UNIVERSIDAD DE LA REPÚBLICA
Facultad de Ingeniería - Instituto de Computación

Reestructura del Testing Funcional en ASSE e implantación de Automatización de Pruebas Funcionales

Informe de Proyecto de Grado presentado al Tribunal Evaluador como requisito de graduación de la carrera Ingeniería en Computación

Michel Camarotta **Christian Pla** **Rodolfo Verocai**

Tutores

Ariel Sabiguero & Mónica Wodzislawski

TRIBUNAL

Carlos Martínez
Leandro Scasso
Jorge Triñanes

Montevideo, Mayo de 2012

Resumen

La Administración de los Servicios de Salud del Estado (ASSE) cuenta con una variedad de sistemas de software que se corresponden con diversos tipos de dominios, con diferentes niveles de complejidad. Para el desarrollo de los sistemas, ASSE involucra a varios proveedores y funcionarios, generando un área de sistemas muy heterogénea mayoritariamente residiendo en la organización.

Las necesidad de verificación y validación de los productos sobrepasa la capacidad de los recursos disponibles del área de testing. Consecuentemente gran parte de los desarrollos no pasan por testing y a lo sumo son validados por los usuarios funcionales expertos en el dominio.

Existe en ASSE una preocupación por la mejora de la calidad de sus sistemas y procesos, planteando particular interés en la automatización de pruebas.

En el contexto del presente proyecto de grado, se estudia la organización y el estado del arte en testing funcional, para generar una propuesta metodológica flexible, y que sea practicable en los proyectos de desarrollo de ASSE. La metodología se pone en práctica en un proyecto piloto para observarla y extraer conclusiones, simulando una reestructura del área de testing de ASSE.

Como trabajos anexos importantes, se presenta un informe de análisis y diagnóstico inicial sobre el estado de la organización frente al testing funcional; y un apartado que supone contextos organizacionales y propone estrategias de cómo utilizar los procesos propuestos.

Palabras Claves
Testing, Testing Funcional, Sistemas de Software, Automatización de Pruebas, Procesos de Testing Funcional, Herramientas de Automatización.

Agradecimientos

Deseamos expresar nuestro agradecimiento a todos aquellos que directa o indirectamente estuvieron involucrados en el desarrollo de este proyecto de grado.

Agradecemos al personal de ASSE. Por la dirección de ASSE agradecemos a Paola Suárez el apoyo brindado. A Carlos Gutiérrez y Fernando Pereira, integrantes del área de testing de ASSE, quienes colaboraron activamente en el transcurso del proyecto, brindando su mejor disposición. A Marcelo Lemos y Ariel Sabiguero, quienes colaboraron con la configuración de entornos y herramientas necesarias. A los diferentes desarrolladores del área de sistemas, tanto los funcionarios como los pertenecientes a empresas residiendo en ASSE.

Agradecemos también a nuestros compañeros y amigos del Centro de Ensayos de Software, quienes siempre se pusieron a las órdenes para colaborar con nosotros. En especial agradecer a nuestro gran amigo y compañero Federico Toledo, por su constante preocupación, invaluables sugerencias y revisiones de este trabajo.

A la Universidad de la República, más precisamente a la Facultad de Ingeniería y el Instituto de Computación, por todo el arduo trabajo que desemboca en nuestra formación en esta carrera por la cuál hemos generado vocación.

A nuestros tutores Ariel Sabiguero y Mónica Wodzislawski, quienes nos apoyaron y guiaron en el transcurso del proyecto, siempre respetando y valorando nuestras preferencias e ideas a la hora de plasmar el conocimiento.

A nuestras familias y amigos, quienes nos apoyaron y entendieron en las diferentes etapas de este proyecto de grado, y en el transcurso en sí de toda nuestra formación académica. En particular a Gustavo, Mary, Florencia, Glací, Fabian, “Los Larvas”, Silvia, Mendex y los diferentes integrantes de “ACME Labs”.

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

En este capítulo se introduce y motiva el trabajo, se plantean objetivos y se describe el enfoque adoptado. Se hace además mención a los aportes del trabajo, los términos utilizados, y la forma en que esta organizado el documento.

1.1. Contexto y motivación

La Administración de Servicios de Salud del Estado (en adelante ASSE), basa su práctica del testing de software en expertos funcionales que ejecutan pruebas manuales y de forma *ad hoc*, ante solicitudes de cambio, nuevas funcionalidades y reparación de incidentes. El conocimiento adquirido en el proceso de verificación se pierde cuando el equipo se encarga del testing de nuevas tareas.

Inicialmente ASSE manifiesta el interés de mejorar la metodología de testing funcional e incorporar el uso de herramientas de automatización de pruebas funcionales.

Los estudiantes que integramos este proyecto de grado, hemos trabajado en diversos proyectos dentro del Centro de Ensayos de Software (CES) [CES10], en diferentes actividades dentro del testing y por lo tanto consideramos apropiado abordar esta propuesta de proyecto de grado. Muchos de los conceptos presentados fueron madurados y desarrollados en varios contextos de trabajo en la industria.

1.2. Objetivos

Entre los objetivos fundamentales del proyecto se encuentran:

- Estudio del estado del arte de los procesos de testing y modelos de mejora de procesos de testing.
- Estudio del estado del arte en automatización de testing funcional.
- Investigación de herramientas para apoyar la automatización de testing funcional acordes a la plataforma de ASSE.
- Propuesta metodológica en base a la situación inicial de ASSE
- Seguimiento de la metodología propuesta para incorporar testing manual y automatizado en un proyecto piloto.

El proyecto busca sentar las bases tecnológicas para contar con una ejecución automática de test de conformidad de componentes centrales de la plataforma de ASSE.

Uno de los objetivos centrales del proyecto, es prototipar testing automatizado en alguno de los sistemas relevantes de la organización.

Resultados alcanzados:

- Estudio del estado del arte en testing funcional.
- Propuesta de la nueva metodología de trabajo.
- Implementación de la propuesta de trabajo en un proyecto piloto.
- Informe de análisis y diagnostico inicial del estado de la organización frente al testing, incluyendo sugerencias de acciones a tomar.
- Selección de herramienta e implementación de casos de prueba testigos.
- Análisis de características de desarrollo que simplifiquen el testing.

1.3. Enfoque del Trabajo

Este trabajo es una investigación general en el área del testing de software. Está nutrido de diversas fuentes que van desde los libros clásicos en el área, hasta los trabajos y tendencias más actuales en cuanto a procesos, metodologías, mejoras de proceso, buenas prácticas y otras producciones[1] de autores relevantes. La meta perseguida es la aplicación del marco de trabajo propuesto partiendo de un contexto real inicial de la organización, y transitar hacia un

[1] Estas producciones referidas, son las que están presentes en medios digitales como lo son blogs y foros.

estado donde es posible implantar testing funcional manual y automatizado.

La propuesta metodológica deberá entenderse en el contexto de la realidad de la organización de estudio (ASSE), pero se busca que sea lo suficiente flexible como para que aplique a otras realidades. Dejamos abierta la posibilidad de mejorar la metodología, y trabajar a futuro para lograr otros tipos de testing, mejorando el proceso de testing de la organización.

1.4. Aportes del Trabajo

Entre los resultados que consideramos aportes, queremos destacar el enfoque adoptado, el marco de trabajo elaborado (al que llamaremos *framework instanciable de testing,* o simplemente por su sigla *FIT*) y el valor agregado de entrenamiento en testing de los integrantes de ASSE.

La elaboración del marco de trabajo apunta a construir un producto flexible, adaptable, y de uso práctico, poniendo el foco en las particularidades del contexto. En el estudio del estado del arte, además de las fuentes básicas, se investigan las nuevas tendencias del testing relacionadas con el manejo de realidades cambiantes.

El framework definido organiza los conceptos que formarán parte de la configuración o *instanciación* particular para cada contexto. La flexibilidad necesaria para adaptarse a un proyecto dado se encuentra en la posibilidad de marcar un camino entre las etapas y actividades de la propuesta, incluso incorporando prácticas avanzadas de testing de software y automatización de pruebas.

Seguir las actividades propuestas en el contexto de un proyecto, colabora al entrenamiento de los integrantes del área de testing de la organización, en la disciplina del testing. Este marco metodológico esta diseñado para que apoye la maduración del área de calidad de ASSE en torno al testing funcional y consecuentemente sus integrantes se encuentren en mejores condiciones de seguir un proceso de testing más completo, como lo es por ejemplo ProTest [Pér06].

La propuesta es implementada en ASSE en un proyecto piloto, comprobando su aplicabilidad en un contexto de la industria nacional.

1.5. Precisión de Algunos Términos Presentes en el Documento

En este apartado, pretendemos justificar el uso de ciertos términos, que pueden necesitar aclaración en cuanto al significado en este trabajo, y su uso se deriva de la jerga cotidiana en la temática tratada por este informe. Muchos de estos términos, no corresponden al idioma español, o no están reconocidos oficialmente por la Real Academia Española[2], por lo tanto es conveniente precisar la intención y significado al utilizarlos en este trabajo. La definición de cada término debe ser entendida en el contexto de este trabajo, esto es, qué significa (para los autores de este informe) el término al ser utilizado, más allá de múltiples otras acepciones que pueda tener. Estos y otros términos están especificados además en el glosario, al final de este documento.

Como en otras disciplinas en el área de informática, muchos términos son extraídos del idioma inglés, y además, algunos no poseen una traducción biunívoca en el idioma español, cayendo muchas veces en ambigüedades, o perdiendo significado al traducirse. Este es el caso del término *testing*, que proviene del verbo en inglés *to-test* que se puede traducir como *probar*. *Probar*, en idioma español, tiene un significado ambiguo, y puede entenderse como *demostrar*. A menos que se haga una demostración formal, o una prueba exhaustiva, no podemos *demostrar* que el software no contiene defectos. Dada la complejidad del software, es poco probable que se pueda realizar una prueba formal, y las pruebas exhaustivas son prácticamente imposibles[3]. La Real Academia Española define *test* como una *"Prueba destinada a evaluar conocimientos o aptitudes, en la cual hay que elegir la respuesta correcta entre varias opciones previamente fijadas"* y como *"Prueba psicológica para estudiar alguna función"*; en el presente trabajo no se utilizará el término *test* de ninguna de estas dos formas.

El significado de *testing* se manejará en el documento para referirse a la actividad, o el conjunto de actividades involucradas en ensayar diferentes componentes de un sistema con la intención de encontrar defectos [Mye04a]. Se encontrará en el documento también acompañado del artículo "el" para referirse también a lo mencionado como *testing*.

De todas formas se utilizará en ocasiones el término *prueba*, indistintamente para referirse a la acción de probar, o a un *caso de prueba*; y nunca se utilizará en su acepción de *demostrar*. Para referirnos a un caso de prueba se podrá ver el uso del termino *"test"*, o *"tests"* en plural. En plural, se encontrará en el documento también el término *pruebas*, para referirse a un determinado conjunto de casos de prueba, y en ocasiones para referirnos a algún subconjunto de actividades durante el transcurso del proyecto de verificación (por ejemplo, podría usarse "durante las pruebas..." para hablar de actividades que ocurren en el proyec-

[2] Ver http://rae.es.
[3] Discusión en la sección 2.7.1

to de testing). A las *pruebas*, también se les tratará como *"testing"* o *"el testing"*.

Para referirnos a la persona que toma algunos de los roles específicos de las actividades de *testing*, utilizaremos *tester*, y *testers*, cuando es en plural. De lo contrario, tendríamos que referirnos a estas personas como "probadores" o quizá "verificadores y validadores", y no son, a nuestro entender, los términos más comunes en las organizaciones actualmente.

1.6. Estructura del documento

El documento esta organizado en capítulos y apéndices. Al inicio encontrara un resumen del trabajo, y a continuación, el capítulo 1, que es la introducción al trabajo y a este informe. Luego en el capítulo 2, se encuentra el estado del arte en testing funcional manual y automatizado, el cuál trata desde temas más generales, como modelos de madurez de procesos de testing, hasta temas más específicos en amplitud, como ser procesos de testing, estrategias, y otros. El informe prosigue en el capítulo 3, que es una reseña del estudio de la organización a modo de diagnóstico inicial de ASSE frente al testing funcional (presentando además algunas recomendaciones). En el capítulo 4 se encuentra la propuesta metodológica y continúa con la documentación de la experiencia piloto llevada adelante en ASSE en el capítulo 5. El último capítulo es el 6 que resume el documento y discute alternativas de trabajo a futuro.

Como anexos al informe se encuentran el apéndice A con comentarios acerca de las características deseables de una herramienta de seguimiento de incidentes, y el apéndice B con ejemplos de instrumentación de la metodología en diferentes supuestos de contextos organizacionales. Al final del documento se encuentran el glosario, conteniendo términos relevantes al trabajo, y un apartado con referencias bibliográficas.

Capítulo 2

Estado del arte

"Lo que sabemos es una gota de agua; lo que ignoramos es el océano"

Isaac Newton

Este capítulo presenta el estudio de los temas del testing de software que consideramos relevantes a la elaboración de una propuesta acorde a los requisitos del proyecto. Se define un conjunto de conceptos relevantes para el contexto bajo estudio, y a partir de allí, se selecciona, en base a nuestra experiencia, el material más apropiado de las fuentes para cada concepto. Los temas tratados en las primeras secciones son amplios y conforme avanza el capítulo, se abarcan temas más específicos; ya sobre el final, se abordan los temas más transversales.

En la sección 2.1 presentamos una reseña sobre los más relevantes modelos de mejora de procesos de testing de software. Luego en la sección 2.2 indagamos en procesos de testing de software. Proseguimos adentrándonos en temas más particulares en la sección 2.3, donde se estudian algunos modelos de incorporación temprana del testing en el proceso de desarrollo de software. La sección 2.4 trata de las estrategias del testing funcional, destacando los enfoques más importantes; mientras que la sección 2.5 cita los puntos más importantes en cuanto a la automatización de testing. Para finalizar, presentamos un informe sobre testeabilidad en la sección 2.6 y una discusión en la sección 2.7 contrastando el testing manual con el automatizado.

2.1. Modelos de Mejora de Procesos de Testing

Los temas tratados en este capítulo se abordan con un enfoque descendente, por lo tanto iniciamos con una reseña a los modelos de mejora de procesos de testing, por ser el tema más amplio.

Un modelo de mejora de procesos, provee de cierta categorización y descripción de puntos clave sobre los cuales se debe trabajar para conseguir la mejora en la manera que recomienda el modelo. Generalmente están diseñados para que una autoridad certificadora pueda evaluar si una organización cumple con los puntos marcados por el modelo y a qué nivel, para posteriormente considerar a la organización en cierto nivel o determinada etapa según el modelo y otorgarle una certificación. Estos modelos son populares en al disciplina de desarrollo de software, y como consecuencia del avance del área del *testing de software*, surgen entonces los modelos para mejoras de procesos de testing.

2.1.1. TMMi: Testing Maturity Model Integration

El TMMi [Fou11] es un modelo detallado de mejora del proceso de testing, que nace como complemento del CMMi [Kne08] (CMMi es usado muchas veces en la industria como modelo estándar de mejora de los procesos de desarrollo de software). TMMi define una jerarquía de cinco niveles (al igual que CMMi), que van desde una etapa con prácticas básicas y no gestionadas de testing, hasta una etapa con testing gestionado, definido, medido y optimizado.

Los niveles definidos por TMMi y sus respectivas áreas de proceso son:

- Nivel 1: Inicial
- Nivel 2: Gestionado
  - Política de Testing y Estrategia
  - Plan de Testing
  - Seguimiento y Control de Testing
  - Diseño y Ejecución de Pruebas
  - Ambiente de Testing
- Nivel 3: Definido
  - Organización de las Pruebas
  - Programa de entrenamiento
  - Ciclo de vida del Testing e Integración
  - Testing no-Funcional
  - Revisión por pares
- Nivel 4: Medido

- Métricas de testing
- Evaluación de la calidad del software
- Revisión por pares avanzada

- Nivel 5: Optimizado
  - Prevención de defectos
  - Optimización del proceso de testing
  - Control de calidad

Nivel 1: Inicial

En este nivel, el testing es caótico, no existen procesos definidos y no se jerarquiza la actividad de testing (por ejemplo, las organizaciones en este nivel, suelen confundir testing con depuración de código). Los casos de éxito a esta altura suelen asociarse a actos notables de integrantes de la organización y no al resultado de la aplicación de un proceso, por lo cual, la situación de éxito no es repetible.

La aspiración de calidad más común en las organizaciones en nivel inicial de TMMi es lograr liberar productos que puedan funcionar sin errores de gravedad alta. Generalmente los productos, no cubren las expectativas de calidad ya sea en rendimiento, estabilidad, disponibilidad, pobreza y ausencia de funcionalidades, nivel de incidentes remanentes, entre otros.

Es común que las organizaciones en este nivel, no dediquen recursos a crear departamentos de pruebas, entornos o ambientes dedicados a pruebas, así como herramientas, o personal capacitado (incluida la falta de entrenamiento).

Nivel 2: Gestionado

Para el nivel dos de TMMi, el testing es un proceso gestionado y diferenciado de las actividades de depuración de código. En las organizaciones que adoptan la disciplina del nivel dos, se suele ver que se mantienen las prácticas aún en momentos de presión y estrés. De todas formas, el testing sigue siendo percibido por los interesados (o *"stakeholders"*) como una etapa posterior a la implementación.

Se cuenta con requerimientos escritos y se construyen planes de testing describiendo las actividades de testing, cómo serán llevadas a cabo y por quién. El plan de testing incluye el enfoque adoptado y análisis de riesgo. Es común utilizar técnicas para la gestión del riesgo, diseño y selección de casos de prueba en base a los requerimientos documentados. El testing se aplica a varias etapas, como ser pruebas de integración, de sistemas, de componentes y aceptación. Las actividades de control y seguimiento del testing están presentes en este nivel,

pudiéndose aplicar medidas correctivas al enfrentar desviaciones.

El objetivo del testing es verificar que el producto cumple con sus requerimientos, por lo cual, es común que los defectos en los requerimientos se propaguen al sistema, a medida que el testing se incluye en etapas tardías en el ciclo de vida del desarrollo.

Nivel 3: Definido

En el nivel tres de TMMi, el testing deja de ser considerado como una etapa posterior a la implementación y pasa a estar integrado totalmente en el proceso de desarrollo.

Existe planificación temprana de testing, y se considera la profesión de Tester. Las revisiones formales son valoradas, y los testers muchas veces son incluidos en la especificación de los requerimientos, permitiendo que se puedan prestar atención además a requerimientos no funcionales como la usabilidad, confiabilidad, entre otros.

El nivel tres de TMMi es una refinación de las áreas de proceso del nivel dos, donde además los procesos, conjunto de estándares de testing, y procedimientos son definidos con mayor rigurosidad. Aplicar áreas de proceso del nivel tres a un proyecto particular, involucra tomar del conjunto de estándares de la organización los que mejor apliquen al contexto dado.

Nivel 4: Medido

El nivel cuatro implica que el proceso de testing es cuidadosamente definido, bien fundado y medible. Se considera al testing como una evaluación, concerniente a todas las actividades del desarrollo del producto y sistemas en funcionamiento relacionados.

Se incorporan métricas para controlar y evaluar el rendimiento del proceso de testing. La calidad del producto es definida en términos de atributos medibles cuantitativamente, identificando las necesidades de calidad y las métricas a utilizar.

Las revisiones e inspecciones cobran importancia y son consideradas como parte del proceso de testing, sobre todo las revisiones por pares. Además, se utilizan para medir la calidad del producto, con criterios cuantitativos de aspectos como confiabilidad, usabilidad y mantenibilidad; así como también como apoyo a la construcción de los planes de testing, establecimiento de la estrategia y enfoque del testing.

Nivel 5: Optimizado

El nivel cinco de TMMi tiene como objetivos la prevención de defectos y la mejora continua. Consta de tres áreas de proceso fuertemente relacionadas: control de calidad, prevención de defectos, y mejora del proceso de testing.

Al alcanzar todos los niveles previos, la organización se supone que cuenta con un proceso de testing, medible y definido, y con una sólida infraestructura organizacional que apoya al testing. La organización es capaz de mejorar su proceso de testing, incorporando mejoras incrementales en tecnología, innovación, nuevas herramientas, y metodologías. El foco está en el ajuste continuo y mejora del proceso con apoyo de herramientas, evaluando la reutilización de la documentación generadas en las pruebas, y derivando nuevos ajustes al proceso.

La prevención de defectos juega un papel fundamental identificando y analizando causas comunes, para establecer acciones que prevengan defectos similares que puedan ocurrir. Parte de la prevención de defectos también es el análisis de algunos resultados del proceso de control de calidad, evaluando el rendimiento del proceso y encaminando acciones correctivas para el futuro. El proceso de testing es gestionado de forma estática por medio del área de proceso de control de calidad, mediante mediciones cuantitativas de aspectos como nivel de confianza y confiabilidad.

En el nivel 5 de TMMi se espera que el proceso de testing:

- Sea gestionado, definido, medido, eficiente y efectivo.
- Sea controlado estáticamente y predecible.
- Tenga foco en prevención de defectos.
- Sea apoyado por automatización siempre y cuando se de un uso efectivo de los recursos.
- Contribuya a la transferencia de tecnología desde la industria a la organización.
- Facilite la reutilización del testing.
- Esté enfocado en cambios de proceso para lograr la mejora continua.

Resumen

TMMi es consecuencia del seguimiento de las prácticas de CMMi y como éste, cuenta con niveles de madurez. Las recomendaciones de los niveles de TMMi son amplias (desde planificación, diseño y ejecución de pruebas; hasta revisiones y prevención de defectos) y son referidas en diferentes procesos. Un posible contexto de aplicación de TMMi, sería una empresa que esté certificada en algún nivel de CMMi y se interese por mejorar sus procesos de testing. Esto

puede resultar muy costoso para pequeñas empresas que quieran incorporar o mejorar sus actividades de testing, y por lo tanto no es el caso más común.

2.1.2. TPI: Test Process Improvement

El modelo de mejora de procesos de testing, TPI [Sog09] es derivado de la experiencia de la aplicación del proceso de testing TMap [vdABR+08] de la empresa SOGETI, definiendo pasos de mejora incrementales y controlables. Abarca diferentes aspectos del proceso de testing desde el uso de herramientas, técnicas de diseño de casos de prueba así como aspectos de comunicación y gestión, entre otros. A estos aspectos se les denomina áreas clave.

Las áreas clave

TPI se basa en 20 áreas clave, y diferentes niveles para cada área. Se pasa de un nivel A a un nivel B, de un B a un C, a mayor eficiencia y efectividad en esa área. El nivel superior representa que se alcanzó la madurez requerida para el nivel actual y todos los niveles inferiores. Para cada área clave se establecen metas o *"checkpoints"* para pasar de nivel.

Las 20 áreas clave[1]:

- Test strategy
- Life-cycle model
- Moment of involvement
- Estimating and planning
- Test specification techniques
- Static test techniques
- Metrics
- Test automation
- Test environment
- Office environment
- Commitment and motivation
- Testing functions and training
- Scope of methodology

[1] No se traducen aquí los nombres de las áreas clave porque algunos términos pierden su verdadera semántica.

- Communication
- Reporting
- Defect management
- Testware management
- Test process management
- Evaluation
- Low-level testing

Resumen

TPI es construido como modelo de mejora inspirado en el proceso estructurado TMap. Es interesante destacar que entre las áreas clave existen muchos aspectos de los procesos de testing y desde ese punto de vista, se pueden ver como una enumeración de los elementos comúnmente presentes en los procesos. Además, cuenta con áreas claves que refieren a aspectos no siempre contemplados en los procesos como ser testing automatizado (*Test automation*), compromiso y motivación (*Commitment and motivation*), o capacitación (*Testing functions and training*).

2.2. Procesos de Testing Funcional

Continuando con el enfoque descendente, tratamos a continuación los procesos de testing funcional.

Un proceso de testing **funcional** se refiere a que el testing del producto será desde el punto de vista de las funcionalidades, acorde con sus requerimientos, en lugar de medir su rendimiento u otras cualidades para-funcionales.

De los procesos existentes, se presentan a continuación dos ejemplos que contrastan bastante. Uno de ellos es un proceso precisamente definido en etapas, y el otro es una tendencia que surge en a partir de la adopción de metodologías ágiles en el desarrollo de software.

2.2.1. ProTest

ProTest [Pér06] es un proceso de testing **funcional independiente**, **orientado a ciclos de prueba** y **basado en análisis del riesgo** del producto.

La **independencia** indica que es posible aplicar y adaptar este proceso de testing independientemente del proceso de desarrollo que siga la organización, aportando una visión objetiva, y pudiéndose incorporar a cualquier altura del proyecto de desarrollo (tanto en etapas tempranas, como con el producto ya

construido).

Los **ciclos de prueba** son el eje del proceso, e implican la ejecución de las pruebas planificadas para una versión dada del producto. Cada ciclo de prueba, se corresponde con una versión del producto, la cual no debe sufrir alteraciones durante la ejecución de las pruebas. Paralelamente, se mejora el producto o arreglan los incidentes detectados en un ambiente separado. Una nueva versión del producto conteniendo los arreglos de los incidentes y eventualmente nuevas funcionalidades, puede ser liberada para la ejecución de un nuevo ciclo de prueba.

El **análisis del riesgo del producto** hace que ProTest tome en cuenta las realidades de la organización. En términos generales, se utilizan las técnicas básicas de análisis de riesgo, elaborando una lista de riesgos priorizada, tratando de mitigar cada uno de ellos, revisándola periódicamente, y manejando la incorporación de nuevos riesgos. Un riesgo es tal, si lo es para el producto, la organización, el negocio, los interesados, o compromete el desarrollo del proyecto. La lista de riesgos puede ser elaborada e identificada por los encargados de llevar adelante el testing, pero siempre debe ser validada y alimentada con la opinión el resto de los actores del proyecto.

ProTest consta de cuatro etapas principales:

- Estudio Preliminar
- Planificación
- Ciclos de Prueba
- Evaluación

En el siguiente diagrama, vemos cómo se relacionan las diferentes etapas.

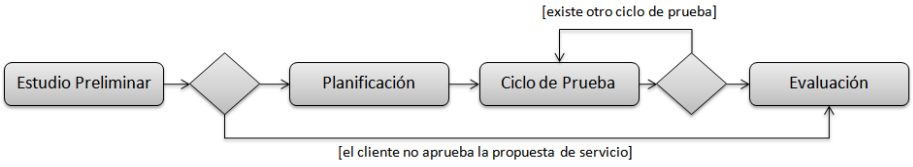

Figura 2.1: Etapas de ProTest

Estudio Preliminar: consiste en el estudio de las características del producto y sus funcionalidades con el propósito de definir el alcance de las pruebas y elaborar una primera estimación de los ciclos de prueba. El siguiente paso es el de calcular la cotización del proyecto de testing. Si la cotización es aceptada la siguiente etapa es la de Planificación, de lo contrario, se da paso a la etapa de Evaluación de los posibles problemas, y se archiva el proyecto para consultarlo a futuro.

