma automática. Así se reduce el tiempo de desarrollo de un sistema ya que se pueden detectar fallos en los requerimientos desde fases más tempranas del desarrollo y porque se puede volver a probar la fiabilidad del sistema más rápidamente ante cambios de dichos requisitos.

3.1.1. Qué es Model Based Testing

MBT es un método que permite probar el comportamiento de un sistema mediante la generación de casos de prueba basados en un modelo [2]. Se basa en la representación del comportamiento esperado de un sistema mediante un modelo abstracto.

MBT es la automatización del diseño de prueba de caja negra. La diferencia con una prueba de caja negra es que en lugar de escribir manualmente las pruebas basadas en la documentación de los requerimientos, en MBT creamos un modelo del comportamiento esperado del SUT, el cual captura algunos de estos requerimientos.

El término MBT se puede usar para diferentes técnicas de generación de pruebas. Existen cuatro enfoques principales según [28]:

- Generación de pruebas de datos de entrada de prueba de un modelo de dominio.
- Generación de casos de prueba de un modelo de entorno.
- Generación de casos de prueba con predicciones del comportamiento del modelo.
- Generación de scripts de prueba a partir de pruebas abstractas.

3.1.2. Cómo funciona Model Based Testing

Lo primero es construir el modelo, el cual debe tener dos características: debe ser pequeño en relación al tamaño del sistema que estamos probando cosa que no sea tan caro producirlo, pero debe ser lo suficientemente detallado para describir exactamente las características que se quieren probar.

Una vez que tenemos el modelo del sistema que queremos probar, usamos una herramienta MBT para generar automáticamente el banco de pruebas desde el modelo.

La salida del generador de casos de prueba será un conjunto de casos de prueba abstractos, cada uno de los cuales es una secuencia de operaciones con sus respectivos valores de entrada y salida esperados (el oráculo). Estos casos de prueba generados serán expresados en términos de operaciones abstractas y valores usados por el modelo.

A continuación se debe transformar los casos de prueba abstractos en scripts de pruebas ejecutables. Esto podría hacerse con una herramienta MBT, utilizando algunas plantillas y tablas de traducción proporcionadas por el ingeniero de pruebas. Estos scripts pueden ser ejecutados para tratar de detectar fallas en el SUT. La ejecución de las pruebas podría ser controlada y monitorizada por una herramienta para ejecución de pruebas.

El funcionamiento de MBT se resume en la Figura 1 obtenida de [28] donde plantea dividir el proceso en cinco pasos principales (para el generador de casos de prueba y el script de pruebas se usan herramientas de pruebas especiales, por tal motivo están resaltados en negrita):

1.- Model: Modelar el SUT y/o su entorno.
2.- Generate: Generar las pruebas abstractas a partir del modelo.
3.- Concretize: Concretizar las pruebas abstractas para hacerlas ejecutables.
4.- Execute: Ejecutar las pruebas sobre el SUT y asignar veredictos.
5.- Analyse: Analizar los resultados de la prueba.

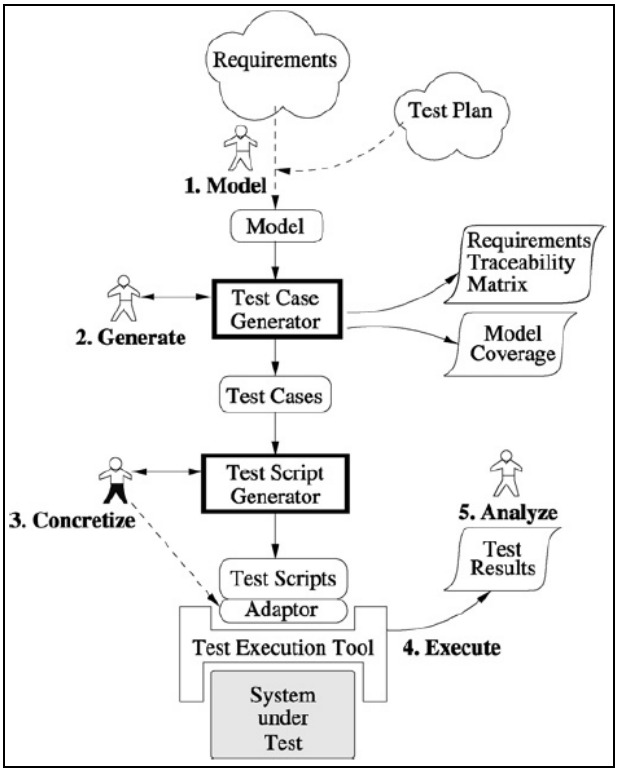

Figura 1: El Proceso de Model Based Testing

En [3] propone una figura para apreciar en detalle las actividades a realizar en el proceso, la misma ha sido enriquecida en la figura 2 con detalle de [28] que no se encontraba.

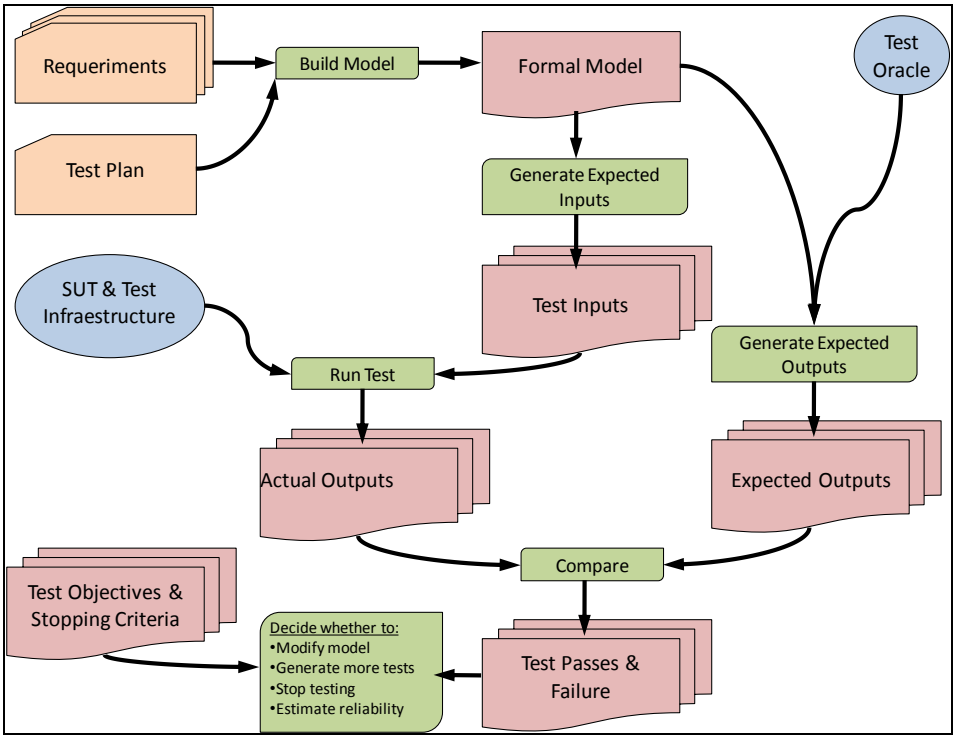

Figura 2: Actividades del Proceso de Model Based Testing

Los pasos principales que plantea son similares a los planteados en [28], con la única diferencia que los pasos tres y cuatro se unen y añaden las actividades para generación de entradas y salida esperadas. Las actividades se realizan en el siguiente orden y forma:

3.1.2.1 Construcción del Modelo

A partir de los requerimientos y el plan de pruebas construimos un modelo del comportamiento del SUT y/o de su entorno. Este es el paso más importante del proceso MBT, debido a que las siguientes actividades dependen siempre del modelo. A más completo sea el modelo serán más efectivos los casos de pruebas generados a partir de este.

Para realizar el modelo del comportamiento de la aplicación se requiere conocer exactamente el comportamiento esperado de la aplicación, y por ello se debe, como paso previo al modelado, entender el sistema y su entorno. Por ello muchas veces, el análisis exhaustivo por tratar de plasmar el comportamiento de la aplicación lleva a detectar ambigüedades y lagunas en los requerimientos de forma temprana. Existen especificaciones formales para los requerimientos como UML, diagramas ER, SDL, JML ayudan a que el modelado sea una tarea algo más sencillo de realizar que si se realizara a partir de requerimientos escritos en lenguaje natural. Para modelar el comportamiento existen diversas técnicas como FSM, Cadena de Markov, Tablas de Decisiones, Diagramas de Estados, entre otros.

Construir el modelo no es una tarea sencilla, ya que necesita habilidades, experiencia e imaginación de parte de los probadores para construir un buen modelo que se comporte de la forma deseada. En [65] se presenta las propiedades que debería tener un buen modelo, las cuales son:

- Debe ser tan abstracto como sea posible sin perder la información importante.
- No debe contener información que sea redundante o no relacionada al propósito de la prueba.

- Debe ser entendible para todos los probadores.
- Debe ser sencillo añadir, eliminar y modificar información sobre él.

Existen diferentes notaciones para la representación del modelo, dependiendo del propósito del modelo es más o menos conveniente usar una determinada notación. Existe un procedimiento general que se puede tener en cuenta en el momento de la construcción del modelo cuando se usan notaciones FSM o cadenas de Markov; las acciones a seguir son:

- Listar todas las entradas.
- Por cada entrada se debe listar las situaciones en las cuales las entradas pueden ser aplicadas y las situaciones en las cuales las entradas no pueden ser aplicadas.
- Para cada entrada, se debe listar las situaciones en las cuales las entradas provocan comportamientos o salidas diferentes, dependiendo del contexto en el cual la entrada es aplicada.

3.1.2.2 Generación de los Casos de Prueba

Una vez que el modelo se encuentra construido, es factible crear a partir de este los casos de prueba. Antes de crear los casos de prueba, los probadores normalmente también deben especificar o proveer información sobre el criterio o el propósito de los casos de prueba, las entradas y en algunos casos información de las salidas esperadas del sistema.

Las dificultades que podrían presentarse en la generación de casos de prueba dependen del modelo. Será más efectivo tener un modelo completo y preciso para generar automáticamente los casos de prueba. Cuando se trata de sistemas complejos es difícil e incluso a veces imposible generar los casos de prueba de forma manual, por lo cual la automatización es la solución para estos casos, pero siempre se debe tener en cuenta que el modelo debe ser lo más completo y preciso para hacer posible la generación automática de los casos de prueba.

Cuando el SUT es complejo normalmente significa que la cantidad de casos de prueba se puede tornar muy grande e incluso infinita. Sin embargo, para mejorar la calidad del sistema, se necesita seleccionar los casos de prueba que realmente pueden ayudar a los probadores a encontrar tantas fallas como sea posible en el sistema, y por supuesto, con un costo aceptable (recordar que a más casos de prueba el costo aumenta). Existen diferentes criterios de cobertura que permiten ayudar a controlar el proceso de generación de casos de prueba, sin embargo seleccionar uno de estos criterios no implica que no se pueda seleccionar otro, al contrario, se recomienda seleccionar más de un criterio para lograr un mejor resultado en la cobertura. En la sección 3.1.7.2 se lista y se presenta brevemente en qué consiste cada uno de estos criterios de cobertura para la generación de casos de prueba.

Generar las salidas esperadas correspondientes a unas determinadas entradas es más complejo que crear las entradas. Para ello se necesita un oráculo que cree salidas válidas y las mapee a casos de prueba. Crear el oráculo es la tarea más compleja.

3.1.2.3 Ejecución de los Casos de Prueba

Para ejecutar los casos de prueba, se necesita trasladarlos en una forma ejecutable. Debido a que los casos de prueba ejecutables pueden ser utilizados durante todo el tiempo de prueba, ellos deben ser escritos de una forma más eficiente que permita que sean reutilizables e incluso que frente a una modificación sean fáciles de generarlos y ejecutarlos. Por ello, se crean script para los casos de prueba.

La traslación del caso de prueba generado a un script será más eficiente si es hecha de forma automática. Después de aplicar los casos de prueba ejecutables (scripts) al SUT se producen las salidas actuales del sistema (estas son las salidas que deben ser contrastadas con las salidas esperadas).

3.1.2.4 Análisis de los Resultados

Tras ejecutar los casos de prueba se obtienen las salidas actuales del sistema, se debe evaluar y analizar los resultados. Para ello, se debe comparar las salidas actuales del SUT con las salidas esperadas especificadas en los casos de prueba. De forma ideal, esta comparación debe ser de forma automática. La evaluación y análisis de los resultados puede dar como resultado: pasa, falla, o inconcluso. Se considera que una prueba pasa cuando las salidas actuales se encuentran acordes a las salidas esperadas (en consecuencia, si las salidas esperadas se encuentran mal determinadas, el veredicto será erróneo, por ellos es necesario poner toda la atención posible en los pasos anteriores). Se considera que falla cuando las salidas actuales no se encuentran acordes a las esperadas. En caso no se tome una decisión en el momento, la prueba es inconclusa.

Tras obtener el veredicto de la prueba, el probador debe determinar si decide generar más casos de prueba, si se debe modificar el modelo, o si la prueba debe ser detenida y debe ponerse en marcha el software del sistema que se está probando. Debido a que no es sencillo encontrar una forma automática para determinar las salidas esperadas, especialmente es sistemas complejos, comprobar la conformidad resulta difícil en MBT. El nivel de dificultad para encontrar un oráculo de pruebas depende del tamaño y complejidad del sistema. Sin embargo, la necesidad e importancia del oráculo de pruebas también aumenta cuando el sistema es más complejo.

3.1.3. Consideraciones para la aplicación o desarrollo de MBT

En diferentes artículos se describen aspectos importantes a considerar durante el desarrollo o aplicación de MBT. En [34] condensa las observaciones de 74 artículos en siete puntos:

1.- El modelo que describe el comportamiento del software es el punto clave.

El comportamiento del modelo debe ser desarrollado cuidadosamente. Sus limitaciones y restricciones deben ser respetadas durante el modelado del SUT. Si el modelo está mal, las pruebas generadas también lo estarán, por lo tanto, la exactitud del modelo es fundamental para comenzar el proceso de generación de casos de prueba.

2.- Las pruebas deben cubrir los criterios de control de flujos y datos.

Los criterios de cobertura de las pruebas definen un conjunto de casos de pruebas seleccionados para la misma. Los casos de prueba pueden ser generados a partir de dos fuentes: criterios de flujo de control o de flujo de datos. En el caso de pruebas aplicadas a grandes sistemas es importante combinar ambos criterios para incrementar la cobertura de la prueba. Normalmente, una estrategia de ejecución de prueba está compuesta por procedimientos para la prueba (describe la secuencia que debe ser seguida desde el flujo de control) y casos de prueba (describe los datos de entrada que ingresarán desde un flujo de datos).

3.- El nivel de automatización debe ser alto.

El nivel de automatización es un factor importante para definir la factibilidad de uso de MBT. Usualmente un enfoque MBT tiene solamente un paso no automatizado, este es el modelado del comportamiento del software utilizado como entrada para el proceso de generación de los casos de prueba. Si hubiera otros pasos no automatizados aparte de este, se incrementaría el esfuerzo, costo y la complejidad de su uso.

4.- Es fundamental las herramientas que soportan la automatización de los pasos.

Las herramientas deben soportar la ejecución de los pasos automatizados. Sin embargo, si no hay una herramienta que soporte un determinado paso, esta tarea podría ser difícil e inviable. Una herramienta debe soportar la integración entre pasos no automatizados (normalmente el único suele ser el de modelado) y automatizados.

5.- Los stakeholders deben entender las habilidades requeridas y la complejidad para usar MBT.

Antes, las personas encargadas de las pruebas necesitaban sólo aprender las técnicas de pruebas para ejecutar las mismas. Actualmente, necesitan conocer técnicas de pruebas, lenguajes de modelado y algunos lenguajes de programación. Cada enfoque MBT requiere la experiencia de los probadores en la notación usada para el modelado del software, y algunas veces de experiencia en criterios de prueba, métricas, o lenguajes para generar los scripts de prueba.

6.- El orden de los pasos a seguir mientras se usa un enfoque MBT.

Cada enfoque define un conjunto de pasos para el proceso de generación de los casos de prueba. Es necesario conocer estos pasos, sus pre-requisitos, sus post-condiciones, y restricciones. Los pasos deben ser ejecutados en la misma secuencia como fueron definidos originalmente por el autor, de otra forma, se podrían generar salidas erróneas.

7.- Transferencia de la tecnología.

Las tecnologías MBT que se transfieren de entornos académicos para ser implementados en proyectos industriales, necesitan ser analizadas cuidadosamente antes de ser ejecutadas. También se debe prevenir el riesgo e impacto que implica esta transferencia. Los investigadores deben soportar la transferencia mediante estudios experimentales y controlados para adaptar el enfoque a las características del entorno industrial.

3.1.4. Beneficios de Model Based Testing

Diversos trabajos [**28**, **30**, **31, 32**] listan los beneficios que tiene elegir MBT para la fase de prueba. A continuación listamos algunos de los beneficios de MBT:

Detección de fallos del SUT

El modelado del comportamiento permite detectar ambigüedades y lagunas que existen en la especificación y el diseño del software.

Reducción del costo y tiempo

MBT contribuye a que disminuya el tiempo y esfuerzo de la fase de prueba. Si bien el tiempo empleado para escribir y mantener el modelo es mayor, el tiempo en la generación de las pruebas es menor que el costo de diseño y mantenimiento de un banco de pruebas.

Mejora de la calidad de las prueba

La calidad de las pruebas cuando el diseño de las mismas se hace manualmente, es muy dependiente de los desarrolladores de las mismas, además, el proceso de diseño no puede ser reproducido, el criterio racional aplicado para cada prueba normalmente no es recordado, y el resultado de los casos de prueba no es fácilmente relacionado a los requerimientos originales del sistema. MBT es capaz de mejorar mucho de estos inconvenientes de las pruebas manuales gracias al generador automatizado de casos de prueba, y debido a que los datos de entrada y los oráculos son generados a partir del modelo, el costo de generar más scripts de prueba ejecutables es sólo el tiempo de máquina requerido para generarlos.