Planificación: tiene por objetivo planificar el proyecto de testing luego de que se acordó con el cliente seguir adelante con el proyecto. En esta etapa se analizan y mejoran los requerimientos, se definen los ciclos de prueba y es elaborado el Plan de Testing considerando las funcionalidades a probar y haciendo énfasis en el riesgo de de cada parte del sistema.

Ciclo de Prueba: etapa en la que se construyen pruebas y son ejecutadas. Un ciclo de prueba se corresponde con una versión dada del sistema. Los ciclos de prueba guían al proyecto de testing.

El ciclo de prueba se compone de:

- **Seguimiento del ciclo:** se trata del seguimiento y control del ciclo de prueba. Consiste de la revisión de la planificación, la planificación detallada de las actividades del ciclo de prueba y ajustes al plan derivados de las estimaciones y mediciones llevadas a cabo en ciclos de prueba anteriores. Al término de la configuración del entorno, diseño de las pruebas y ejecución de las pruebas, se vuelve a seguimiento y control con el fin de evaluar la finalización del ciclo, o la continuación con el ciclo, agregando más pruebas.

- **Configuración del entorno:** los entornos y ambientes de testing son configurados, y la versión del producto es instalada, así como las herramientas a utilizar. Se apunta a lograr un ambiente de testing separado del ambiente de desarrollo.

- **Diseño de pruebas:** consiste en la construcción de un conjunto de casos de prueba acordes a los requerimientos. Estos casos de prueba pueden ser derivados de la aplicación de una o varias técnicas de diseño de pruebas.

- **Ejecución de las pruebas:** es la actividad de interactuar con el producto, ejecutando los casos de prueba diseñados. La realidad se compara con los resultados esperados, y se reportan los posibles incidentes, consultas u observaciones al producto.

Evaluación: a modo de cierre, en esta etapa se elabora el informe final y se evalúa el proceso de testing para identificar puntos donde se puede mejorar. En la evaluación se busca conocer el grado de satisfacción del cliente. Finalmente,

se archiva el proyecto y sus elementos relacionados para recurrir a él en futuros proyectos a modo de consulta.

2.2.2. Testing Ágil

Esta sección, trata el tema desde la visión de un artículo muy interesante publicado en la revista *Avances en Sistemas e Informática* [Zap10] por la Universidad Nacional de Colombia, sede Medellín.

¿Qué son las Pruebas Ágiles?

El testing ágil es una práctica (o conjunto de prácticas de testing), que se ha ido consolidando como propuesta de complemento a las metodologías ágiles y en respuesta a la necesidad de las organizaciones de adaptar el testing a estos métodos.

Como sucede con otros procesos de desarrollo, a la hora de incorporar testing en el ciclo de vida de los métodos ágiles de desarrollo, es común encontrar obstáculos y limitaciones, como la mala comunicación entre los equipos y dentro de los equipos, y la carencia de automatización de las pruebas en diferentes fases del ciclo de vida de desarrollo.

El testing ágil, sigue los principios del manifiesto ágil[2], teniendo prioridad de satisfacer al interesado por medio de las entregas tempranas de software. Como otras metodologías de testing, también promueve la **colaboración continua entre equipo interesado y equipo desarrollador** y se procura la **mejora continua en el proceso**, **respondiendo al cambio y a las nuevas expectativas** del interesado.

Adoptar metodologías ágiles requiere un cambio cultural, que algunas organizaciones no son capaces de afrontar. Las organizaciones más adecuadas para implantar metodologías ágiles suelen tener integrantes con un pensamiento independiente, no tienen estructura jerárquica establecida y por sobre todo ponen énfasis en políticas de calidad.

El testing ágil es una de las metodologías que apunta a un entorno colaborativo, con fuertes interacciones interpersonales. Hace foco en la auto-gestión y la integración de usuarios, programadores y desarrolladores en el proceso de testing, instaurando una responsabilidad por la calidad de todo el equipo, en lugar de que sea vista como pertenencia exclusiva del área de testing.

Dentro de un equipo ágil, debe existir comunicación continua entre los desarrolladores e interesados, definiendo los requisitos, las pruebas y el comportamiento esperado del producto. Además se manifiesta el concepto de roles, sugiriendo

[2]Se puede acceder a la versión en español del Manifiesto por el Desarrollo Ágil de Software en http://http://agilemanifesto.org/iso/es/, accedido en última oportunidad en marzo de 2011

que cada miembro asuma varios roles para eliminar la dependencia entre individuos, y favoreciendo la proactividad.

Tanto como otras metodologías, el testing ágil tiene entre sus prácticas recomendadas las pruebas por pares (par tester y desarrollador, tester y usuario, o dos testers trabajando juntos), pruebas de regresión automatizadas, y la utilización de herramientas adecuadas (como un sistema de reporte y seguimiento de incidentes). También recomienda como buena práctica, considerar la importancia del rol del usuario o cliente en la elaboración y ejecución de pruebas de aceptación del producto.

Conceptos alrededor del testing ágil

- **Tester:** persona que puede desempeñar varios roles en un proyecto de software. Tiene habilidades técnicas y destrezas para el desarrollo y ejecución de pruebas.
- **Interesado:** se apoya en un analista con el fin de proponer una solución al problema. Tiene conocimiento del negocio, y es quien es entrevistado para descubrir los requisitos, los cuales encaminan la solución.
- **Equipo ágil:** expertos en áreas técnicas de software y en el dominio del negocio, que trabajan orientados hacia un mismo fin en pequeños ciclos de tiempo, llamados **iteraciones**.
- **Metodología ágil:** proceso de desarrollo de software que busca construir aplicaciones de manera rápida teniendo en cuenta en todo momento al interesado. Estas metodologías enfatizan las comunicaciones cara a cara en vez de la documentación.
- **Prueba:** procedimiento para validar una funcionalidad de un determinado software.
- **X-Unit:** entorno utilizado para automatizar las pruebas unitarias.
- **Bug Tracking System:** herramienta utilizada para el reporte y seguimiento de los incidentes.

Como sugieren la mayoría de las metodologías para el rol del tester, un *tester ágil*, se caracteriza por contar con habilidades técnicas en testing, y cumplir un rol articulador entre las necesidades del interesado y el equipo desarrollador. El tester está en la posición adecuada para lograr la colaboración de todas las partes.

Por último, dentro de las prácticas más importantes recomendadas en el testing ágil, se encuentra la automatización. Contar con **pruebas de regresión automatizadas**, e incluso **pruebas de aceptación automatizadas**, es fundamental para el éxito del testing ágil. El ritmo de desarrollo supera rápidamente la capacidad de ejecución de pruebas manuales, por lo tanto, la estrategia debe estar enfocada a la automatización.

2.3. Incorporación Temprana del Testing

Comenzar las actividades de testing en etapas tempranas del desarrollo de un producto, tiene muchas consecuencias positivas[3]. Los modelos a continuación, discuten cuándo hacer qué tipo de testing, según la información disponible en cierta etapa del proceso de desarrollo.

2.3.1. Modelo V

El modelo V (o vee) [IAB97] es un modelo de proceso que se opone a la idea de que el testing puede comenzar solamente cuando el producto está terminado. Es lógico pensar que no se puede testear algo que no existe, pero, tratar así al testing es desconocer que el conjunto de actividades relacionadas va más allá de sólo la ejecución de las pruebas.

Tomando en cuenta el modelo V simplificado de la figura 2.2, podemos ver cómo cada actividad de desarrollo se corresponde con una actividad de testing. Las diferentes organizaciones pueden nombrar a las etapas de diferentes maneras, pero según el modelo V, cada actividad de la parte izquierda, se corresponde, y alimenta de información, a una actividad de testing en la parte derecha del modelo.

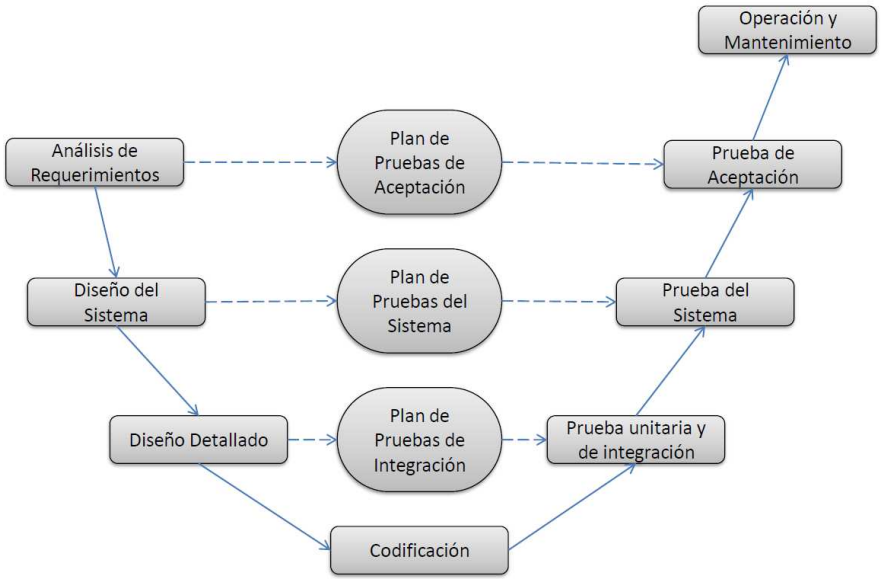

Figura 2.2: Modelo V simplificado.

[3]Por ejemplo, sabemos que el costo de arreglar un error en la definición es siempre mucho más bajo que hacer cambios en el producto ya elaborado.

Si tomamos las actividades de testing y de desarrollo, vemos como avanzan de forma opuesta las etapas de construcción del producto y convergen en el vértice del modelo. Este es precisamente el gran aporte del modelo V que sostiene que las actividades de testing correspondientes a las actividades de desarrollo, cuentan con la información suficiente para diseñar las pruebas correspondientes. De esta manera, estas etapas están incorporando testing de manera temprana en el ciclo de vida del desarrollo del software, encontrando defectos en etapas iniciales del producto, impidiendo que dichos errores se propaguen a etapas posteriores, donde es mucho más costoso corregirlos.

A modo de ejemplo consideremos un proceso de desarrollo con las etapas de la figura 2.2, un proyecto en sí o una iteración dentro de un ciclo de vida de un producto. En las etapas más tempranas, se definen los requerimientos, esta información es suficiente para diseñar y negociar pruebas de aceptación con los usuarios finales. Más adelante, se especificará funcionalmente el análisis de forma más detallada, con una visión funcional; de esta manera la actividad de testing correspondiente está en condiciones de diseñar pruebas de aceptación. Al llegar a un diseño detallado de la solución, para cierto requerimiento, estamos en condiciones de diseñar pruebas de integración entre los diferentes componentes a construir. Por último, en el punto de convergencia del modelo, la construcción de las pruebas unitarias, se alimenta del mismo diseño o especificación (o partes de él) que requieren las unidades o módulos.

Críticas al modelo V

El modelo V presentado, hace foco en la inclusión temprana del testing en el ciclo de vida del producto, lo cual es siempre más efectivo que incluirlo en etapas tardías. La base del modelo V, es que hay una correspondencia entre cada etapa del modelo en la parte izquierda, y las actividades de la parte derecha.

Según algunos autores, hay algunos problemas con el modelo V. La experiencia práctica indica que la correspondencia entre las etapas a la izquierda del modelo y las actividades de testing no siempre se da. Por ejemplo, la especificación funcional muchas veces no está completa, o no provee la suficiente información para elaborar pruebas de sistema. En una situación del mundo real, el testing se nutre de diversas fuentes de requerimientos, considerando aspectos del negocio, de la arquitectura, o del diseño físico de la solución inclusive.

Otra crítica hacia el modelo V, es que no toma en cuenta el potencial valor agregado de las pruebas estáticas (inspecciones, revisiones de documentos, análisis estáticos de código, entre otros). Esta se puede considerar una de las grandes omisiones del modelo; además de tratar a las actividades de testing solamente como actividades anexas a las actividades de desarrollo.

2.3.2. Modelo W

El modelo W intenta subsanar los puntos débiles del modelo V. Introducido por Paul Herzlich en 1993 [Her93], el modelo W intenta establecer actividades intermedias entre las correspondientes etapas en ambas ramas del modelo V, en vez de hacer hincapié en actividades específicas del testing dinámico; logrando productos en sí mismos.

Cada actividad de desarrollo es acompañada por una actividad de testing, la cual busca determinar si la actividad de desarrollo alcanzó sus objetivos, y los entregables asociados satisfacen sus requerimientos de entrada.

Cada entregable de desarrollo (por ejemplo documento de análisis), tiene su actividad de testing específica (siguiendo con el ejemplo, sería testear el documento de análisis). Si las etapas de desarrollo son más o menos, la única adaptación necesaria, es agregar una actividad de testing que se encargue de verificar los entregables. Como contraparte a la debilidad del modelo V ante las **técnicas de testing estático**, el modelo W acepta el uso gran cantidad de ellas; como ser inspecciones de documentos, revisiones, análisis de escenarios, análisis estáticos, animación de requerimientos y por supuesto, se puede avanzar con el diseño de casos de prueba de forma temprana.

El modelo W sugiere focalizar en aquellos puntos más riesgosos, dónde el testing temprano (estático y diseño de testing dinámico) puede ser más efectivo.

En cuanto a las etapas de **testing dinámico**, existen diferentes tipos que se pueden aplicar (pruebas unitarias, pruebas de integración, pruebas de sistema, pruebas de aceptación); esta parte del modelo V se conserva en el W.

El modelo W, no exige contar con la misma cantidad de etapas de testing dinámico que etapas de desarrollo, ni restringe a que el testing deba contar con determinado documento; sino que permite obtener información de varias etapas, y de diferentes áreas de la información. Por ejemplo, si hay cinco etapas de desarrollo para la definición, diseño y construcción del producto, entonces puede que alcance con tener solamente tres etapas de testing dinámico; y se podría lograr un buen diseño de pruebas combinando la información adquirida desde un manual de usuario, conocimiento teóricos acumulados en técnicos funcionales y la experiencia acumulada en usuarios que utilicen el sistema en la práctica.

Una posible forma aplicar el modelo W sería:

- Identificar los riesgos relevantes.
- Especificar los componentes, productos y entregables que se van a testear.
- Seleccionar técnicas estáticas de testing, y los tipos de pruebas dinámicas a llevar a cabo para tratar los riesgos.

- Planificar las actividades de testing lo más cercanas posibles a las actividades de desarrollo que generen algún producto verificable.

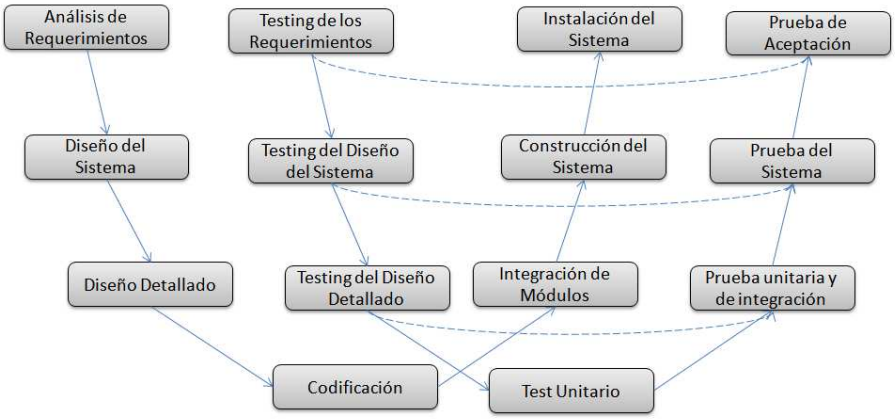

Figura 2.3: Posible caso del Modelo W.

2.4. Estrategias de Testing Funcional Manual

Una estrategia de testing consiste en la forma en la que se llevará adelante el testing para determinado producto. Define de qué manera y en qué momento se analizarán requerimientos y funcionalidades del software, cómo y cuándo se diseñarán y ejecutarán las pruebas.

Las estrategias más comunes son la de testing exploratorio (que confiere un aprendizaje simultáneo del producto a la vez que se ejecutan pruebas) y la de testing planificado (donde se analiza el producto y los requerimientos, se diseña casos de prueba y luego se ejecutan). La estrategia puede incluir la decisión de contar con pruebas automatizadas.

Existen estrategias más, o menos complejas. Con la experiencia en el manejo de las estrategias de testing, es posible formular nuevas estrategias, como ser híbridas, tratando de tomar lo mejor de cada una en un determinado contexto, sin encasillar la prueba a los límites de unas u otras.

2.4.1. Testing Diseñado

El testing diseñado (o planificado) está fuertemente relacionado con el análisis de los requerimientos. Formalmente, depende en gran medida de que estos requerimientos estén construidos con un grado de avance cercano al final.

El objetivo es logar un conjunto estático de casos de prueba que luego de especificados (con datos de prueba), puedan ser ejecutados por cualquier tester. En organizaciones donde es primordial la negociación del alcance de las pruebas, la validación de producciones intermedias y las auditorias; el diseño aporta un gran valor, dado que a partir del conjunto de casos de prueba generado, se pueden apoyar las instancias de validación, priorización y negociación del alcance de las pruebas, con una visión objetiva y discreta.

Según ProTest [Pér06], **el testing diseñado involucra:**

- **Análisis de requerimientos:** estudio de las funcionalidades documentadas en requerimientos con el fin de obtener conocimiento en el dominio.
- **Diseño de casos de prueba:** a partir de los requerimientos analizados, se aplican diversas técnicas de testing funcional para derivar un conjunto de casos de prueba.
- **Selección y priorización de casos de prueba:** en conjunto con las técnicas, se seleccionan y priorizan los casos de prueba basándose en diferentes criterios, como ser la criticidad de la funcionalidad, la complejidad de implementación o uso de la funcionalidad, el riesgo asociado al producto, el tiempo disponible para la ejecución de las pruebas, entre otros.
- **Ejecución de pruebas:** es la interacción con el sistema siguiendo los casos de prueba especificados en el diseño, luego de ser priorizados y seleccionados. La ejecución se considera finalizada cuando el conjunto de casos negociados y validados fueron ensayados en el sistema.

2.4.2. Testing Exploratorio

El testing exploratorio es una estrategia que supone un aprendizaje simultáneo a la ejecución de las pruebas. Es ampliamente utilizada, sobre todo en contextos ágiles donde el foco está en cubrir la mayor cantidad de funcionalidades posibles, en lugar de contar con documentación estructurada y detallada.

Es posible llevarlo adelante de diferentes formas, como ser el testing exploratorio conocido como estilo libre y el testing exploratorio basado en sesiones. Esto no quiere decir que no existan otras maneras, ni que no puedan desarrollarse nuevos estilos en el futuro. Una reseña de otros estilos de testing se puede encontrar en el libro *"Exploratory Software Testing: Tips, Tricks, Tours, and Techniques to Guide Test Design"*, de James A. Whittaker [Whi09a].

Se considera al testing exploratorio en sí, como una estrategia muy poderosa a la hora de encontrar incidentes rápidamente. Uno de los objetivos (y muchas veces virtud) del testing exploratorio se puede considerar en justamente encontrar los incidentes más importantes de forma temprana en el transcurso de las pruebas [Bac02].

Un punto particular a tener en cuenta, es que el testing exploratorio depende fuertemente de las habilidades del tester, su experiencia, su capacidad de tomar decisiones en el acto, manejo de la lógica y capacidad de crear modelos mentales del funcionamiento del sistema. Por lo tanto, es importante para los testers incrementar las habilidades mediante la práctica y la incorporación de nuevas técnicas, y nuevas habilidades en diferentes áreas, como ser el diseño de casos de prueba, la comunicación, relaciones interpersonales, capacidad técnica, auto-gestión, entre otras.

Testing Exploratorio Estilo Libre

El estilo libre implica la exploración *ad hoc* del sistema sin necesariamente tener que dejar trazas o documentación explicita de la ejecución de las pruebas. Es en ocasiones utilizado para familiarizarse con el producto, aprender de él e investigarlo, así como también para las *"pruebas de humo"*[4]. El estilo libre no sigue ningún patrón específico ni reglas. No es recomendable utilizar este estilo como la única implementación de la estrategia de testing exploratorio, dado que las pruebas resultantes no dejan evidencia, se suelen ejecutar en cualquier orden, y se pierde el control sobre la cobertura de las funcionalidades [Whi09b].

Según James Bach en su artículo *"Exploratory Testing Explained"* [Bac02] la gestión del testing estilo libre pude ser llevada adelante de manera *delegante*, o de forma *participativa*.

Gestionar en la forma delegante: implica que el líder tendrá que tener en cuenta el caudal de responsabilidad que está depositando en los testers y a la vez saber que esta comprometiendo parte del éxito del proyecto de testing. La relación con el equipo debe ser muy estrecha para que se comprenda que cada integrante es dueño y gestor de su tiempo, y responde por el éxito de su parte. El líder, en este caso, se limita a establecer los charters, dividir el trabajo y hacer el seguimiento; debe ser capaz de reconocer a quienes delegar los trabajos más importantes, más riesgosos según el dominio, o más triviales. A los testers más habilidosos y comprometidos, se les suelen encargar las tareas más importantes y complejas, en tanto al resto se les asocian tareas más cortas y controlables.

[4]Las pruebas de humo tienen por objetivo encontrar incidentes graves rápidamente. Se aplican generalmente para saber si un producto está lo suficientemente estable para practicarle testing más riguroso.

Gestionar por participación: es la modalidad de gestión del testing exploratorio estilo libre en la cuál, los líderes de testing trabajan muy apegados a los testers, inclusive como un tester más. La situación práctica deseable, es que los líderes tengan sus responsabilidades administrativas y reuniones, delegadas en otras personas. Dirigir el equipo de testing por participación, permite que el líder establezca en todo momento los ajustes a la estrategia, explicitando el comportamiento esperado del equipo. Un líder es responsable por el desempeño de su equipo, y apoyarse en este estilo de dirección lo ubica en una posición favorable a la hora de lograr sus objetivos. Los mitos que rodean a la ineficiencia del testing exploratorio, tienden a desaparecer cuando un líder está estrechamente vinculado al resto del equipo y al transcurso del proyecto de testing.

Reporte del testing exploratorio estilo libre: además de las herramientas con la que se gestionen los incidentes, la coordinación del equipo de testing se hace de forma verbal en reuniones, que según las recomendaciones, deberían ser al menos semanales. Bach comenta en el mismo artículo [Bac02] que Cem Keaner[5], sugiere reuniones regulares con testers para discutir sus procesos al menos una vez por semana. Él encuentra útil abrir la reunión con una pregunta estándar del estilo: "¿Cuál es el bug más interesante que han encontrado recientemente?. Muestrenmeló!".

Testing Exploratorio Basado en Sesiones

El testing exploratorio basado en sesiones, es un enfoque del testing exploratorio ideado por los hermanos James y Jonathan Bach, quienes han escrito varios artículos, y han colaborado en diversos libros sobre testing de software, haciendo foco en la enseñanza e investigación del testing; particularmente, se los considera referentes en el testing exploratorio. Este enfoque incluye directivas de organización, gestión de las pruebas e incidentes, así como control de la cobertura y tipos de testing. En el artículo *"Session-Based Test Management"* [Bac00], Jonathan Bach considera que el grado en que se involucra el tester con el producto es mayor, al encontrarse en la dinámica de aprender del dominio, diseñar y ejecutar pruebas simultáneamente. Como consecuencia, se consiguen mejores resultados en una menor cantidad de tiempo.

Sesiones en el Testing Exploratorio

Con el fin de gestionar el testing exploratorio, nace de la necesidad de separar el testing de todo lo que no lo es. Para este objetivo, se utiliza el concepto de *sesión*. La sesión es un intervalo de tiempo dedicado a practicar testing exploratorio, sin interrupciones (por ejemplo mail, chat, celulares), con el foco puesto en el testing. La duración recomendada de una sesión según el artículo puede

[5]Otro gran referente en testing, más información en su web personal, *http://kaner.com/* (accedida en diciembre de 2010)

ser: *short* (una sesión "corta" de aproximadamente 45 minutos), *normal* (una sesión de duración "normal" de aproximadamente 90 minutos) o *long* (equivale a una sesión "larga" de dos horas de duración aproximadamente).

La sesión puede registrarse en texto plano, siguiendo la plantilla de la figura 2.4[6].

CHARTER	TEST NOTES
----------	----------
Analyze MapMaker's View menu functionality and report on areas of potential risk.	View: Welcome Screen Navigator Locator Map Legend Risks: - Incorrect display of a map element. - Incorrect display due to interrupted repaint.
#AREAS OS | Windows 2000 Menu | View Strategy | Function Testing Strategy | Functional Analysis START	BUGS
----------	----------
5/30/00 03:20 pm TESTER	#BUG 1321 Zooming in makes you put in the CD 2 when you get to a certain level of granularity (the street names level) -- even if CD 2 is already in the drive.
----------	#BUG 1331
Jonathan Bach TASK BREAKDOWN ᴸ	Zooming in quickly results in street names not being rendered.
----------	#BUG <not_entered>
#DURATION short #TEST DESIGN AND EXECUTION 65 #BUG INVESTIGATION AND REPORTING 25 #SESSION SETUP 20 #CHARTER VS. OPPORTUNITY 100/0 DATA FILES	instability with slow CD speed or low video RAM. Still investigating. ISSUES
----------	----------
#N/A	#ISSUE 1 How do I know what details should show up at what zoom levels? #ISSUE 2 I'm not sure how the locator map is supposed to work. How is the user supposed to interact with it?

Figura 2.4: Ejemplo de hoja de reporte de sesión de testing exploratorio.

Las secciones que componen el reporte de la sesión son:

- **Session Charter:** se puede traducir como "hoja de ruta". Es una descripción verbal de lo que se pretende en la sesión. Puede indicar algunos elementos deseables, ideas, o mencionar heurísticas a utilizar durante la exploración. Es común que contenga algunos pasos generales de las pruebas y documentación útil para la ejecución de la sesión. Esta "hoja de ruta", es generalmente escrita por el tester que esté gestionando las actividades de testing exploratorio, o bien por el mismo tester que ejecutará la prueba. Aquí se encuentra la **Misión**, que es la descripción verbal del objetivo de una prueba. Puede consistir en probar una funcionalidad, o conjunto

[6]La ventaja de seguir este formato, es que se puede utilizar una herramienta de Bach para recopilar la información, y obtener métricas interesantes, información sobre la cobertura, entre otros. Más información de este y otros recursos en *http://www.satisfice.com/sbtm/*

de funcionalidades relacionadas, así como también cumplir determinados criterios o situaciones por las cuales transitar durante la prueba.