Detección de los defectos y lagunas en los requerimientos

Los requerimientos son plasmados en amplios documentos basados en lenguaje natural que pueden contener omisiones, contraindicaciones, y requerimientos poco claros. Pero en MBT el primer paso es el modelado del comportamiento del SUT. El modelo tiene una semántica muy precisa, de esta forma, puede ser analizada por una máquina. Una vez que se comienza a elaborar el modelo, la precisión requerida aumenta la cantidad de preguntas acerca de los requerimientos. Entonces, la fase de modelo expone diversas cuestiones sobre los requisitos.

Trazabilidad

La trazabilidad se entiende como la habilidad para relacionar cada caso de prueba al modelo, al criterio de selección, e incluso a los requerimientos informales del sistema. Esto ayuda a explicar los casos de prueba así como justificar el porqué fueron creados. MBT provee trazabilidad entre el modelo y los casos de prueba generados. Los algoritmos de generación de pruebas deben saber qué partes del modelo serán empleadas en cada caso de prueba y salidas.

Evolución de los requerimientos

Si los requerimientos del sistema cambian, en MBT basta con actualizar el modelo y las pruebas serán generadas. Debido a que el modelo es usualmente más pequeño que el banco de pruebas, toma menos tiempo actualizar el modelo que actualizar todos los casos. Esto se resume en respuesta más rápida a cambio en los requerimientos.

Reusabilidad del modelo

El modelo incorpora la información sobre el comportamiento del software. Este puede ser reutilizado en futuras pruebas, incluso cuando las especificaciones cambien.

3.1.5. Problemas y Limitaciones de Model Based Testing

Si bien MBT proporciona diversos beneficios, también es cierto que se pueden presentar algunos problemas o inconvenientes durante su uso. Además, MBT tiene algunas limitaciones. A continuación se presenta los problemas y limitaciones del uso de MBT.

Problemas

- Es necesario que los probadores en MBT tengan ciertas habilidades, por ejemplo, deberían estar familiarizados con la notación usada en el modelo. Y dependiendo de la notación podría ser necesario que estén familiarizados con matemática lógica y discreta. Por otra parte, deberían tener experiencia con herramientas, scripts, y lenguajes de programación para que sean capaces de ejecutar las tareas de MBT.

- Cuando se está frente a un sistema complejo, el número de estados, reglas gramaticales, o combinaciones variable-valor es más amplio. Adicionalmente, existe el riesgo de una explosión de la cantidad de casos de prueba generados a partir del modelo.

- Es compleja la automatización de la generación del oráculo de pruebas a partir del modelo, especialmente en sistemas complejos. Cuando el tamaño del sistema aumenta, la necesidad de automatización también se incrementa.

- Existen diversas notaciones de modelado del sistema, pero cada una es para un / unos determinados tipos de aplicación. Es decir, es necesario desarrollar un modelo de software específico para un dominio de aplicación específico.

- Aparece la necesidad de adoptar un lenguaje formal para representar los requerimientos formalmente, ya que el modelo se obtiene a partir de ellos, y si estos están en lenguaje natural suelen presentar ambigüedades que dificultan la exactitud en el modelado. Cabe resaltar que la mayoría de empresas usan el lenguaje natural.

Limitaciones

- No es posible modelar todos los aspectos del comportamiento del SUT, por lo tanto es necesario el refinamiento o la aparición de notaciones de modelado que cubran dichos aspectos. [57]

- Los proyectos de software evolucionan, y estos nuevos requerimientos quedan fuera de fecha. Si se ha comenzado con el modelado a partir de los requerimientos antiguos, se construirá un modelo erróneo y se encontrar gran cantidad de errores en el SUT.

- Algunas partes del SUT podrían ser difíciles de modelar, y es posible que sea más rápido probarlas usando pruebas diseñadas manualmente. El riesgo que existe, es que se necesita experiencia para saber qué aspectos del SUT deben ser modelados, y cuáles deben ser probados manualmente.

- Cuando falla una prueba generada, se debe decidir si la falla es causada por el SUT, por el adaptador de código, o por un error en el modelo. El inconveniente es que la secuencia de pruebas generadas en MBT podría ser más compleja y menos intuitiva que las diseñadas manualmente, por lo tanto, podría ser más difícil determinar la causa.

- En muchas empresas, los probadores tienen menos conocimientos técnicos que los desarrolladores. Además, los probadores no se encuentran involucrados en el desarrollo del sistema hasta que el sistema ha sido diseñado y codificado. En consecuencia, los probadores sólo tienen un corto tiempo para encontrar bugs en los casos de prueba, lo cual significa que no tienen suficiente tiempo para construir modelos y generar casos de prueba. Sin imaginar que esto último podría resultar más rentable. [43]

- Cuando las pruebas son diseñadas manualmente, se suele usar la cantidad de casos de prueba diseñados como medida para conocer el progreso de las pruebas. Pero en MBT es fácil generar una gran cantidad de pruebas, lo que genera que esta métrica deje de ser útil. Aparece la necesidad de usar otras métricas más efectivas como la cobertura del código del SUT, de los requerimientos y del modelo.

3.1.6. Cuando elegir Model Based Testing

Luego de conocer los beneficios, problemas y limitaciones que tiene MBT, se puede definir algunos puntos a tener en cuenta si se piensa usar MBT. Estos puntos se encuentran relacionados con aspectos del proyecto y los probadores:

- Debe existir documentación lo suficientemente buena para poder llevar a cabo el modelado del comportamiento del SUT.

- Si ya existiera un modelo, mucho mejor.

- Los probadores deben tener un buen conocimiento del sistema para ser capaces de implementar y mantener un entorno MBT.

- Los probadores deben comprender cómo se espera que reaccione el software. Para ello deben comprender el entorno del software.

- Los probadores deben estar familiarizados con lenguajes formales de modelado, teoría de autómatas, teoría de gráficos, estadística elemental, e incluso en algunos casos con matemática lógica y discreta.

- Los probadores deben conocer bien el enfoque MBT que se vaya a aplicar. Si cuenta con experiencia en el mismo, mucho mejor.

- El manager del proyecto debe ser consciente que la fase de pruebas con MBT comienza desde la toma de requerimientos ya que se debe modelar el comportamiento del SUT a partir de ellos, por lo tanto no se puede esperar que la fase de pruebas comience después del desarrollo. Hacerlo tras el desarrollo podría significar modificar los requerimientos y en consecuencia el desarrollo.

3.1.7. Taxonomía de Model Based Testing

La taxonomía que se presenta se basa en la planteada por Utting en el 2006 [46], la cual desde entonces es citada por diversos trabajos [32, 40, 54]. La taxonomía identifica tres clases generales que se relacionan con los procesos principales de MBT: Modelo, Generación de Prueba y Ejecución de la Prueba. Estas clases se subdividen en categorías, y éstas a su vez en opciones, las cuales pueden ser mutuamente exclusivas o complementarias. Adicionalmente, existe una tesis [40] que añade a la clasificación original una categoría llamada "Evaluación de las Pruebas" enriqueciendo de esta forma la taxonomía planteada por Utting. En la Figura 3 se muestra la taxonomía.

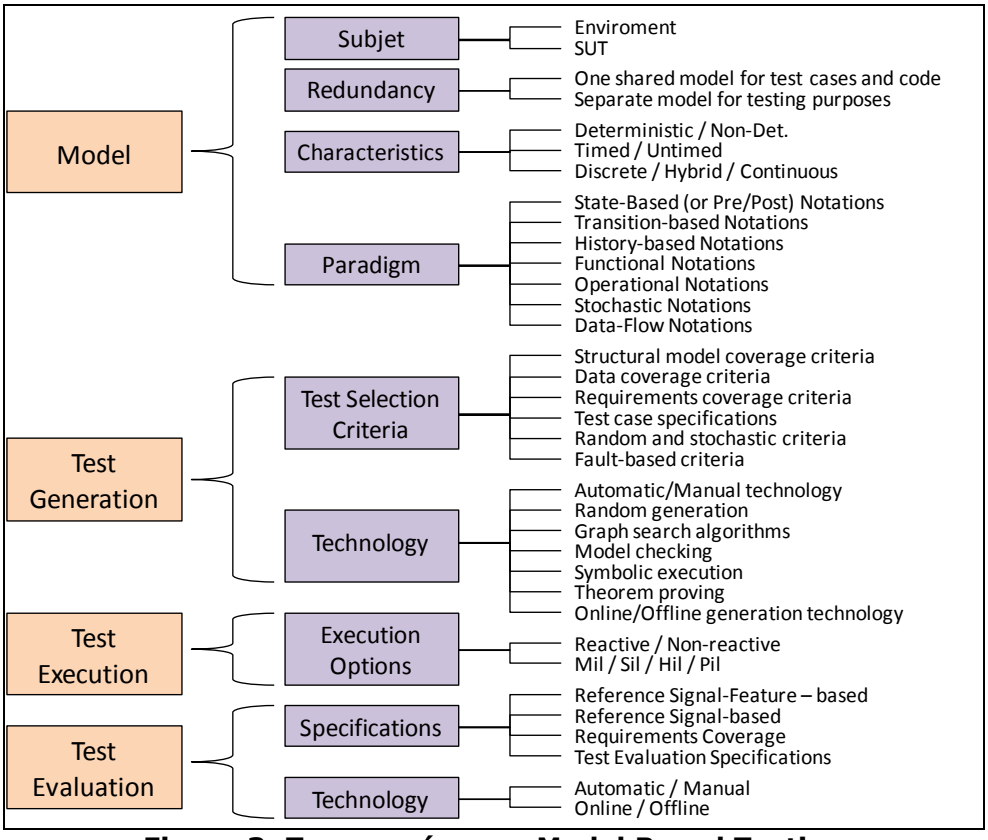

Figura 3: Taxonomía para Model Based Testing

A continuación se presenta un breve detalle de cada una de las categorías y opciones.

3.1.7.1 Modelo

Se distinguen cuatro categorías a tener en cuenta:

Subject

Denota el comportamiento del SUT o el posible comportamiento del entorno del SUT. A menudo ambos serán utilizados.

- **Enviroment**: El modelo del entorno se utiliza para restringir las posibles entradas al modelo, de esta forma, restringe un conjunto de posibles comportamientos del modelo del SUT, y en este sentido actúa como un criterio de selección de pruebas.

- **SUT**: El modelo del SUT tiene dos propósitos, el primero es actuar como un oráculo para el SUT en el cual se codifica el comportamiento previsto, y el segundo es que su estructura puede ser aprovechada para la generación de los casos de prueba.

Redundancy

MBT puede ser aplicado en diferentes escenarios. A grandes rasgos, estos se diferencian en el nivel de redundancia que hay entre los modelos para probar y / o para implementar.

- **One shared model for test cases and code**: Un mismo modelo es usado tanto para la generación de los casos de prueba como para la generación del código.

- **Separate model for testing purposes**: El modelo del SUT es construido manualmente basado en una especificación informal que ya existe. Por otro lado, un modelo de pruebas es derivado manualmente de la especificación y usado para la generación de las pruebas. Debido a que los casos de pruebas y el código no son generados desde el mismo documento formal, se genera la redundancia.

Characteristics

Se refiere a tres aspectos como son el no determinismo, información del tiempo y a la naturaleza continua o discreta de los eventos del modelo. Estas características afectan la notación usada para describir el modelo, el criterio de selección de pruebas, y cómo los casos de prueba son generados y ejecutados.

- **Deterministic / Non-Det.**: Los modelos no deterministas son la mayoría de los modelos de sistemas reactivos. En este caso el modelo presenta variabilidad en el dominio de tiempo o valor, y esto puede ser manejado cuando el veredicto se construye.

- **Timed / Untimed**: EL tiempo es una característica importante en la gran mayoría de sistemas en tiempo real. Debido al grado de libertad que tienen estos sistemas, son difíciles de probar. Esta característica continúa en investigación.

- **Discrete / Hybrid / Continuous**: Se refiere a la dinámica del modelo. La mayoría de MBT se enfocan en sistemas de eventos discretos, sin embargo los modelos continuos e híbridos son comunes en muchos sistemas embebidos. Esta característica continúa en investigación.

La distinción entre las diferentes características es importante porque impacta en la selección del paradigma de modelado, en la tecnología para la generación de los casos de prueba, y el entrelazado de la generación y ejecución de las pruebas.

Paradigm

Se determina el paradigma y la notación que serán usados para describir el modelo. Existen diferentes notaciones de modelado que han sido usados para modelar el comportamiento de los sistemas para la generación de pruebas. Las notaciones indicadas en la taxonomía son:

- **State-Based (or Pre/Post) Notations**: Modelan un sistema como una colección de variables, las cuales representan una foto del estado interno del sistema, además también modelan algunas operaciones que afectan a estas variables. En vez de definir operaciones usando código como con un lenguaje de programación, cada operación es usualmente definida mediante una pre-condición y una post-condición. Algunas de estas notaciones son Z, B, VDM, y JML.

- **Transition-based Notations**: Se centra en la descripción de las transiciones entre diferentes estados de un sistema. Normalmente, son notaciones gráficas de nodos y arcos, como las máquinas de estados finitos (FSMs), donde los nodos de los FSM representan los principales estados del sistema y los arcos representan las acciones u operaciones del sistema. Algunas de estas notaciones son FSMs en sí mismo, gráfica de estados (UML State Machines, gráfica de estados Statemate, y diagramas Simulink Stateflow), sistemas de transición etiquetadas y autómatas de entrada y salida.

- **History-based Notations**: Modelan un sistema describiendo los trazos permitidos de su comportamiento en un tiempo determinado. Variadas notaciones de tiempo pueden ser usada (discretas o continuas, lineares o ramificadas, puntos o intervalos, y otras), conduciendo a diferentes tipos de lógicas temporales. También se incluyen diagramas de secuencia de mensajes y formalismos relacionados, que son básicamente notaciones gráficas o textuales para especificar una secuencia de interacciones entre componentes.

- **Functional Notations**: Describen un sistema como un conjunto de funciones matemáticas. Las funciones podrían ser de primer orden solamente, como en el caso de especificaciones algebraicas, o de un orden más alto, como en el caso de las notaciones como HOL. Las especificaciones algebraicas tienden a ser más abstractas y más difíciles de escribir que otras notaciones, este es el motivo por el cual no son ampliamente usadas en MBT.

- **Operational Notations**: Describen un sistema como un conjunto de procesos ejecutables, ejecutándose en paralelo. Ellos son particularmente adecuados para describir sistemas distribuidos y protocolos de comunicación. Como ejemplos se incluye álgebra de procesos como CSP o CSS, y como notación la red de Petri.

- **Stochastic Notations**: Describen un sistema mediante un modelo probabilístico de eventos y valores de entrada, y tiende a ser usado para modelar el entorno del SUT. Como ejemplo de esta notación tenemos la cadena de Markov.

- **Data-Flow Notations**: Se concentran en describir el flujo de datos. Algunos ejemplos de este tipo de notaciones son Lustre y los diagramas de bloque tal como se utiliza, por ejemplo, en Matlab Simulink.

3.1.7.2 Generación de la Prueba

El proceso de generación de pruebas comienza en la toma de requerimientos del sistema tomando en cuenta los objetivos de la prueba. Es definido en un contexto de prueba determinado y lidera la creación de los casos pruebas. Un número de enfoques existe dependiendo del criterio de selección de la prueba y tecnología para la generación. Ellos son revisados abajo.

Criterio de Selección de la Prueba

Define las facilidades que son usadas para controlar la generación de las pruebas. Ellas ayudan a especificar las pruebas y no depender del código del SUT. Con el fin de lograr una mejor cobertura de la prueba se debería combinar los diferentes métodos a fin que estos se complementen. Por lo tanto no existe un criterio más adecuado para generar la especificación de la prueba. Las herramientas MBT pueden ser clasificadas de acuerdo al criterio de selección de prueba que ellas soportan. Estos criterios sin usados para controlar el proceso de generación de los casos de prueba. Las opciones que existen para la generación de las pruebas que plantea la taxonomía son las siguientes:

- **Structural model coverage criteria:** Explota la estructura del modelo como los nodos y arcos de los modelos basados en transiciones, o sentencias condicionales en un modelo de notaciones pre-post.

- **Data coverage criteria**: Plantea dividir el rango de datos en clases equivalentes y seleccionar un representante de cada clase, pero se espera que los elementos de una determinada clase sean realmente equivalentes en términos de su capacidad para detectar fallas.

- **Requirements coverage criteria**: Cuando los elementos del modelo pueden ser asociados explícitamente con requerimientos informales del SUT, entonces la cobertura del modelo puede ser aplicada también a esos requerimientos. Por ejemplo, unos requerimientos pueden ser plasmados en transiciones de una máquina de estado UML o sentencias dentro de post condiciones de un modelo pre-post.

- **Test case specifications**: Si se realiza una especificación de casos de prueba bajo una notación formal, ésta se puede utilizar para determinar las pruebas que se generarán. La notación usada para expresar estos objetivos podría ser la misma que la notación usada para el modelo. Algunas de las notaciones usadas para estos objetivos de pruebas son FSMs, UML Testing Profile (UTP), expresiones regulares, fórmulas de lógica temporal, restricciones, y cadenas de Markov.

- **Random and stochastic criteria**: Son mayormente aplicados para modelar el entorno, debido a que es el entorno quien determina los patrones usados del SUT. Las probabilidades de acciones se modelan directa o indirectamente.

- **Fault-based criteria**: Son aplicados mayormente al modelado del SUT, debido a que el objetivo de estas pruebas es encontrar las fallas que hay en el SUT. Uno de los criterios más comunes es la cobertura de la mutación. Este involucra la mutación del modelo, luego generar pruebas que distinguirían entre el modelo mutado y el original.

Tecnología para la Generación de Pruebas

Una de las características más atractivas de MBT más atractivas es el potencial que tiene para la automatización. La generación automática de casos de prueba por lo general requiere la existencia de los tipos de especificaciones de casos de prueba. Las opciones que existen para las tecnologías de generación de las pruebas que plantea la taxonomía son las siguientes:

- **Automatic/Manual technology**: La generación automática de las pruebas consiste en que los casos de prueba sean generados automáticamente desde la fuente de información basada en un criterio dado. La generación manual de la pruebas consiste en que los casos de prueba son producidos de forma manual.

- **Random generation**: Es realizada por muestreo del espacio de entrada de un sistema. Es fácil de implementar, pero necesita mucho tiempo para llegar a obtener un cierto nivel de satisfacción de la cobertura del modelo.

- **Graph search algorithms**: Se basa en un algoritmo de búsqueda sobre una gráfica. Incluye algoritmos de cobertura de nodos o arcos. En [30] se muestra la teoría básica para este tipo de algoritmos de búsqueda, y se brindan ejemplos como el problema de puentes en la ciudad de Prussian en 1976, o el problema del repartidor de cartas en China en 1962.

- **Model checking**: Es una tecnología para la verificación o falsificación de propiedades de un sistema. Primero formula las especificaciones de casos de prueba como propiedades que pueden llegar a pasar. Una propiedad normalmente expresa una situación no deseada. El verificador de modelo constata si la situación puede ser alcanzada.

- **Symbolic execution**: Ejecuta un modelo ejecutable con un conjunto de valores de entrada y no con un único valor. Estos son representados como restricciones. De esta forma se generan trazos simbólicos. La instanciación con los valores concretos debe ser ejecutada para obtener los casos de prueba del SUT.

- **Theorem proving**: Generalmente son usados para verificar la satisfactibilidad de las fórmulas que ocurren directamente en los modelos.

- **Online/Offline generation technology**: Con la generación de pruebas online, los algoritmos pueden reaccionar a las salidas reales del SUT durante la ejecución de la prueba. Esta misma idea es usada para la aplicación de las pruebas reactivas. Las pruebas offline generan los casos de prueba antes que se ejecuten. En este último caso un conjunto de casos de prueba es generado una sola vez y puede ser ejecutado muchas veces; además la generación y ejecución de la prueba puede ser ejecutada en diferentes máquinas, diferentes niveles de abstracción, o incluso en diferentes entornos. Finalmente si el proceso de generación de pruebas es más lento que la ejecución de las pruebas, entonces hay una ventaja evidente para hacer la fase de generación de pruebas sólo una vez.

3.1.7.3 Ejecución de la Prueba

Opciones para la Ejecución

Tenemos las siguientes opciones para la ejecución de la prueba.

- **Reactive / Non-reactive**: Las pruebas reactivas son pruebas que toman cualquier señal o dato derivado de las salidas del SUT o del sistema de prueba mismo, para influenciar la señal de alimentación del SUT. En consecuencia, la ejecución de los casos de prueba reactivos varían dependiendo del comportamiento del SUT, a diferencia de la ejecución de pruebas no reactivas, donde el SUT no influencia la prueba.

- **Mil / Sil / Hil / Pil**: La ejecución de las pruebas es gestionada mediante las llamadas plataformas de pruebas. El propósito de estas plataformas es estimular el objeto de prueba (por ejemplo el SUT) con entradas, y observar y analizar las salidas del SUT. La plataforma de prueba la podemos asemejar a un coche con un conductor. El conductor es el que determina las entradas del SUT a través de los escenarios de prueba y observa la reacción del coche. Los Model-in-the-Loop (Mil) están basados en el modelo del sistema en sí mismo. Durante los Software-in-the-Loop (Sil)el SUT es el software probado en un circuito abierto o cerrado. El Processor-in-the-Loop (Pil) es similar a Sil, pero el software embebido se ejecuta en un target board. Cuando probamos los sistemas embebidos sobre el nivel Hardware-in-the-Loop (Hil) el software se ejecuta al final del ECU (Unidad de control electrónico).

3.1.7.4 Evaluación de Prueba

Explota la prueba del oráculo. Este mecanismo analiza la salida del SUT y decide sobre el resultado. Los resultados del SUT son comparados con los esperados y se diseña un veredicto en base a ello. Un oráculo podría ser un sistema existente, una especificación de prueba, o un conocimiento especializado de un individuo en particular.

Especificación

La especificación de los algoritmos de evaluación de pruebas podría ser basada en diferentes fundamentos que cubran el mismo criterio. Normalmente forman un tipo de modelo o un conjunto ordenado de señales o datos de referencia para especificar escenarios. Las opciones que plantea la taxonomía son las siguientes:

- **Reference Signal-Feature – based**: Se basa en las pruebas de señales de referencia del comportamiento del SUT comparando los resultados del SUT con las referencias previamente especificadas.

- **Reference Signal-based**: Se basa en las pruebas de las características de las señales de referencias del comportamiento del SUT comparando los resultados del SUT particionados en características con los valores de referencias especificados previamente para esas características.

- **Requirements Coverage**: Su objetivo es cubrir todos los requerimientos informales del SUT, pero con respecto al comportamiento del SUT esperado especificado en los mismos.

- **Test Evaluation Specifications**: Se refiere a la especificación de las salidas esperadas del SUT luego de la ejecución de los casos de prueba. Cuando se definen los escenarios de prueba en alguna notación formal, estos pueden ser usados para determinar cómo, cuándo y qué pruebas serán evaluadas.

Tecnología

La tecnología de la especificación de las pruebas permite un proceso automático o manual, mientras que la ejecución de la evaluación de la prueba ocurre de forma online u offline.

- **Automatic / Manual**: Puede ser entendida en dos aspectos, desde la perspectiva de la definición de la evaluación de la prueba, o desde su ejecución. En cuanto a la perspectiva de la especificación de la evaluación de la prueba, cuando las salidas del SUT son definidas manualmente, entonces, es un proceso de especificación de prueba manual. En cuanto a la perspectiva de su ejecución, cuando las salidas esperadas del SUT son derivadas automáticamente, entonces, es automática.

- **Online / Offline**: La evaluación de pruebas online ocurre durante la ejecución del SUT. Esta evaluación de pruebas online hace posible extender el concepto de control de las pruebas y prueba de reactividad. La evaluación de pruebas offline significa todo lo contrario.

3.1.8. Herramientas para Model Based Testing

En el mercado existen diversas herramientas para MBT. Las mismas se diferencian según la notación de modelado que usan para modelar el comportamiento del SUT, algunas de ellas se ciñen a la generación de datos de entrada, otras a la generación de casos de prueba, y otras a los scripts de pruebas.

Las herramientas MBT ha sido motivo de desarrollo de diversos estudios, podríamos citar el esquema de caracterización desarrollado para herramientas MBT en 2010 en [45] que se enfoca en herramientas orientadas a modelos basados en estados, si bien tiene un alcance de estudio bastante limitado, lista características importantes que deben ser consideradas al momento de elegir una herramienta MBT, como por ejemplo:

- **Criterios de la Prueba:** Considera el criterio de flujo de modelo, de flujo de script, de datos, y de requerimientos.
- **Criterio de Soporte para las Actividades Relacionadas:** Considera la creación del modelo, la verificación del modelo, depuración de los casos de prueba, trazabilidad de los requerimientos.
- **Otros Criterios:** Considera la notación del modelo, la categoría de la herramienta (si es comercial, académica), y el lenguaje de programación.

En la Tabla 1 se muestra un listado de algunas de las herramientas que existen para MBT, muchas de ellas se encuentran referenciadas en [28, 33, 40, 45, 54].

Para el listado de la Tabla 1 se ha definido el criterio tipo, que considera el tipo de generación que realiza y la fuente para dicha generación, y clasifica las herramientas en cuatro tipos:

- **Tipo 1:** Generación de Datos de Entrada desde el Dominio del Modelo
- **Tipo 2:** Generación de Casos de Prueba desde un Modelo del Entorno
- **Tipo 3:** Generación de Casos de Prueba con Oráculos desde el Comportamiento del Modelo
- **Tipo 4:** Generación de Scripts de Pruebas desde Pruebas Abstractas

Tabla 1: Listado de Herramientas MBT

Nombre de la Herramientas	Académica / Comercial	Tipo	Notación del Modelo	Página Web
AETG	Comercial	1	Modelo de Dominio de Datos de Entrada	aetgweb.argreenhou se.com
AGEDIS	Académica	N/D	UML/AML	-
AsmL	Académica	N/D	AsmL	-
Autofocus	Académica	2	Autofocus	-
Case Maker	Comercial	1	Modelo de Dominio de Datos de Entrada	www.casemakerinter national.com
Conformance Kit	Académica	2	EFSM	-
Conformiq Test Generator	Comercial	3	Diagrama de Estados UML	www.conformiq.com
ConformiQtronic	Comercial	2	Máquina de Estados UML con bloques Java o C	-
Cooper	Académica	N/D	LTS (Basic LOTOS)	-
CTesK, JTesK	Comercial	3	Extensiones Pre-Post de Lenguajes de Programación	www.unitesk.com
GATeL	Académica	N/D	Lustre	-
GOTCHA-TCBeans	No especifica	N/D	FSM	www.haifa.ibm.com/projects/verification/gtcb/index.html
JUMBL	Académica	2	Modelos que usan Cadena de Markov	-
LTG	Comercial	2	Diagrama de Estados UML, Notación B	-
Lurette	Académica	N/D	Lustre	-
Lutess	Académica	N/D	Lustre	-
MaTeLo	Comercial	2	Cadena de Markov	www.all4tec.net
mbt	No especifica	N/D	EFSM	mbt.tigris.org
MOTES	No especifica	N/D	EFSM	www.elvior.ee
ParTeG	No especifica	N/D	Clases UML Diagramas de Máquina de Estados	parteg.sourceforge.net
Phact	Académica	N/D	EFSM	-
Qtronic	No especifica	N/D	Máquina de Estados UML y Lenguaje Scripting	www.conformiq.com
Rave	Comercial	3	Notación Tabular	www.t-vec.com
Reactis	Comercial	3	Mathlab, Simulink, Stateflow	www.reactive-systems.com
Simulink -T-Vec Tester for	Comercial	3	Simulink and MATRIXx	www.t-vec.com
SmartTest	Comercial	1	Modelo de Dominio de Datos de Entrada	www.smartwaretech nologies.com/smartt estprod.htm
Spec Explorer 2010	Comercial	2	FSM y ASM	msdn.microsoft.com/ennus/devlabs/ee692301.aspx
StatemateStatemate Automatic Test Generator / Rhapsody Automatic Test Generator (ATG)	Comercial	3	Diagrama de Estados UML	www.llogix.com
STG	Académica	N/D	NTIF	-
TAF	Comercial	2	Tabla SCR convertida a T-VEC	-
TAU Tester	Comercial	4	TTCN-3	www.telelogic.com/products/tau/tester/index.cfm
Test Cover	Comercial	1	Modelo de Dominio de Datos de Entrada	www.testcover.com
Test Designer	No especifica	N/D	Clases UML, Máquinas de Estados y Diagrama de Objetos	www.smartesting.com
TestOptimal	No especifica	N/D	FSM	testoptimal.com
TGV	Académica	N/D	LTS-API (LOTOS, SDL, UML)	-
TorX	Académica	2	LTS (LOTOS, Promela, FSP)	-
TVEDA	Académica	N/D	SDL, Estelle	-
Uppaal Cover	Académica	N/D	TA	-
Uppaal Tron	Académica	N/D	TA	-
ZigmaTEST Tools	Comercial	3	Máquina de Estado Finita	www.atssoft.com/pr oducts/testingtool.h tm

3.1.9. Casos de Éxito utilizando Model Based Testing

La primera herramienta comercial para soportar MBT fue lanzada en 1996. Sin embargo en 1995 un programa de generación de programas de pruebas para verificaciones funcionales de procesadores PowerPC en IBM ya había sido puesto en marcha logrando encontrar 1530 bugs en 3 procesadores PowerPC.

A continuación detallamos brevemente algunos casos de éxito de uso de MBT en diversas empresas:

MBT en IBM [5]

IBM reporta una reducción significativa de los costos también como una buena capacidad para encontrar fallas en dos casos de estudios de sistemas industriales donde usaban un generador MBT llamado GOTCHA-TCBeans. En ambos casos, modelos FSM fueron derivados de especificaciones informales de los requerimientos en idioma inglés. Otra herramienta que usaron fue TCtranslator para trasladar las pruebas abstractas generadas en Scripts de pruebas en lenguaje C y Java. En uno de los casos lograron reducir en un 17% el costo de MBT y en otro lograron cubrir las sentencias del SUT en un 78%.

MBT en Microsoft [52]

Microsoft ha tenido al menos tres generaciones de herramientas MBT desarrolladas y usadas dentro del mismo Microsoft. En la primera generación encontramos Test Model Toolkit (TMT). En la segunda y tercera generación encontramos ASML/T y Spec Explorer). Uno de los resultados que se obtuvo fue el aumento de la cobertura de la implementación del 60 al 70%.

MBT en la Industria de las Smart Card [6]

Las experiencias en este campo empiezan en el 2000 con varios casos de estudio y pruebas de aplicación de MBT para software de Smart Cards. Actualmente empresas como Axalto, Gemplus, Giesecke& Devrient, usan MBT en la validación de sus procesos. En esta industria podemos ver las características que hacen fácil integrar MBT en los procesos de prueba existentes y obtener un alto nivel de retorno de la inversión. Estas características son: potente validación de las necesidades, alto nivel de la automatización de la ejecución de las pruebas, y un buen nivel de madurez de desarrollo (Gemplus alcanzó un nivel 3 de CMMI).

MBT en la Industria Automotriz [4]

Pretschner y su equipo aplican MBT en un proceso de la industria automotriz. Logran detectar errores en los requerimientos y errores severos en el SUT. Logran incrementar en seis veces el tamaño de los bancos de prueba generados automáticamente y esto traer como consecuencia un incremento del 11% en los errores detectados.

3.2. Pruebas Funcionales

Para definir lo que son las pruebas funcionales primero se definirá lo que son las pruebas de software. Luego se verá las dos técnicas principales que existen para las pruebas de software ya que una de ellas está estrechamente ligada con las pruebas funcionales. A continuación se verá la clasificación de las pruebas de software y dónde se ubican las pruebas funcionales dentro de la clasificación seleccionada. También se verá dentro de la clasificación seleccionada el alcance que tiene MBT. Finalmente se definirá las pruebas funcionales dentro del contexto de MBT.

3.2.1. Qué son las Pruebas de Software

A continuación citamos algunas definiciones para pruebas de software:

"La prueba es una actividad ejecutada para evaluar la calidad del producto, y para mejorarlo, a través de la identificación de problemas y defectos." [10]

"Las pruebas de software consisten en verificaciones dinámicas del comportamiento de un programa en un conjunto finito de casos de prueba, adecuadamente seleccionado usualmente a partir de un dominio de ejecuciones infinito, contra el comportamiento esperado." [28]

Se dice que es dinámico porque se puede ejecutar un programa con valores de entrada específicos para encontrar fallas en su comportamiento. Una prueba exhaustiva no es posible para la mayoría de programas reales debido al amplio número de entradas permitidas para cada operación, entradas inválidas o no esperadas, y las posibles secuencias de operaciones usualmente es infinito. Por lo tanto se debe seleccionar un número más pequeño de pruebas de forma tal que se pueda ejecutar las pruebas en un determinado tiempo, la clave está en cómo se seleccionan las pruebas que tienen más posibilidades de exponer fallas en el sistema. Luego de ejecutar las pruebas, se debe decidir si el comportamiento esperado del sistema falla o no. Este es el llamado problema del oráculo, el cual es frecuentemente resuelto de forma manual mediante inspección de las salidas de la prueba.

3.2.2. Técnicas para las Pruebas de Software

Existen dos técnicas básicas para las pruebas de software. Estas son las pruebas de caja blanca y caja negra.

3.2.2.1 Prueba de Caja Blanca

En [11] la IEEE define la prueba de caja blanca como:

"Pruebas que tienen en cuenta los mecanismos internos de un sistema o componente."

También es llamada prueba estructural o prueba de caja de cristal [1]. Tiene en cuenta la estructura del código fuente del programa a partir del cual selecciona casos de prueba, guiándose por la existencia de sentencias de tipo condicionales, bucle, ramas, y otras, es decir, los casos de prueba son desarrollados usando criterios de cobertura para el código del programa. De esta forma trata el SUT como una caja de cristal que permite ver su estructura. Se dice que la prueba de caja blanca prueba lo que el programa hace y no lo que se supone que debería hacer.

3.2.2.2 Prueba de Caja Negra

En [11] la IEEE define la prueba de caja negra como:

"Pruebas que ignoran los mecanismos internos de un sistema o componente y se enfocan solamente en las salidas generadas como respuesta a las entradas seleccionadas y a las condiciones de ejecución."

También son llamadas pruebas funcionales. Trata el SUT como una caja negra, por lo tanto no usa información de su estructura interna (no se considera cómo fue implementado). En lugar de ello, las pruebas son diseñadas a partir de los requerimientos del sistema, los cuales describen el comportamiento esperado de la caja negra. En [28] Utting considera que MBT es la automatización de la prueba de caja negra, y que para lograr esta automatización es necesario que los modelos sean formales dado que es la única manera de permitir realizar múltiples

análisis mecánicos sobre el texto de los modelos, mientras que la prueba de caja negra tradicional, como se ha mencionado, prueba a partir del documento que contiene los requerimientos. Se puede encontrar mayor detalle sobre la técnica de caja negra en [18].

3.2.3. Clasificación de las Pruebas de Software

Las pruebas de software se pueden clasificar en diferentes niveles dependiendo de las características del SUT y del sistema de prueba.

En [51] su objetivo es probar los sistemas de comunicación y para ello clasifica las pruebas teniendo en cuenta los objetivos (estático, estructural, funcional y no funcional), el alcance (componente, integración y sistema) y la distribución.

En [47] reemplaza la clasificación de las pruebas basadas en la distribución por una clasificación basada en las fases de desarrollo de las pruebas.

En [12] considera que los sistemas se encuentran compuestos de subsistemas, módulos y subrutinas, y que por lo tanto las pruebas deben clasificarse bajo dicha estructura. Plantea cinco tipos de pruebas basados en la estructura: pruebas de unidad, de módulos, de subsistemas, de sistemas, y de aceptación. A su vez los agrupa en tres grandes grupos, donde las pruebas de unidad y módulos se encuentran dentro de las pruebas de componentes; las pruebas de subsistemas y sistema se encuentran dentro de las pruebas de integración; y las pruebas de aceptación dentro de las pruebas de usuario.

En [40] se muestra una clasificación más completa pero orientada a sistemas embebidos. Considera cinco criterios para clasificar las pruebas: el alcance de la prueba (estático, estructural, funcional, no funcional), el nivel de abstracción de la prueba (abstracto, no abstracto), la plataforma de ejecución de la prueba (Mil, Sil, Hil, Pil), la capacidad de reacción (reactivos, no reactivos), y el alcance de la prueba (componente, integración, sistema).