- **Tester name(s):** el nombre del tester o los testers que ejecutarán la sesión.
- **Date and time started:** fecha y hora en que comienza la sesión de testing exploratorio
- **Task breakdown:** incluye la duración de la sesión, y tres porcentajes relacionados que expresan la porción de tiempo dedicada a diseño y ejecución de pruebas, investigación de bugs y reporte, y configuración (preparación) de la sesión. Luego, una relación de porcentaje que indica qué porción del tiempo fue dedicado a la misión en sí, y qué porción se dedicó a otros temas por fuera de la misión. A esto último se le llama "oportunidad", como en el caso de la analogía del tour,[7] es todo aquello que exploramos por afuera de nuestra "ruta" trazada, pero luego, volviendo a enfocarnos en la misión.
- **Data files:** eventuales archivos relacionados con la sesión (de cualquier índole, como imágenes, archivos conteniendo datos de prueba anexos, entre otros).
- **Test notes:** esta sección puede contener anotaciones varias, como ser los pasos ejecutados para la prueba, algún comentario detallando particularidades de las funcionalidades, o cualquier otro comentario que resulte interesante para la ejecución y documentación de la sesión.
- **Issues:** incidentes de menor porte, consultas, sugerencias y observaciones varias.
- **Bugs:** aquí se documentan los errores encontrados durante la sesión.

2.4.3. Discusión: Testing Exploratorio Vs. Testing Diseñado.

Así como manifiestan muchos de los autores referenciados, no hay buenas prácticas en general, sino una mejor práctica dado un contexto. No sería correcto por lo tanto, contraponer dos estrategias de testing, esperando sacar la conclusión de que una es mejor que la otra. Inclusive separar el mundo de las estrategias entre exploratorio y diseñado, sería un enfoque bastante arbitrario que dejaría por fuera gran parte de la realidad a la que se enfrenta cualquier tester profesional. ¿Por qué entonces presentamos esta discusión?, por el motivo de que

[7] La analogía se trata de una recorrida turística, en la cual planificamos pasar por ciertos puntos, esta sería nuestra "misión". Cuando algo nos llama la atención de la ciudad, nos apartamos momentáneamente para explorar ese lugar no planificado, esto sería la oportunidad". Luego, regresaríamos a nuestra ruta trazada desde un principio, volviendo a hacer foco en la misión.

son dos grandes y renombradas estrategias base en el testing de software en las cuales se suelen catalogar los enfoques. Otras estrategias y heurísticas no entran en la discusión por considerarse complementarias desde un principio, o por no ser tan populares.

Una estrategia que incluye testing exploratorio, suele ser más flexible, adaptándose al cambio más rápidamente que el testing diseñado. Un conjunto diseñado de casos de prueba, se vuelve más "débil" conforme cambia el contexto y el producto. Es necesario re-planificar y diseñar nuevos casos de prueba, contemplando el nuevo estado del producto, si queremos mantener o incrementar la efectividad y la probabilidad de encontrar errores. El testing exploratorio, está en permanente cambio, y supone un diseño y ejecución de pruebas simultáneos, irremediablemente tomando en cuenta el estado actual del software y del contexto en cada instante.

En líneas generales; el testing exploratorio es recomendable cuando el objetivo es dejar trazas de **ciclos funcionales,** encontrar gran cantidad de errores en poco tiempo, no se cuenta con buena documentación de requerimientos y no es necesario contar con la validación de los casos de prueba generados por el equipo de testing. En cambio, el testing diseñado, es más recomendable cuando se están siguiendo procesos más rígidos y estructurados, se cuenta con documentación de requisitos aceptable (o es posible reconstruirla a partir de distintos referentes expertos del dominio), se puede planificar y estimar el proyecto de testing, se debe solicitar y negociar la validación de los casos de pruebas con autoridades o expertos en el dominio, y relacionado con lo anterior, se desea dejar evidencia de la negociación concreta de la cobertura específica de la ejecución de las pruebas.

La idea no es encasillar las estrategias de testing en dos únicas versiones. La complejidad del software actualmente, y la potencialmente infinita cantidad de casos de prueba posibles para una aplicación, nos ubica indefectiblemente en la situación de tener que verificar desde múltiples estratégicas básicas, o inclusive con una combinación de diferentes estrategias. Adoptar una mezcla de estrategias incluyendo ideas de análisis del riesgo y sectores críticos del negocio, puede enriquecer enormemente las actividades de testing, salvando la posible pérdida de información derivada de tener una perspectiva desde una sola estrategia. Dentro de las habilidades deseables del tester, está la de conocer las ventajas y desventajas de cada estrategia, y sacar lo mejor de cada una para reunir la información necesaria.

2.4.4. Otro enfoque complementario: Testing Basado en Riesgo

En el artículo *"Heuristic Risk-Based Testing"* [Bac99], James Bach describe la heurística del testing basado en riesgo, que consta de:

- Crear una lista priorizada de riesgos
- Llevar adelante pruebas que exploren cada riesgo
- Ajustar el esfuerzo de testing para mantener el foco en los riesgos actuales; los riesgos anteriores son mitigados y se identifican nuevos.

El riesgo siempre esta asociado al testing, pero llevar adelante testing basado en riesgo implica centrar las actividades en los riesgos identificados, manejarlos, identificar nuevos riesgos y direccionar las actividades de testing a su mitigación.

El artículo menciona dos enfoques para el análisis, **"de adentro hacia afuera"**, y **"de afuera hacia adentro"**. Ambos enfoques son complementarios y tienen distintas fortalezas.

De adentro hacia afuera, identificando los riesgos asociados a cada situación:

- Vulnerabilidades: ¿Qué debilidades o posibles fallas hay en estos componentes?.
- Amenazas: ¿Qué entradas o situaciones pueden haber que puedan explotar una vulnerabilidad y lanzar una falla en este componente?.
- Víctimas: ¿Quién, o qué puede ser impactado por potenciales fallas, y cuán malo puede ser?.

En una sesión con desarrolladores, se obtiene información sobre el funcionamiento del sistema, luego, el tester va conociendo el comportamiento esperado del sistema y formulando interrogantes acerca de las diferentes partes del sistema y sus interacciones. Se recomienda estar muy atento al tono de voz, la seguridad con que se contestan ciertas preguntas, o se demuestra dominio sobre determinados puntos; todo puede ser una pista para encontrar incidentes. A modo de ejemplo, en una supuesta sesión con desarrolladores, se cuenta con un pizarrón donde se esquematiza el funcionamiento del producto en diagramas informales (con cuadros y flechas), luego el tester formularía preguntas del estilo:

- ¿Qué sucede si la función en este cuadro falla? (señalando cierto cuadro en el diagrama).
- ¿Esta función puede ser siempre invocada en un momento incorrecto?.
- ¿Qué chequeo de errores se hacen aquí? (apuntando algún lugar del diagrama).

- ¿Qué significa esta flecha? (señalando una flecha) ¿Qué sucedería si se corta?.
- Si los datos en esta dirección, se corrompen, ¿Cómo lo identificaría? ¿Qué pasaría? (señalando un flujo de datos).
- ¿Cuál es la mayor carga que puede soportar este proceso?.
- ¿De qué componentes externos, servicios, estados o configuraciones depende este proceso?.
- ¿Puede algún otro recurso o componente diagramado aquí ser manipulado o influenciado por algún otro proceso?.
- ¿Este es el panorama general? ¿Qué no se ha representado en el diagrama?.
- ¿Cómo probarías esto si lo juntamos con esto otro?.
- ¿Qué es lo más preocupante? ¿Qué piensa que debo testear?.

De afuera hacia adentro: En esta forma de proceder con el análisis de riesgo, se consulta una lista predefinida de riesgos decidiendo si aplican o no. Este es un enfoque relativamente más sencillo que el anterior.

Bach menciona algunos tipos de listas de riesgo que utiliza comúnmente, como ser: "Categorías de criterios de Calidad", "Lista Genérica de Riesgos" y "Catálogos de Riesgos"

Categorías de criterios de calidad

Estas categorías se ajustan con diferentes tipos de requerimientos, investigando qué podría pasar si el requerimiento asociado a las diferentes categorías no se alcanzara.

- Capacidad: ¿Puedo efectuar las funciones requeridas?.
- Fiabilidad: ¿Funcionará bien y resistirá fallas en todas las situaciones requeridas?.
- Usabilidad: ¿Cuán fácil es para un usuario real usar el producto?.
- Performance: ¿Cuán veloz debe ser y responder el sistema?.
- Instalabilidad: ¿Cuán fácil puede ser instalado en la plataforma objetivo?.
- Compatibilidad: ¿Qué tan bien trabaja con componentes y configuraciones externas?.
- Soportabilidad: ¿Cuán económico resulta proveer soporte?.
- Testeabilidad: ¿Qué tan efectivamente puede ser testeado el producto?.

- Mantenibilidad: ¿Cuán económico será construir, reparar y mejorar el producto?.
- Portabilidad: ¿Cuán económico será trasladar o reutilizar el producto?.
- Localizabilidad: ¿Qué tan económico será publicar el producto en otro idioma?.

Lista genérica de riesgos

Este tipo de lista es universal para cualquier sistema. Algunos riegos:

- Complejidad: cualquier elemento desproporcionalmente largo, intricando y complicado.
- Nuevo: todo lo que no tiene antecedentes en el producto.
- Cambiado: cualquier elemento que haya sido manipulado o mejorado.
- Dependencia de flujo ascendente: elementos cuya falla puede causar fallas en cascada al resto del sistema.
- Dependencia de flujo descendente: algo que es especialmente sensible a las fallas del resto del sistema.
- Crítico: todo que al fallar pueda causar un daño de importancia.
- Preciso: cualquier elemento del cual podamos conocer sus requerimientos exactamente.
- Popular: algo que será muy usado.
- Estratégico: cualquier cosa que tenga especial importancia en el negocio, como ser una funcionalidad que distingue al producto de los competidores.
- De terceros: algo que se utilice en el producto que halla sido desarrollado fuera del proyecto.
- Distribuido: cualquier elemento que esté repartido en el tiempo o espacio; también elementos que deben trabajar juntos.
- Con errores: algo que es sabido que tiene muchos errores.
- Falla reciente: cualquier cosa que tenga una historia de fallas recientes.

Catálogos de riesgos

Los catálogos de riesgos son punteos de riesgos que pertenecen a un dominio particular. Derivan de la experiencia de interactuar con determinadas circunstancias frecuentemente. Se deben interpretar como si comenzaran: “hemos tenido problemas con...”.

Algunos ejemplos que formarían parte de una catálogo de riesgos de instalación:

- Archivos instalados incorrectamente.
  - Archivos temporales no se borran.
  - Archivos viejos no se eliminan después de una actualización
  - Se instalan archivos innecesarios.
  - No se instalan archivos necesarios.
  - Un archivo correcto se instala en un lugar incorrecto.
- Archivos afectados.
  - Archivo viejo reemplaza un archivo más reciente.
  - Se afectan datos del usuario durante la instalación
- Otras aplicaciones afectadas.
  - Un archivo compartido con otra aplicación es afectado.
  - Archivos pertenecientes a otro producto son eliminados.
- Hardware configurado incorrectamente.
  - El hardware es afectado por otras aplicaciones.
  - El hardware no es suficiente para las aplicaciones instaladas
- El protector de pantalla interrumpe la instalación.
- No detección de aplicaciones incompatibles.
  - Aplicaciones ejecutando actualmente.
  - Aplicaciones ya instaladas.
- Instalador reemplaza o modifica archivos críticos o parámetros sin avisar.
- El proceso de instalación es demasiado lento.
- El proceso de instalación requiere seguimiento constante del usuario.
- El proceso de instalación es confuso

A partir del uso de las listas de riesgo se puede decidir qué componentes analizar, reuniendo información y haciendo un seguimiento a los riesgos priorizándolos y así mantenerlos bajo control e incorporando nuevos.

Con el fin de organizar el testing basado en riesgo, se proponen en el artículo algunas formas, como ser el seguimiento periódico de la lista de riesgos, o vinculado los riesgos a tareas específicas para atacarlos, así como también relacionando los riesgos a diferentes componentes, como ser partes de código, o cualquier otro elemento que pueda ser testeado.

2.5. Testing Automatizado

Esta sección tiene como fundamental referencia el libro "Software Test Automation" [FG99a] de Mark Fewster y Dorothy Graham. Las siguientes subsecciones abarcan diferentes partes del texto de Fewster y Graham, y se mencionará como referencia la parte del libro que trata el tema, indicándolo al inicio de cada subsección.

2.5.1. Selección de Casos de Prueba a Automatizar

Si tenemos un conjunto de casos de prueba, algunos serán automatizables y otros no. Incluso si todos son automatizables, hay que decidir cuáles de ellos conviene automatizar primero. Se debe considerar si el esfuerzo de automatizar será redituable frente a la ejecución manual de una prueba [FG99b].

Supongamos estos costos para un determinado caso de prueba:

Testing	Costo de prueba	Frecuencia de ejecución	Costo primer año	Costo en 5 años
Manual	10 minutos	Una vez al mes	Dos horas	10 horas
Automatizado	10 horas	Una vez al mes	120 horas	Al menos 10 horas

Cuadro 2.1: Ejemplo de retorno de la inversión de pruebas automatizadas.

Tendría que correrse esta prueba al menos cinco años a una frecuencia de una vez por mes, para empezar a igualar los costos de la automatización. Por otra parte, supongamos que el caso de prueba fuera muy difícil de correr manualmente. Aunque se ejecute en diez minutos, es propenso a errores (por ejemplo, un formulario extenso con campos y validaciones complejas), y por lo tanto ese tiempo de ejecución empieza en la práctica a ser mucho mayor. En este caso, sí valdría la pena automatizar.

Automatización de Diferentes Tipos de Pruebas

Cada tipo de prueba tiene ciertas particularidades que la hacen más o menos recomendable de automatizar en cada contexto. Es necesario evaluar para un proyecto particular, los pros y contras de automatizar cierto tipo de prueba acorde a las características de los diferentes tipos de requisitos.

Funcionalidades: son uno de los objetivos principales para el testing manual y automatizado. Las pruebas de funcionalidades consisten generalmente en interactuar con una interfaz gráfica, proporcionarle entradas y esperar un

resultado claro y visible; por lo tanto suelen ser las más fáciles de automatizar (de contar con las herramientas adecuadas).

Performance: este tipo de pruebas evalúan aspectos no funcionales de la calidad del sistema, incluyendo la correcta respuesta del sistema frente a determinados volúmenes de carga, ejecutando sobre cierta infraestructura, plataforma y configuración. Simulan el uso real del sistema para verificar que tanto el hardware, las configuraciones y el sistema en si, soporten el uso estimado. Son candidatas directas a la automatización[8].

Cualidades para-funcionales: aún cuando los sistemas respondan correctamente acorde a sus requerimientos funcionales, puede ser que otro tipo de requerimientos no se estén satisfaciendo: performance, mantenibilidad, portabilidad, testeabilidad, usabilidad, entre otros[9]. Muchos de estos requerimientos pueden apoyarse en pruebas automatizadas, pero otros requieren de la opinión de los usuarios, o de la inspección manual. Por ejemplo, en usabilidad, un test automatizado puede responder si el color que se muestra en la pantalla es el "#E1E4F2", pero no puede decir si el color "#E1E4F2" se ve "desagradable" en determinado monitor.

¿Qué automatizar primero?

El objetivo sigue siendo obtener los mejores beneficios de la automatización. Algunos de los factores a tomar en cuenta la hora de decidir cuáles pruebas se van a automatizar:

- Las pruebas más importantes, de mayor prioridad.
- Una muestra en amplitud del sistema.
- Pruebas de las funcionalidades más importantes.
- Las pruebas más fáciles de automatizar.
- Las pruebas que retornen la inversión más rápidamente.
- Pruebas que se ejecutarán más frecuentemente.

A la hora de automatizar es mejor ir de a pequeñas áreas manejables, a la vez que se consolida la experiencia, obteniendo buenos resultados en corto período. Tratar de iniciar un gran proyecto de automatización sin tomar estas precauciones, ni contar con la experiencia y los ajustes necesarios, puede tornar al proyecto en inmanejable, conduciéndolo a un posible fracaso.

[8] Ver *"performance"*, *"performance requeriment"*, *"performance specification"* y *"performance testing"* en [IEE91]

[9] Para más información, ver en el glosario de IEEE [IEE91], los términos "performance", "maintainability", "portability", "testability" y "usability".

2.5.2. Criterios de Selección de Herramientas de Automatización

Elegir una herramienta de automatización de pruebas funcionales, es uno de los puntos neurálgicos para que un proyecto de automatización de testing tenga éxito. No hay herramientas que solucionen todos los problemas, ni herramientas perfectas para cualquier proyecto. El proceso de selección puede ser tan formal y detallado como lo requiera la dimensión del proyecto, y en todos los casos el objetivo es llegar a la herramienta apropiada para la organización, que será usada en forma efectiva y con el menor costo posible [FG99c].

En primer lugar, es necesario evaluar los requerimientos propios de la organización y no basarse en las recomendaciones del mercado. La forma en que la herramienta es elegida es muy importante. Si se opta por la herramienta equivocada o que no cumple las expectativas, entonces se pueden enfrentar serios problemas al intentar hacerla funcionar en la organización.

Además de elegir la herramienta correcta, se debe elegir en la forma correcta. Esto es, puede ser que determinada herramienta cumpla con los requisitos técnicos y económicos, pero si las personas que la van a utilizar no participan en su proceso de selección, entonces, puede suceder que se genere cierta resistencia a utilizarla, porque desde sus perspectivas, no se eligió la herramienta correcta en definitiva. Por lo tanto, incluir a las personas que van utilizarán la herramienta en el equipo de selección, es un factor muy importante a considerar.

Seleccionar la herramienta de automatización es un **proyecto en sí mismo**, y como tal se le debe asignar recursos humanos y materiales. No debe ser un proyecto de gran escala[10], pero puede afectar los cronogramas de otros proyectos, por tal motivo, no se recomienda encaminarlo cuando los tiempos de desarrollo y testing están muy comprometidos.

Para seleccionar la herramienta, se debe formar un equipo específico que trabajará en el proyecto de selección y estará compuesto por un **líder** y **representantes de diferentes áreas**.

El **líder** es el responsable de gestionar el proceso de evaluación y selección. Debe contar con habilidades de gestión y ser capaz de construir un equipo de gente de diferentes áreas de la organización. Es ideal que sea alguien que tenga una visión global de la organización y sus integrantes; y que además sea respetado. Esta persona debería ser quién esté más motivado con la incorporación de la automatización en la organización.

[10] La bibliografía menciona que un proyecto de selección de herramienta para una organización mediana, típicamente toma de cuatro a seis semanas/persona de esfuerzo, y puede involucrar de tres a diez personas.

El **resto del equipo** debe incluir **representantes de cada área** que quieran automatizar sus pruebas. Puede contar con desarrolladores interesados en automatizar pruebas unitarias, usuarios que quieran automatizar pruebas de aceptación. Entre los integrantes también pueden haber testers funcionales que quieran automatizar sus pruebas de regresión, automatizadores de testing, e incluso líderes de proyecto, auditores internos de testing y coordinadores de testing, entre otros.

Identificar los requerimientos de la organización implica reconocer los problemas a solucionar. Muchos inconvenientes pueden tener que ver con el testing, o pueden ser restricciones tecnológicas y no tecnológicas a la decisión de la herramienta a elegir.

Un primer paso es elaborar una lista de problemas, los cuales pueden ser solucionados (o fácilmente manejables) con la incorporación de una herramienta. Lograr una especificación de los problemas, asegura que los integrantes del equipo están alineados en el camino de la automatización y que son capaces de detallar sus requerimientos de automatización (si no se pueden especificar las necesidades, es poco probable que se puedan solucionar por el mero hecho de incorporar una herramienta).

Algunos ejemplos de ítems que podrían estar presentes en una lista de problemas a solucionar:

1. **Problemas de testing manual:** lleva mucho tiempo, es una tarea repetitiva y fatigante o acarrea muchos errores humanos (en comparar resultados por ejemplo).

2. **No hay tiempo para pruebas de regresión:** por ejemplo, al tener muchos cambios en el producto, o tiempos de desarrollo muy inferiores a los de la ejecución de pruebas de regresión.

3. **Configuración de datos de pruebas o casos de prueba es muy propenso a errores.**

4. **La documentación no es adecuada para el testing.**

5. **Se carece de información acerca del software a probar.**

6. **El testing no es efectivo**

Luego de elaborada la lista, habría que ordenarlos por importancia y cuanto cuestan para la organización. Todos estos problemas no tiene por qué ser solucionables por herramientas de automatización de pruebas, pero puede ser que se trate mejor con ellos al contar con las adecuadas. Por otra parte, es importante tener alternativas a la automatización, **explorar diferentes soluciones**, dado que para determinados contextos puede ser que una solución que no incluya

herramientas, sea más efectiva y menos costosa. Inclusive, los problemas detectados indican que en realidad son necesarios ajustes a ciertas áreas, en lugar de automatizar pruebas.

Identificar las restricciones de la organización es el paso siguiente. Aún cuando se tengan claros los problemas y las posibles soluciones, no vale la pena invertir recursos en investigar herramientas que serán descartadas en primera instancia.

Posibles restricciones:

1. **Restricciones de ambientes:** hardware disponible, y necesario para la herramienta; software de base (sistemas operativos libres y propietarios, manejadores de bases de datos).

2. **Integración e interacción con el software a probar:** si la herramienta debe convivir con el software en el mismo ambiente, puede ser necesario prestar atención al nivel de intrusión de la herramienta en el uso de los recursos.

3. **Restricciones de proveedores comerciales:** el soporte comercial de la herramienta ante cualquier problema puede ser un factor decisivo a la hora de seleccionar una. Pesan elementos como la experiencia de otras organizaciones con la herramienta, la permanencia en el mercado del proveedor, la historia de la herramienta.

4. **Restricciones de costo:** estas restricciones van desde el costo de la herramienta por licencia, por equipo, por uso, costo del soporte técnico adicional, hasta el costo de aprendizaje de la herramienta para los futuros usuarios.

5. **Restricciones políticas:** puede ser que existan restricciones de este tipo, que suelen estar por encima de las otras en ocasiones. Por ejemplo, se puede tener cierta restricción a adquirir herramientas creadas por *partners* o en determinada región geográfica, o pertenecientes a cierta asociación. Descuidar los factores políticos es un grave error.

6. **Restricciones de calidad:** pueden ir desde requerimientos funcionales, hasta aspectos no funcionales. Facilidad de uso, facilidad de aprendizaje, cantidad de usuarios simultáneos que soporta, frecuencia de fallas, confiabilidad en el manejo de los datos, interoperabilidad con otras herramientas; son algunos de los atributos que pueden requerirse.

Habiendo cumplido las restricciones, es aconsejable elaborar una lista de características que las herramientas deben cumplir. Al menos debería ser una lista con dos categorías: características excluyentes, y características no excluyentes. Para **evaluar las características de las herramientas**, se pueden usar categorías como:

1. **Características Excluyentes:** la herramienta seleccionada debe contar con estas características para ser considerada.

2. **Características Deseables:** esta categoría puede ser usada para comparar entre herramientas que cubren las funcionalidades esperadas.

3. **Características que no interesan:** muchas herramientas pueden incluir buenas características desde el punto de vista del marketing, pero que realmente aportan poco a la funcionalidad en sí, o simplemente no aplican a la realidad de las necesidad de automatización de la organización.

4. **Características que cambian de categoría:** en el proceso de selección de herramientas puede suceder que la lista de características cambie, que surjan nuevas, o que algunas categorías cambien de una a otras.

5. **Tipos de características:** existen características que pueden estar ausentes o no; y de ser necesaria su presencia en la herramienta, se puede descartar en el proceso. Pero otras características pueden estar presentes en mayor o menor medida. Atributos como la facilidad de uso o aprendizaje, pueden ser expresados en alguna escala.

No es difícil conseguir una **gran lista de proveedores y herramientas**. El siguiente trabajo consiste en considerar las restricciones y controlar la lista de características para **construir una lista corta de herramientas candidatas.**

Cada herramienta de la lista candidata es evaluada frente a las demás. Las herramientas de lista corta son comparadas una contra otra en detalle. Es de ayuda contar con reportes de evaluación y sitios de referencia, para tener una idea de las experiencias de otras personas con las herramientas. Si es apropiado, se organizan demostraciones in situ con cada vendedor. Las herramientas deben ser testeadas con casos de pruebas realistas, debe ser tenido en cuenta el manteamiento de los casos de prueba para cuando el software cambia y si se necesita se puede evaluar y contrastar las dos herramientas identificadas con mejores prestaciones. La decisión de compra debe ser basada en un caso de negocio, desde una relación de costo beneficio previamente estimada.

2.5.3. Tipos de Herramientas Para Apoyar al Testing

En la figura 2.5, vemos un modelo V simplificado, que además está subdividido y relacionado con las diferentes clases de herramientas que pueden brindar apoyo a las actividades de testing [FG99d].

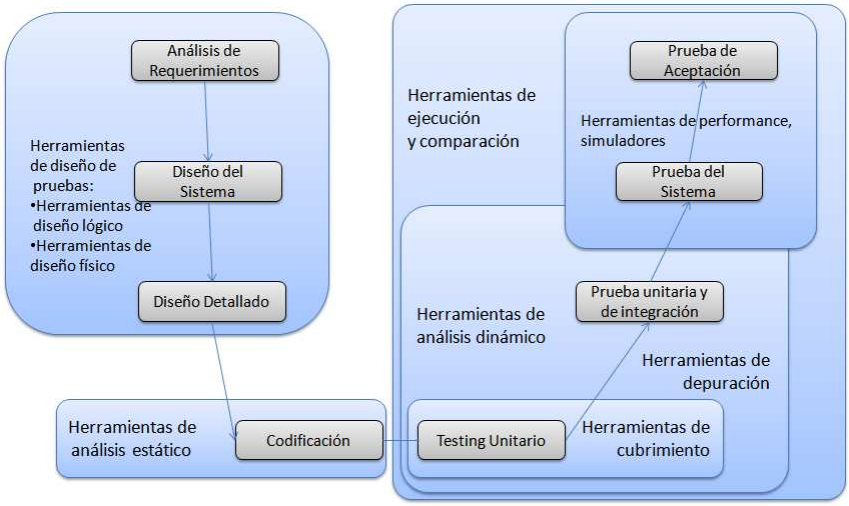

Figura 2.5: Modelo V y los diferentes tipos de herramientas relacionadas.