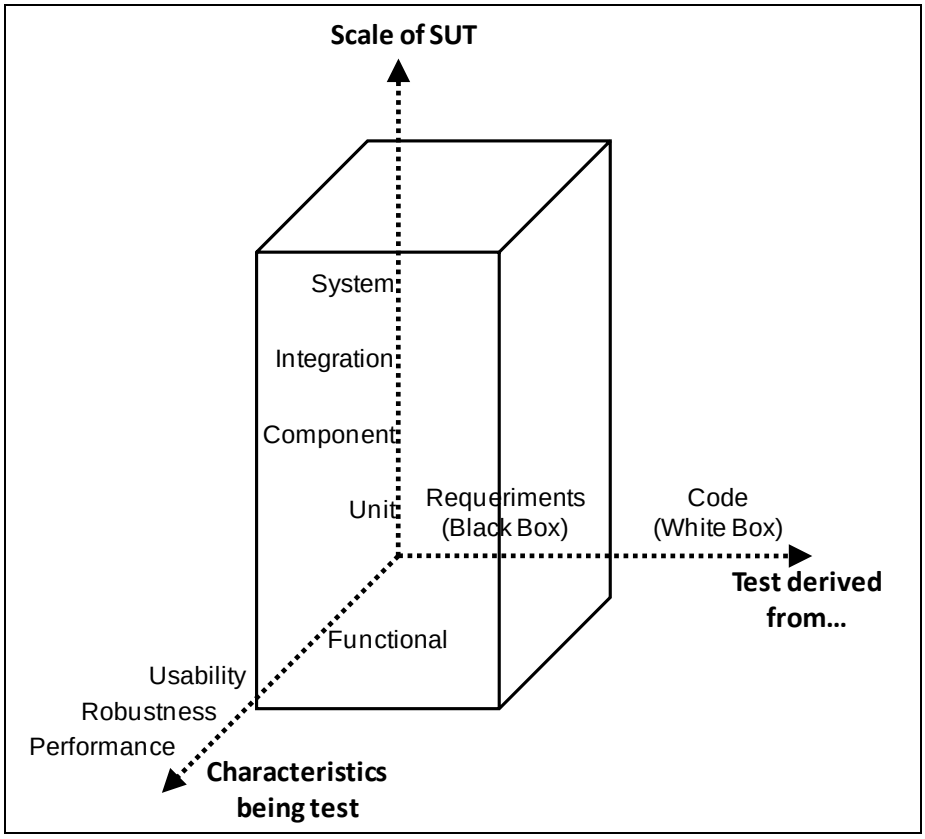

Figura 4: Clasificación de las Pruebas Funcionales y Alcance de MBT

En [29] plantea clasificar las pruebas en fases (pruebas de regresión, aceptación, de sistema, de integración, unitarias) directamente relacionadas a las fases de desarrollo de software basándose en el modelo V, y hace un refinamiento de dicho modelo con el modelo W (en este muestra un mayor detalle de las pruebas).

Podríamos seguir citando clasificaciones desarrolladas para casos particulares, pero existe una planteada dentro del contexto de MBT. En [28] muestra una clasificación adaptada al contexto de MBT basada en la clasificación de [56]. Considera tres criterios para clasificar las prueba: las características que van a ser probadas, la escala del SUT, y la fuente a partir de la cual son derivadas las prueba. Esta será la clasificación que se tome como referencia debido a que se encuentra adaptada al contexto de MBT y muestra claramente el alcance que este tiene. En la Figura 4 de [28], podemos ver la clasificación según los tres criterios mencionados y el alcance de MBT dentro de ellos.

Según la escala del SUT encontramos:

- **Pruebas Unitarias**: Prueba una única unidad a la vez, como por ejemplo un procedimiento o una clase.

- **Pruebas de Componente**: Prueba cada componente o subsistema de forma separada.

- **Pruebas de Integración**: Su objetivo es asegurar que muchos componentes trabajan juntos de forma correcta y cumplen con la funcionalidad establecida.

- **Pruebas de Sistema**: Prueba la funcionalidad y atributos de calidad el sistema como un todo.

Según el tipo de información que se usa para diseñar, las pruebas pueden ser:

- **Prueba de Caja Negra**: Se trata el SUT como una caja negra, por lo tanto no se usa la información de su estructura interna. Los casos son diseñados a partir de los requerimientos del sistema, los cuales describen el comportamiento esperado.

- **Prueba de Caja Blanca**: Usa el código de la implementación como base para el diseño de las pruebas.

Cuando nos referimos a probar las características del sistema pensamos en probar la funcionalidad, usabilidad, eficiencia, portabilidad, mantenibilidad, robustez, desempeño o escalabilidad. En [28] se enfoca en cuatro características:

- **Pruebas Funcionales**: También llamadas como pruebas de comportamiento. El objetivo es encontrar errores en la funcionalidad del sistema.

- **Pruebas de Robustez**: Su objetivo es encontrar errores en el sistema bajo condiciones inválidas como por ejemplo entradas inesperadas, falta de disponibilidad de las aplicaciones de las que depende, y fallas en el hardware y la red.

- **Pruebas de Rendimiento**: Prueban el rendimiento del sistema bajo una carga pesada de procesamiento.

- **Pruebas de Usabilidad**: Se centran en encontrar problemas de la interfaz de usuario que podrían hacer que el software sea difícil de utilizar o podrían causar que los usuarios mal interpreten las salidas.

MBT puede ser aplicado a cualquiera de los niveles de la escala del SUT. La técnica de prueba de caja negra es comúnmente la usada para el diseño de las pruebas funcionales y de robustez.

Se dice que MBT es una forma de la prueba de caja negra debido a que las pruebas son generadas a partir de un modelo, el cual es derivado a su vez de la documentación de los requerimientos.

Aunque MBT se usa principalmente para generar pruebas funcionales, también puede ser usado para otros tipos de características. En los últimos años se ha comenzado a investigar en características como el rendimiento [37] y la usabilidad [49].

3.2.4. Las Pruebas Funcionales

También son llamados pruebas de caja negra [18]. La especificación se hace a alto nivel y a partir de los requerimientos como se ha mencionado anteriormente. Evalúan el comportamiento funcional del SUT contra los requerimientos funcionales. Es decir, los probadores tienen en cuenta la especificación de los requerimientos de cliente y el diseño a alto nivel para plantear los casos de prueba que aseguren que el sistema hace lo que se supone que debería hacer.

MBT es la automatización de las pruebas funcionales o de caja negra según [28]. Para lograr dicha automatización es necesario que los modelos sean formales dado que es la única manera de permitir realizar múltiples análisis mecánicos sobre el texto de los modelos, mientras que la prueba de caja negra tradicional, como se ha mencionado, prueba a partir del documento que contiene los requerimientos.

3.3. Aplicaciones Web

3.3.1. Qué son las Aplicaciones Web

Las aplicaciones web son aquellas aplicaciones que se comunican con clientes a través de protocolos web como por ejemplo HTTP, WSDL o SOAP [7]. Las aplicaciones web son comúnmente definidas como un conjunto de páginas web conectadas lógicamente que son gestionadas por una única entidad que es accedida a través de un navegador Web sobre una red como Internet o Intranet [36].

Su presencia y predominio son resultados directos de la presencia de la World Wide Web para alojar sus aplicaciones. Además predominan porque presentan ventajas como:

- Ahorro de tiempo para desarrollar tareas sencillas donde en la mayoría de casos no se necesita descargar ni instalar ningún programa.
- Actualizaciones inmediatas, ya que el software lo gestiona el propio desarrollador, y cada vez que uno se conecta siempre accede a la última versión lanzada.
- Dado que toda o gran parte de la aplicación no se encuentra en el cliente, muchas de las tareas que se realiza no consumen recursos del cliente.
- Es multiplataforma, ya que se pueden usar desde cualquier sistema operativo (salvo pocas excepciones) ya que basta con tener un navegador para poder acceder a ellas.
- Tienen una alta disponibilidad en tiempo y espacio. Una aplicación web puede ser accedida desde cualquier parte del mundo donde haya conexión a Internet y un navegador web.

Pero también presentan algunos inconvenientes como:

- Ofrecen menos funcionalidades que las aplicaciones de escritorio. Esto se debe a que las funcionalidades que se pueden realizar desde un navegador web son más limitadas que las que se pueden realizar desde un sistema operativo. Sin embargo, ha medida que pasan los años los navegadores están cada vez más superando estas deficiencias. La aparición de HTML 5 es un hito importante en tal sentido.

- La disponibilidad de la aplicación depende de un tercero como el proveedor de la conexión a Internet, o del que provee el enlace entre el servidor de la aplicación y el cliente.

3.3.2. Dificultades para la Prueba de Aplicaciones Web e Impacto

El desarrollo de aplicaciones web, en esencia, no es diferente a cualquier otra aplicación de software, ya que incluso es factible utilizar las mismas metodologías de desarrollo de software en ambos casos. La diferencia entre una aplicación web y otra no web, se ve claramente durante la puesta en marcha de la aplicación; ya que si una aplicación web presenta fallos, tiene el potencial de lanzar un diluvio de quejas e insatisfacción casi inmediatamente, y lo peor, generar pérdidas inmediatas a los stakeholders.

Como consecuencia del incremento de la complejidad de las aplicaciones web surgen nuevos paradigmas como la Arquitectura orientada a servicios (SOA), la cual permite interoperabilidad y flexibilidad de las aplicaciones distribuidas. Todas las grandes compañías de tecnologías de información como IBM, Tibco, Software AG, Oracle, han realizado enormes inversiones en SOA en los últimos años, lo que representa un presupuesto global estimado en 2 billones de dólares americanos en 2007, y que se espera aumente a 9.1 billones para el 2014 [8]. Y más allá de estos números está el hecho que las empresas en prácticamente cualquier dominio como los bancos, los gobiernos, los hospitales, las academias, el ocio y las agencias de viaje, están siendo progresivamente desplazadas hacia los servicios en línea del mercado.

Esto lleva a plantearse qué se debería hacer para asegurar que la aplicación web que se va a poner en marcha funciona según los requerimientos definidos. Se podría asegurar llevar a cabo una excelente fase de análisis, de diseño, o de implementación, pero siempre la fase de prueba termina siendo la fase previa a la puesta en marcha de la aplicación, con lo que resulta siendo la última oportunidad para detectar posibles fallos que puedan terminar, en ocasiones, en terribles catástrofes y penosas pérdidas. En el caso de las aplicaciones web, no sólo se trata de pérdidas económicas, sino también de pérdida de prestigio, de fiabilidad, de credibilidad por parte de nuestros usuarios o clientes.

Realizar las pruebas en una aplicación web no es tarea sencilla, ya que muchas veces no se conoce el perfil de la persona que se encuentra frente a la aplicación, u otras veces estas interactúan con otros sistemas mediante interfaces. Básicamente se trata que alguien cuyo perfil no se conoce, interactúa con la aplicación, con lo cual, existen un sinfín de flujos posibles que se pueden llevar a cabo durante la interacción.

Es importante resaltar que ha medida que han ido pasando los años las aplicaciones web se han hecho cada vez más complejas, lo que conlleva a que las pruebas de las mismas también hayan ido incrementado en complejidad. Las aplicaciones web ya no son simples páginas HTML, sino que tienen interfaces gráficas. En 1995 las aplicaciones web eran casi en 100% interfaces (HTML), en 1998 casi un 90% de ellas lo era, pero en 2000 solamente un 50% de las aplicaciones web eran interfaces [9]. Hoy en día han aparecido los servicios web, SOA, y otras tecnologías que hacen más compleja la fase de prueba. También hay que tener en cuenta que una aplicación web puede ser construida con componentes escritos en diferentes lenguajes, puede estar basada en diferentes modelos de programación, y por último puede ejecutarse en diversas plataformas de hardware y software.

4. Desarrollo del Esquema de Caracterización

El presente trabajo de caracterización ha sido guiado por el procedimiento que plantea [48], donde indica que el proceso tiene tres actividades principales:

1.- Planificación. En esta actividad el estudio se planifica y delimita, y se define un protocolo específico para él. Se define clara y completamente el objetivo del estudio, así como el entorno y las fuentes para identificar los enfoques del estudio.

2.- Realización de la Revisión. En esta actividad, tras haber especificado las restricciones iniciales del estudio, el tema de estudio debe ser identificado y revisado en diferentes fuentes. Luego, se detecta el conjunto de enfoques relevantes para ser analizados. [48] plantea establecer un criterio de caracterización común para definir cada enfoque antes que el estudio comparativo se lleve a cabo. Esto permite obtener una definición uniforme de cada enfoque que podría facilitar el estudio comparativo.

3.- Reporte de la Revisión. Por último, una vez que los enfoques han sido estudiados y se ha llevado a cabo el análisis, se procede con el reporte de los resultados obtenidos de la revisión.

4.1. Planificación y Realización de la Revisión

A continuación se procede a establecer el contexto bajo el cual se realizará el estudio, los objetivos que se pretende alcanzar con la realización del estudio, el método que se usará para llevar a cabo el estudio, y se define la forma cómo se mostrarán los resultados del estudio.

4.1.1. Contexto

El alcance del estudio son los enfoques MBT orientados a pruebas funcionales. Si bien muchos de ellos son aplicados a diferentes tipos de sistemas, se identificarán cuáles de estos enfoques pueden ser aplicados en el entorno de las aplicaciones web.

Los enfoques pueden orientarse a cualquiera de las fases del proceso MBT, es decir, al modelado del SUT, a la generación de las pruebas, a la ejecución de las pruebas, o a la evaluación de las pruebas.

4.1.2. Objetivos

El presente estudio define los siguientes objetivos:

- Identificar los enfoques MBT orientados a las pruebas funcionales y a aplicaciones web.
- Caracterizar los enfoques MBT identificados a partir de la literatura disponible sobre la base de lo que es relevante para MBT.
- Analizar en base a un conjunto de indicadores las características principales que presenta el enfoque.
- Identificar si los enfoques tienen aplicaciones reales o casos prácticos asociado al uso del mismo.
- Identificar posibles lagunas en la investigación actual.

4.1.3. Método

La estrategia de búsqueda para el estudio consiste en tres fases principales: identificación de trabajos similares al presente estudio, definición y ejecución del proceso de búsqueda para encontrar otros enfoques adicionales a los mencionados en los trabajos similares, y búsqueda en las referencias de las últimas fuentes encontradas.

4.1.3.1 Identificación de Trabajos Similares

Se identifican aquellos trabajos publicados similares al presente. En consecuencia, se encontraron cuatro trabajos similares y uno orientado exclusivamente a la caracterización de herramientas MBT. Si bien no existen trabajos que se ciñan al contexto exclusivo de las pruebas funcionales, y menos aún, que se orienten exclusivamente a identificar enfoques para aplicaciones web, los cuatro trabajos similares que se encontraron presentan caracterizaciones de enfoques MBT que sirven de punto de partida para el presente estudio.

A continuación se detalla brevemente cada uno de los trabajos similares encontrados:

Título: "Characterization of Model-based Software Testing Approaches"
Autor: Arilo Claudio Dias Neto, Rajesh Subramanyanb
Tipo de Publicación: Technical Report
Año: 2007

El propósito principal de la revisión sistemática que realiza es proveer de una base para nuevas actividades de investigación y resumir las evidencias relacionadas con MBT. Por otro lado, define un protocolo de revisión para guiar la ejecución de la revisión de la literatura. Responde a la pregunta sobre qué enfoques MBT han sido descritos en la literatura técnica y cuáles son sus principales características.

Es elaborado en diciembre del 2006 y publicado en agosto de 2007, y realiza una completa investigación basada en un esquema de caracterización de los enfoques existentes desde 1990 hasta 2006, con el objetivo de identificar los enfoques que aplican UML. De la búsqueda que realiza se obtienen 406 trabajos, los cuales son agrupados según sean funcionales o estructurales. De esta forma tras analizar los enfoques logra identificar 119 enfoques para pruebas funcionales, 83 enfoques para pruebas estructurales, y el resto como no relacionado con la descripción de un enfoque MBT. En el Appendix B – Model-based Testing Approaches Classification del trabajo, identifica el dominio del software de aplicación MBT para los enfoques estudiados y si son orientados para pruebas estructurales y funcionales, identificando de esta forma sólo un trabajo [23] relacionado con aplicaciones web y pruebas funcionales.

Título: "A Survey on Model-based Testing Approaches: A Systematic Review"
Autor: Arilo C. Dias Neto, Rajesh Subramanyan
Tipo de Publicación: Artículo
Año: 2007

Su objetivo es caracterizar los enfoques MBT a partir de la literatura técnica disponible al público que sea basada en lo que es relevante para MBT. Responde a la pregunta de cuáles son los enfoques MBT publicados y cuáles son sus principales características.

Título: "Supporting the Selection of Model-based Testing Approaches for Software Projects"
Autor: Arilo Claudio Dias Neto, Guilherme Horta Travassos
Tipo de Publicación: Artículo
Año: 2008

Presenta los resultados iniciales con miras a la caracterización y la construcción de un conjunto de conocimientos sobre los enfoques MBT. Su objetivo es que estos resultados sirvan de base para proponer una infraestructura de apoyo a la selección de un enfoque MBT para un determinado proyecto de software.

Título: "Surveying Model Based Testing Approaches Characterization Attributes"
Autor: Arilo Claudio Dias Neto, Guilherme Horta Travassos
Tipo de Publicación: Artículo
Año: 2008

Describe la planeación, ejecución y el análisis de los resultados iniciales de un estudio realizado con los investigadores y profesionales relacionados con MBT. El estudio tiene dos propósitos: por un lado observar los atributos que podrían adecuarse a la caracterización de enfoques MBT, y por otro lado, definir lo que es relevante para la selección de un enfoque MBT para proyectos de software. Detalla una lista inicial de los atributos con su importancia identificada. Trata que los resultados obtenidos consoliden una base de conocimiento para apoyar a los ingenieros de software usando enfoques MBT en proyectos de software.

Título: "A Systematic Review of Model Based Testing Tool Support"
Autor: Muhammad Shafique, Yvan Labiche
Tipo de Publicación: Technical Report
Año: 2010

Presenta una revisión sistemática de las herramientas de apoyo MBT más importantes que se basan en modelos basados en estados. Provee a los profesionales en MBT las pautas en la selección de la herramienta más apropiada según sus necesidades. Responde a las preguntas como cuáles son las herramientas MBT comerciales y de investigación, cuáles son las capacidades de esas herramientas en términos de la cobertura de la prueba, nivel de automatización, y construcción de la prueba.