Herramientas de diseño: ayudan a derivar entradas o datos para el testing a través del modelado de las diferentes partes o funcionalidades del sistema.

Herramientas de diseño lógico: son también conocidas como generadores de casos de prueba, y logran derivar casos de prueba a partir de cierta especificación lógica, una interfaz, o desde código. Un ejemplo sería una herramienta que dada una especificación, nos genere entradas para testear dicha lógica.

Herramientas de diseño físico: manipulan datos existentes o generan datos para las pruebas. Por ejemplo, una herramienta que pueda extraer registros aleatorios de una base de datos, o poblar automáticamente con datos de prueba generados con algún criterio.

Herramientas de gestión de testing: van desde las herramientas para asistir en la planificación, hasta las que mantienen una trazabilidad entre requerimientos, casos de prueba ejecutados, e incidentes detectados. Entre los ejemplos más comunes, están las herramientas de gestión en propósito general, y las más específicas como las de reporte y seguimiento de defectos, mejorando la trazabilidad de las pruebas a requerimientos, diseño y código.

Herramientas de análisis estáticos de código: este tipo de herramientas detectan cierto tipo de defectos en el código fuente, mucho más efectivamente que otros tipos de ejecución.

Herramientas de cobertura: responden cuanto ha sido probado el software dado un conjunto de casos de prueba. Son comúnmente usadas a nivel de testing unitario. Por ejemplo, dado un caso de prueba en x-Unit, y un criterio de cobertura de decisión, la herramienta nos puede decir si nuestros casos de prueba satisfacen dicho criterio para el código ensayado. A nivel de requerimientos, hay herramientas que pueden apoyar el seguimiento de las diferentes funcionalidades alcanzadas en un determinado momento por el equipo de prueba[11].

Herramientas de depuración: permiten ir paso a paso dentro del código controlando la ejecución, inclusive sentencia a sentencia; permitiendo controlar el estado de las variables. No son realmente herramientas de testing, dado que la depuración no es parte del testing[12]. Sin embargo, estas herramientas son utilizadas en testing en ciertas ocasiones, por ejemplo, cuando se intenta aislar un defecto a muy bajo nivel.

Herramientas de análisis dinámico: inspeccionan el sistema en tiempo de ejecución, recopilando información acerca de utilización de los recursos (memoria, procesador, y características especiales del sistema operativo). Por ejemplo, herramientas que detectan *memory leaks*[13] en tiempo de ejecución.

Simuladores: son herramientas que habilitan probar partes de sistemas de maneras que no sería posible en el mundo real, o sería muy difícil de lograr. Por ejemplo, la ejecución simultánea de cientos de usuarios concurrentes en un sistema web.

Herramientas de testing de performance: miden el tiempo tomado por un conjunto de eventos. Por ejemplo pueden medir la respuesta de un sistema ante condiciones típicas o extremas.

Herramientas de carga: generan tráfico en el sistema. Por ejemplo, pueden generar un número de transacciones que representen una cantidad promedio, o picos máximos. Pueden ser usadas para pruebas de estrés[14].

Herramientas de ejecución y comparación: Comparación los resultados obtenidos con los esperados automáticamente. Este tipo de herramientas aplican para ejecución de testing a cualquier nivel.

[11] Una herramienta para gestionar y obtener mediciones acerca de la cobertura de las pruebas se encuentra en la web de satisfice http://www.satisfice.com/sbtm/sessions.exe

[12] Testing identifica los defectos, y la depuración los remueve. Por este motivo, la depuración es una actividad de desarrollo y no de testing.

[13] Memory leak sucede cuando un programa no libera los bloques de memoria cuando debe. En un caso extremo, si el programa se apodera de toda la memoria libre, no quedaría espacio disponible para la carga de otros programas, causando que el sistema se bloquee.

[14] Según [IEE91], las pruebas de estrés (*"stress testing"*) es el testing que evalúa el sistema al límite, o más de los límites especificados en los requerimientos.

2.5.4. Limitaciones del Testing Automatizado

Es importante tener en cuenta que la automatización no es la solución perfecta para todas las situaciones. Incluso hay situaciones donde será dificultoso recuperar la inversión en automatización. A continuación se nombran alguna limitaciones de la automatización [FG99e].

La Automatización no Reemplaza al Testing Manual

No es posible ni deseable, automatizar todas las actividades de testing. Siempre habrá algo que es mucho más fácil de hacer manualmente que automáticamente. O que es muy difícil de automatizar, y en definitiva implica que no es redituable económicamente encarar la automatización.

Es probable que algunas pruebas no deban ser automatizadas:

1. Pruebas que correrán raramente una vez. Por ejemplo si una prueba se corre una vez al año; probablemente no valga la pena automatizarla
2. Cuando el software es muy volátil. Por ejemplo, si la interfaz gráfica y la funcionalidad cambia más allá de lo que se puede reconocer de una versión a la siguiente, entonces el esfuerzo de cambiar las pruebas automatizadas correspondientes probablemente no sea redituable en cuanto a costo.
3. Cuando los resultados son fácilmente verificables por un humano, pero es difícil, o cercano a imposible de automatizar. Por ejemplo, la correctitud de un esquema de colores, la apariencia estética de una distribución de la pantalla, o cuando el sonido correcto se escucha cuando un objeto es seleccionado.
4. Pruebas que involucren interacción física, tales como deslizar una tarjeta a través de un lector, desconectar algún equipo, interactuar con dispositivos electromecánicos.

No todo el testing manual debe automatizarse, solo los mejores, o aquellos que se van a correr frecuentemente.

Cuando el software es inestable, el testing manual encuentra más defectos que el testing automatizado.

El Testing Manual Encuentra más Defectos que el Testing Automatizado

Es más probable que una prueba manual encuentre defectos la primera vez que se corre. En cambio, una prueba automatizada tendrá que ejecutarse varias veces para verificar su correctitud. La prueba automatizada, esta destinada a permanecer dentro de ciertos parámetros fijos, o a lo sumo, consumir diversas fuentes de datos.

Gran Dependencia de la Calidad de las Pruebas

Una herramienta sólo puede identificar diferencias entre la salida actual y la esperada. Por lo tanto, se debe invertir en la tarea de verificar la correctitud de las salidas esperadas en el testing automatizado. Es decir, la calidad de las pruebas automatizadas tiene que ser confiable. Las pruebas automatizadas pueden ser revisadas e inspeccionadas para asegurar su calidad. La inspección es la forma de revisión más poderosa y es muy efectiva al usarse en documentación de testing.

La Automatización no Mejora la Efectividad

Automatizar un conjunto de pruebas no las hace más efectivas (o ejemplares) que aquellos mismos tests corridos manualmente. La automatización puede mejorar la eficiencia de los tests, es decir, cuánto cuesta correrlos o cuánto tiempo toma correrlos.

La Automatización Puede Limitar el Desarrollo de Software

El testing automatizado es más frágil que el testing manual. Las pruebas automatizadas pueden ser afectadas por cambios inocuos en el software. Requiere mucho más trabajo configurar las pruebas automatizadas que las manuales y el esfuerzo en mantenimiento es mucho mayor también. Muchas veces, esto se vuelve una restricción a cambiar o mejorar el sistema bajo pruebas.

En ocasiones es conveniente desalentar a la inclusión en el software de cambios con alto impacto en la automatización, por razones económicas.

Las Herramientas no Tienen Imaginación

Una herramienta es sólo software, y por lo tanto lo único que puede hacer es ejecutar instrucciones obedientemente. Ambas, herramientas y tester, pueden seguir instrucciones para ejecutar un conjunto de casos, pero una persona puede comportarse diferente para una misma tarea, por ejemplo derivando nuevos casos de prueba dada una situación, aplicando la creatividad, evaluando la conveniencia de ejecutar cierta parte del sistema en el momento mismo de la ejecución (lo cual es un comportamiento muy difícil de modelar para una herramienta, sino imposible). Por ejemplo, los humanos pueden decidir y reaccionar ante situaciones inesperadas de una secuencia de pasos definida.

2.6. Testeabilidad

Uno de los términos menos utilizados en las actividades que involucran desarrollo de software y testing, es "testeabilidad" (*"testability"*).

- ¿Cómo construir un software, pensando en que alguien lo tiene que probar?

- ¿Cuál es la probabilidad de que un software falle?, ¿Cuál es la probabilidad de que este software falle si contiene un defecto?
- ¿Qué tipo de testing puedo mejorar?
- ¿Hay alguna medida relativa a la testeabilidad?, por ejemplo, ¿Se puede medir el aumento de la testeabilidad?
- ¿Hay factores que obstaculizan el testing?

Estas interrogantes, entre otras, están relacionadas con el aporte que nos puede dar manejar el concepto de testeabilidad en sus diferentes acepciones.

2.6.1. Concepto de Testeabilidad

Podemos interpretar a la testeabilidad como la facilidad con la que un software es probado. O asistiendo a la terminación *-bilidad* [Aca01], del idioma español, sería la capacidad con la que un software puede ser probado, en un determinado contexto[15]. La IEEE aporta una definición bastante estricta, [IEE91] que refiere a la facilidad con la que un software nos permite establecer criterios de testing, y cómo se pueden ejecutar los casos de prueba, y así medir si se han alcanzado esos criterios. La discusión se movería al plano entonces, de cuáles serían estos criterios y cómo establecer la medición. El estándar SQuaRE ISO/IEC 25000(SQuaRE) [20005], define el término *testeabilidad*[16] como una sub-característica del atributo de calidad referente a la facilidad de mantenimiento[17].

En términos más prácticos, otro grupo de autores están alineados con la idea de que todo aquello que hace más fácil la tarea de testear un software, ya sea porque es más fácil diseñar los casos de prueba, o podemos testar de forma más eficiente; favorece la testeabilidad [Bol09].

En definitiva, mediante estos enfoques, podemos decir que:

Testeabilidad es la capacidad que tiene un software de ser probado, en un determinado contexto.

Acorde con estas ideas, diremos que un software con mejor capacidad de ser probado en un contexto, es más testeable. Y como consecuencia, un software más testeable, requiere menos esfuerzo de testing.

[15] Se esta tomando la palabra *testeabilidad*, dado que es el término que se usa coloquialmente, aunque no exista en el idioma español.
[16] Referido en [20005] como *"testability"*
[17] Referido en [20005] como *"maintainability"*

Testeabilidad como Visibilidad y Control

Otro enfoque de otro de los autores, dice que hablar de testeabilidad es hablar en términos de visibilidad y control sobre el software [Pet09].

Visibilidad, es la capacidad de observar los estados, salidas, recursos usados y otros efectos secundarios del sistema que estamos probando, y **Control** es la capacidad que tenemos de poder darle entradas al sistema o situarlo en diferentes estados conocidos.

Para concretizar, supongamos que tenemos un proceso batch, con cierta oscuridad hacia el testing, es decir, no sabemos mucho acerca de las entradas, las salidas, ni los diferentes estados. Supongamos que existen además, cálculos complejos. Entonces, algunos ejemplos de testeabilidad, como visibilidad y control:

Visibilidad

- "Hago visible" cierta parte de código asegurándonos que el testing está ejecutándola.
- "Hago visible" los resultados, valores de diferentes variables, campos relevantes.
- "Hago visible" las variables de entrada de cierto código.
- "Hago visible" los datos que tomará, o sobre los que actuará el proceso (puede ser mediante una grilla que representa el estado actual de cierta tabla, o cierto conjunto de datos).

Control

- Puedo ejecutar el proceso.
- Puedo controlar los diferentes estados del proceso, ya sea modificando valores en tiempo de ejecución o cambiando el contexto.
- Puedo pasar parámetros al proceso, simulando condiciones específicas de ejecución.
- Puedo dar entradas, o modificar las entradas, de donde el proceso va a tomar los datos.

En el entendido anterior, podemos pensar en la testeabilidad desde el diseño, lo que favorecería la interrelación del equipo y además reduciría los costos.

Ejecución, Defectos y Propagación

Voas & Miller, definen la testeabilidad en términos probabilísticos[Mil95], motivados por la experiencia ante la interacción con sistemas críticos, donde se tiene que hacer énfasis en la confiabilidad de los sistemas. Por ejemplo, sistemas de misiones espaciales, que comanden un transbordador, o similar. Como en el estándar sobre Reliability de IEEE [IEE06], se busca predecir, dada las características del sistema, cuál es el tiempo para que ocurra una próxima falla, el tiempo medio entre fallas, tratando de predecir que la falla no ocurra dentro del tiempo destinado a una misión. Tal rigurosidad se aplica, en general, a este tipo de sistemas críticos.

Estos autores se han estudiado y modelado cómo los errores se "esconden del testing", manejando términos de "enmascarado de errores" (*error masking* en la bibliografía), y la pérdida de información en el transcurso de la ejecución de los casos de prueba. Se refiere a que, en ocasiones, un software es ejecutado, el software contiene defectos, pero como consecuencia de ello, no se manifiesta una falla u error observable en el sistema. Es decir, como característica conjunta del software con defecto, el contexto y determinado conjunto de casos de prueba, no se manifiesta una falla. Aumentar la testeabilidad, es aumentar la probabilidad de dado un conjunto de casos de prueba, un contexto y un software con defecto, dicho defecto se propague una falla.

En los artículos referenciados, se cita el ejemplo de la división entera, donde claramente se pierde información, dado que la división de cuatro entre dos, da el mismo resultado que cinco entre dos.

2.6.2. Diseñar para la Testeabilidad

En el artículo *"Design for Testability"* [Pet09] que hemos estado citando, se maneja además cómo se puede relacionar la testeabilidad y el diseño del producto, desde dos perspectivas: *"desde dentro"* y *"por fuera"*.

Diseñar para la Testeabilidad desde Dentro.

Diseñar para la testeabilidad, es construir el software pensando que alguien tiene que probarlo. Aportar visibilidad y control desde el diseño, identificando las necesidades del testing, construyendo puntos de verificación, trazas y log de eventos, monitoreos, interfaces de testing en general.

Diseñar para la Testeabilidad por Fuera

El no tener presente la testeabilidad desde la concepción o desde el diseño del producto, no es impedimento para obtener beneficios de este concepto. Es posible brindar visibilidad y control con construcciones externas al software, ya sea mediante interfaces gráficas que controlen ciertas variables, o que muestren resultados o diferentes estados de las variables. Pero, no solamente

es necesario contar con complejas piezas de software, sino que muchas veces los desarrolladores en sus pruebas de integración o modulares elaboran sus propias herramientas que les hacen más fácil la tarea de probar sus códigos, y ese material puede ser en muchas ocasiones reutilizado por el tester. Incluso, los usuarios suelen construirse herramientas para apoyar sus tareas diarias, en planillas electrónicas por ejemplo; información que es de extrema utilidad al tester, dado que puede llegar a motivar nuevas pruebas, enriquecer los oráculos y en definitiva facilitar la tarea de testear el software.

2.6.3. Diseñar para la Testeabilidad de la Automatización

Verificar software que no ha sido diseñado para que sea fácil de testear requiere mucho más esfuerzo. Por ejemplo, si el sistema genera un conjunto complejo de datos, hace validaciones de las entradas de usuarios y chequeo cruzado de valores, antes de ingresar un conjunto de datos a la base o algún sistema, entonces puede ser útil verificar esos datos justo antes de que sean enviados. Es decir, es más fácil verificar estos datos justo en la salida, que cuando ya están ingresados en diferentes lugares en la base de datos [FG99f].

Acceder a datos intermedios es una forma de colocar *"test hooks"* ("arnés" o "gancho" de testing podrían ser posibles traducciones)[18]. de donde se puede servir la automatización para simplificar la verificación. La cantidad de estos *test hooks* y su granularidad, determinan cuán testeable es el sistema. Además, estos puntos de acceso a datos intermedios aportan información. Un testing carente de información se torna poco eficiente y poco efectivo.

El software bajo prueba, también puede ser mejorado en pro de la automatización. Por ejemplo, si la aplicación incluye un "modo testing" o similar, donde se pueda consumir datos desde un archivo en lugar de la pantalla, entonces la aplicación puede configurarse incluso para adelantar camino en la automatización, como si se "testease a si misma" mediante configuraciones especiales en un archivo.

El mejor camino para aportar testeabilidad es pensar acera de cómo va a ser verificado el software desde los inicios, desde el diseño y en la implementación. Pensar qué debe ser testeado y cómo puede ser testeado puede resultar en mejores diseños y mejores aplicaciones, y además ayudará a las tareas de depuración de código.

2.6.4. Discusiones sobre Testeabilidad

A continuación mencionamos brevemente cómo se relacionan testeabilidad y seguridad, y luego discutiremos algunos beneficios que nos puede brindar la testeabilidad.

[18] Una definición más precisa en [IEE91], bajo *"test harness"* o *"test driver"*.

Interfaces de Testing y Seguridad

Un tema importante, es que las interfaces de testing pueden transformarse en baches de seguridad. Es necesario prestar especial atención a las construcciones específicas de testing, que aportan más información que la que las políticas de seguridad de una organización quiere o debe revelar, para que no lleguen finalmente a producción, y que además, su no inclusión, u ocultación no ocasione nuevos incidentes, ni sea fácil de "quebrar". Imaginemos que contamos con una interfaz de testing que monitorea los estados de una transacción bancaria, y por algún motivo la ocultamos en producción, y algún atacante puede tomar su control; claramente, estamos ante un muy grave agujero de seguridad que puede evitarse fácilmente teniendo control sobre cuáles son las interfaces de testing y cómo y cuándo están disponibles.

Beneficios de la Testeabilidad

Del análisis de la bibliografía, y derivado de la experiencia, podemos concluir un conjunto interesante de beneficios relacionados con la testeabilidad. Como ser la obtención de **productos más testeables o más fáciles de probar**, en un determinado contexto, redundando en que es posible enfocar el trabajo en encontrar los errores más importantes, presumiblemente, de manera más temprana. Logramos **interfaces más fáciles de automatizar** y guionar, **más puntos de verificación**, más lugares de donde la automatización se pueda servir, mejorando en eficiencia. Los testers encuentran un buen camino para **incrementar sus habilidades** técnicas, y poder crear sus propias herramientas. La comunicación y la colaboración mejoran, lo que favorece la **consolidación del equipo**. Los documentos y reportes de incidentes se vuelven más certeros y claros, así como también aquellos interesados, o líderes, quienes tengan que tomar decisiones basados en informes de testing, obtienen **mejor y más relevante información**.

Podríamos incluso, hasta llegar a hablar de **requerimientos de testeabilidad**, por ejemplo, para que las organizaciones incluyan en sus licitaciones[19]. En definitiva, estamos logrando un mejor compromiso con la calidad, reforzando el fin común de tener un mejor producto.

[19]Es cada vez más común que las organizaciones adquieran productos adaptables, en lugar de soluciones llave en mano, dada la complejidad de los dominios actuales. Sobre todo para complejos productos de gran porte. Usualmente la implementación de estos productos en una organización es un proceso largo y costoso, que involucra desarrollo particular. Contar con requerimientos de testeabilidad puede ayudar al desarrollo del proceso de implantación.

Enumerando algunos de los beneficios (categorizados por actor implicado) podemos mencionar:

- **Para los Testers**
  - Obtener productos más testeables o más fáciles de testear.
  - Enfocar su trabajo en encontrar los errores más importantes.
  - Poder automatizar una interfaz más fácilmente.
  - Desarrollar sus habilidades técnicas, crear.
  - Mejor comunicación con el equipo de desarrollo entendiendo el diseño del software revisando dando a conocer las funcionalidades que necesita.
- **Para los Desarrolladores**
  - No reciben reportes de incidentes triviales.
  - Mejor comunicación con los técnicos de testing.
  - Mejores reportes de incidentes, reportes más claros, con más información, más detallados.
  - Considerando la testeabilidad en el diseño.
  - Compartiendo conocimiento.
    - Código.
    - Documentos de diseño.
    - Catálogos de error, entre otros.
- **Líderes de Proyecto**
  - Equipo más sólido y colaborativo.
  - Un mejor compromiso con la calidad del producto, reforzando el fin común.
  - Información más relevante para la toma de decisiones en cuanto a los incidentes, coberturas, entre otros.

2.7. Discusión: Testing Manual y Automatizado

"Lo mejor de los dos mundos"

Presentamos en esta sección algunas consideraciones entre el testing manual y automatizado. Analizando qué relación existe entre la calidad de un caso de prueba y su automatización, y qué consecuencias tiene esto para el testing.

2.7.1. Relación entre Testing Manual y Automatizado

Antes de contrastar ambos conceptos, hablaremos de los *Atributos de Calidad de un Caso de Prueba,* [FG99g], lo que aportará precisión a la comparación.

Ya sea un caso de prueba rigurosamente derivado de la aplicación de alguna técnica, sea generado automáticamente por algún algoritmo, o provenga de la creatividad de un tester aplicando estrategias exploratorias; en términos generales, podemos definir la calidad de un caso de prueba en los siguientes cuatro atributos:

Efectividad para encontrar defectos. Es uno de los más importantes atributos, y refiere a si el test encuentra o no defectos, o al menos si tiene alguna probabilidad de encontrar defectos.

Atributo de Ejemplaridad. Este atributo habla de cuando un caso de prueba abarca más de un aspecto del software a la vez, lo cual favorece a que el conjunto de casos de prueba final tenga menos elementos.

Atributo económico de un caso de prueba. Mediante este atributo se expresa que tan económico es ejecutar, analizar y depurar un caso de prueba.

Caso de prueba Evolucionable. Por último, este atributo nos dice en qué medida el caso de prueba es mantenible, adaptable, evolucionable acorde a los cambios del software. Hay autores que manifiestan que un caso de prueba que no cambia con las sucesivas versiones de un producto, pierde efectividad, dado que a lo sumo encontrará la misma clase de incidentes que encontró en un principio [Bac02][20].

Si bien la calidad es un concepto de difícil medición; para un contexto dado, se busca seleccionar un conjunto de casos de prueba que balancee estos cuatro atributos, en pro de encontrar la mayor cantidad de defectos a un costo razonable.

Como es sabido, es imposible o cercanamente imposible ejercitar todos los casos de prueba aún para una pieza muy sencilla de código [Mye04b][21]. Con los años,

[20] Bajo la sección "Exploratory Testing is a Profoundly Situational Practice", página cuatro de la versión 1.3 del artículo.

[21] Myers muestra la discusión un ejemplo muy sencillo de un triángulo, que tiene una cantidad inabarcable de casos de prueba.

el software se ha vuelto cada vez más complejo, y la cantidad de combinaciones, situaciones, y en definitiva, casos de prueba es prácticamente infinita para un software dado en determinado contexto. **El testing manual**, tiene gran parte de su éxito comprometido en la habilidad de seleccionar de ese universo de casos de prueba, el subconjunto más adecuado; que detecte la mayor cantidad de defectos posible en el software, dadas las circunstancias (consideraciones del negocio, cobertura, tiempo disponible, entre otros).

En definitiva, la destreza en el testing manual radica en balancear los cuatro atributos de calidad de un caso de prueba (caso de prueba efectivo, ejemplar, económico, evolucionable), logrando que los casos de prueba a ejecutar encuentren una gran proporción de defectos a un costo *"razonable"*.

La habilidad requerida para el **testing automatizado**, y su aporte difiere enormemente del testing manual. Es fácil notar que el costo de automatizar una prueba, es bastante más alto que el costo de ejecutarlo manualmente; simplemente desde el hecho de que alguien se tiene que dedicar a especificar los pasos y los detalles de la ejecución de las pruebas, puntos de verificación, resultados esperados, entre otros; y ese costo es al menos igual o mayor que el costo de ejecutarlo una vez manualmente. La experiencia indica que el costo de automatizar, es bastante alto, por lo tanto las pruebas a automatizar deben ser elegidas cuidadosamente.

El punto a dejar en claro es que **la calidad de la automatización, es independiente de la calidad de las pruebas.** Es decir, un caso de prueba que no es **efectivo**, al ser automatizado, no cambiará este atributo, solamente se ejecutará más rápido; pero seguirá siendo pobre en efectividad.

La gran diferencia de peso es que, una vez automatizado, un caso de prueba se vuelve mucho más **económico**. Es de esperar, naturalmente, que con las sucesivas pruebas, el costo de correr el caso de prueba automatizado, sea extremadamente menor al de correr la misma prueba manualmente. Por otra parte, es mucho más costoso crear, mantener y **evolucionar** las pruebas automatizadas. Es conveniente que se dedique esfuerzo a las primeras etapas de la automatización, seleccionando los casos de pruebas adecuados, con el fin de abaratar costos de mantenimiento en el futuro. Las pruebas automatizadas también son software, por lo tanto implican verificación y mantenimiento. Si este trabajo se vuelve muy costoso, la prueba automatizada empieza a perder fuerza como alternativa a la prueba manual.

El insumo fundamental para conseguir un buen conjunto de pruebas automatizadas, es partir de *"buenos casos de prueba"*[22].

[22] Por ejemplo en el entendido de los cuatro atributos de calidad de un caso de prueba, antes mencionados.

El encargado de implementar y mantener las pruebas automatizadas, es el *"automatizador de pruebas"* o *"tester automatizador"*. Inclusive, este rol puede ser llevado adelante por una persona con habilidades técnicas de desarrollo, dado que, si está bien especificada la prueba, pueden implementar el código de la prueba automatizada sin grandes inconvenientes. Inclusive, si los testers carecen de conocimiento técnico, de todas formas, se puede trabajar en esta dinámica conjunta de definición de los casos de prueba desde la perspectiva del negocio e implementación de los mismos desde el punto de vista de la programación de las pruebas automatizadas.

Como conclusión, la buena o mala calidad del testing automatizado, es independiente de la buena o mala calidad de la automatización. Si el automatizador de testing logra pruebas fáciles de mantener y consigue que sea posible agregar nuevas pruebas a las suites con facilidad; entonces la automatización brindará mejores beneficios (atributo **económico**, y atributo **evolucionable**), lo cual no quiere decir nada acerca de cuan **ejemplares**, y **efectivos** sean los casos de prueba.

2.8. Resumen

En este capítulo hemos presentado una síntesis del estado del arte del testing funcional, mostrando modelos de mejora de procesos de testing, procesos de testing, estrategias y otros temas. Inspirados en estos temas y en el marco de la reestructura del área de testing de ASSE; planteamos el desafió de crear una metodología particular que se ajuste a las necesidades de ASSE en cuanto a testing funcional. El capítulo siguiente, muestra un informe de diagnóstico inicial de la situación de ASSE en cuanto al testing funcional. Luego en el capítulo 4 se detalla la metodología desarrollada a partir del estado del arte en testing funcional y el diagnóstico inicial de la organización.

Capítulo 3

Informe de Análisis y Diagnostico Inicial

3.1. Introducción

En este capítulo se detallan las tareas desarrolladas y las conclusiones obtenidas para estudiar el departamento de pruebas de ASSE en relación con los proyectos de desarrollo, tomando como caso de estudio el Sistema de Gestión de Salud (SGS) principalmente.

Para estudiar la organización se desarrollaron reuniones con personal involucrado con el SGS, lo que aporta información para profundizar en el análisis, obteniendo además propuestas por parte de los integrantes de ASSE, trabajando en conjunto para entender la problemática y generar ideas de cómo solucionarlas.

Se estudió también documentación entregada por ASSE de los diferentes sistemas y documentos que definen la forma de trabajo a la que apunta el área de testing.

En este documento se presentan los inconvenientes detectados y sugerencias para solucionarlos. Las recomendaciones se construyen en base a las sugerencias de los integrantes de ASSE y el análisis hecho por el grupo de estudiantes de este proyecto de grado.

3.2. Actividades desarrolladas

En una primera etapa se llevaron a cabo las siguientes actividades para evaluar el estado de la organización desde el punto de vista del testing:

1. Entrevistas con los responsables de áreas estratégicas.

2. Análisis de los casos de las funcionalidades más usadas.
3. Análisis de la documentación existente
4. Análisis de fortalezas y debilidades

Se contó con alrededor de cinco instancias de reunión con modalidad entrevista guionada[1], con los distintos actores involucrados al SGS. Participaron además, en su mayoría, Carlos Gutiérrez y Fernando Pereira con motivo de seguir y aprovechar el avance del testing de este proyecto e involucrarse con la interna del desarrollo de estos sistemas.

3.2.1. Entrevistas

Se mantuvo entrevistas con Carlos Gutiérrez y Fernando Pereira, ambos involucrados directamente en la formación del departamento de testing; Homero Muñoz, referente funcional del SGS; Diego Lorenzo, líder de desarrollo por parte de la empresa Hexa y Ariel Sabiguero como apoyo general del proyecto.

3.2.2. Análisis

Complementando la información obtenida de las entrevistas, se analizó una serie de documentos y una herramienta, que fueron puestos a nuestra disposición.

Los documentos vistos y la herramienta analizada son:

1. Documento de casos de uso del sistema SGS
2. Planilla con ranking de casos de uso más ejecutados en el sistema SGS
3. Documento de Arquitectura Simplificado del sistema SGS
4. Mantis, usada para seguimiento de incidentes y pedidos de los requerimientos
5. Propuesta de Proceso de Testing en etapa de elaboración, por parte de los testers de ASSE.
6. Manual de Usuario del sistema SGS.

3.3. Diagnóstico

En esta sección presentamos los resultados obtenidos del análisis de la organización. Se quiere destacar que el énfasis se puso en detectar las dificultades y proponer alternativas para sortearlas, más que en destacar el trabajo ya avanzado y logrado en ASSE.

[1] La entrevista guionada se basa en la elaboración para cada reunión de un guión con los puntos fundamentales a tratar, sin perder el aporte libre de cada participante. De esta manera se logra un productivo intercambio de ideas. En todos los casos se solicitó previamente información documentada para estudiarla y lograr un mejor entendimiento de la realidad.

3.3.1. Estado Actual del Testing

La división informática de ASSE está enfrentándose a cambios, habiéndose incorporado y redistribuido recursos. También está teniendo un aumento importante en la cantidad de usuarios finales de su principal sistema, el SGS.

ASSE tiene la preocupación de instrumentar y mejorar sus actividades de testing, dado que lo considera como un elemento importante en el desarrollo para mejorar la calidad de los productos y lograr más confianza en el software al momento de salir a producción.

Durante el desarrollo se hacen pruebas básicas a diferentes niveles (unitarias, pruebas de integración y de sistema), pero no muy frecuentemente. La información de las pruebas de desarrollo es a menudo consultada por el equipo de testing para mitigar la falta de requerimientos y especificación.

3.3.2. Control y alcance de las pruebas

En el desarrollo de las aplicaciones, es poco frecuente la incorporación de nuevo código, dado que se re-utiliza cierta funcionalidad ya implementada y se hacen los cambios pertinentes. Por la parte de testing, no se lleva un control de lo que se probó y es dificultoso identificar las versiones del producto. No se exige a los equipos de desarrollo un mínimo de pruebas, quedando en última instancia a criterio del desarrollador.

3.3.3. Área de testing

En un principio y debido a los limitados recursos, solamente se asignan a desarrollo. Luego ASSE asume la responsabilidad del testing y crea el área de testing, integrada por dos personas con perfil técnico-informático. Este equipo se encarga de todas las actividades de verificación, lo cual representa una sobrecarga de trabajo y trae como consecuencia que muchos proyectos de desarrollo no cuenten con las pruebas necesarias antes de salir a producción. De los sistemas que son verificados, en muchas oportunidades las pruebas se reducen a una leve verificación de usuarios funcionales destinatarios de la aplicación. El rol de los testers está muy diluido entre actividades de soporte a usuarios, ya sea por el conocimiento acumulado en los productos como otras actividades no vinculadas con el testing en sí.

Capacitación técnica en testing

En nuestro análisis, no detectamos antecedentes de instancias de capacitación o talleres en la temática del testing funcional ni testing funcional automatizado. El conocimiento residente en los integrantes es merito propio, a raíz de interés y motivación personal.

3.3.4. Proceso de testing

A la hora de efectuar las pruebas, se sigue un proceso de testing *ad hoc* no documentado. Con la inclusión de recursos específicos para testing, se inicia la tarea de documentación de la metodología seguida y algunas propuestas con las actividades deseables a seguir; con el fin de explicitarlas y mejorarlas. Esta formalización todavía está en curso y en etapas iniciales por parte de ASSE.

3.3.5. Procesos de desarrollo

El desarrollo de los sistemas está distribuido entre varios proveedores, y desarrollo interno a la organización. A los efectos, muchas de las empresas que tienen a su cargo desarrollos parciales de los diferentes sistemas, funcionan como parte íntegra de un equipo de desarrollo heterogéneo y ligeramente gestionado. Cada proveedor decide y lidera sus actividades de testing, y resulta dificultoso identificar las diferentes actividades como un único proceso.

3.3.6. Especificación y documentación de requerimientos

En cuanto a los requerimientos, no está especificado qué interesa documentar y hasta qué punto. Se generan pedidos en la herramienta de seguimiento de incidentes (Mantis), que oficia a estos efectos, de repositorio de requerimientos; en general es la única documentación de requerimientos con la que se cuenta.
Existe documentación bastante avanzada sobre la arquitectura; además de casos de uso, con los flujos típicos, y en ocasiones cuentan con flujos alternativos.
Entre los temas relacionados, detectamos interés en mantener trazabilidad entre casos de uso, módulos y objetos.

3.3.7. Diversas fuentes de requerimientos

Dependiendo del destino de los sistemas, existen distintos tipos de requerimientos de diversas y múltiples fuentes. Por ejemplo, los hospitales del interior y capital pueden tener diferencias de funcionamiento. Por lo tanto se debe personalizar además, una misma funcionalidad para cada ubicación. Esto dificulta tener una versión única y que los cambios ocurridos no impacten sobre el sistema de otros hospitales.

3.3.8. Requerimientos de calidad para tercerización

A la hora de licitar y contratar empresas para implementar los diferentes sistemas; no se establecen parámetros claros de calidad; esto es, no se incluye en las licitaciones temas que especifiquen en qué estado deben estar los entregables, qué tiempo deben cumplir las entregas, cómo se negocian y se encargan los desarrollos. En definitiva, cuáles son los atributos de calidad de interés, y qué se considera satisfactorio.

3.3.9. Problemas en el sistema SGS

Del análisis de la información relevante al sistema SGS, se identifican los siguientes puntos como destacables:

1. Los datos, tanto de la base local del SGS como los del padrón están corruptos. Esto se debe mayormente al poco control en la interfaz de ingreso.

2. El sistema de seguridad es muy débil. Se quiere avanzar sobre esto, encriptación de contraseñas, política de contraseña, etc..

3. En el caso de la farmacia, la complejidad del negocio es muy alta, tanto sea para modelarla en software, como para testearla.

    a) El VadeMecum es una tabla que indica cuales medicamentos son crónicos y cuáles no. Pero esto es un problema ya que en distintas instituciones el mismo medicamento puede ser crónico o no.

    b) La presentación de los medicamentos es variable, dependiendo del laboratorio que lo produzca.

3.4. Recomendaciones

En esta sección, mostramos las recomendaciones de acciones a tomar para los temas identificados en el diagnóstico inicial. Cada punto del diagnóstico cuenta con una o más recomendaciones, compuestas por un título y una descripción.

3.4.1. Control y alcance de las pruebas

Establecer una política de gestión de la configuración.

Como punto de partida se debería poder identificar los productos y sus versiones, para poder tener noción de qué versiones están instaladas en los diferentes ambientes, y en qué estado están, es decir, si se puede ejecutar pruebas sobre ellas.

Generar contratos entre testing y desarrollo para la entrega de los productos

Exigir a los diferentes equipos de desarrollo como parte de su entrega la ejecución de pruebas de humo, implementación de pruebas unitarias y de integración con el fin de asegurar la integridad del producto y reducir los ciclos de consultas y respuestas entre testing y desarrollo al evitar los incidentes más básicos.

3.4.2. Área de testing

Incorporación de recursos para testing

Para reforzar el área de testing se recomienda incorporar personal capacitado. Es necesario contar con personal fijo, además de brindarles capacitación permanente y eventual apoyo de especialistas que puedan guiar en temas puntuales.

Se recomienda como mínimo integrar un Tester Líder, con experiencia en gestión de proyectos, un Tester Automatizador, y dos Testers de Profesión.

Capacitación en Testing

Incorporar programas de capacitación e incluso certificación de los todo el equipo de testing para asegurar su continua evolución en la disciplina.

Los beneficios de la capacitación son conocidos, pero es de destacar que promueve la nivelación de los integrantes en los conocimientos y conceptos, manejo de vocabulario común e incorporación de nuevas habilidades.

Dedicación total de los integrantes al testing

Las actividades que no estén relacionadas con la disciplina se deben delegar a otras personas fuera del área de testing. Diversificar las tareas de los integrantes reduce aún más los recursos disponibles y desvía el foco del testing.

3.4.3. Proceso de testing

Definición de proceso de testing

Contar con una consultoría en testing que defina una forma de trabajo acorde a la organización que integre a los equipos de desarrollo, usuarios funcionales y a los diferentes actores involucrados. Reutilizar el material iniciado por los testers de ASSE como insumo para la propuesta de la forma de trabajo.

3.4.4. Procesos de desarrollo

Definir forma de trabajo entre equipos

Con el fin de organizar el proceso de desarrollo y su integración con el proceso de testing a seguir, se sugiere el modelado de un flujo de trabajo que involucre la gestión de los cambios solicitados, los diferentes proyectos, los incidentes relacionados y la incorporación gradual de una herramienta capaz de integrar este flujo de trabajo para que sea fácil de seguir, gestionar, y mantener.

Ante la realidad de varios proveedores con diferentes formas de trabajar, se recomienda definir un proceso estándar que deben seguir los equipos de desarrollo e incluir esto en las licitaciones y contratos. Para este fin, se

recomienda contar con una consultoría en temas relacionados a ingeniería de software.

3.4.5. Especificación y documentación de requerimientos

Definir el nivel de interés de la organización por la documentación

Llegar a un entendido de qué tipo de información se requiere documentar, para que los productos ganen en mantenibilidad y a su vez la organización pueda auditar y controlar sus procesos. Esta definición puede ser que no interesa contar con documentación detallada, pero no se recomienda que quede a criterio de cada equipo. Es importante tener en cuenta que la documentación requiere una carga trabajo importante, por lo tanto la documentación exigida y el nivel de detalle debe ser estudiado detenidamente por la organización.

Exigir los niveles de documentación acordados

Establecer con los diferentes equipos (y de ser tercerizados, en sus contratos) la obligatoriedad de presentar la documentación definida por la organización junto con los productos construidos.

Participación de Testers en definición de requerimientos

Se sugiere incorporar a los integrantes del equipo de testing en etapas tempranas de definición de requerimientos siempre que sea posible y esto no obstaculice otras actividades. La visión de los encargados de las pruebas es enriquecedora, dado que difiere del punto de vista de quienes tienen que construir un producto. Es común que se disipen ambigüedades, zonas grises, y se encuentren errores en la definición en estas etapas, sólo por el hecho de contar con la opinión del equipo de testing.

Herramientas específicas para modelar pedidos de cambio y asignación de tareas

Contar con herramientas específicas para determinado fin, (o una herramienta capaz de modelar fielmente toda la realidad).

Las herramientas de seguimiento de incidentes, pueden compartir información con otras herramientas de asignación de tareas y control de cambios, pero no es recomendable utilizar el flujo de trabajo y los conceptos definidos para un fin, con otro fin diferente.

3.4.6. Diversas fuentes de requerimientos

Documentación de las configuraciones particulares

Registrar la personalización particular de los sistemas distribuidos y sus funcionalidades; validar dicha información con las autoridades pertinentes.

Exigir sistemas configurables

Exigir a los proveedores que sus sistemas sean diseñados para que sean fácilmente configurables y extensibles, mitigando el riesgo de que una configuración particular afecte a los sistemas ya implantados.

3.4.7. Requerimientos de calidad para tercerización

Definir atributos de calidad e incluirlos en licitaciones y contratos

La organización debería identificar cuáles son los parámetros de calidad que necesita a la hora de aceptar un entregable y qué nivel se acepta de cada uno. Esto debe ser claramente especificado en licitaciones y contratos.

Algunos atributos y sus valores podrían ser:

1. Tiempo de entrega: en fecha, fuera de fecha, rango de tolerancia.
2. Estado de las funcionalidades y el producto: en desarrollo, inestable, estable, cerrado y verificado.
3. Documentación: cumplir con el mínimo exigido, no se exige documentación, manuales de usuario y requerimientos, documentación de las pruebas ejecutadas, y muchas otras combinaciones de documentos que se puedan requerir.
4. Soporte: requisitos de disponibilidad de soporte, asistencia a usuarios, etc.
5. Antecedentes: por ejemplo si se requiere experiencia en proyectos similares o no.

3.4.8. Problemas en el sistema SGS

Revisar la política de entrega de software

Establecer y negociar la política de entrega de software mediante la definición de atributos de calidad. Exigir que las funcionalidades complejas cuenten con documentación detallada de su funcionamiento.

Definir los atributos de seguridad y de uso del software

La organización debe tener claro los requerimientos de uso (usabilidad, facilidad de uso) y los atributos de seguridad deseados acorde a la información que se maneja. Para este fin se puede contar con profesionales que estudien el sistema desde este punto de vista, identificando problemas en el uso e interacción con el producto, así como también baches de seguridad.

Contar con equipo de Testing dedicado a sistemas críticos

Es deseable que por un período de tiempo prolongado se cuente con recursos de testing dedicados a la verificación de los sistemas críticos y complejos, dado que implica una ventaja contar con más integrantes que conozcan el comportamiento del sistema. De ser posible, que se puedan incorporar en etapas tempranas del proceso de desarrollo.

3.5. Conclusiones

El área de desarrollo de ASSE presenta una alta heterogeneidad, los sistemas actúan sobre dominios complejos, y el área de testing cuenta con muy reducidos recursos para afrontar la responsabilidad de llevar adelante el testing de todos los productos.

El software construido es generalmente propenso a errores y no existen procesos definidos en cuanto a desarrollo y testing de software.

Las recomendaciones se basan en atacar los problemas relativos a la carencia de personal en el área de testing, definición y mejora de los procesos de testing y desarrollo, así como también acuerdos de calidad que mitiguen el riesgo de contar con entregables de software demasiado inestables, tanto para las empresas proveedoras como los equipos internos de ASSE.

En el siguiente capítulo se detalla la metodología desarrollada para ASSE a partir del estado del arte y la conclusiones extraídas del diagnóstico inicial de la organización frente al testing funcional.

Capítulo 4

Framework Instanciable de Testing

"La perfección es una pulida colección de errores"

Mario Benedetti

4.1. Introducción

La metodología propuesta estará contenida en el Framework Instanciable de Testing, (también referido en adelante por su sigla *FIT*). Este marco de trabajo, o *framework* esta dividido en diferentes módulos. Estos módulos tratan temas estrechamente relacionados. El módulo inicial, corresponde a la identificación del estado de la organización en cuanto al testing. Los siguientes módulos definen procesos de testing funcional manual y automatizado. Se puede extender esta forma de organizar los conceptos en nuevos módulos. A modo de discusión, se proponen algunos de los posibles módulos que pueden ser agregados, involucrando tareas de gestión e integración de procesos globales de desarrollo y testing.

4.2. Justificación de Módulos y Procesos

El Framework Instanciable de Testing, está organizado en módulos. Los módulos tienen diferentes características. El primero supone una identificación del estado inicial de la organización, no cuenta con actividades en sí mismo, sino que es un punto de partida simbólico. Los restantes módulos contienen procesos específicos, es decir *una secuencia de pasos llevados adelante con un propósito dado*[1].

[1] Ver *process* (1) en [IEE06].

En esta propuesta es posible transitar por los diferentes módulos, involucrando algunas etapas y actividades de los procesos definidos, sin tener que necesariamente cumplir con todas ellas. Cada proyecto particular motiva un camino a seguir entre las etapas y actividades. Este camino es la instancia del marco de trabajo propuesto y es llamado *"roadmap"*. Cada organización que ponga en práctica el FIT, podría por ejemplo, definir un *roadmap* genérico a modo de plantilla para proyectos de similares características.

La instanciación de la metodología en un *roadmap* se puede ejemplificar en dos dimensiones, una dimensión se puede interpretar como las etapas que se van a ejecutar, y dentro de esas etapas cuáles actividades; otra dimensión, puede entenderse como la rigurosidad o profundidad con la que se ejecuta esa etapa o actividad, esto es, si se sigue su propósito detalladamente y se documenta estrictamente el proceso, o si se transitan las etapas y actividades tomando en cuenta su punto central, sin detenerse a registrar el proceso abordado. En el apéndice B de instrumentación se muestran algunos ejemplos de qué tipo de factores considerar para construir el *roadmap* dadas ciertas características y restricciones de un proyecto dentro de una organización. En esta última dimensión, además de las exigencias del contexto (por ejemplo decisiones de la organización, externas al alcance de testing) depende también del conocimiento y experiencia del tester, dado que como entrenamiento, el proceso puede aplicarse de manera formal, hasta hacerlo más laxo, dónde las etapas y actividades se interpretan más como una lista de tareas en las cuales es necesario pensar a la hora de encarar un proyecto de testing.

Los módulos definidos para organizar el marco de trabajo son:

- FIT Módulo I: Testing Ad-Hoc
- FIT Módulo II: Proceso de Testing Funcional Manual
- FIT Módulo III: Proceso de Testing Funcional Automatizado

Esta forma de organizar la propuesta metodológica está pensada para que sea posible extender el marco de trabajo con otros módulos.

Como se discute en el capítulo 6 se podrían agregar nuevos módulos, a modo de ejemplo, en la figura 4.1 se muestran en rectángulo punteado dos de los posibles:

- FIT Módulo IV: Proceso de Testing Gestionado
- FIT Módulo V: Proceso de Desarrollo y Testing Integrados

La figura 4.1 ejemplifica cómo debe interpretarse la división en módulos del FIT[2].

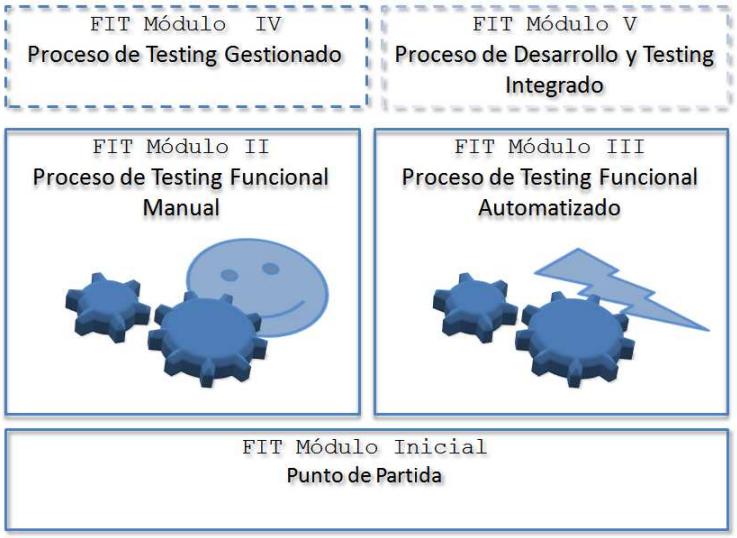

Figura 4.1: FIT con sus módulos.

Un *roadmap* consiste en las actividades y etapas correspondientes a los procesos de los diferentes módulos, que aplican para un proyecto de testing particular en un contexto dado. El objetivo es llegar a un camino que conduzca al éxito del proyecto. En la figura 4.2, se muestra una representación conceptual de un camino que parte del módulo inicial y atraviesa los módulos II y III (en el gráfico, se muestra con una flecha quebrada atravesando los rectángulos etiquetados con el nombre de cada módulo).

[2]Los módulos definidos tienen borde sólido, mientras que los ejemplos de futuros módulos están delimitados con borde punteado.

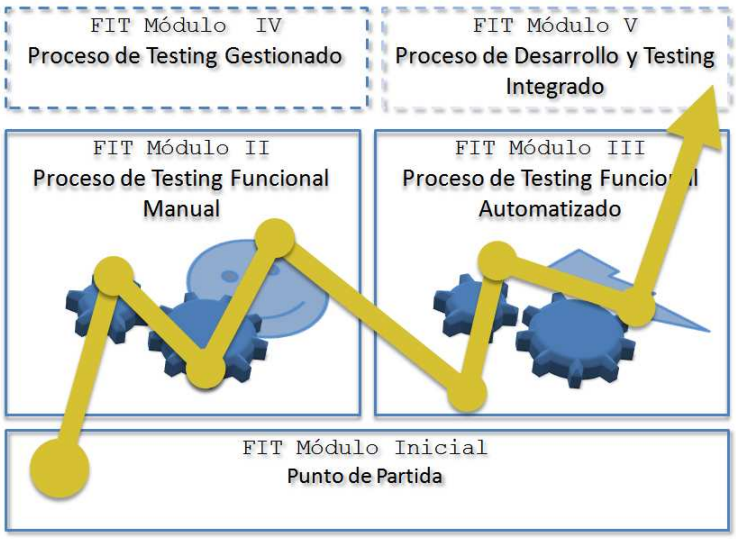

Figura 4.2: FIT con sus módulos y representación gráfica del roadmap.

Cada proceso cuenta con diferentes **etapas** las cuales contienen **actividades**. Las actividades constan de un identificador, un título y una descripción. Hay ciertas actividades que son **requeridas** en su etapa, esto significa que la actividad debe ser ejecutada, aunque no sea en detalle; por ejemplo, como se ve en la sección 4.4.1, en la actividad *"Seleccionar estrategia y enfoque"* se puede optar por no hacer testing diseñado, o por no tomar una estrategia exploratoria para el transcurso de las pruebas, pero es explícito que se debe ejecutar la actividad, y definir una estrategia.

Algunas etapas tienen precedencia con otras, así como algunas actividades dentro de una etapa. De ser necesario para aportar claridad, se especifica con diagramas la precedencia de las etapas, y de las actividades entre sí. Las etapas y actividades se indican con rectángulos. Una actividad o etapa que pueda ser ejecutada en paralelo a las demás, se indica con un rectángulo que no está en la precedencia sino que está presente a la izquierda o derecha del diagrama, cuyo largo abarca todos los elementos que se ejecutan en paralelo.

Cada elemento en el diagrama está etiquetado por su identificador y su título. Una flecha con dirección desde un elemento A hacia un elemento B, indica que el elemento A debe (o se sugiere) ser completado antes que el elemento B[3]. Las actividades y etapas requeridas tendrán un asterisco junto a su identificador (ejemplo, SEE* en la etiqueta de un elemento, indica que la actividad de identificador SEE es requerida).

[3]B podría necesitar como entrada resultados generados en A.

4.3. FIT Módulo I: Punto de Partida

Es común que las organizaciones tengan diferentes percepciones del testing, por ejemplo, a menudo suele tomarse como una actividad secundaria, anexa al desarrollo y muchas veces es confundido con depuración de código. También sucede que se desconocen los diferentes niveles en los cuales se puede implementar testing, y cómo hacerlo. Algunas organizaciones recurren a los usuarios del sistema y expertos del dominio para colaborar con el testing, pero son involucrados de forma tardía y muchas veces es el único testing que se práctica sobre el sistema.

Este es el módulo inicial, y su alcance es muy amplio. El objetivo del módulo es establecer un punto de partida a través de la identificación de las características y restricciones de la organización y el proyecto particular. El análisis de cada contexto pone en manifiesto cuáles los puntos fuertes y las carencias en la forma en que el testing es llevado a cabo hasta ese momento.

Usualmente, el punto de partida detecta que el testing funcional se da en un contexto donde no están claras las actividades, los pasos no son repetitivos o medibles, los roles y las responsabilidades están débilmente delimitadas y el testing es tomado como una actividad posterior al desarrollo y no como un proyecto o disciplina en sí misma.

Si bien las organizaciones pueden estar preocupadas por la calidad, y aunque puedan tener la intención de hacer testing, es común que al no seguir un proceso se presenten carencias, por ejemplo, no saber cómo implementar tareas de testing, cómo organizarse, cómo controlar las pruebas ejecutadas y el avance, y conocer qué funcionalidades fueron cubiertas.

En el módulo inicial, se debe conocer cómo trabaja actualmente la organización, con la intención de rescatar las buenas prácticas y tareas actuales que se ajustan con los procesos de testing funcional manual y automatizado, potenciando su uso. Es importante detectar las carencias o ausencia de actividades, así como también prácticas no deseadas.

La identificación del contexto inicial, consta de:

- **Identificar actividades existentes**
  - **Prácticas de desarrollo:** la forma en que se lleva adelante el desarrollo y mantenimiento de los sistemas.
  - **Ingeniería de requerimientos:** cómo se elaboran los requerimientos en la organización, cómo se definen, quién los valida, quienes participan.
  - **Prácticas de testing:** en qué manera se ejecuta la verificación y validación de los productos, quiénes participan, qué peso tiene en la organización.
  - **Forma de trabajo en general:** aspectos de cómo interaccionan los diferentes actores y equipos, cómo se comunican, qué jerarquías implícitas y explícitas existen en la organización.
- **Identificar restricciones del proyecto**
  - **Técnicas:** cierto producto puede requerir la interacción con determinado componente, condicionar a las herramientas de automatización de pruebas o determinada plataforma base e infraestructura.
  - **Recursos disponibles:** se trata de la inversión destinada al proyecto, tanto en recursos materiales, hardware, software, como recursos humanos dedicados, como ser personal técnico y especialistas.
- **Identificar restricciones de la organización**
  - **Políticas:** la organización puede tomar la decisión de apoyar ciertos proyectos, trabajar o no con determinado proveedor, o requerir seguimiento especial de los procesos.
  - **Económicas:** se trata de los recursos materiales y humanos dedicados al proyecto específico, y al testing en general.
  - **Temporales:** es el margen de tiempo destinado al proyecto, o plazos exigidos para cumplir con los objetivos pautados.