4.1.3.2 Definición y Ejecución del Proceso de Búsqueda

Para llevar a cabo el proceso de búsqueda de trabajos relacionados con enfoques MBT se identificaron primero las palabras clave de la búsqueda, el idioma en el cual podrían estar los trabajos, las fuentes de búsqueda de trabajos y las posibles salidas que se obtendrían al realizar la búsqueda. Además, se ha tomado como año de partida para la búsqueda el 2006, debido a que existe un trabajo de caracterización de enfoques de MBT completo sobre todos los enfoques que existen hasta septiembre de 2006.

Palabras Clave

Las palabras clave que se usaron para las búsquedas fueron:

- Model Based Testing
- Model Based Testing Functional
- Model Based Testing Web Application
- Testing Web Application

Idioma

Sólo se considera aquellos trabajos que se encuentren en idioma inglés o español.

Lista de Fuentes

Las fuentes consultadas son las siguientes:

- Biblioteca Digital ACM
- Google Scholar
- Google
- IEEEXplore
- Springerlink
- EI Compendex
- Artículos disponibles en http://www.geocities.com/model_based_testing/. Esta es la página oficial de Harry Robinson para MBT.

- Artículos listados en http://react.cs.uni-sb.de/mbt2010/. Esta es la página oficial del Sixth Workshop en MBT.

Salidas

Los trabajos obtenidos tras la realización de la búsqueda se pueden catalogar según el tipo de información que contiene y según el tipo de fuente que sea.

Según la información referente a MBT que contiene podría ser:

- Enfoque
- Detalle sobre Herramienta

Según el tipo de fuente podría ser:

- Artículos de Investigación
- Reportes Técnicos
- Manuales
- Tesis

Una vez ejecuta la búsqueda de trabajos, se analiza mediante una inspección rápida su contenido para identificar los trabajos relevantes para el estudio identificando la temática que trata y si es relevante se procede a registrarlo en una Excel (título, año, temática), así mismo se le asigna un código para manejo de la bibliografía.

Se procede a identificar los posibles duplicados en la obtención de la bibliografía y se eliminan. También se detectan los trabajos que no son válidos para el presente estudio.

4.1.3.3 Búsqueda en las Referencias

A partir de los trabajos relevantes obtenidos en los pasos anteriores, se procede a revisar sus referencias para identificar posibles trabajos adicionales relevantes para el presente estudio, y se procede a registrarlos en la Excel de la misma forma que los anteriores. En este punto se identifican los trabajos repetidos y se eliminan.

4.1.4. Resultados

Se han encontrado 17 trabajos tras realizar el proceso de búsqueda, donde uno de ellos corresponde al único enfoque orientado a pruebas funcionales y aplicaciones web que fue obtenido a partir del estudio de 2006, y el resto de trabajos son obtenidos a partir del proceso de búsqueda.

Lo siguiente es validar de los 17 trabajos cuáles son realmente enfoques, y si están orientados a pruebas funcionales y a aplicaciones web. Para determinarlo, se analiza cada uno de los trabajos obtenidos y se obtiene la información especificada en la Tabla 2 por cada trabajo.

Tabla 2: Ficha para Análisis de Trabajos

Título:	
Autores:	
Tipo de Trabajo:	[Artículo], [Tesis], [Manual], [Otro]
Año de Publicación:	
Resumen:	
Tipo de Prueba:	[Pruebas Funcionales], [Pruebas Estructurales], [Otro]
Tipo de Contenido:	[Enfoque], [Herramienta], [Otro], [No definido]
Dominio del Software:	
Relacionado con MBT:	[Sí], [No]

En el Apéndice A se muestra el listado de los 17 trabajos encontrados con su detalle según la ficha especificada en la Tabla 1. Tras analizar las fichas se determina lo siguiente:

Según el tipo de prueba hay 11 trabajos orientados a pruebas funcionales, 3 orientados a pruebas estructurales y el resto es de otro tipo de prueba o no se encuentra definido.

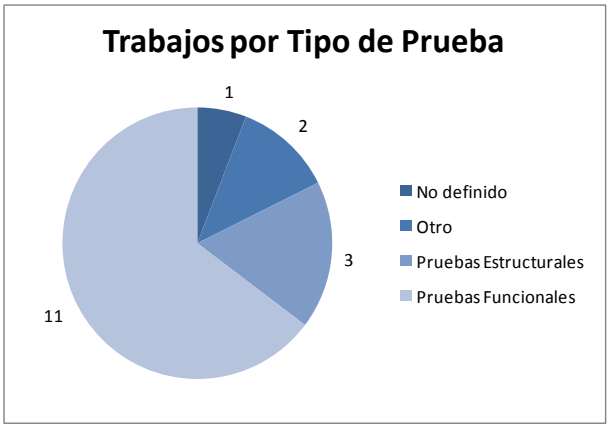

Figura 5: Trabajos Agrupados por Tipo de Prueba

Según el tipo de contenido 13 son realmente enfoques, el resto contiene información de herramientas u otro tipo de información.

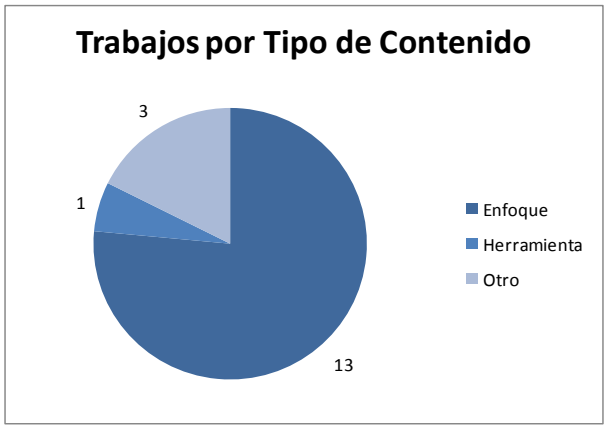

Figura 6: Trabajos Agrupados por Tipo de Contenido

Según el dominio de software sobre el que se aplica sólo 12 se aplican sobre aplicaciones web.

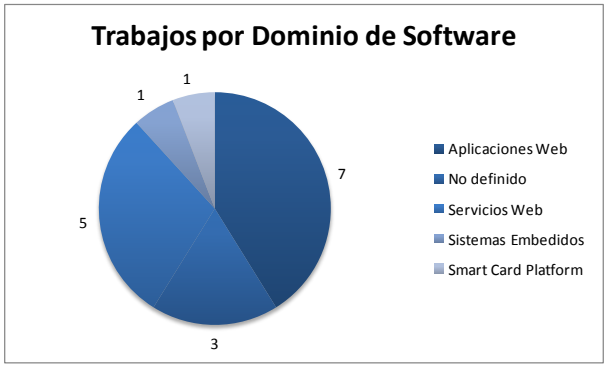

Figura 7: Trabajos Agrupado por Dominio de Software

En base a los tres criterios anteriores y teniendo en cuenta que debe estar relacionado con MBT, se selecciona los trabajos a estudiar que sean enfoques, orientados a pruebas funcionales y que su dominio de software sean las aplicaciones web (aplicaciones o servicios web según definición de aplicaciones web). Por consiguiente se obtienen 8 trabajos que cumplen con los criterios de selección tal como se ve en la Tabla 3.

Tabla 3: Trabajos por Criterios de Selección

Relacionado con MBT	Tipo de Prueba	Tipo de Contenido	Dominio del Software	Total
No	Otro	Otro	Servicios Web	**1**
Sí	No definido	Otro	No definido	**1**
		Enfoque	Smart Card Platform	**1**
	Pruebas Estructurales	Enfoque	Aplicaciones Web	**1**
			Servicios Web	**1**
		Otro	Servicios Web	**1**
	Pruebas Funcionales	**Enfoque**	No definido	**2**
			Aplicaciones Web	**6**
			Servicios Web	**2**
		Herramienta	Sistemas Embedidos	**1**
Total general				**17**

En las siguientes secciones se desarrollará el esquema de caracterización para los 8 trabajos seleccionados que se muestran en la Tabla 4.

Tabla 4: Listado de Trabajos Seleccionados para la Caracterización

Año de Publicación	Título	Referencia
2005	Testing Web Applications by Modeling w ith FSMs	[23]
2006	Tow ards Model Based Testing of Web Services	[62]
2006	A Model Based Approach for Testing the Performance of Web Applications	[37]
2007	Testing de Migración de Aplicaciones Distribuidas a Entornos Web	[61]
2007	Automated Functional Conformance Test Generation for Semantic Web Services	[35]
2009	Model Based Testing of Web Applications using NModel	[55]
2009	Improving Web Application Testing Using Testability Measures	[64]
2010	Model Based Testing Of Web Applications	[53]

4.2. Un Esquema de caracterización

De acuerdo a la definición del esquema de caracterización, el siguiente paso es definir un patrón común para representar cada enfoque. El esquema definido permite el almacenamiento de información de cada enfoque en base a un patrón común, de esta forma, es más fácil la comparación entre ellos.

El esquema de caracterización definido responde las preguntas propuestas en el estudio para poder dar respuesta a la pregunta principal del estudio de investigación, que en este caso es: ¿Cuáles son los enfoques MBT existentes que se encuentran orientados a las pruebas funcionales y para aplicaciones web? Las preguntas son:

Q1: ¿Cuáles son los enfoques que se encuentran descritos en la literatura que se encuentran orientados a pruebas funcionales y que son aplicados a aplicaciones web, y cuáles son sus características principales? Se desea saber si el enfoque ofrece una visión global en el estudio, cuales son las entradas, cuál es la notación usada para el modelo, a qué fase del proceso MBT se orienta, etc.

Q2: ¿Ofrece una adecuada herramienta de soporte? Se desea saber si la herramienta, en caso que el enfoque la ofreciera, es una herramienta comercial o netamente académica, si existen casos prácticos de uso o incluso si existen casos reales de uso de la misma, si tiene documentación asociado al uso de la herramienta.

Q3: ¿Está el enfoque bien documentado? Algunas veces es imposible aplicar un apropiado enfoque debido a que no se encuentra bien documentado.

Para dar respuesta a las preguntas planteadas se definen varios indicadores que serán aplicados sobre cada enfoque. Ellos se encuentran resumidos en la Tabla 5. Para determinar cuáles serían los elementos, atributos y dominios de la caracterización, se ha tomado como referencia los trabajos anteriores, tales como [34, 38, 39, 42, 45]. Así mismo se ha añadido algunos atributos que se consideraban relevantes para dar respuesta a las preguntas.

Tabla 5: Esquema de Caracterización para Enfoques MBT

Elementos	Atributos	Dominio
Técnica	Notación del Modelo	Texto
	Criterio de Cobertura de la Prueba	[Structural model coverage criteria], [Data coverage criteria], [Requirements coverage criteria], [Test case specifications], [Random and stochastic criteria], [Fault-based criteria], [otro]
	Tecnología para Generación de la Prueba	[Automatic/Manual technology], [Random generation], [Graph search algorithms], [Model checking], [Symbolic execution], [Theorem proving], [Online/Offline generation technology], [otro]
	Nivel de la Prueba	[Sistema], [Integración], [Unidad/Componente], [Regresión]
	Nivel de Automatización	[Total], [Parcial]
	Aspectos de la Aplicación Web	Texto
Herramienta	Especifica su Propia Herramienta	[Sí], [No]
	Nombre de Herramienta	Texto
	Tipo de Licencia	Texto
	Disponible	[Sí], [No]
	Grado de Automatización	[Total], [Parcial]
	Entorno Soportado	Texto
	Problemas/Limitaciones	Texto
Documentación	Referencia a Proyectos	[Sí], [No]
	Formato	[Reporte Técnico], [Artículo de Investigación], [Tesis]
	Tamaño	Número
	Referencias	Texto
	Dominio de Aplicación del Caso Práctico	Texto

En la Tabla 5, cada atributo es formalizado definiendo cada uno de los posibles valores, es decir el dominio, o en otros casos su formato. Los dominios indicados como texto pueden incluir cualquier tipo de descripción, y no se ha podido establecer un dominio debido a la amplia variedad de posibles valores que pudiera incluirse para dicho atributo. El dominio tamaño se especifica como numérico al ser un valor de cantidad, tal como se indica en la Tabla 5. A continuación se provee la descripción de cada atributo definido en la Tabla 5, considerando como referencia la pregunta a la que se encuentra asociado para dar respuesta.

En la Tabla 5 encontramos primero el elemento Técnica, el cual describe las características generales asociadas al enfoque, tales como modelado del comportamiento del sistema, el proceso de generación de casos de prueba, y los aspectos que tiene la aplicación web referenciada en el enfoque. Para dar respuesta a la pregunta Q1 se han planteado dentro del elemento Técnica los siguientes atributos:

- **Notación del Modelo:** Especifica cuál ha sido la notación usada en el modelo que posteriormente será usado para la generación de las pruebas.

- **Criterio de Cobertura de la Prueba:** Indica el criterio de selección usado para la selección de las pruebas, pudiendo ser: Structural model coverage criteria, Data coverage criteria, Requirements coverage criteria, Test case specifications ,Random and stochastic criteria, Fault-based criteria, u otro.

- **Tecnología para Generación de la Prueba:** Indica cuál es la tecnología que se ha usado para generar las pruebas, pudiendo ser: Automatic/Manual technology, Random generation, Graph search algorithms, Model checking, Symbolic execution, Theorem proving, Online/Offline generation technology, u otro.

- **Nivel de la Prueba:** Define cuál es el nivel de abstracción usado por el enfoque MBT para verificar el comportamiento del sistema. Los niveles usados son: Sistema, Integración, Unidad / Componente, Regresión.

- **Nivel de Automatización:** Indica si los pasos se encuentran automatizados en su totalidad o de forma parcial. Se tiene en cuenta que el paso de modelado es manual, y se consideran sólo el resto de pasos tras el modelado.

- **Aspectos de la Aplicación Web:** Describe los lenguajes, frameworks, tecnologías consideradas en el enfoque que se encuentran relacionadas directamente con las aplicaciones web.

El segundo grupo de atributos son los referentes al elemento Herramienta y permiten dar respuesta a la pregunta Q2. Los atributos seleccionados permiten identificar los aspectos relevantes de la herramienta. Los atributos seleccionados para este caso son:

- **Especifica su Propia Herramienta:** Indica si el enfoque especifica una herramienta propiamente desarrollada para el mismo.

- **Nombre de Herramienta:** Describe el nombre de la herramienta utilizada en el enfoque.

- **Tipo de Licencia:** Presenta información sobre las políticas de licenciamiento de la herramienta.

- **Disponible:** Indica si es posible obtener la herramienta desde un sitio web o desde el mismo autor.

- **Grado de Automatización:** Indica si el proceso ha sido implementado soportado sólo por la herramienta o solamente una parte del procesos es ejecutado automáticamente. Puede ser total o parcial según se encuentre o no totalmente automatizado.

- **Entorno Soportado:** Describe el entorno de ejecución soportado por la herramienta como sistema operativo, librerías, etc.

- **Problemas / Limitaciones:** Describe los problemas detectados en el enfoque a partir del estudio. Así mismo describe las limitaciones detectadas en el enfoque.

- **Referencia a Proyectos:** Indica si la documentación incluye referencias a proyectos reales o experiencias empíricas (casos prácticos de uso), en casos donde el enfoque fue utilizado.

Finalmente, encontramos los atributos para el elemento Documentación que permitirán responder la pregunta Q3. Estos atributos describen la documentación encontrada acerca de cada enfoque y aspectos relevantes para considerar que el enfoque se encuentra correctamente documentado. Los atributos para este elemento son:

- **Formato:** Describe el formato de la documentación disponible para cada enfoque, pudiendo ser: reporte técnico, artículo de investigación, tesis.

- **Tamaño:** Analiza la cantidad de documentación en base al número de páginas de cada enfoque. Se especifica en formato numérico.

- **Referencias:** Lista las principales referencias de cada enfoque.

- **Dominio de Aplicación del Caso Práctico:** Describe brevemente el dominio de aplicación del caso práctica en caso existiera. Por ejemplo, e-learning, industria automotriz, etc.

4.3. Caracterización de Enfoques

En base al esquema de caracterización presentado en la Tabla 5, en esta sección se provee una instancia del esquema por cada enfoque de la Tabla 4. Al mismo tiempo, cada enfoque es presentado con sus aspectos más relevantes y una descripción global del mismo, la cual es completada con el esquema de caracterización definido.

Sin embargo, un enfoque de la Tabla 4 no se encuentra caracterizado debido a que no incluye ningún proceso o idea común al resto de enfoques. Al finalizar la descripción de los esquemas de caracterización se muestra un detalle de este último enfoque.

4.3.1. Testing Web Applications by Modeling with FSMs (2005)

Este enfoque propone una técnica de pruebas a nivel de sistema que combina la generación de pruebas basadas en FSMs con restricciones. Utiliza un enfoque jerárquico para modelar las aplicaciones web potencialmente grandes. El enfoque consiste en dos fases:

- La primera fase, construye un modelo de la aplicación a ser probada.
- La segunda fase, genera las pruebas a partir del modelo.

El enfoque se basa en jerarquías de FSMs que modela el subsistema de las aplicaciones web, y luego genera los requerimientos de prueba como consecuencia de los estados de la máquina. Estas subsecuencias son combinadas y refinadas para formar pruebas ejecutables completas. Las restricciones son utilizadas para seleccionar un reducido conjunto de entradas con el objetivo de reducir la explosión del espacio de estados inherentes al uso de FSMs.

En la Tabla 6 se muestra el esquema de caracterización para este enfoque.