Con la información recopilada, se elabora una visión de la organización en cuanto al testing y se elabora un documento que informe los resultados de este análisis. Este documento esta enfocado al testing particularmente aunque puede identificar otros problemas. Se identifican los inconvenientes y se sugieren acciones para lidiar con ellos. Un ejemplo elaborado para el caso de estudio del proyecto de grado, esta presente en el capítulo 3.

4.3.1. Roles

Es común encontrar patrones muy diversos en la forma en que las personas asumen diferentes tareas. En un proyecto de desarrollo de software de mediano porte, con alguna intención de incluir testing, existen al menos un referente para cada área involucrada. A continuación se destacan los roles más comunes que suelen estar presentes para lograr implantar algún tipo de testing.

Tester

Este rol lleva adelante las pruebas de piezas de software o documentos solicitadas por otros actores de la organización, ya sea encargados de desarrollo, usuarios funcionales particulares, u otro actor designado por una determinada autoridad. Reportan los incidentes al encargado o referente de desarrollo.

Usuarios Funcionales

Son los encargados de concretar pruebas de aceptación de los productos. Puede ser en conjunto con los testers o no. Son los clientes internos, y quienes deciden si un producto o determinado cambio cumple con sus expectativas. También ofician de oráculo, al ser expertos en el dominio; poseen el conocimiento del negocio, y muchas veces, son los destinatarios finales del producto.

Encargado o referente de desarrollo

En relación al testing funcional, notifican al *tester* de un nuevo producto, cambio o liberación para que reciban pruebas. Coordinan la reparación de los incidentes. Toman decisiones a partir de las pruebas de los testers o de los usuarios funcionales.

4.4. FIT Módulo II: Proceso de Testing Funcional Manual

Entre los objetivos de este módulo se encuentra el de otorgar importancia a la práctica del testing de software, aportando un mínimo de metodología y un conjunto de conceptos que hagan del testing una actividad repetible, justificada y profesional.

Este módulo define un proceso de testing funcional manual, con etapas y actividades que acercan a la organización a seguir una metodología. En el proceso definido los testers son capaces de analizar los requerimientos, establecer criterios de testing, construir e investigar los requerimientos faltantes y articular las actividades necesarias para llevar adelante el testing. Además se espera de que sean capaces de comunicarse con los demás integrantes de forma eficaz y puedan formular las estrategias de testing, aplicar priorización basada en riesgo, y apoyarse en técnicas de diseño, llegando a un conjunto de casos de prueba que garantice una determinada cobertura, o cumpla con cierto criterio de finalización de pruebas previamente establecido o negociado con los actores involucrados (usuarios funcionales, responsables en general).

4.4.1. Etapas del Proceso de Testing Manual

Como se ve al inicio del capítulo 4.2, los procesos se componen de etapas, y estas de actividades. Las tres letras iniciales previas al nombre una etapa o actividad corresponde a su identificador. Los identificadores y nombres son usados indistintamente para referirse explícitamente a una determinada etapa o actividad.

Este proceso se compone de las siguientes etapas:

- PLN - Planificación Inicial.
- ANR - Análisis de Requerimientos.
- DSP - Diseño de Casos de Prueba.
- PEJ - Preparación de Ejecución.
- EJP - Ejecución de Pruebas.
- ADD - Análisis de Defectos.
- SAP - Seguir y Actualizar el Plan.
- AYR - Alcance y Análisis de Riesgo.
- VDD - Verificación de Documentos.
- INR - Informar Resultados.

Un proyecto de testing debería contar como mínimo con las etapas de planificación (PLN), análisis (ANR), diseño (DSP), ejecución de pruebas (EJP) y la etapa de comunicar los resultados (INR). La preparación de la ejecución (PEJ) es una etapa importante, pero hay realidades en las cuales se pueden ejecutar pruebas sin tener un manejo especial de los entornos y ambientes. Las etapas de análisis de riesgo y alcance (AYR), verificación de documentos (VDD) y de seguimiento y actualización del plan (SAP), contienen actividades muy recomendadas, pero no son absolutamente necesarias para la concepción de un proyecto de testing.

Veremos reflejado en el siguiente diagrama, que las etapas INR, AYR, VDD y SAP son transversales, es decir, ocurren durante el transcurso del proyecto. El diagrama muestra cómo se relacionan las diferentes etapas, su precedencia, paralelismo y si son requeridas o no.

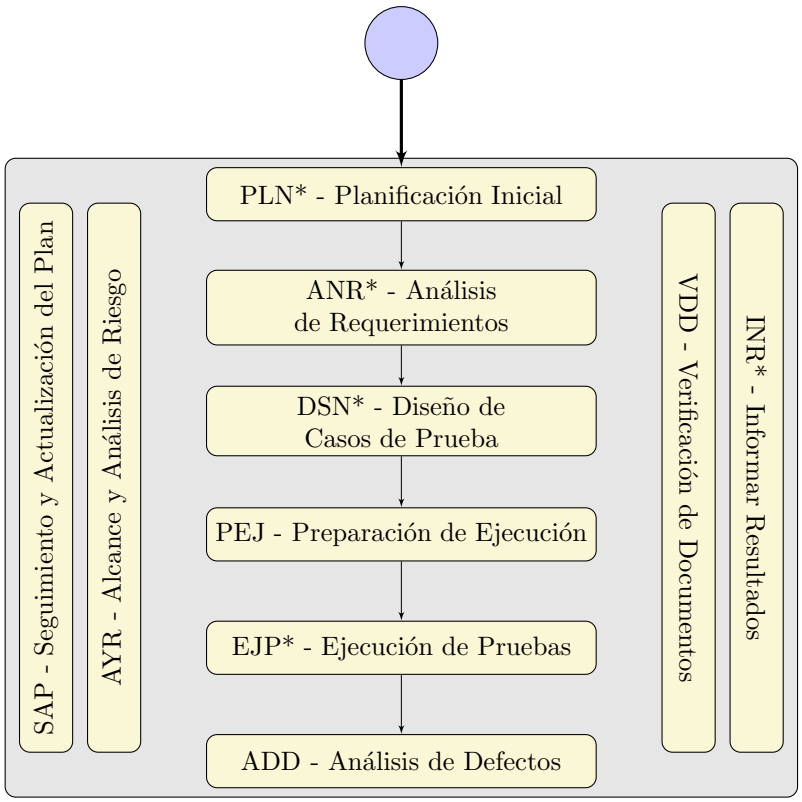

Figura 4.3: FIT Módulo II: Proceso de Testing Funcional Manual

Cada etapa está relacionada con la construcción o mantenimiento de documentos que registran la información relevante de las actividades. La utilidad de un documento particular en un contexto dado, determinará su contenido, profundi-

dad e incluso su existencia. En las siguientes secciones, al examinar cada etapa, mencionaremos las características más relevantes de cada documento.

El cuadro 4.1 presenta las diferentes etapas del Módulo II del FIT, con su identificador, su nombre y los documentos involucrados.

Identificador	Nombre de la Etapa	Documento Involucrado
PLN	Planificación	Plan de Testing
ANR	Análisis de Requerimientos	Documento de Análisis de Testing
DSP	Diseño de Pruebas	Casos de Prueba diseñados / Documentos de Misiones
PEJ	Preparación de Ejecución	Documento de entornos y ambientes
EJP	Ejecución de Pruebas	Documento de ejecución / Casos de Pruebas diseñados / Documentos de Misiones
ADD	Análisis de Defectos	Informe de Análisis de Defectos
SAP	Seguir y Actualizar el Plan	Plan de Testing
AYR	Alcance y Análisis de Riesgo	Plan de Testing
VDD	Verificación de Documentos	Informe de Consultas y Observaciones
INR	Informar Resultados	Informes de Avance / Informe Final

Cuadro 4.1: Documentos del Proceso de Testing Funcional Manual.

El cuadro 4.2 presenta las diferentes etapas del proceso Módulo II del FIT, con su identificador, su nombre y los roles involucrados en cada una de ellas. El primer rol en el cuadro es el rol principal y cuenta con el apoyo de los siguientes.

Identificador	Nombre de la Etapa	Rol Involucrado
PLN	Planificación	Tester Profesional / Tester Funcional Líder / Líder de Proyecto
ANR	Análisis de Requerimientos	Tester Profesional / Tester Funcional / Líder de Proyecto / Desarrollador
DSP	Diseño de Pruebas	Tester Profesional / Tester Funcional
PEJ	Preparación de Ejecución	Tester Funcional Líder / Tester Profesional / Líder de Proyecto
EJP	Ejecución de Pruebas	Tester Funcional / Tester Profesional
ADD	Análisis de Defectos	Tester Profesional / Tester Funcional
SAP	Seguir y Actualizar el Plan	Tester Funcional Líder / Tester Profesional
AYR	Alcance y Análisis de Riesgo	Tester Funcional Líder
VDD	Verificación de Documentos	Tester Profesional / Tester Funcional
INR	Informar Resultados	Tester Funcional Líder / Tester Profesional

Cuadro 4.2: Roles en el Proceso de Testing Funcional Manual.

Planificación Inicial

La etapa de planificación consiste en la elaboración de un plan que exprese las decisiones más importantes del transcurso del proyecto de testing. Puede incluir la introducción, las estrategias seleccionadas para atacar las diferentes pruebas, las estimaciones, junto con diagramas y descripción de la forma en que se hará el seguimiento y control del proyecto. Los proyectos de testing, tienen algunas particularidades que los diferencian de proyectos de otra índole, y por lo tanto se deben manejar ligeramente diferente. Entre las particularidades, se puede decir que proyecto de testing:

- Provee información, desde una perspectiva de servicios: el testing sirve a diferentes partes, actuando como nexo y aportando a cada uno de forma diferente.
- Suele ser de las etapas finales dentro de un proyecto mayor: carga con la responsabilidad de detectar los incidentes antes de que pasen a producción. El proyecto debe estar enfocado a la transparencia y a la comunicación transparente y fluida.

- Puede re-alimentar etapas de un proyecto paralelo o mayor: los productos de trabajo generados en testing pueden ser insumo, soporte, o material de consulta para otras áreas, como ser desarrollo de documentación de usuario, desarrolladores, entre otros.

- Puede dividirse en subproyectos que deben ser consistentes: en ocasiones el proyecto de testing avanza en paralelo con el proyecto de desarrollo, o se suceden muchos subproyectos de testing en torno a un producto, o fases de un producto. Conforme avanzan los subproyectos, estos deben ser consistentes con el proyecto de testing mayor que los contiene. Es probable que cada subproyecto deba dejar información a los demás, o que utilice la información generada en otros anteriores, o que están ejecutándose en paralelo con él.

En el cuadro 4.3, se ven las diferentes actividades relacionadas con la etapa, mostrando el identificador de cada actividad, y si es obligatoria. El objetivo es la elaboración de un *plan de testing*.

Identificador	Nombre de la Actividad	¿Es requerida?
SEE	Seleccionar Estrategia y Enfoque	SI
EST	Formular Estimaciones	NO
ADR	Análisis de Riesgo según Requerimientos y Negocio	NO
STD	Selección de Técnicas de Diseño	NO
ALC	Alcance de las Pruebas	SI
SCP	Seguimiento y Control del Proyecto	NO
CRA	Criterios de Aceptación	NO

Cuadro 4.3: Etapa de Planificación Inicial

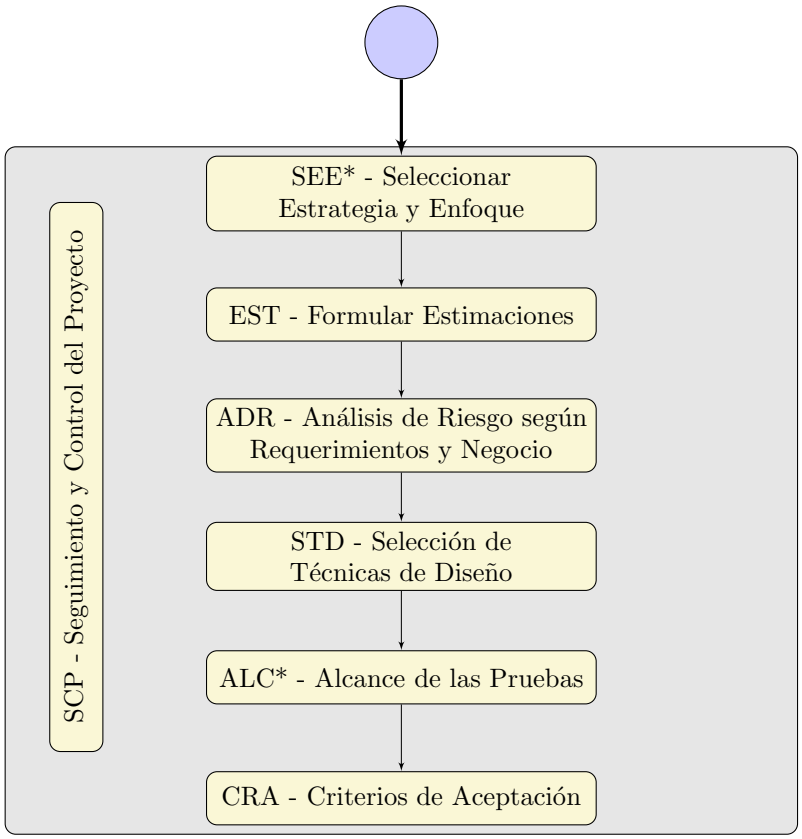

Figura 4.4: Etapa Planificación Inicial

Productos de trabajo involucrados en la etapa

- *Plan de pruebas:* en su forma más completa contiene un apartado para cada actividad de la etapa. Se registra la decisión acerca de la estrategia a utilizar, las estimaciones del proyecto de testing, el análisis de riesgo, las técnicas de diseño de pruebas a utilizar, el alcance de las pruebas, ajustes al plan y criterios de aceptación pautados para las pruebas. El plan puede incluir comentarios acerca de si la automatización es parte de la estrategia. El plan es una herramienta muy útil para organizar el trabajo, documentar el proceso y negociar el alcance de las pruebas y tener como referencia para la gestión del proceso. Los planes más formales suelen ser más difíciles de mantener, por eso es común que los planes de testing se vuelvan rápidamente obsoletos. El plan de testing es un documento dinámico y por lo tanto tiene que ser consultado y actualizado. Planes menos rigurosos en formalismos, pero que hacen foco en los puntos relevantes al proceso suelen ser más realistas y realmente aportar en contextos de alto dinamismo[4].

[4]Con *alto dinamismo* nos referimos a contextos muy cambiantes, que provocarían ajustes

Seleccionar Estrategia y Enfoque: consiste en elegir, construir, o adaptar una estrategia de testing para las pruebas. Se puede optar por el testing planificado, testing exploratorio, o inclusive en la automatización de ciertos componentes (como se ve en el Proceso de Testing Funcional Automatizado, del Módulo III del FIT en la sección 4.5 de este capítulo).

Las estrategias con tenor exploratorio son adecuadas a situaciones en las que los tiempos son acotados, comprometiendo algo de rigurosidad en la documentación. La productividad de la estrategia de testing exploratorio depende en gran medida de las habilidades del tester, favoreciendo el uso de la creatividad y toma de decisiones en el momento, siendo por esto último más flexible que el enfoque de testing planificado.

El testing planificado requiere un mínimo de análisis y diseño de pruebas antes de pasar a la ejecución. Es conveniente para contextos en los cuales es necesario documentar rigurosamente el proceso, por ejemplo para que dicha información sea auditable. Eventualmente, el diseño de las pruebas puede estar a cargo de un tester experimentado y la ejecución a cargo de otro tester con menos experiencia o conocimiento del dominio, por lo cual la aplicación de esta estrategia tendrá éxito por motivo de un buen diseño de pruebas y no tanto por las habilidades del tester que las ejecute. Como desventaja frente a la estrategia de testing exploratorio, el testing planificado es más rígido y es afectado por los cambios en el proyecto, que derivan en un rediseño de casos de prueba ante una nueva realidad.

Enfocar el proyecto alrededor de la automatización de pruebas es una estrategia que en muchas realidades depende de la naturaleza del proceso de desarrollo seguido. Como se describe en la sección 2.5.1, la automatización puede ser una alternativa para asegurar la no regresión del producto debido a una alta frecuencia de liberaciones del producto, puede ser la forma de manejar grandes volúmenes de información, o suplantar pruebas que manualmente tomarían mucho tiempo o son muy tediosas.

La decisión de la estrategia no tiene por que ser un extremo, sino que la estrategia puede ser un híbrido de las mencionadas, dado que en un sistema de gran porte, los diferentes componentes pueden requerir enfoques particulares, o la estrategia puede directamente ser algo diferente de lo mencionado.

Roles involucrados

- *Tester Profesional, Tester Funcional Líder:* identifican las particularidades del proyecto y definen una estrategia para el proyecto de testing. Para esto se apoyan en la información preliminar que se pueda recopilar del proyecto, ya sea de documentos o de otros actores, como Usuarios Funcionales y Referentes de Proyecto.

frecuentes al plan, que de ser un documento muy estricto, se vuelve una tarea costosa.

- *Usuarios Funcionales:* proveen de información al equipo de testing sobre el sistema, mencionando sus aspiraciones, los resultados deseables, y toda información relevante a la realidad del producto.

Formular Estimaciones: luego de tener idea del alcance y el dominio del problema, es posible plantear tiempos estimados para las diferentes etapas seleccionadas para el proyecto de testing particular. Es recomendable acompañar estas estimaciones con algún diagrama estilo Gantt. Esta actividad, suele tener una participación fuerte al principio, y luego en algunos hitos intermedios, donde se pueden ajustar, corregir y volver a estimar, acorde a la realidad. Es una herramienta fundamental para el seguimiento y el control, ayudándonos a tomar acciones correctivas.

Roles involucrados

- *Tester Profesional, Tester Funcional Líder:* plantean estimaciones en base al dimensionamiento inicial del proyecto. La estimación debe ser revisada y ajustada como respuesta al seguimiento. Una buena estrategia es ajustar la estimación al contar con una idea de los casos de prueba a ejecutar, o las funcionalidades y flujos que se desean ensayar en el sistema.
- *Responsables de desarrollo:* mediante el conocimiento del sistema y de la complejidad de sus diferentes funcionalidades, los responsables de desarrollo colaboran a la construcción de una estimación más acertada.
- *Usuarios Funcionales:* conocen el dominio del problema, los puntos críticos del negocio, y muchas veces el modo de uso del sistema. Esto los suele dotar de un gran conocimiento del dominio y la complejidad de las funcionalidades. Esta información contribuye a la mejora de las estimaciones.

Análisis de Riesgo según Requerimientos y Negocio: el objetivo es plantear una lista inicial de riesgos, que apoyará la toma de decisiones en cuanto al alcance y el foco de las pruebas. El equipo de testers identificará los riesgos a partir del análisis preliminar de los requerimientos, la interacción con otros actores del proyecto de desarrollo y el contexto del proyecto en sí. Esta lista puede ser elaborada y mantenida mediante mecanismos clásicos de gestión del riesgo, u otros, como la heurística de testing basado en riesgo [Bac99] mencionada en la sección 2.4.4.

Roles involucrados

- *Tester Profesional, Tester Funcional Líder:* elaboran una lista de riesgos a partir de la información aportada por los diferentes actores, conocimiento previo del sistema, conocimiento del equipo de trabajo y desarrollo, experiencias en funcionalidades o productos similares, entre otros. Es frecuente que se necesiten varias instancias de reunión para llegar al

conjunto de riesgos adecuado. Dependiendo de la rigurosidad del proceso, esta actividad deberá contar con la validación de quien tenga la visión del negocio, o tenga la responsabilidad de responder por la correcta mitigación de estos riesgos.

- *Responsables de desarrollo, equipo de desarrollo:* son consultados para evaluar el impacto técnico y riesgos asociados a la construcción del producto. Frecuentemente el equipo de desarrollo y responsables pueden informar de los riesgos asociados a la complejidad técnica, acoplamiento e impacto de un cambio en otras funcionalidades, complejidad de mantenimiento, complejidad de comportamiento de cierta funcionalidad, grado de dificultad que representa automatizar o testear unitariamente cierta parte del producto, y otros riesgos.

- *Usuarios Funcionales:* entre la información que aportan, es de destacar que son la fuente principal para identificar los riesgos que provienen de los aspectos del dominio (funcionalidades críticas, importantes o accesorias) y de particularidades del uso del sistema. En ciertos contextos, también pueden participar de la validación de la lista de riesgos. Muchas veces estos usuarios además han tenido que gestionar riesgos mediante algún mecanismo, por lo tanto esa información puede ser de suma utilidad en esta actividad.

Productos de trabajo involucrados

- *Documento de Riesgos Identificados:* contiene los riesgos que se destacan en el proyecto. Suele contar con una priorización y se puede presentar por ejemplo en forma de lista priorizada, o matriz, como se veía en la sección 2.4.4, entre otras técnicas. El material presente en este documento se puede anexar como sección al *plan de pruebas* elaborado.

Selección de Técnicas de Diseño de Pruebas: del conocimiento residente en los testers, o como resultado de investigación y contraste con un análisis primario de los requerimientos funcionales, se eligen las técnicas a utilizar para derivar casos de prueba (de aplicar testing diseñado). Es común recaer sobre técnicas populares como análisis de valores límites, particiones de equivalencia, árboles de decisión, todos los pares; pero de optar por proceder con esta actividad, estamos ante una buena oportunidad para conocer e investigar sobre nuevas técnicas y heurísticas[5].

Roles involucrados

- *Tester Profesional, Tester Funcional Líder:* seleccionan técnicas adecuadas a cada funcionalidad y según la estrategia seguida. Para el testing

[5] Existe material base, como el libro de Mayers [Mye04b], así como muchos buenos recursos en la web, como el blog de testing de Google[JAWea11], o el blog de James Bach [Bac11], que han marcado tendencias en la disciplina

diseñado se evalúa la aplicabilidad de las técnicas conocidas por los testers y para el testing exploratorio, se puede optar por seguir las técnicas mencionadas en la sección 2.4.2, u otras[6].

Productos de trabajo involucrados

- *Plan de Pruebas:* el conocimiento generado en esta actividad puede ser incluido en este documento.

Alcance de las Pruebas: basándose en los requerimientos analizados, se formula un alcance y una cobertura para las funcionalidades del producto bajo testing. La cobertura expresa para cierto criterio, el grado de alcance que tendrá la prueba. El alcance además sirve para ajustar los tiempos del proyecto en base a las estimaciones. Se suele recortar el alcance de las pruebas menos críticas, las que, luego del análisis de priorización y riesgo, sean menos prioritarias.

Roles involucrados

- *Tester Profesional, Tester Funcional Líder:* establecen una propuesta de alcance para las pruebas, las que luego validarán con los diferentes actores.
- *Responsables de Desarrollo:* revisan junto a los testers el alcance preliminar propuesto y eventualmente lo validan.
- *Usuarios Funcionales:* pueden participar en la validación del alcance de las pruebas, sobre todo en etapas tempranas en lo que respecta a las pruebas de aceptación del producto.

Productos de trabajo involucrados

- *Plan de Pruebas:* el conocimiento generado en esta actividad puede ser incluido en este documento.

Seguimiento y Control del Proyecto: es el nexo con la etapa de seguimiento y actualización del plan (SAP), la cual es transversal al transcurso del proyecto. A través de esta actividad, se contrasta la realidad con la planificación y se efectúan los ajustes necesarios. Aunque la etapa SAP no sea llevada a cabo, es posible implementar ajustes al plan mediante esta actividad. Los ajustes pueden ser a cualquier punto del plan, como las estimaciones, el alcance, comentarios sobre las técnicas usadas, entre otros.

Roles involucrados

[6] Además de las heurísticas mencionadas, James Wittaker presenta en su libro acerca de testing exploratorio [Whi09c], otras heurísticas para llevar a la práctica esta estrategia de testing.

- *Tester Profesional, Tester Funcional Líder:* mantienen el plan de pruebas, representando los ajustes necesarios para adaptarlo a la nueva realidad.

Productos de trabajo involucrados

- *Plan de Pruebas:* el plan puede ser eventualmente enriquecido y ajustado en esta actividad. En contextos más rigurosos en cuanto a la documentación, se registran los ajustes y en en una sección aparte se incluyen los desvíos al plan junto con sus causas.

Criterios de Aceptación: en proyectos que se requiera una validación formal de los elementos generados por las diferentes etapas, se puede establecer, negociar y validar los criterios de aceptación del producto, entre diferentes partes involucradas. Entre estos criterios pueden estar las instancias de pruebas de aceptación, la cantidad y/o gravedad de los incidentes conocidos tolerados, así como consideraciones de cobertura y el conjunto mínimo de casos de prueba a ejecutar, entre otros.

Roles involucrados

- *Tester Profesional, Tester Funcional Líder:* elaboran junto a los Usuarios Funcionales y Responsables de Desarrollo la lista de criterios con sus diferentes niveles de aceptación o rechazo del software.

- *Responsable de Desarrollo:* colabora en la definición de los criterios de aceptación a ser negociados con los interesados y Usuarios Funcionales. Es importante contar con sus aportes para definir la traducción de un criterio de aceptación a parámetros técnicos medibles.

- *Usuarios Funcionales:* ayudan a la definición de criterios desde el punto de vista conceptual, con la perspectiva del dominio del producto. En la mayoría de los contextos, tienen la última palabra acerca de los criterios, y son quienes pueden dar el visto bueno al respecto.

Productos de trabajo involucrados

- *Plan de Pruebas:* el plan puede contener los criterios de aceptación definidos.

Análisis de Requerimientos

Es la etapa que consiste del estudio de los requerimientos funcionales con el fin de identificar los puntos de verificación y ganar conocimiento sobre el dominio. Los requerimientos no siempre están completamente definidos, identificados y documentados. En ocasiones el trabajo de los testers consisten colaborar a la definición y sintetización de los requerimientos. El requerimiento es un insumo fundamental para el testing, pero, como veremos más adelante, el análisis de requerimientos puede llegar a ser simultáneo con el diseño, la ejecución de pruebas y el aprendizaje sobre producto y el dominio; en estrategias particulares como la de testing exploratorio. De una u otra manera, siempre estaremos ante algún tipo de análisis, ya sea contando con el tiempo necesario para estudiar los requerimientos o relevarlos y construirlos a partir del conocimiento de los diferentes actores, así como también en paralelo y simultáneamente con la ejecución de pruebas. Es necesario transitar por alguna de las actividades de esta etapa y según la estrategia formulada, será cómo se ejecutarán las actividades.

La siguiente tabla indica las actividades con su identificación y si son requeridas dependiendo del tipo de estrategia[7].

Identificador	Nombre de la Actividad	¿Es requerida?
ANR	Análisis de requerimientos	SI
VFR	Verificación de requerimientos	NO
CRF	Construcción de requerimientos faltantes	NO
VDR	Validación de requerimientos relevados	NO

Cuadro 4.4: Etapa de Análisis de Requerimientos

[7] **Referencias de la tabla:**
TD: actividad es requerida para testing diseñado, y opcional para testing exploratorio.
TE: actividad requerida para testing exploratorio y opcional para testing diseñado.
SI: actividad requerida para estrategias exploratorios y de testing diseñado.
NO: actividad opcional para ambas estrategias.

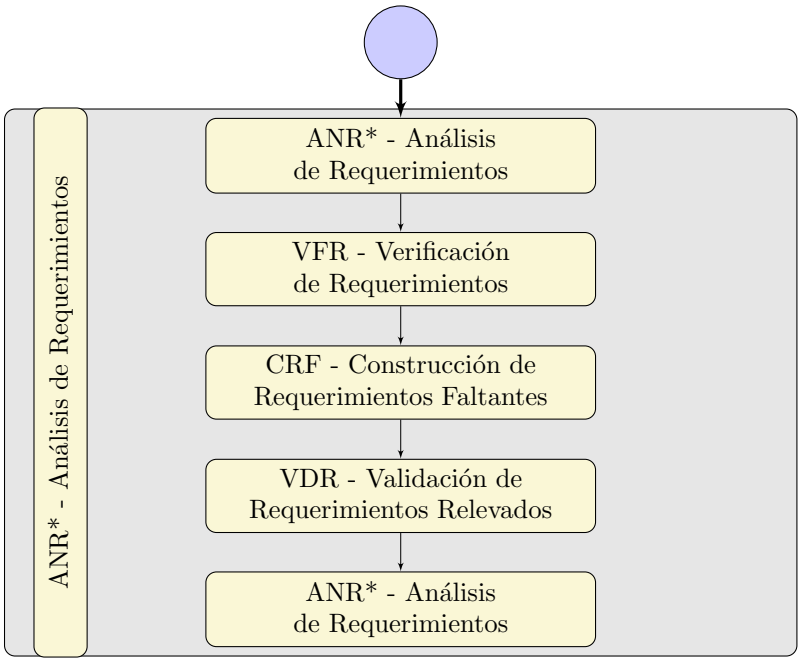

Figura 4.5: Etapa Análisis de Requerimientos

Análisis de requerimientos: estudio de los requerimientos funcionales identificando las entradas, las salidas y el comportamiento esperado del sistema. Como los requerimientos pueden o no estar escritos, también conlleva un aprendizaje del dominio. Esta actividad es fundamental en esta etapa, y es el principal insumo para las tareas de testing, sobre todo para el testing planificado. Es una actividad que puede llevarse a cabo simultáneamente con las demás actividades de esta etapa, y es la única actividad requerida. En diagrama podemos ver, mediante el orden de precedencia, que se inicia y finaliza con esta actividad. Además la actividad también se va dando progresiva y simultáneamente, retroalimentándose de los resultados intermedios de las demás actividades de la etapa.

Para estrategias exploratorias, el análisis del dominio y los requerimientos se abordan en la profundidad suficiente como para delinear los aspectos más importantes de las misiones del testing exploratorio. El análisis más profundo se da simultáneamente con la ejecución de las pruebas, siendo los límites entre análisis, diseño y ejecución, mucho más difusos en este caso.

Roles involucrados

- *Tester Profesional, Tester Funcional Líder:* construyen el análisis de las funcionalidades del producto a verificar. Coordinan reuniones o generan instancias donde poder consultar la información necesaria. Es común que

las dudas planteadas por el equipo de testing, ayude a clarificar la solución al problema en sí, por lo tanto es recomendable, siempre que sea posible, que las etapas de análisis del problema comiencen lo antes posible, incluso desde el momento mismo de la definición de los requisitos.

- *Responsable de Desarrollo, equipo de desarrollo:* cuentan la visión técnica de la funcionalidad. Dado que para desarrollar cierta funcionalidad, se tiene que comprender el problema, los caminos seguidos por testers y desarrolladores son similares en este aspecto, diferenciándose en el punto de vista. La colaboración enriquece mucho, sobre todo para interactuar con los expertos del dominio, ayudándoles a definir mejor sus requisitos.

- *Usuarios Funcionales:* dado que son los expertos en el dominio, su papel es fundamental en la definición de los requisitos. La interacción con el equipo de testing a menudo favorece la correcta formulación de lo que se espera de cierto producto y su funcionamiento. Los Usuarios Funcionales encuentran muchas veces en los testers una buena forma de canalizar sus expectativas, dado que los especialistas técnicos, suelen tener su visión y forma de comunicación muy ligada al mundo y vocabulario técnico (configurándose una barrera a la comunicación con Usuarios Funcionales).

Productos de trabajo involucrados

- *Documento de Análisis de Requerimientos de Testing:* se puede construir el documento que registra la visión de testing del sistema a verificar. Este documento oficia como base de conocimiento y muchas veces complementa la documentación existente de los requerimientos del producto. Si se opta por testing diseñado, es recomendable contar con documentos de este estilo que pueda ser consultado, compartido y validado con diferentes actores y que sea una entrada única para la definición de casos de prueba y escenarios.

Verificación de requerimientos: consiste en el análisis de los documentos con el fin de identificar si están claros, si la información es completa, y si es posible diseñar casos de prueba o llevar adelante una estrategia de prueba con la información disponible. Esta actividad es recomendada como mecanismo de incorporación de testing a etapas temprana de desarrollo del producto. Es común encontrar inconsistencias en la definición de los requerimientos, lo que redunda en menos esfuerzo de testing y desarrollo al prevenir futuros incidentes. Consecuentemente los documentos son más completos y precisos.

Roles involucrados

- *Tester Profesional, Tester Funcional Líder:* revisan los documentos de la manera descrita. Se pueden dar varias interacciones con los autores de los documentos.

- *Responsable de Desarrollo, Usuarios Funcionales, otros autores de documentos:* revisan las sugerencias de cambios, se reúnen con los testers para evacuar dudas, e impactan las posibles modificaciones a los documentos, motivando eventuales nuevas verificaciones por parte de los testers.

Productos de trabajo involucrados

- *Documentos a verificar:* cualquier documento puede ser verificado si se considera que influye en el correcto desarrollo del producto. Entre estos pueden estar los documentos de análisis de requerimientos, manuales de usuario, documentos de análisis de riesgo e informes en general.

Construcción de requerimientos faltantes: es la actividad de completar la especificación de los requerimientos funcionales, si no estuviese completa. El testing diseñado en su forma más estricta, requiere de documentos completos o lo más cercano posible. Mediante reuniones con expertos funcionales, investigación del producto, investigación de documentación anterior y lectura de manuales de usuario; el tester elabora su análisis y enriquece o completa los documentos.

Roles involucrados

- *Tester Profesional, Tester Funcional Líder:* si se le requiere y autoriza, el tester es capaz de complementar los documentos con la información a la que ha podido acceder (probablemente por la ejecución de la actividad de Verificación de Requerimientos).

- *Responsable de Desarrollo, equipo de desarrollo:* revisan que los documentos complementados con los testers reflejen lo esperado. Estas instancias de validación y enriquecimiento de documentos puede generar varias interacciones entre los equipos.

- *Usuarios Funcionales, otros autores de documentos:* al igual que los desarrolladores, estos revisan la información agregada por los testers, y pueden tener instancias de coordinación con ellos. Como se mencionaba anteriormente, el objetivo perseguido es eliminar las ambigüedades e inconsistencias de los documentos, para que sean más comprensibles y hagan más sencilla la tarea para el equipo de desarrollo.

Productos de trabajo involucrados

- *Documento de requerimientos a enriquecer:* es el documento que el tester tratará de completar con la información que ha podido recopilar.

Validación de requerimientos relevados: esta actividad está estrechamente ligada a la etapa de construcción de requerimientos faltantes. El trabajo de investigación de los requerimientos llevado adelante por los testers debe ser validado por algún responsable o líder de proyecto; potencialmente se podrían estar incluyendo cambios importantes de definición, y dicha información puede haber sido consultada con diferentes actores, con múltiples visiones. Es importante por lo tanto, que exista una única versión validada en a la que todos los involucrados puedan referirse como la información oficial.

Roles involucrados

- *Tester Profesional, Tester Funcional Líder:* presentan la información recopilada y procesada, haciendo énfasis en qué significa esta para el equipo de testing.

- *Responsable de Desarrollo, equipo de desarrollo, Usuarios Funcionales, otros autores de documentos:* colaboran en la definición y puesta en común de la información presentada por el equipo de testing, dando su consentimiento de la correctitud de los datos relevados, así como también corregir y agregar más información.

Productos de trabajo involucrados

- *Documento de Análisis de Requerimientos de Testing:* de ser construido, suele contener la fuente de la información que los testers van a impactar en los documentos de requerimientos. Puede también ser validado con los diferentes actores.

- *Documento de requerimientos a validar:* es el documento modificado por los testers, que luego puede ser revisado y validado por los autores, así como también regresar a testing para nuevas revisiones, como ser la verificación de las correcciones o las respuestas a las eventuales dudas y la confirmación de la correctitud de la información agregada.

Diseño de Pruebas

El desarrollo de esta etapa depende de la estrategia seleccionada. En las estrategias con un enfoque de testing planificado, se aplicaran técnicas de diseño logrando un conjunto de casos de prueba inicial. En las estrategias con tenor exploratorio, la etapa se desarrolla identificando los grandes puntos y flujos funcionales, con el fin de establecer misiones y trazando algunos hitos que interesa recorrer en la exploración.

La siguiente tabla indica las actividades con su identificación y si son requeridas dependiendo del tipo de estrategia[8].

Identificador	Nombre de la Actividad	¿Es requerida?
DCP	Diseño de Casos de Prueba	TP
ESP	Especificación de Casos de Prueba	TP
DMS	Diseño de Misiones y Sesiones	TE
EMS	Especificación de Misiones y Sesiones	TE

Cuadro 4.5: Etapa Diseño de Pruebas

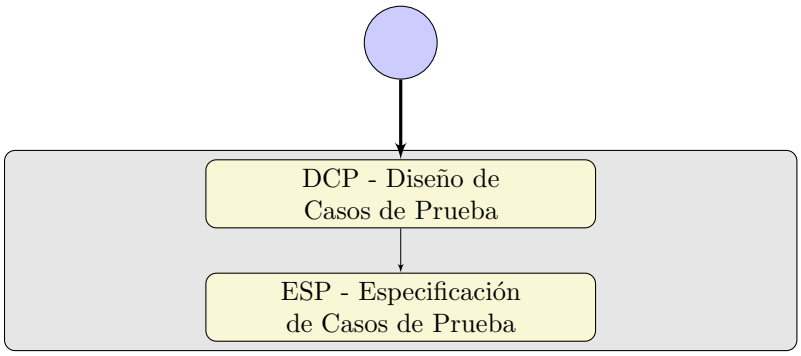

Figura 4.6: Etapa Diseño de Pruebas del Testing Planificado

[8] **Referencias del cuadro 4.5:**
TP: actividad es requerida para testing planificado, y opcional para testing exploratorio.
TE: actividad requerida para testing exploratorio y opcional para testing planificado.
SI: actividad requerida para estrategias exploratorios y de testing diseñado.
NO: actividad opcional para ambas estrategias.

Diseñar casos de prueba: esta actividad consta en la derivación de casos de prueba a partir de la aplicación de unas o varias técnicas. El objetivo es llegar a un conjunto de casos de prueba que puede ser insumo para negociación de alcance, priorización, ajuste de estimaciones y seguimiento de cobertura de funcionalidades. Se estila que los casos de prueba diseñados incluyan un identificador, las variables involucradas, el resultado esperado ante la ejecución y algún comentario o descripción; incluso, pueden llevar una explicación detallada paso a paso de su ejecución, y los puntos de verificación durante la ejecución [9].

Especificación de Casos de Prueba: es la tarea de instanciar un caso de prueba a una situación particular de ejecución. Por ejemplo, mediante la inclusión exacta de los valores que se van a utilizar en las pruebas. Esta etapa puede estar vinculada con la búsqueda de datos apropiados, un detalle no menor, y que puede llegar a consumir mucho tiempo. Las estrategias de testing con inclinación al testing diseñado, ven muy comprometido su éxito en lograr un conjunto de casos de prueba de calidad, como veíamos en la sección 2.7.1.

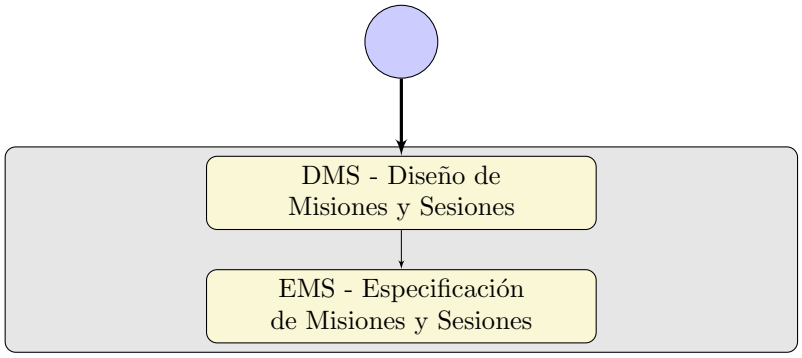

Figura 4.7: Etapa Diseño de Pruebas del Testing Exploratorio

Diseñar Misiones y Sesiones: mediante este diseño se elaboran las descripciones verbales de las misiones y se organizan las sesiones de testing exploratorio. A partir de la información funcional obtenida, se delimitan los grandes puntos. Un enfoque podría ser el separar las misiones con diferente granularidad, para explorar grupos de funcionalidades relacionadas, operaciones y usos. Luego, se asigna una o varias sesiones que aborden las misiones diseñadas. Estas ideas provienen de la técnica de testing exploratorio basado en sesiones (visto en la sección 2.4.2), que puede ser aplicada más rigurosamente siguiendo las recomendaciones de la técnica.

[9] Por ejemplo, un caso de prueba puede ser la descripción en pasos de un flujo funcional, identificando los puntos dónde se debe validar y verificar información, especificando qué resultados se esperan. Esta forma de documentación es muy útil para la automatización de pruebas (ver etapa Seleccionar Pruebas en la seccion 4.5.1

Especificación de Misiones y Sesiones: consiste en detallar la información relativa a las sesiones, armar archivos donde se reportarán, y organizar material para tener alguna trazabilidad entre misiones, sesiones, testers e incidentes encontrados. Aquí podemos: asignar las misiones y sesiones a los testers, especificar la rigurosidad esperada en la ejecución de pruebas, describir el tipo de prueba y establecer delineamientos de la cobertura esperada, entre otros.

Roles involucrados en la etapa

- *Tester Profesional, Tester Funcional Líder:* son los encargados de ejecutar las actividades de diseño. Es importante que los testers cuenten con conocimiento en técnicas de diseño de pruebas y enfoques para testing exploratorio, sabiendo para qué realidades aplican y así poder evaluar cuáles son las más adecuadas para cada contexto.

Productos de trabajo involucrados en la etapa

- *Casos de Prueba diseñados:* documenta los casos de pruebas que pueden derivar de la aplicación de una o varias técnicas. Estos documentos pueden llegar hasta el detalle de contar con los datos de pruebas específicos que se van a utilizar para la ejecución. El detalle del documento puede depender de la experiencia que tenga el tester que ejecutará las pruebas (ya sea en el dominio del producto, como en la disciplina de testing en sí).

- *Documentos de Misiones:* representa un enumerado de las misiones a ejecutar por los testers. Si bien se esta basando este documento en la técnica de James Bach, mencionada en la sección 2.4.2, también es una buena práctica para cualquier otro enfoque adoptado, en algún momento registrar las intenciones que tendrán las pruebas acerca de ensayar determinada funcionalidad.

Alcance y Análisis de Riesgo

Esta es una etapa no requerida. Si bien no es imprescindible, abordar su ejecución denota cierta madurez en el proceso seguido, al involucrar actividades secundarias en pro de la mejora de la calidad del proceso de testing en sí. Las técnicas de manejo y gestión del riesgo utilizadas pueden ser diversas,algunas de ellas mencionadas en 2.4.4. Se hace presente en la actividad un componente fuerte de negociación, contraposición de ideas y manejo de la interacción con los diferentes actores del proyecto.

Identificador	Nombre de la Actividad	¿Es requerida?
PRF	Priorizar Requerimientos y Funcionalidades	NO
SPC	Seleccionar y Priorizar Casos de Prueba	NO
NVA	Negociar y Validar Alcance	NO

Cuadro 4.6: Etapa Alcance y Análisis de Riesgo

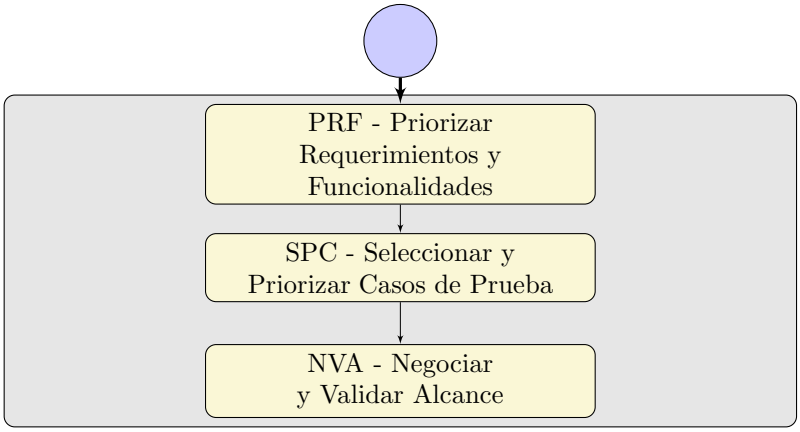

Figura 4.8: Etapa Alcance y Análisis de Riesgo

Roles involucrados en la etapa

- *Tester Profesional, Tester Funcional Líder:* construyen las listas para priorizar con el resto de los actores, ya sea de requerimientos y funcionalidades como de los casos de prueba tentativos.
- *Responsable de Desarrollo:* aportan la visión técnica a la hora de valorar las funcionalidades y casos de prueba a incluir en las tareas de testing. En algunas organizaciones son también quienes pueden validar el alcance de las pruebas negociando con el equipo de testing.
- *Usuarios Funcionales:* ayudan a la priorización de casos y funcionalidades desde su visión de expertos en el dominio. También pueden ser determinantes a la hora de definir el alcance de las pruebas. En otras organizaciones son autoridades de mayor jerarquía incluso que los responsables de desarrollo de los distintos productos, y por lo tanto son quienes tienen la última palabra en la negociación del alcance.

Productos de trabajo involucrados en la etapa

- *Plan de Testing:* el resultado de la negociación se puede documentar en una sección del plan de testing. De ser necesario una formalización mayor del acuerdo, se puede acompañar este documento de otro que exprese la conformidad de todos los actores con el alcance pautado.

Priorizar Requerimientos y Funcionalidades: mediante un análisis primario del equipo de testers, y luego con los demás integrantes del proyecto, se elabora una lista de requerimientos y eventualmente funcionalidades asociadas a algún indicador de prioridad. Independientemente de la escala utilizada, la idea es que se logre un insumo importante para la negociación de alcance, manejo de los riegos, ajustes de estimaciones de esfuerzo, entre otros. La priorización enfocada en el riesgo puede utilizar cualquiera de las técnicas comentadas en la sección 2.4.4 sobre manejo de riesgo, entre otras.

Seleccionar y Priorizar Casos de Prueba: en contraste con la priorización y tomando como punto de partida un conjunto de casos de prueba, el objetivo de esta actividad es sintetizar un conjunto de casos de prueba lo más adecuado posible al contexto del proyecto. Ya sea por extensión, complejidad, relevancia o riesgo; los casos de prueba deben ser catalogados y etiquetados con una prioridad adecuada, o simplemente ser descartados ante casos de prueba de mejores características. Es conveniente que el equipo que discute sobre estos temas tenga un representante de cada actor relevante (en ejemplo: desarrolladores, testers, usuarios y líderes de proyecto), dependiendo también de la rigurosidad del proceso en sí de la organización (la formalidad en las validaciones por ejemplo).

Negociar y Validar Alcance: los casos de prueba diseñados y priorizados (también aplicaría a las sesiones y misiones de testing exploratorio) pueden ser validados con responsables de diferentes áreas, o con quienes tengan la responsabilidad de aceptar o no un componente modificado o creado. Si el proyecto requiere una formalidad superior, esta actividad produce documentos al estilo de contrato o acuerdo, donde las partes se comprometen a cierto patrón de transcurso de las pruebas y su resultado esperado. Al validar el alcance se están validando aspectos de cobertura de las pruebas diseñadas.

Preparación de Ejecución

Esta es la etapa en la que debemos asegurarnos que todo esta listo para pasar a la ejecución. Puede ser tan compleja como el contexto lo requiera. Es necesario tener debidamente instaladas y configuradas las versiones de los productos a probar, así como también todas las herramientas de apoyo a la ejecución de pruebas y administración del proceso de testing. De ya contar con ambientes y herramientas de apoyo al testing, las actividades de esta etapa serán menos profundas y se desarrollarán más al nivel de configuración y puesta a punto.

Identificador	Nombre de la Actividad	¿Es requerida?
CET	Configurar Entorno de Testing y Ambientes	NO
BDP	Búsqueda de Datos para las Pruebas	NO
CHA	Configuración de Herramientas de Apoyo	SI

Cuadro 4.7: Etapa Preparación de Ejecución

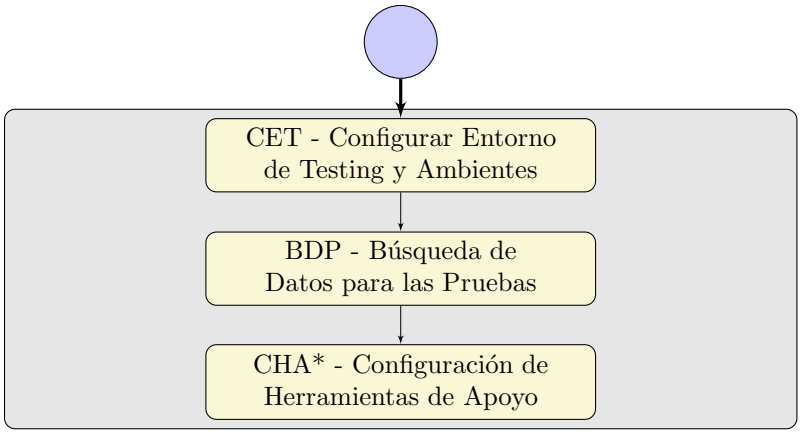

Figura 4.9: Etapa Preparación de Ejecución

Roles involucrados en la etapa

- *Tester Profesional, Tester Funcional Líder:* definen los requisitos de software y hardware para generar un ambiente de ejecución de pruebas. Muchas veces, las condiciones están dadas y puede ser que el equipo de testing tenga que adaptar sus necesidades a los recursos disponibles. Como

mínimo tiene que asegurarse cierta independencia de datos y ambientes, de tal manera que las pruebas no se vean afectadas, por ejemplo por la existencia de pruebas del equipo de desarrollo en el mismo ambiente. Si el tester no tiene control sobre el ambiente y sus datos, no tendrá certeza sobre el resultado de las pruebas.

- *Especialista en Infraestructura:* asiste en la instalación y configuración de las herramientas de base a utilizar para la ejecución de las pruebas, así como los ambientes y las versiones del producto a verificar.

Productos de trabajo involucrados en la etapa

- *Documento de Entornos y Ambientes:* detalla los ambientes preparados y sus características como ser el software de base, herramientas de apoyo instaladas y sus versiones, manejadores de bases de datos, características de la información presente en las bases de datos de testing (calidad, origen y calidad de los datos, fidelidad con respecto a los datos reales que maneja el producto, vigencia de la información, entre otros) y escenarios simulados por cada ambiente. Este documento sirve de referencia a todos los equipos para saber dónde están los datos de las pruebas, cómo se manejan y qué propósito tiene cada ambiente preparado; y como constancia de los requisitos para la ejecución de pruebas.

Configurar Entorno de Testing y Ambientes: para el desarrollo normal de las pruebas, es importante contar con un ambiente de pruebas separado del ambiente de desarrollo. Se torna muy dificultoso para las actividades de testing interactuar con versiones que sufren cambios, o con datos inestables que no pueden ser controlados. Consecuentemente las pruebas pierden validez, y no es posible generar confianza en el producto bajo testing. Si ya contamos con un ambiente de testing aislado del de desarrollo, se debe poner en condiciones para la ejecución de pruebas. Esto incluye la instalación del producto en la versión que se desea verificar, la configuración de software de base, manejadores de bases de datos y cualquier otra aplicación que interopere con el producto o que sean herramientas anexas para la ejecución de pruebas. Algunas de estas tareas las puede llevar a cabo un tester, pero para la mayoría, es probable que se necesite el apoyo de expertos en infraestructura e incluso los desarrolladores; quienes suelen contar con mucho conocimiento técnico en configuración de entornos y ambientes.

Búsqueda de Datos para las Pruebas: es la investigación de los entornos y ambientes con el objetivo de extraer datos para especificar los casos de prueba, para conocer el perfil de los datos cargados en el dominio, y saber a grandes rasgos si será posible ejecutar los casos de prueba diseñados. Si no encontramos datos en los entornos que sean consistentes con las pruebas que se intentan ejecutar, entonces hay que intentar modificar las pruebas, o lo que es más común, intentar modificar los datos, creando el juego de datos necesario para la ejecución de las pruebas. Dependiendo del contexto, esta actividad puede llegar

a ser muy compleja, y puede llegar a condicionar la ejecución de ciertos casos en determinadas situaciones. Por ejemplo, supongamos que tenemos un diseño de pruebas o una misión que tiene por objetivo verificar el comportamiento de la aplicación ante un valor poco frecuente de una variable identificada, entonces, si el valor consumido de cierto entorno, no está cargado correctamente, ese escenario no se puede ejecutar en un principio con el mecanismo tradicional de ejecución de la aplicación; lo que en definitiva nos estaría anulando los casos de prueba relacionados con ese escenario[10].