Tabla 6: Esquema de Caracterización para el Enfoque "Testing Web Applications by Modeling with FSMs"

Elementos	Atributos	Dominio
Técnica	Notación del Modelo	Transition-based Notations - Máquina de Estados Finito
	Criterio de Cobertura de la Prueba	Test case specifications
	Tecnología para Generación de la Prueba	Automatic technology, Graph search algorithms
	Nivel de la Prueba	Sistema
	Nivel de Automatización	Parcial
	Aspectos de la Aplicación Web	HTML, JavaScript, Java Servlet, Java Beans, JSPs
Herramienta	Especifica su Propia Herramienta	Sí
	Nombre de Herramienta	Define su propia herramienta
	Tipo de Licencia	No especifica
	Disponible	No
	Grado de Automatización	Parcial
	Entorno Soportado	No especifica
	Problemas/Limitaciones	• No incluye el oráculo que es el que decide si los resultados sin correctos. • Tiene soporte limitado para las transiciones operacionales. • El proceso de generación está en proceso de refinamiento especialmente en el reconocimiento y uso de las relaciones de las entradas como restricciones.
Documentación	Referencia a Proyectos	Sí
	Formato	Artículo de Investigación
	Tamaño	28
	Referencias	[13, 14, 44, 58]
	Dominio de Aplicación del Caso Práctico	Sistema de Información de Estudiantes

4.3.2. Towards Model Based Testing of Web Services (2006)

Muestra un ejemplo sobre cómo los protocolos podrían servir como entradas para MBT de Servicios Web. Propone usar Sistemas de Transiciones Simbólicas y la teoría de pruebas subyacentes para la coordinación del modelado y las pruebas del enfoque. Usa las especificaciones de la interfaz como entradas del proceso MBT.

Especifica su propia herramienta que incorpora el framework Audition. Usa Sistemas de Transición Simbólica (STS) que permite explotar los algoritmos y teorías de las pruebas basadas en STS descritos en [15]. En la Tabla 7 se muestra el esquema de caracterización para este enfoque.

Tabla 7: Esquema de Caracterización para el Enfoque "Towards Model Based Testing of Web Services"

Elementos	Atributos	Dominio
Técnica	Notación del Modelo	Transition-based Notations - Sistema de Transición Simbólica
	Criterio de Cobertura de la Prueba	Test case specifications
	Tecnología para Generación de la Prueba	Automatic technology, Theorem proving
	Nivel de la Prueba	Sistema
	Nivel de Automatización	Parcial
	Aspectos de la Aplicación Web	Servicios Web
Herramienta	Especifica su Propia Herramienta	Sí
	Nombre de Herramienta	Define su propia herramienta
	Tipo de Licencia	No especifica
	Disponible	No
	Grado de Automatización	Parcial
	Entorno Soportado	No especifica
	Problemas/Limitaciones	No especifica
Documentación	Referencia a Proyectos	Sí
	Formato	Artículo de Investigación
	Tamaño	17
	Referencias	[15, 16]
	Dominio de Aplicación del Caso Práctico	Sistema de Abastecimiento de Suministros

4.3.3. A Model Based Approach for Testing the Performance of Web Applications (2006)

El enfoque simplifica la generación de cargas de trabajo sintéticas usadas en las pruebas de desempeño. Usa un modelo formal para captar el comportamiento de la aplicación. El modelo puede ser usado para obtener varias secuencias de solicitudes representando cómo los usuarios interactúan normalmente con la aplicación basada en la web planteada. La secuencia a su vez puede ser utilizada para construir cargas de trabajo con características específicas. Las ventajas que posee frente a otros enfoques son las siguientes:

- Provee soporte automatizado para características de control de carga de trabajo fine-grained.

- Menor esfuerzo es necesario para adaptar las herramientas de generación de cargas de trabajo sintéticas para manejar los cambios hechos para un sistema dado o para utilizarlos para pruebas de otros sistemas. Esta mejora de la portabilidad es un resultado indirecto de la proporcionada por el modelo de la aplicación. El modelo de la aplicación hace que la herramienta de generación de carga de trabajo sea independiente del SUT.

En la Tabla 8 se muestra el esquema de caracterización para este enfoque.

Tabla 8: Esquema de Caracterización para el Enfoque "A Model Based Approach for Testing the Performance of Web Applications"

Elementos	Atributos	Dominio
Técnica	Notación del Modelo	Transition-based Notations - Máquina de Estados Finita Extendida
	Criterio de Cobertura de la Prueba	Test case specifications
	Tecnología para Generación de la Prueba	Automatic technology, Graph search algorithms
	Nivel de la Prueba	Sistema
	Nivel de Automatización	Parcial
	Aspectos de la Aplicación Web	Página Web
Herramienta	Especifica su Propia Herramienta	Sí
	Nombre de Herramienta	Define su propia herramienta
	Tipo de Licencia	No especifica
	Disponible	No
	Grado de Automatización	Parcial
	Entorno Soportado	No especifica
	Problemas/Limitaciones	Requiere cierto esfuerzo especificar el modelo de la aplicación.
Documentación	Referencia a Proyectos	Sí
	Formato	Artículo de Investigación
	Tamaño	8
	Referencias	[17, 60]
	Dominio de Aplicación del Caso Práctico	Carrito de Compras

4.3.4. Testing de Migración de Aplicaciones Distribuidas a Entornos Web (2007)

El enfoque propuesto se basa en la reutilización de casos de pruebas existentes, resultantes de la trazabilidad con los casos de uso planteados para el sistema distribuido, y la automatización de las pruebas unitarias, de integración y de regresión.

Estudia la reutilización de requisitos de aplicaciones distribuidas a aplicaciones Web, a partir de la trazabilidad de casos de uso de ambas aplicaciones. Además, define metodologías de trabajo para la prueba de los resultados de las migraciones a la Web de aplicaciones distribuidas no realizadas bajo entorno Web, a partir de la consideración de las propiedades que se quieren preservar de estas últimas. Documenta la información recopilada en función de los resultados obtenidos luego de realizar las pruebas de migración. Para llevar a cabo todo ello se centra en los siguientes tres puntos:

- Construcción de metamodelos para abstraer propiedades en común de modelos de aplicaciones Web y de modelos de aplicaciones tradicionales, los que sirven de base para mapear casos de uso utilizados en aplicaciones distribuidas no Web a casos de uso de aplicaciones basadas en tecnología Web.

- Formulación de una metodología de análisis para la migración de aplicaciones distribuidas a entornos Web, basada en un enfoque de pruebas con reutilización de casos de prueba y que prevea la utilización de herramientas automáticas para la ejecución y reutilización de los casos de prueba generados.

- Aplicación de la metodología propuesta tomando como caso de estudio el Sistema de Gestión Académica de una institución universitaria a efectos de comprobar su desempeño.

En la Tabla 9 se muestra el esquema de caracterización para este enfoque.

Tabla 9: Esquema de Caracterización para el Enfoque "Testing de Migración de Aplicaciones Distribuidas a Entornos Web"

Elementos	Atributos	Dominio
Técnica	Notación del Modelo	Transition-based Notations - UML
	Criterio de Cobertura de la Prueba	Test case specifications
	Tecnología para Generación de la Prueba	Manual technology, Graph search algorithms
	Nivel de la Prueba	Unidad / Componente, Integración, Regresión
	Nivel de Automatización	Parcial
	Aspectos de la Aplicación Web	Página Web
Herramienta	Especifica su Propia Herramienta	Sí
	Nombre de Herramienta	Define su propia herramienta
	Tipo de Licencia	No especifica
	Disponible	No
	Grado de Automatización	Parcial
	Entorno Soportado	No especifica
	Problemas/Limitaciones	Sólo es aplicable es migraciones.
Documentación	Referencia a Proyectos	Sí
	Formato	Tesis
	Tamaño	217
	Referencias	[18, 19, 20, 50, 59, 63]
	Dominio de Aplicación del Caso Práctico	Sistema de Autogestión de Alumnos

4.3.5. Automated Functional Conformance Test Generation for Semantic Web Services (2007)

Presenta un enfoque automatizado para generar pruebas funcionales de conformidad para semántica de servicios web los cuales son definidos usando el paradigma Entradas, Salidas, Precondiciones, Efectos (IOPEs). Para cada servicio web, el enfoque produce un objetivo de prueba usando un conjunto de modelos de falla. Un nuevo componente planificador acepta los objetivos de las pruebas para generar una secuencia de invocaciones a servicios web como casos de prueba.

Tabla 10: Esquema de Caracterización para el Enfoque "Automated Functional Conformance Test Generation for Semantic Web Services"

Elementos	Atributos	Dominio
Técnica	Notación del Modelo	State-Based (or Pre/Post) Notations - IOPEs - Inputs, Outputs, Preconditions, Effects
	Criterio de Cobertura de la Prueba	Test case specifications
	Tecnología para Generación de la Prueba	Automatic technology, Random generation
	Nivel de la Prueba	Sistema
	Nivel de Automatización	Total
	Aspectos de la Aplicación Web	Servicios Web
Herramienta	Especifica su Propia Herramienta	No especifica
	Nombre de Herramienta	No especifica
	Tipo de Licencia	No especifica
	Disponible	No especifica
	Grado de Automatización	No especifica
	Entorno Soportado	No especifica
	Problemas/Limitaciones	• Asume servicios web atómicos. • Los casos de prueba son producidos a priori.
Documentación	Referencia a Proyectos	Sí
	Formato	Artículo de Investigación
	Tamaño	8
	Referencias	[16, 21, 22]
	Dominio de Aplicación del Caso Práctico	Sistema de Abastecimiento de Suministros

Otra característica resaltante del enfoque es la generación de las secuencias de verificación. Por último el enfoque permite la generación y ejecución de los casos de pruebas los cuales pueden ser aplicados a varias interfaces a través de las cuales los servicios web pueden ser accedidos. En la Tabla 10 se muestra el esquema de caracterización para este enfoque.

4.3.6. Model Based Testing of Web Applications using NModel (2009)

Muestra cómo MBT puede ser aplicado en el contexto de las aplicaciones web usando un kit de herramientas llamado NModel [27]. El caso concreto de estudio es un sistema comercial basado en la web. El principal objetivo es probar el correcto funcionamiento del sistema bajo circunstancias normales y durante el arranque y apagado del sistema. En la Tabla 11 se muestra el esquema de caracterización para este enfoque.

Tabla 11: Esquema de Caracterización para el Enfoque "Model Based Testing of Web Applications using NModel"

Elementos	Atributos	Dominio
Técnica	Notación del Modelo	Transition-based Notations - Máquina de Estados Finito
	Criterio de Cobertura de la Prueba	Test case specifications
	Tecnología para Generación de la Prueba	Automatic technology, Online/Offline generation technology, Random generation
	Nivel de la Prueba	Sistema
	Nivel de Automatización	Parcial
	Aspectos de la Aplicación Web	Página web (el ejemplo usa HTML, JavaScript)
Herramienta	Especifica su Propia Herramienta	No
	Nombre de Herramienta	NModel [27]
	Tipo de Licencia	Open Source - Microsoft License for Nmodel
	Disponible	Sí
	Grado de Automatización	Total
	Entorno Soportado	No especifica, pero se sabe que funciona sobre Windows
	Problemas/Limitaciones	• No posee una herramienta para coger mensajes. • No tiene una herramienta que ayude a generar los escenarios que corresponden con casos de prueba abstractos.
Documentación	Referencia a Proyectos	Sí
	Formato	Artículo de Investigación
	Tamaño	16
	Referencias	[23, 24]
	Dominio de Aplicación del Caso Práctico	Sistema de Posicionamiento

4.3.7. Model Based Testing Of Web Applications (2010)

Presenta un enfoque para el modelado del SUT, generación de los casos de prueba y para la ejecución de los mismos. Para llevar a cabo todas estas tareas hace uso de la herramienta MBT TestOptimal [26]. Para explicar el enfoque hace uso de un caso práctico basado en una aplicación web para servicios de salud. Al inicio del artículo, identifica los problemas que presentan las pruebas de aplicaciones web. En secciones posteriores del artículo muestra cómo dar solución a los problemas encontrados. Provee ventajas como:

- Capacidad para la generación de pruebas altamente adaptables.
- Actualización rápida del modelo y regeneración y ejecución de las pruebas.
- Provee soporte completo para pruebas de regresión.

En la Tabla 12 se muestra el esquema de caracterización para este enfoque.

Tabla 12: Esquema de Caracterización para el Enfoque "Model Based Testing Of Web Applications"

Elementos	Atributos	Dominio
Técnica	Notación del Modelo	Transition-based Notations - Máquina de Estados Finito
	Criterio de Cobertura de la Prueba	Test case specifications
	Tecnología para Generación de la Prueba	Automatic technology, Graph search algorithms
	Nivel de la Prueba	Sistema
	Nivel de Automatización	Total
	Aspectos de la Aplicación Web	Página Web
Herramienta	Especifica su Propia Herramienta	No
	Nombre de Herramienta	TestOptimal [26]
	Tipo de Licencia	Licencia Personalizada
	Disponible	Sí
	Grado de Automatización	Total
	Entorno Soportado	• JDK 1.6 or later • Windows XP, VISTA, 2003, Windows 7 or Linux • 512mb de memoria o más, y 200mb de espacio libre en disco • Navegador con javascript habilitado y bloqueador de popup deshabilitado
	Problemas/Limitaciones	No especifica
Documentación	Referencia a Proyectos	Sí
	Formato	Artículo de Investigación
	Tamaño	28
	Referencias	[25, 41]
	Dominio de Aplicación del Caso Práctico	Sistema de Servicios para la Salud

4.3.8. Otros Enfoques

Como se mencionó, existe un enfoque de la Tabla 4 sobre el cual no se ha instanciado el esquema de caracterización. El enfoque es: "Improving Web Application Testing Using Testability Measures". Y no se ha caracterizado por las siguientes razones:

- No especifica el modelado de la aplicación.
- No especifica la generación de casos de prueba.
- Se refiere a definición de métricas para probarlas durante el proceso de pruebas automatizadas.

Como se puede ver, no se refiere específicamente al desarrollo de un enfoque MBT, sino a la definición de un framework de métricas para procesos de pruebas automatizadas, razón por la cual carece de información respecto a la forma de modelar la aplicación y la generación de pruebas.

4.4. Análisis

Con la revisión del estado del arte y la definición del esquema de caracterización para cada uno de los enfoques seleccionados para el estudio que se encuentran listados en la Tabla 4, esta sección analiza los resultados obtenidos y da respuesta a las preguntas planteadas previo al proceso de definición del esquema de caracterización.

Como pregunta principal se definió la siguiente: ¿Cuáles son los enfoques MBT existentes que se encuentran orientados a las pruebas funcionales y para aplicaciones web?

Tras desarrollar los esquemas de caracterización para los enfoques seleccionados, y excluir uno de ellos ya que no se podía aplicar sobre este el esquema de caracterización debido al planteamiento que hacía, se concluye que, los enfoques existentes que se encuentran

orientados a las pruebas funcionales y que son aplicados en el contexto de las aplicaciones web son los que se encuentran listados en la Tabla 13.

Tabla 13: Listado de Enfoques orientados a Pruebas Funcionales y aplicados en el contexto de las Aplicaciones Web

Año de Publicación	Título	Referencia
2005	Testing Web Applications by Modeling w ith FSMs	[23]
2006	Tow ards Model Based Testing of Web Services	[62]
2006	A Model Based Approach for Testing the Performance of Web Applications	[37]
2007	Testing de Migración de Aplicaciones Distribuidas a Entornos Web	[61]
2007	Automated Functional Conformance Test Generation for Semantic Web Services	[35]
2009	Model Based Testing of Web Applications using NModel	[55]
2010	Model Based Testing Of Web Applications	[53]

En las siguientes secciones se analizarán los atributos seleccionados para ayudar a dar respuesta a las tres preguntas específicas que se plantearon.

4.4.1. Análisis de la Técnica usada en los Enfoques

Se definió el elemento Técnica con la finalidad de dar respuesta a la primera pregunta específica: ¿Cuáles son los enfoques que se encuentran descritos en la literatura que se encuentran orientados a pruebas funcionales y que son aplicados a aplicaciones web, y cuáles son sus características principales?

Para dar respuesta a la pregunta se planteaba analizar si el enfoque ofrecía una visión global en el estudio y cuáles eran las características del enfoque respecto a:

- Notación usada en el modelo.
- Criterio de cobertura para selección de la prueba.
- Tecnología usada para la generación de la prueba.
- Nivel de prueba basado en la escala del SUT sobre el que era aplicado el enfoque.
- Nivel de automatización.
- Aspectos considerados de la aplicación web.

Respecto a la notación usada, se puede ver que si bien se usan diferentes lenguajes de notación como se especifica en la Tabla 14, la mayoría de estos lenguajes corresponden con el paradigma de Notación Basada en Transiciones como se puede apreciar en la Figura 8, y sólo un enfoque usa una notación cuyo paradigma está basado en notaciones Pre-post.

Tabla 14: Detalle de la Notación de los Enfoques

Enfoque	Paradigma de Notación	Notación de Modelo
Testing Web Applications by Modeling with FSMs (2005)	Transition-based Notations	Máquina de Estados Finito
Towards Model Based Testing of Web Services (2006)	Transition-based Notations	Sistema de Transición Simbólica
A Model Based Approach for Testing the Performance of Web Applications (2006)	Transition-based Notations	Máquina de Estados Finita Extendida
Testing de Migración de Aplicaciones Distribuidas a Entornos Web (2007)	Transition-based Notations	UML
Automated Functional Conformance Test Generation for Semantic Web Services (2007)	State-Based (or Pre/Post) Notations	IOPEs - Inputs, Outputs, Preconditions, Effects
Model Based Testing of Web Applications using NModel (2009)	Transition-based Notations	Máquina de Estados Finito
Model Based Testing Of Web Applications (2010)	Transition-based Notations	Máquina de Estados Finito

En lo que respecta al criterio de cobertura para la selección de la prueba, todos se basan en Especificaciones de Casos de Prueba tal como se ve en la Tabla 15, es decir, se realiza una especificación de casos de prueba bajo una notación formal, y esta misma, puede ser utilizada para determinar las pruebas que se generarán. Dentro de las notaciones usadas para este tipo

de criterio de selección se encuentran: FSMs, UML Testing Profile (UTP), expresiones regulares, fórmulas de lógica temporal, restricciones, y cadenas de Markov.

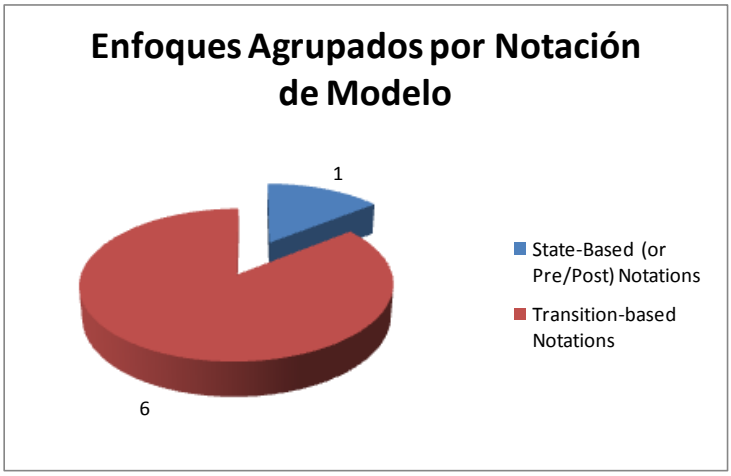

Figura 8: Enfoques Agrupados por Notación de Modelo

También se puede apreciar en la Tabla 15 que se usan varias tecnologías para generación de los casos de prueba. Tal como se mencionó en la sección referente a la taxonomía para MBT, se especificaba que varias tecnologías pueden ser usadas al mismo tiempo. Para el caso de los enfoque todos a excepción de uno genera automáticamente los casos de prueba. Además, la mayoría usa algoritmos de búsqueda gráfica, la cual como se vio, se basa en un algoritmo de búsqueda sobre una gráfica que incluye algoritmos de cobertura de nodos o arcos.

Tabla 15: Detalle de los Criterios y Tecnologías de los Enfoques para la Generación de las Pruebas

Enfoque	Criterio de Cobertura de la Prueba	Tecnología para Generación de la Prueba
Testing Web Applications by Modeling with FSMs (2005)	Test case specifications	Automatic technology, Graph search algorithms
Towards Model Based Testing of Web Services (2006)	Test case specifications	Automatic technology, Theorem proving
A Model Based Approach for Testing the Performance of Web Applications (2006)	Test case specifications	Automatic technology, Graph search algorithms
Testing de Migración de Aplicaciones Distribuidas a Entornos Web (2007)	Test case specifications	Manual technology, Graph search algorithms
Automated Functional Conformance Test Generation for Semantic Web Services (2007)	Test case specifications	Automatic technology, Random generation
Model Based Testing of Web Applications using NModel (2009)	Test case specifications	Automatic technology, Online/Offline generation technology, Random generation
Model Based Testing Of Web Applications (2010)	Test case specifications	Automatic technology, Graph search algorithms

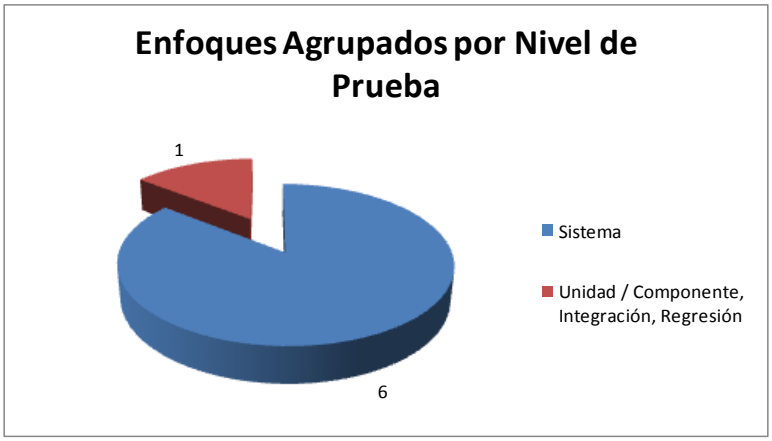

Figura 9: Enfoques Agrupados por Nivel de Prueba

En lo que respecta al nivel de prueba sobre el que se aplica el enfoque, se puede apreciar en la Figura 9 que seis enfoques son aplicados en pruebas de sistema, y sólo uno desarrolla el enfoque sobre pruebas unitarias, de componente, de integración y regresión.

Sobre el nivel de automatización de los enfoques, sólo dos presentan una automatización total. Los que presentan automatización parcial es porque no cubren en la mayoría de casos la automatización para la generación de las pruebas.

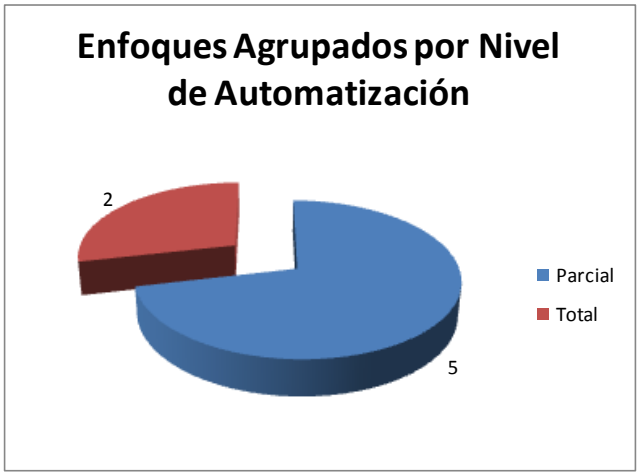

Figura 10: Enfoques Agrupados por Nivel de Automatización

Cabe resaltar que lo más importante en MBT es la generación y ejecución automática de los casos de prueba. Tener ambas tareas automatizadas implica como se vio un decremento del esfuerzo y tiempo empleado para las pruebas.

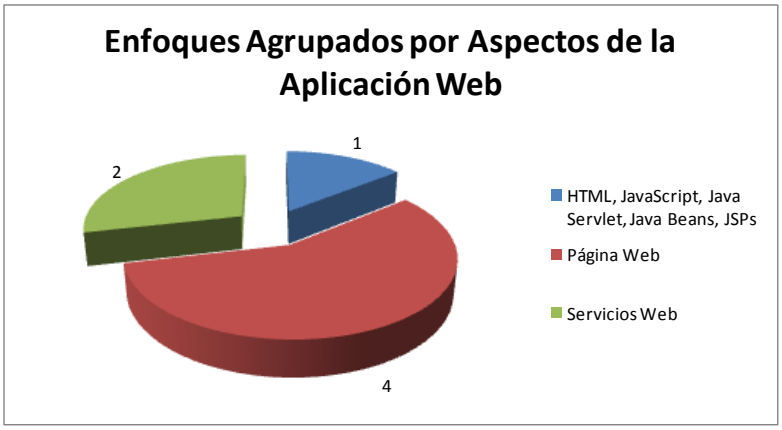

Figura 11: Enfoques Agrupados por Aspectos de la Aplicación Web

Los enfoques seleccionados se aplican sobre aplicaciones web, pero dentro de las aplicaciones web hay una gama de posibilidades. En la Figura 11 se puede apreciar que cuatro enfoques especifican que se desarrollan sobre páginas web, luego hay dos que se orientas a las pruebas de los servicios web, y sólo un enfoque especifica la tecnología sobre la cual es aplicado.

4.4.2. Análisis de la Herramienta usada en los Enfoques

Se definió el elemento Herramienta con la finalidad de dar respuesta a la segunda pregunta específica: ¿Ofrece una adecuada herramienta de soporte?

Para dar respuesta a la pregunta se planteaba analizar si la herramienta usada en el enfoque, en caso que el enfoque usara alguna, era comercial o netamente académica, si existían casos prácticos de uso o incluso si existían casos reales de uso de la misma, si tenía documentación asociado al uso de la herramienta. Por ello se definieron los siguientes puntos respecto a la herramienta:

- Especifica el enfoque su propia herramienta.
- Nombre de la herramienta.
- Tipo de licencia de la herramienta.
- Se encuentra disponible la herramienta.
- Grado de automatización de la herramienta.
- Entorno soportado por la herramienta.
- Problemas y limitaciones que presenta la herramienta.

Tal como se muestra en la Figura 12, sólo cuatro enfoques especificaban el uso de de una herramienta creada especialmente para el desarrollo del enfoque. Dos de los enfoques usaban una herramienta ya existente para desarrollar el proceso. Un enfoque no mencionaba el uso alguno de herramienta.

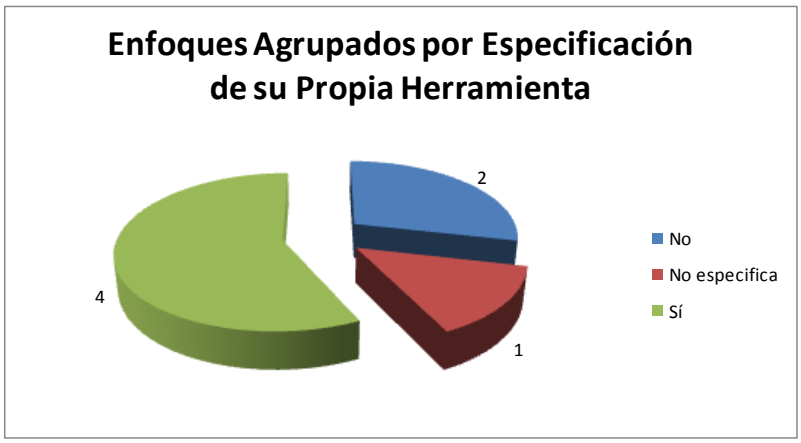

Figura 12: Enfoques Agrupados por Especificación de su Propia Herramienta

Si bien son cuatro los enfoque que especifican el uso de una herramienta desarrollada especialmente para el desarrollo del enfoque, ninguno de ellos brinda mayor información sobre el nombre de la herramienta, la licencia, o el entorno sobre el que es soportado. Sólo dos enfoques tal como se ve en la Tabla 16, hacen explícito el nombre de la herramienta que utilizan para desarrollar el enfoque.

Tabla 16: Enfoques con Detalle de la Herramienta Utilizada

Enfoque	Especifica su Propia Herramienta	Nombre de la Herramienta
Testing Web Applications by Modeling with FSMs (2005)	Sí	Define su propia herramienta
Towards Model Based Testing of Web Services (2006)	Sí	Define su propia herramienta
A Model Based Approach for Testing the Performance of Web Applications (2006)	Sí	Define su propia herramienta
Testing de Migración de Aplicaciones Distribuidas a Entornos Web (2007)	Sí	Define su propia herramienta
Automated Functional Conformance Test Generation for Semantic Web Services (2007)	No especifica	No especifica
Model Based Testing of Web Applications using NModel (2009)	No	NModel
Model Based Testing Of Web Applications (2010)	No	TestOptimal

Sólo las herramientas NModel [27] y TestOptimal [26] se encuentran disponibles en la web, tal como se constató, y brindan soporte para el desarrollo de pruebas con MBT. TestOptimal es una herramienta comercial, pero ofrece una versión trial que tras cumplir el periodo permitido restringe el uso de la funcionalidad de la herramienta. Mientras que NModel es una herramienta OpenSource aunque también tiene su propia licencia bajo el nombre de "Microsoft License for NModel".

El grado de automatización de las herramientas sólo es total para NModel y TestOptimal, esto quiere decir que dan soporte tanto para la generación de los casos de prueba como para la ejecución de los mismos.

Tabla 17: Detalle de las Herramientas MBT usadas en los Enfoques

Nombre de Herramienta	Disponible	Tipo de Licencia	Grado de Automatización	Entorno Soportado
Define su propia herramienta	No	No especifica	Parcial	No especifica
NModel	Sí	Open Source - Microsoft License for Nmodel	Total	No especifica, pero se sabe que funciona sobre Windows
TestOptimal	Sí	Licencia Personalizada	Total	• JDK 1.6 or later • Windows XP, VISTA, 2003, Windows 7 or Linux • 512mb de memoria o más, y 200mb de espacio libre en disco • Navegador con javascript habilitado y bloqueador de popup deshabilitado

Respecto al entorno soportado por las herramientas, sólo en el caso de TestOptimal se especifican los requisitos previos a la instalación, mientras que en la página web de NModel sólo mencionaba que no existía ninguna especificación especial, sin embargo podemos asegurar que la herramienta corre sobre entorno Windows.

4.4.3. Análisis de la Documentación usada en los Enfoques

Se definió el elemento Documentación con la finalidad de dar respuesta a la tercera pregunta específica: ¿Está el enfoque bien documentado?

Para dar respuesta a la pregunta se planteaba analizar características de la documentación asociada al enfoque. Las características definidas fueron:

- Referencia a casos prácticos de uso del enfoque.
- Formato en el que se encuentra el enfoque.
- Tamaño en número de páginas, con la finalidad de tener una referencia.
- Referencias principales usadas en el enfoque.
- Dominio de aplicación del caso práctico.

Tabla 18: Detalle de la Documentación asociada a cada Enfoque

Referencia	Referencia a Proyectos	Formato	Tamaño
Testing Web Applications by Modeling with FSMs (2005)	Sí	Artículo de Investigación	28
Towards Model Based Testing of Web Services (2006)	Sí	Artículo de Investigación	17
A Model Based Approach for Testing the Performance of Web Applications (2006)	Sí	Artículo de Investigación	8
Testing de Migración de Aplicaciones Distribuidas a Entornos Web (2007)	Sí	Tesis	217
Automated Functional Conformance Test Generation for Semantic Web Services (2007)	Sí	Artículo de Investigación	8
Model Based Testing of Web Applications using NModel (2009)	Sí	Artículo de Investigación	16
Model Based Testing Of Web Applications (2010)	Sí	Artículo de Investigación	28

Tabla 19: Dominios de Aplicación de los Casos Prácticos

Dominio de Aplicación del Caso Práctico	Total
Carrito de Compras	1
Sistema de Abastecimiento de Suministros	2
Sistema de Autogestión de Alumnos	1
Sistema de Información de Estudiantes	1
Sistema de Posicionamiento	1
Sistema de Servicios para la Salud	1
Total general	**7**

Tal como se muestra en la Tabla 18, un enfoque que se encuentra en formato tesis, el resto se encuentra en formato artículo.

Todos los enfoques desarrollan un caso práctico tal como lo muestra la Tabla 18, y el dominio sobre el cual se desarrolla el caso práctico se puede apreciar en la Tabla 19. Existen dos enfoques [46, 35] cuyo dominio de aplicación del caso práctico es un Sistema de Abastecimiento de Suministros, ya que se enfocan a MBT para servicios web, y dentro de este dominio se trata las aplicaciones B2B con servicios web.

5. Conclusiones

Si bien hasta el 2006 existía sólo un enfoque para MBT orientado a aplicaciones web y pruebas funcionales, en estos últimos cinco años han aparecido más enfoques dentro de ese alcance. La explosión de las aplicaciones web ha significado un aumento significativo de las investigaciones, entre ellas, el uso de MBT para la fase de pruebas del proceso de desarrollo de software.

Los enfoques seleccionados para el estudio [23, 35, 37, 53, 55, 61, 62] se centran en dos tipos de aplicaciones web: servicios web y páginas web. Pero en ninguno de los dos casos hacen explícita una determinada tecnología a usar sobre cada uno de ellos. En su mayoría, estos enfoques tienen un nivel de automatización parcial, y ninguno de ellos, tanto con automatización parcial o total, toma en cuenta el uso de un oráculo, debido a la complejidad del mismo. Son preferidos por los enfoques el tipo de notación gráfica, debido a que facilita el modelado de la aplicación y resulta ser más comprensible y sencillo a la hora de modelar. Todos lo enfoques muestran un caso práctico sobre el uso del mismo, pero ninguno referencia a proyectos reales donde se haya aplicado el enfoque. Los enfoques que desarrollan su propia herramienta para el uso de MBT no brindan mayor información sobre ella, sólo indican que está en proceso de investigación, lo cual hace difícil aplicar el enfoque sin contar con la herramienta a la que se referencia. Sólo dos enfoques de los seleccionados hacen uso de herramientas MBT disponibles en la web, sin embargo, estas herramientas han sido utilizadas fuera del alcance definido en el presente estudio con resultados exitosos.

En la mayoría de los enfoques seleccionados para el estudio existe una laguna en lo referente a la ejecución de las pruebas y a la evaluación de las mismas. La evaluación de los resultados obtenidos en las pruebas es un tema de investigación que se encuentra abierto. El uso del oráculo dentro de la evaluación de las pruebas, parece ser lo que hace tan complejo este proceso, ya que supone un conocimiento adicional al que se necesitaba para el modelado de la aplicación.

6. Trabajos Citados

1. **Myers, G.** *The Art of Software Testing.* s.l. : Wiley-Interscience, 1979.

2. *Model-based testing in practice.* **S. Dalal, A. Jain, N. Karunanithi, J.M. Leaton, C. M. Lott, G. C. Patton, B. M. Horowitz.** 1999. ICSE'99. págs. pp. 285-294.

3. **Series, DACS Gold Practice Document.** Software Acquisition Gold Practice, model-based testing. [En línea] 2006. https://www.goldpractices.com/practices/mbt/index.php.

4. *One evaluation of model-based testing and its automation.* **A. Pretschner, W. Prenninger, S. Wagner.** s.l. : ACM Press, 2005. 27th International Conference on Software Engineering. págs. 392–401.

5. *Using a model-based test generator to test for standard conformance.* **E. Farchi, A. Hartman, S. S. Pinter.** IBM Systems Journal, págs. 41(1):89–110.

6. *Requirement traceability in automated test generation: Application to smart card software validation.* **F. Bouquet, E. Jaffuel, B. Legeard, F. Peureux, M. Utting.** s.l. : ACM Press, 2005. ICSE International Workshop on Advances in Model-Based Software Testing.

7. Página Oficial de World Wide Web Consortium. [En línea] http://www.w3.org.

8. SOA infrastructure market shares, market strategy, and market forecasts. *Global Information.* [En línea] 2008. http://www.the-infoshop.com/study/wg64381-soa-infra-mkt.html.

9. *Designing Web Usability.* **Nielsen, Jakob.** s.l. : New Riders Publishing, 2000.

10. **IEEE.** *Software Engineering Book of Knowledge IEEE.* s.l. : IEEE, 2004.

11. *IEEE standard glossary of software engineering terminology.* **F. Jay, R.Mayer.** 1990, pág. IEEE Std 610.

12. **Sommerville, Ian.** *Software engineering.* 5th. Redwood City : Addison Wesley Longman Publishing Co., 1995.

13. *A model-based approach for testing Web applications.* **D. Kung, C. H. Liu, P. Hsia.** Chicago, IL : s.n., 2000. Twelfth International Conference on Software Engineering and Knowledge Engineering.

14. *Generating test cases for XML-based Web component interactions using mutation analysis.* **Offutt, Suet Chun Lee and Jeff.** Hong Kong China : IEEE Computer Society Press, 2001. 12th International Symposium on Software Reliability Engineering. págs. 200-209.

15. *Test generation based on symbolic specifications.* **Frantzen, L., J. Tretmans, T. Willemse.** 2005, FATES, págs. 1–15.

16. *Audition of web services for testing conformance to open specified protocols.* **A. Bertolino, L. Frantzen, A. Polini, and J. Tretmans.** [ed.] Springer-Verlag. 2006, Architecting Systems with Trustworthy Components, number 3938 in LNCS.

17. *Principles and Methods of Testing Finite State Machines .* **D. Lee, M. Yannakakis.** 1996. IEEE. págs. Vol. 84, pp. 1090-1123.

18. **Beizer, B.** *Black-Box Testing: Techniques for Functional Testing of Software and Systems.* New York : John Wiley & Sons, 1995.

19. **Linz T., Daigl M.** *Capture and Replay GUI Testing Tools.* Germany : s.n., 1998. Results of the ESSI PIE 24306.

20. **Rosaria S., Robinson H.** *Applying Models in your Testing Process.* s.l. : Intelligent Search Test Group. Microsoft Corporation, 2001.

21. *A web services framework for testing web services.* **Zhu, H.** 2005. IEEE COMPSAC'2005. págs. 34–39.

22. *Automatic conformance testing of web services.* **Mariani, R. Heckel and L.** 2005. Fundamental Approaches to Software Engineering (FASE 05). págs. 34–48.

23. *Testing web applications by modeling with FSMs.* **A. A. Andrews, J. Offutt, and R. T. Alexander.** 2005, Software and Systems Modeling, págs. 4:326–345.

24. *Protocol modeling with model program composition.* **Schulte, M. Veanes and W.** [ed.] FORTE. 2008, Lecture Notes in Computer Science, Springer, págs. 324–339.

25. **NIST.** Finite State Machine Definition. [En línea] http://xlinux.nist.gov/dads/HTML/finiteStateMachine.html.

26. Página Oficial para Herramienta MBT TestOptimal. [En línea] http://testoptimal.com/.

27. Página Oficial para Herramienta MBT NModel. [En línea] http://nmodel.codeplex.com/.

28. **M. Utting, Bruno Legeard.** *Practical Model-based Testing.* s.l. : Elsevier, 2007.

29. *Model-Driven Testing: Using the UML Testing Profile.* **Paul Baker, Zhen Ru Dai, Jens Grabowski.** 2008, Springer-Verlag.

30. *Graph Theory Techniques in Model-Based Testing.* **Robinson, Harry.** 1999. International Conference on Testing Computer Software.

31. *Model-based Testing: Next Generation Functional Software Testing.* **Legeard, Bruno.** Dagstuhl Seminar Proceedings 10111. Vol. http://drops.dagstuhl.de/opus/volltexte/2010/2620.

32. **Kull, Andres.** Model-Based Testing of Reactive Systems. *Tesis Faculty of Information Technology, Tallin University of Technology.* 2009.

33. **Youakim Badr, Richard Chbeir, Ajith Abraham, Aboul-Ella Hassanien.** *Emergent Web Intelligence: Advanced Semantic Technologies.* 1st. s.l. : Springer, 2010.

34. **A. C. Dias Neto, Rajesh Subramanyan, Marlon Vieirab.** *Characterization of Model-based Software Testing Approaches.* Brazil : s.n., 2007. Siemens Corporate Research, Technical Report at PESC/COPPE/UFRJ.

35. *Automated Functional Conformance Test Generation for Semantic Web Services.* **Amit M. Paradkar, Avik Sinha, Clay Williams.** [ed.] IBM Software Group Toronto. 2007. IEEE International Conference on Web Services.

36. *Model-Based Testing of Reactive Systems.* **Broy, M.** s.l. : Berlin Heidelberg, 2005, Springer-Verlag, págs. 391-438.

37. *A Model-Based Approach for Testing the Performance of Web Applications.* **Mahnaz Shams, Diwakar Krishnamurthy, Behrouz Far.** Canada : s.n., 2006. Third International Workshop on Software Quality Assurance (SOQUA'06), Department of Electrical and Computer Engineering, University of Calgary.

38. *A Survey on Model-based Testing Approaches: A Systematic Review.* **Arilo C. Dias Neto, Rajesh Subramanyan, Marlon Vieira.** Atlanta Georgia, USA : s.n., 2007. WEASELTech'07.

39. *Supporting the Selection of Model-based Testing Approaches for Software Projects.* **Arilo C. Dias Neto, Guilherme Horta Travassos.** 2008, ACM.

40. **Zander-Nowicka, Justyna.** Model-based Testing of Real-Time Embedded Systems in the Automotive Domain. *Tesis de Faculty IV – Electrical Engineering and Computer Science Technical University Berlin.* 2009.

41. **B., Beizer.** *Software Testing Techniques.* 2nd. 1990.

42. *Surveying Model Based Testing Approaches Characterization Attributes.* **Arilo C. Dias Neto, Guilherme H. Travassos.** Kaiserslautern, Germany : s.n., 2008, ACM.

43. *Obstacles and opportunities for model-based testing in an industrial software environment.* **Robinson, H.** 2003. 1st European Conference on Model Driven Software Engineering. págs. 118-127.

44. *An object-oriented Web test model for testing Web applications.* **D. Kung, C. H. Liu, and P. Hsia.** Taipei, Taiwan : s.n., 2000. IEEE 24th Annual International Computer Software and Applications Conference (COMP-SAC2000). págs. 537-542.

45. **Muhammad Shafique, Yvan Labiche.** *A Systematic Review of Model Based Testing Tool Support.* Software Quality Engineering Laboratory, Department of Systems and Computer Engineering, Carleton University. Canada : s.n., 2010. Technical Report SCE-10-04.

46. **M. Utting, A. Pretschner, B. Legeard.** *A taxonomy of model-based testing.* Computer Science, The Universiy of Waikato. New Zealand : s.n., 2006. Technical Report 04/2006.

47. **Dai, Z. R.** An Approach to Model-Driven Testing with UML 2.0, U2TP and TTCN-3. *PhD thesis, Technical University Berlin, ISBN: 978-3-8167-7237-8. Fraunhofer IRB Verlag.* 2006. ISBN: 978-3-8167-7237-8.

48. **Group, Software Engineering.** *Guidelines for Performing Systematic Literature Reviews in Software Engineering Version 2.3.* School of Computer Science and Mathematics, Keel University and Department of Computer Science, University of Durham. United Kingdom : s.n., 2007. EBSE Technical Report. EBSE-2007-01.

49. **Esteban Robles Luna, José Ignacio Panach, Julián Grigera.** *Incorporating Usability Requirements in a Test/Modeldriven Web Engineering Approach.* Centro de Investigación en Métodos de Producción de Software, Universidad Politécnica de Valencia. España : s.n., 2010.

50. **N., Escalona M.J. & Koch.** Ingeniería de Requisitos en Aplicaciones para la Web: Un estudio comparativo. [En línea] 2002. http://lsiweb.lsi.us.es/docs/informes/LSI-2002-4.pdf.

51. **Neukirchen, H. W.** Languages, Tools and Patterns for the Specification of Distributed Real-Time Tests. *Georg-August-Universiät zu Göttingen, http://webdoc.sub.gwdg.de/diss/2004/ neukirchen/index.html.* 2004. PhD thesis.

52. *Model based testing in practice at Microsoft.* **Stobie, K.** s.l. : Elsevier Electronic Notes in Theoretical Computer Science, 2005. Workshop on Model Based Testing. Vol. 111.

53. **Achkar, Hani.** Model Based Testing Of Web Applications. [En línea] http://testoptimal.com/ref/ModelBasedTestingOfWebApplications.pdf.

54. **Ahmad Saifan, Juergen Dingel.** *Model-Based Testing of Distributed Systems.* School of Computing Queen's, University Kingston. Ontario, Canada : s.n., 2008. Technical Report 2008-548.

55. *Model-Based Testing of Web Applications using NModel.* **Juhan Ernits, Rivo Roo, Jonathan Jacky.** [ed.] ACM. 2009. TESTCOM '09/FATES '09 Proceedings of the 21st IFIP WG 6.1 International Conference on Testing of Software and Communication Systems and 9th International FATES Workshop.

56. *Model-based testing: Property checking for real.* **Tretmans, J.** http://www-sop.inria.fr/everest/events/cassis04 : s.n., 2004. International Workshop for Construction and Analysis of Safe Secure, and Interoperable Smart Devices.

57. *Perspectives of Model-Based Testing.* **Ed Brinksma, Wolfgang Grieskamp, and Jan Tretmans.** IBFI, Schloss Dagstuhl, Germany : s.n., 2005. Dagstuhl Seminar Proceedings. Vol. 04371, págs. 5-10.

58. **Hower, Rick.** Software QA and Testing Resource Center. [En línea] Web site test tools and site management tools, 2002. www.softwareqatest.com/qatweb1.html.

59. *Software Testing Research and Practice.* **A., Bertolino.** 2003, Springer Link.

60. **Krishnamurthy, D.** Synthetic Workload Generation for Stress Testing Session-Based Systems. *PhD. Thesis, Department of Systems and Computer Engineering, Carleton University.* Ottawa, Canada : s.n., 2004.

61. **Ciolli, María E.** Testing de migración de aplicaciones distribuidas a entornos Web. *Tesis de la Universidad Nacional de la Plata.* 2007.

62. *Towards Model-Based Testing of Web Services.* **Frantzen, Lars.** Instituto di Scienza e Tecnologie della Informazione “Alessandro Faedo” Consiglio Nazionale delle Ricerche, Pisa – Italy : s.n., 2006. International Workshop on Web Services Modeling and Testing.

63. **Pinheiro da Silva P., Paton N.** *User Interface Modelling with UML.* University of Manchester. Manchester : s.n., 2000.

64. **Nadia Alshahwan, Mark Harman, Alessandro Marchetto.** *ImprovingWeb Application Testing Using Testability Measures.* Kings College London–CREST Centre,Strand, London, UK : s.n., [En línea] 2009. http://www.cs.ucl.ac.uk/staff/mharman/wse09.pdf.

65. **Ibrahim K. El-Far**. *Enjoying the perks of model-based testing*. In Proceedings of the Software Testing, Analysis, and Review Conference (STARWEST 2001), 2001.

7. Listado de siglas, abreviaturas y acrónimos.

ACM: Association for Computing Machinery
CSP: Communicating Sequential Processes
CSS: Calculus of Communicating Systems
ER: Entity-Relationship
FSM: Finite States Machine
Hil: Hardware-in-the-Loop
HTML: HyperText Markup Language
HTTP: Hypertext Transfer Protocol
IOPEs: Inputs, Outputs, Preconditions and Effects
JM: Java Modeling Language
JSP: Java Server Pages
MBT: Model Based Testing
Mil: Model-in-the-Loop
PiL: Processor-in-the-Loop
SDL: Specification and Description Language
SiL: Software-in-the-Loop
SOA: Service-Oriented Architecture
SOAP: Simple Object Access Protocol
STS: Symbolic Transition Systems
SUT: System Under Testing
TMT: Test Model Toolkit
UML: Unified Modeling Language
UTP: UML Testing Profile
VDM: Vienna Development Method
WSDL: Web Service Definition Language

Apéndice A – Listado de los Trabajos Identificados

Título	Autores	Tipo de Trabajo	Año de Publicación	Resumen	Tipo de Prueba	Tipo de Contenido	Dominio del Software	Relacionado con MBT
Testing Web Applications by Modeling with FSMs	Anneliese A. Andrews, Jeff Offutt, Roger T. Alexander	Artículo	2005	Propone una técnica para pruebas a nivel de sistemas que combina la generación de pruebas basadas en máquinas de estados finitos con restricciones. Usa el enfoque para modelar aplicaciones web.	Pruebas Funcionales	Enfoque	Aplicaciones Web	Sí
Towards Model-Based Testing of Web Services	Lars Frantzen, Jan Tretmans, René de Vries	Artículo	2006	Analiza sobre la ejecución de un ejemplo cómo los protocolos podrían servir como entradas para Model Based Testing de servicios web. Propone usar Sistemas de Transisición Simbólica y teorías base de modelados para modelar y probar la coordinación. Además identifica lagunas en teorías y técnicas a fin de proponer temas de investigación.	Pruebas Funcionales	Enfoque	Servicios Web	Sí
A model-based approach for testing the performance of web applications	Mahnaz Shams, Diwakar Krishnamurthy, Behrouz Far	Artículo	2006	Propone un enfoque para model based testing que prueba el desempeño de las aplicaciones web usando un modelo de aplicación que captura dependencias para un sistema basado en la web.	Pruebas Funcionales	Enfoque	Aplicaciones Web	Sí
Model-based functional conformance testing of web services operating on persistent data	Avik Sinha, Amit Paradkar	Artículo	2006	Propone un enfoque MBT para generación de pruebas funcionales para servicios web los cuales funcionan con la presencia de datos persitentes. Propone el uso de la técnica para generación de pruebas basada en Extended Finite State Machine (EFSM).	Pruebas Estructurales	Enfoque	Servicios Web	Sí
Testing de migración de aplicaciones distribuidas a entornos Web	María Elena Ciolli	Tesis	2007	Formula una metodología para el análisis lógico y físico de las aplicaciones distribuidas a migrar a entornos Web, y se la pone en práctica aplicándola a un caso de estudio. Este caso corresponde a un sistema distribuido desarrollado mediante el uso de una metodología de análisis y diseño estructurado y la aplicación migrada a la Web fue desarrollada mediante el uso de alguna metodología basada en UML.	Pruebas Funcionales	Enfoque	Aplicaciones Web	Sí
WebServiceTestFramework WithTTCN-3	Stefan Troschütz	Tesis	2007	Presenta un framework para las pruebas se servicios web con la especificación de pruebas estandarizada y el lenguaje de omplementación TTCN-3. Además el mapeo de la descripción de los servicios web a bancos de pruebas abstractas TTCN-3.	Pruebas Estructurales	Otro	Servicios Web	Sí
Automated Functional Conformance Test Generation for SemanticWeb Services	Amit M. Paradkar†, Avik Sinha, Clay Williams	Artículo	2007	Presenta un enfoque automatizado para generar pruebas funcionales conforme a la semántica de los servicios web. Para cada servicio web el enfoque produce los objetivos de las pruebas los cuales son refinados con las pre-condiciones de los servicios web usadas como un conjunto de modelos para fallas.	Pruebas Funcionales	Enfoque	Servicios Web	Sí
Toward Automated WSDL based Testing of Web Services	Cesare Bartolini, Antonia Bertolino, Eda Marchetti, Andrea Polini	Artículo	2008	Introduce una propuesta para automatizar las pruebas basadas en WSDL, las cuales combinan la cobertura de las operaciones con la generación de casos de prueba dirigida por datos.	Otro	Otro	Servicios Web	No
Generating security tests in addition to functional tests	Jacques Julliand, Pierre-Alain Masson, Régis Tissot	Artículo	2008	Trata sobre la generación de las pruebas de seguridad, además de las pruebas funcionales generadas previamente por un enfoque de evaluación basado en modelo.	Otro	Enfoque	Smart Card Platform	Sí
LTS-BT a tool to generate and select functional test cases for embedded systems	Emanuela G. Cartaxo, Wilkerson L. Andrade	Artículo	2008	Presenta la herramienta LTS-BT para sistemas embedidos enfocándose en la selección de notaciones para especificaciones de comportamiento para generación y selección de casos de prueba.	Pruebas Funcionales	Herramienta	Sistemas Embedidos	Sí
Model Based Testing of Web Applications using NModel	Juhan Ernits, Rivo Roo, Jonathan Jacky, and Margus Veanes	Artículo	2009	Muestra cómo model based testing puede ser aplicado en el contexto de las aplicaciones web usando el toolkit Nmodel. El principal objetivo es generar pruebas funcionales correctas.	Pruebas Funcionales	Enfoque	Aplicaciones Web	Sí
Improving Web Application Testing Using Testability Measures	Nadia Alshahwan, Mark Harman, Alessandro Marchetto	Artículo	2009	Propone un framework para un conjunto de métricas de prueba durante el proceso de automatización de las pruebas. Muestra un reporte sobre la implementación de un prototipo de herramienta para pruebas de aplicaciones web.	Pruebas Funcionales	Enfoque	Aplicaciones Web	Sí
Model Based Functional Testing using Pattern Directed Filmstrips	Tony Clark	Artículo	2009	Describe un enfoque que permite que muchos patrone OCL sean expresandos como patrones de foto que corresponden directamente con los diagramas de modelo de información. El comportamiento es construido como una cadena de fotos.	Pruebas Funcionales	Enfoque	No definido	Sí
MBT Next Generation Functional Software Testing	Bruno Legeard	Manual	2010	Da una visión realista de model based testing y sus beneficios esperados. Discute lo que es model based testing y cómo se debe organizar los procesos y el equipo para usar model based testing, además detalla ejemplo de requerimiento de negocio para automatizar el repositorio de pruebas ysando un completo soporte de procesos model based testing.	No definido	Otro	No definido	Sí
Model Based Testing Of Web Applications	Hani Achkar	Manual	2010	Expone las características de model based testing, su proceso, y modela el comportamiento de una aplicación web en términos de sus estados y sus acciones que varían. Usa la máquina de estados finito.	Pruebas Funcionales	Enfoque	Aplicaciones Web	Sí
Modeling and Testing of Web-Based Systems	Ana Cavalli, Mounir Lallali, Stephane Maag	Artículo	2010	Presenta dos enfoques para pruebas de sistemas basado en la web. Se enfoca en composición de servicios web y aplicaciones web. Propone los pasos para una metodología que consiste en describir formalmente el sistema que responde a los requerimientos de información; luego a partir del modelo desarrolla métodos para generar los casos de prueba y finalmente ejecuta las pruebas en una implementación real.	Pruebas Estructurales	Enfoque	Aplicaciones Web	Sí
Automated functionality testing through GUIs	Duc Hoai Nguyen, Paul Strooper, Jörn Guy Süß	Artículo	2010	Propone varios criterios de cobertura basado en enlaces entre acciones abstractas y secuencia de eventos.	Pruebas Funcionales	Enfoque	No definido	Sí