Configuración de Herramientas de Apoyo: mediante esta actividad se instalan y configuran las herramientas anexas que darán soporte al resto de las actividades de testing. Entre estas herramientas pueden estar los sistemas de seguimientos de incidentes (como se ve en el apéndice A), herramientas de gestión en general, productos de ofimática y herramientas auxiliares al testing como captores de pantalla y editores avanzados de texto plano. Si las herramientas están instaladas y configuradas, esta actividad se focaliza en poblarlas de la información del proyecto de testing.

Ejecución de Pruebas

La etapa de ejecución de pruebas consiste en interactuar con el sistema bajo testing corriendo las pruebas planificadas sobre el ambiente de pruebas, con la intención de obtener información y detectar posibles defectos. Esta información será debidamente documentada e informada según la estrategia seguida. En el proceso de ejecución es posible que la misma interacción con el producto revele la necesidad de ajustar las pruebas.

Identificador	Nombre de la Actividad	¿Es requerida?
ECP	Ejecución de Casos de Prueba	SI
RDC	Rediseño y Adaptación de Casos	NO
DEJ	Documentación de Ejecución	NO

Cuadro 4.8: Etapa Documentación de Ejecución

[10]Para bajar más a tierra, supongamos que tenemos una aplicación que sugiere medicamentos genéricos compatibles a partir de las drogas especificadas. Puede suceder que cierta droga, nos condicione todo un escenario de prueba, pero, que no esté cargada en la base de datos de las drogas disponibles. En este caso, si no fuera posible generar o emular la extracción de esa información, entonces, los casos de prueba relacionados no se podrían ejecutar.

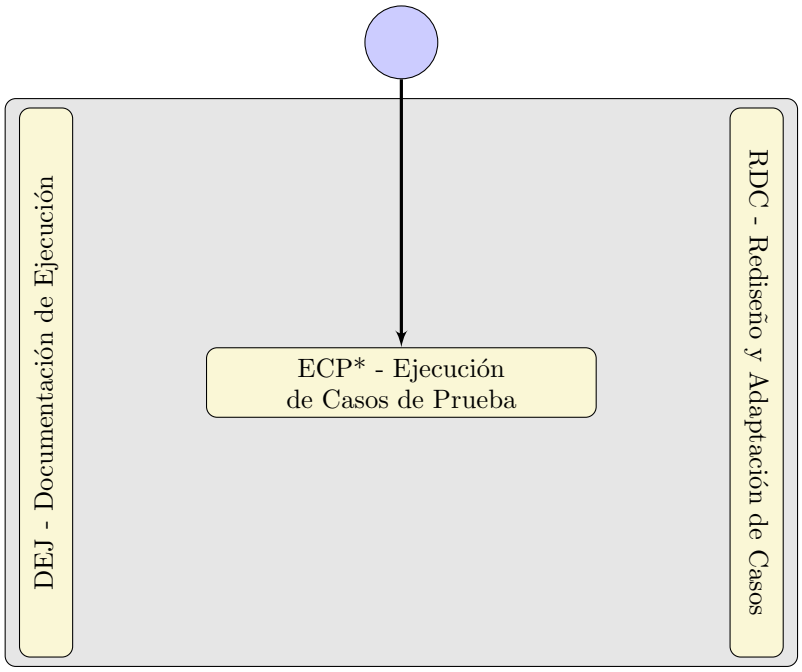

Figura 4.10: Etapa Ejecución de Pruebas

Roles involucrados en la etapa

- *Tester Profesional, Tester Funcional Líder:* se encargan de la ejecución de las pruebas, registro de incidentes y documentación de la ejecución. También quedan a su cargo las actividades relacionadas con la adaptación y rediseño de casos de prueba.
- *Tester:* también ejecutan y documentan las pruebas y registran incidentes. Las pruebas que ejecutan son diseñadas por los testers más experientes, pero es recomendable incluirlos en las actividades de diseño para que adquieran conocimiento y puedan profesionalizarse.

Productos de trabajo involucrados en la etapa

- *Casos de Pruebas Diseñados:* sirven como guía de ejecución de pruebas diseñadas, y pueden además transformarse en registro de ejecución de pruebas.
- *Documento de Misiones:* es el que indica los lugares por donde transitará a grandes rasgos el testing exploratorio (dependiendo de la estrategia o estilo abordado).
- *Documento de Ejecución:* se registra la evidencia de las pruebas. Puede ser un anexo a los casos de pruebas diseñados y especificados, agregando

la información del resultado de la prueba contra el resultado esperado. Otra forma de documentar la ejecución de las pruebas es contando con un documento aparte que muestra la ejecución de cada caso de prueba diseñado (expresado en el documento de Casos de Pruebas Diseñados, o Documento de Misiones) vinculando ambos documentos mediante el identificador del caso de prueba o de la misión. Este documento es muy útil a la hora de planificar las pruebas automatizadas. Se puede además anexar la información de los incidentes encontrados por cada caso de prueba, o en determinada ejecución de testing exploratorio.

Ejecución de Casos de Prueba: se trata de la actividad de ejecutar las pruebas en el sistema. Depende de la estrategia seguida. Si contamos con casos de prueba diseñados, tendremos una especificación de cómo se deben ejecutar las pruebas y los resultados esperados. Si tenemos un diseño más flexible o una estrategia estilo exploratoria, entonces seguiremos una ejecución que tiene una componente que se beneficia del valor agregado de la participación de un tester con más o menos experiencia, dejando lugar para la aplicación de modelos mentales, intuición y capacidad analítica de situaciones en el momento. La actividad de ejecutar pruebas implica el eventual registro de los incidentes detectados en el *sistema de seguimiento de incidentes*. En el apéndice A, se destacan los aspectos deseables de la herramienta que oficie de sistema de gestión de incidentes.

Rediseño y Adaptación de Casos: en el transcurso de las pruebas puede suceder que al contrastar el diseño de las pruebas, o las misiones planificadas contra la realidad del producto, nos encontremos con que no son del todo adecuadas. En estos casos se debe reconsiderar el conjunto de pruebas planificados, y hacer los ajustes necesarios, que pueden tanto ser leves, como de gran importancia, incluso impactando en estimaciones y transcurso del proyecto en sí. No tiene sentido ejecutar casos de prueba que no están acordes al estado actual del producto, ya sea porque el software está demasiado inestable, porque no maneja las funcionalidades de la manera que se consideró en la planificación de las pruebas, porque el mismo sistema no permite ciertos escenarios, entre otras muchas circunstancias. Las modificaciones pueden ir desde temas como modificar los datos de entrada, los resultados esperados, recortar o extender el alcance de las pruebas y cobertura, o incluso cambiar de tipo de prueba (por ejemplo, si el sistema está muy inestable, podríamos dar un enfoque más de *"prueba de humo"* que las pruebas estrictas diseñadas, que pueden no tener sentido en el estado actual de madurez del producto).

Documentación de ejecución: es una actividad intensamente ligada a la actividad de ejecución, pero es separada dado que se puede ejecutar casos de prueba, eventualmente, sin dejar evidencias documentadas. El objetivo es justamente, el de llevar un documento que respalde los resultados de la ejecución. Estos documentos, suelen ser un buen apoyo cuando se deben ejecutar

pruebas de regresión, o pruebas sobre funcionalidades ya verificadas. Incluso, en metodologías ágiles, pueden llegar a constituir respaldo documental de los requerimientos. Para el testing exploratorio, dependiendo de la rigurosidad, se va dejando documentación de los puntos visitados, los incidentes encontrados, las observaciones, datos de prueba y cualquier comentario que el tester considere relevante. El testing exploratorio basado en sesiones (ver en sección 2.4.2) tiene su propia forma de gestionarse y de documentarse, que se recomienda seguir si se desea aplicar con éxito este mecanismo y obtener sus beneficios. El documento de ejecución que esté acorde a un conjunto de casos diseñados, generalmente expresado en un documento de casos de prueba, se sugiere esté separado en secciones y resalte una sección especial para cada identificador de caso de prueba, correspondiente en ambos documentos. De esta manera, se podrá ver un panorama general de la ejecución en el documento de diseño de casos de prueba (especificados o no) y luego acceder a su evidencia mediante la búsqueda del identificador correcto en el documento de ejecución de pruebas.

Seguir y Actualizar el Plan

Ejecutándose en paralelo a las demás, esta etapa es la que corresponde al control frecuente de las actividades que se van sucediendo dentro de las diferentes etapas. Como documento base, utiliza el plan de testing, extrayéndose datos para construir mediciones que se contrastan con la realidad. Los desvíos al plan pueden ser manejados, reprocesando las estimaciones para las tareas faltantes, ajustando el plan y reasignando el esfuerzo para mantener el foco en los objetivos.

Identificador	Nombre de la Actividad	¿Es requerida?
RPT	Reestimar Proyecto de Testing	NO
AEF	Ajustar Esfuerzo	NO
APL	Ajustar Plan	SI

Cuadro 4.9: Etapa Seguir y Actualizar el Plan

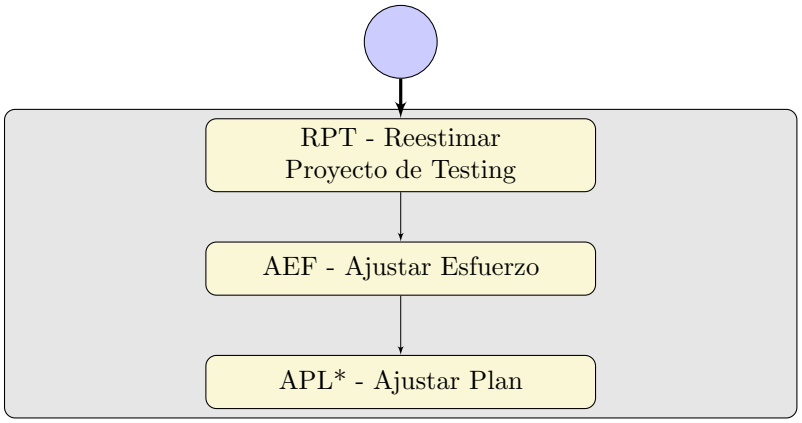

Figura 4.11: Etapa Seguir y Actualizar el Plan

Roles involucrados en la etapa

- *Tester Profesional, Tester Funcional Líder:* evalúan los cambios de contexto y lo contrastan contra la planificación actual. De ser preciso se pueden reunir con diferentes responsables con el fin de obtener mejor información, y presentar la replanificación. Los ajustes se pueden impactar en el documento plan de testing.

Productos de trabajo involucrados en la etapa

- *Plan de Testing:* es impactado con los cambios pertinente en cuanto a los ajustes efectuados a la planificación. También puede documentarse en él los desvíos y sus motivos, con el propósito de que en un futuro pueda servir como material de análisis en pro de mejorar los procesos en sí (conocer las causas por las cuales el camino para lograr los objetivos se aparta del planificado, puede derivar en tomar acciones preventivas en futuros proyectos).

Reestimar Proyecto de Testing: es la tarea de ajustar la estimación ante la presencia de desvíos y el cambio del contexto. Los proyectos suelen cambiar sus prioridades, lo que impacta en todas las áreas involucradas. Como todo plan, el Plan de Testing debe ser un documento dinámico acorde con la realidad. Ante los cambios del proyecto, se debe revisar el panorama y contemplar los factores que han cambiado y que motivan nuevas estimaciones con sus ajustes. Los diferentes factores internos y externos que pueden atacar el transcurso del proyecto, deben ser manejados y documentados en el plan. Replantear los objetivos, y modificar los cronogramas y actividades, nos mantendrá en el foco nuevamente de lograr el éxito del proyecto. Forzar un proyecto a cumplir un cronograma o planificación estática, que ya está obsoleta y es dispar con la realidad; lo conducirá a un probable fracaso.

Ajustar Esfuerzo: mediante esta actividad se reorganizan los recursos disponibles para orientarlos a la nueva realidad. Esta es una tarea de gestión que se ve beneficiada por las habilidades administrativas del Tester Líder que la ejecute. La estrategia es reasignar los diferentes recursos a las tareas pendientes, para que en el transcurso del tiempo del proyecto se cumplan los objetivos.

Ajustar Plan: es la actividad que se encarga de impactar las nuevas decisiones, los ajustes de esfuerzo y estimaciones en el documento Plan de Testing. El plan es la herramienta de seguimiento más importante y por lo tanto debe estar actualizado con la información más reciente. Tanto las nuevas decisiones como los desvíos al plan original pueden ser documentados, información que en un futuro análisis del proyecto puede ser útil para establecer mejoras al proceso de testing seguido.

Informar Resultados

Las etapas de comunicación son fundamentales para la articulación del proyecto. Se destacan algunas actividades particulares de comunicación relevantes al proyecto de testing. Esta etapa es el nexo del proyecto de testing con el proyecto de desarrollo, y los diferentes actores.

Identificador	Nombre de la Actividad	¿Es requerida?
CAO	Comunicar Avances y Obstáculos	SI
OBM	Obtener Mediciones	NO
CIN	Crear Informes	SI

Cuadro 4.10: Etapa Comunicar Avances y Obstáculos

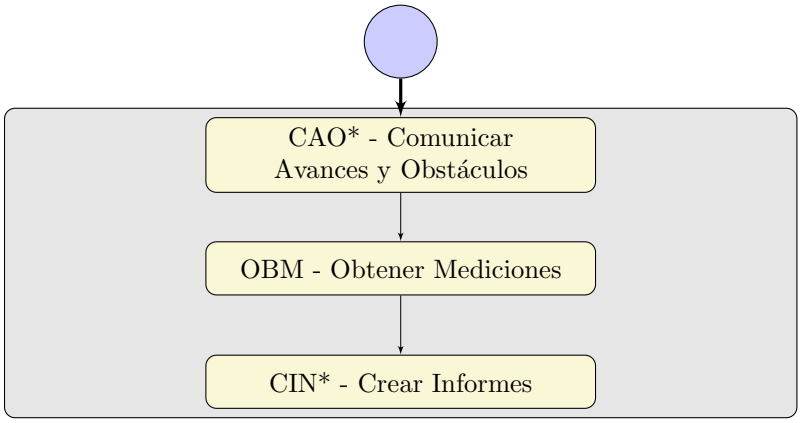

Figura 4.12: Etapa Informar Resultados

Roles involucrados en la etapa

- *Tester Funcional Líder:* elaboran los informes a partir de los hechos y las mediciones obtenidas según los patrones acordados. Estos roles suelen participar en la creación de informes finales y de estado del equipo frente a las tareas asignadas. Son los encargados de definir y acordar con los restantes equipos qué tipos de informes serán entregados.

- *Tester Profesional, Tester:* se encargan de completar los informes solicitados por el Tester Líder. Estos informes pueden ser destinados a Usuarios Finales, Desarrolladores, y otros perfiles por ejemplo de índole

política o dirección de la organización; por lo tanto el tester debe ser capaz de adaptar el vocabulario utilizado y visión del documento a cada interlocutor.

Productos de trabajo involucrados en la etapa

- *Informe de Avances y Obstáculos:* contiene el estado del proyecto de testing en un período particular de tiempo. Estos informes pueden ser diarios, semanales, o con la frecuencia que el Tester Líder defina o acuerde. Debe contar con la información de los logros y las dificultades del proyecto, aportando una visión general que pueda ser interpretada y rápidamente manejada por aquellos que necesiten estar informados del estado del proyecto, o quienes tengan las facultades para remover los obstáculos identificados.
- *Informe Final:* detalla los acontecimientos del proyecto de testing y es elaborado al finalizarlo. Contiene información acerca de los incidentes encontrados, las pruebas ejecutadas, las funcionalidades cubiertas, y comentarios sobre el trabajo completado.
- *Otros informes solicitados:* pueden existir documentos propios de la organización que se requiera completar al encarar actividades de testing. Deben ser documentos que puedan ser enriquecidos por la información derivada de las prácticas de testing, de lo contrario, sería recomendable revisar su inclusión en el proceso abordado.

Comunicar Avances y Obstáculos: a modo de resumen se informa a los responsables de las diferentes áreas, líderes y demás integrantes del equipo de los avances y puntos alcanzados, así como también se puede alertar de obstáculos encontrados en la ejecución del proyecto. Se estila que esta actividad sea llevada a cabo por cada integrante del equipo de testing cada cierto período de tiempo acordado.

Obtener Mediciones Relevantes Para el Contexto: se trata de la extracción de información a partir de la documentación generada en el proyecto de testing, y posterior procesamiento de esos datos para elaborar estadísticas y comunicar resultados. La información extraída puede ser de los incidentes, teniendo en cuenta su prioridad, estado de ejecución, criticidad, estado de reparación, funcionalidad afectada; o pude obtenerse información acerca de los desvíos del plan, de la cobertura de las funcionalidades, entre otros. Volviendo al caso particular del Testing Exploratorio Basado en Sesiones, hay muchas métricas[11] interesantes que se pueden obtener aplicando la herramienta que

[11] Algunas de las métricas propuestas por la técnica de Bach:

- Número de sesiones completas
- Número de problemas encontrados
- Áreas funcionales cubiertas

propone el método. Estas mediciones serán insumo importante de los informes presentados.

Crear Informes: es la elaboración de documentos informativos, de los cuales los interesados y líderes podrán servirse para tomar decisiones. Estos informes pueden ser informes intermedios de avance, o informes finales de proyecto. Suelen contener la información de la cobertura alcanzada, las estadísticas de los incidentes reportados, con su gravedad, origen y prioridad. Puede ir acompañado de alguna sugerencia o comentario de los testers respecto a la impresión general del comportamiento y estado del producto.

Verificación de Documentos

La etapa de verificación de documentos se compone de actividades que pueden aplicarse desde el inicio del proceso de desarrollo. El objetivo es encontrar ambigüedades, carencias en claridad, formato y redacción de los documentos que servirán de entrada para diferentes equipos de la organización. Esta etapa no es obligatoria, pero es muy recomendable que sea ejecutada en contextos donde los documentos son insumo fundamental para la construcción del producto, la verificación, así como cuando son material importante en la toma de decisiones del proyecto.

Roles involucrados en la etapa

- *Equipo de Testing:* verifican los documentos y envían sus dudas y observaciones a los autores. Si las observaciones generan correcciones, el documento puede volver a manos del equipo de testing para un nuevo ciclo de verificación. De tratarse de documentos de definición de requerimientos y funcionalidades, es sabido que los errores encontrados son ventajosos en costo para la organización frente a los errores encontrados en etapas posteriores, cuando el producto está construido, y por lo tanto es recomendable (no obligatorio) que el equipo de testing participe de la revisión de los documentos.

Productos de trabajo involucrados en la etapa

- *Informe de Consultas y Observaciones:* contiene las dudas y puntos observados en la verificación efectuada por el equipo de testing. Las observaciones pueden contener sugerencias de redacción, formato, identificación de ambigüedades, faltas ortográficas y principalmente errores en definición (de requerimientos por ejemplo). Las herramientas informáticas proveen mecanismos para elaborar la corrección mediante control de cambios, por

---

- Porcentaje del tiempo de la sesión dedicado a preparar el testing
- Porcentaje del tiempo de la sesión dedicado al ejecutar pruebas
- Porcentaje del tiempo de la sesión dedicado a investigar problemas

lo tanto muchas veces este documento está incluido en una versión revisada del documento a verificar. Aunque se pueda contar con la versión revisada y corregida, es buena práctica extraer las dudas y observaciones en un documento aparte que pueda ser fácilmente consultado en el futuro (o que sirva de insumo para la comprensión del contexto, y corrección de otros documentos).

- *Documentos a verificar:* son los documentos que estarán en el circuito de verificación del equipo de testing, y corrección por parte de los autores. Los incidentes encontrados a los documentos pueden incluso registrarse en el sistema de seguimiento de incidentes si el proceso lo requiere.

Identificador	Nombre de la Actividad	¿Es requerida?
ESP	Establecer Parámetros Observables	NO
RVD	Revisión de Documentos	SI
ROB	Registrar Observaciones a Documentos	NO
IRD	Crear Informe de Revisión de Documentos	SI

Cuadro 4.11: Etapa Verificación de Documentos

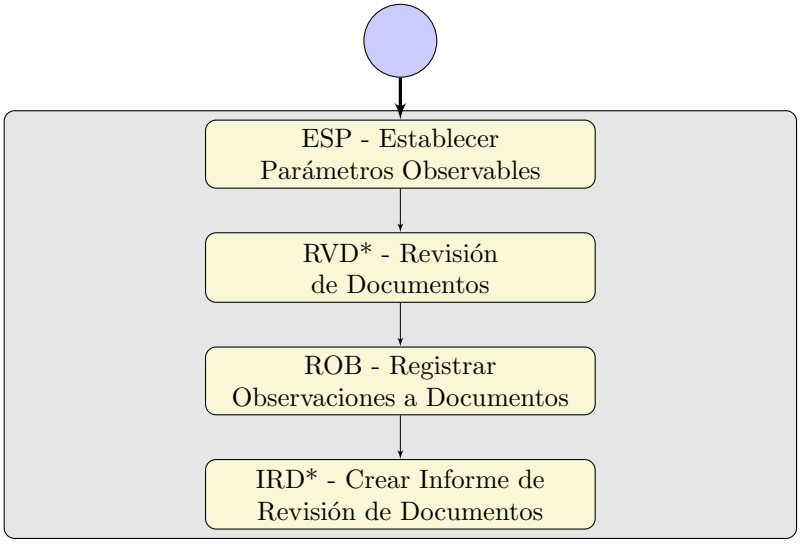

Figura 4.13: Etapa Verificación de Documentos

Establecer Parámetros Observables: se construye una lista de elementos a los que se pondrá especial énfasis en la verificación de documentos. Los aspectos pueden involucrar la visión del testing, de otros equipos de desarrollo, parámetros de expertos del negocio entre otros involucrados en el proyecto. Para la revisión de documentos que no están directamente involucrados con las actividades de testing, o que se desea un enfoque desconocido por los testers, basta con acordar con el equipo de testing cuáles son los puntos a tener en cuenta durante la revisión, así como la información esperada del documento.

Revisión de Documentos: actividad que corresponde a la lectura e investigación de los documentos con la intención de detectar puntos débiles. En cuanto al testing, se verifica la documentación para corroborar que los requisitos sean verificables, estén completamente definidos, no tengan problemas de redacción, ambigüedades u otras carencias. No solamente los documentos que contienen requerimientos pueden ser verificados, sino también todo documento para el que se considere importante que cuente con revisiones imparciales, independientes de su autor original. Una buena estrategia que apoya actividades de testing temprano es que el equipo de testing esté verificando los documentos involucrados en la construcción del producto y que serán insumo para la construcción del futuro *testware* y por consiguiente de la ejecución de las actividades de testing.

Registrar Observaciones a Documentos: se trata del registro de las dudas, consultas y observaciones que el tester identifica en el documento. Muchas veces las mismas herramientas de ofimática proveen funcionalidades para el control de cambio. En proyectos de mayor escala, con procedimientos de documentación estrictos, puede ser necesario registrar estas observaciones directamente como incidentes en un sistema de seguimiento de incidentes, rigiéndose por las mismas reglas que el ciclo de prueba de un incidente del producto construido en sí.

Crear Informe de Revisión de Documentos: es la manera en la que se comunican los resultados de la revisión. Puede ser un documento aparte, destacando las observaciones, dudas y consultas por cada sección del documento revisado. Es común que este informe, sea un versionado del mismo documento, aprovechando las capacidades de revisión y control de cambios que brindan las herramientas de ofimática (haciendo seguimiento de modificaciones, sugerencias, comentarios, entre otras funcionalidades).

Análisis de Defectos

La información almacenada de los incidentes puede ser analizada para extraer conclusiones interesantes. Esta etapa puede ser parte de un análisis *post mortem* del proyecto al finalizar un periodo de pruebas, o como para reforzar la estrategia de pruebas de regresión. El objetivo común es aprender de los acontecimientos y tomar ventaja de ellos en el futuro. Así, podremos identificar puntos débiles y puntos fuertes, teniendo material para generar alertas, mejorar prácticas o tomar decisiones ventajosas para el contexto.

Identificador	Nombre de la Actividad	¿Es requerida?
AIN	Analizar Información de Incidentes	SI
PIN	Procesar Información de Incidentes	NO
CIA	Crear Informes de Análisis de Incidentes	SI

Cuadro 4.12: Etapa Análisis de Defectos

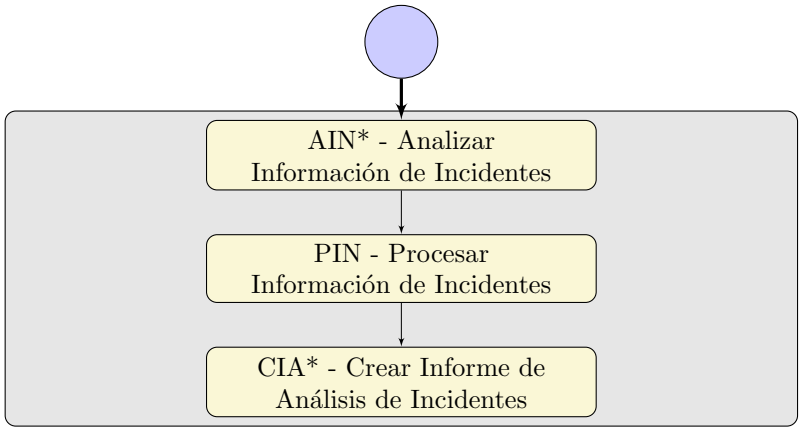

Figura 4.14: Etapa Análisis de Defectos

Roles involucrados en la etapa

- *Tester Profesional, Tester Funcional Líder:* analizan la información de los incidentes registrados, tratando de encontrar patrones o elementos notables que se puedan aprovechar para mejorar el proceso. Son los encargados de sintetizar y comunicar esta información.

Productos de trabajo involucrados en la etapa

- *Informe de Análisis de Defectos:* presenta los datos estadísticos de los incidentes y las conclusiones que de ellos se pueden extraer. Se pueden mencionar aspectos sobre qué funcionalidades son más propensas a errores, cuáles son los tipos de errores que se cometen más, dónde está la fuente del error en el proceso (por ejemplo en la especificación de los requerimientos, en la construcción del producto, en el diseño de la solución) entre otros factores.

Análisis de la Información de Incidentes: es la actividad de recopilar y seleccionar la información de los incidentes que se considere necesaria para extraer conclusiones. Comienza por la investigación del repositorio de incidentes y también estudia las particularidades del proyecto, revisando requerimientos, o información de las prioridades, riesgos identificados y manejados, y elementos presentes en el Plan de Testing. También involucra la tarea de decidir cuáles son los criterios para clasificar la información, cuáles mediciones queremos obtener, y qué factores deseamos investigar. A modo de ejemplo, el análisis puede estar enfocado a: investigar qué sucedió con un conjunto de funcionalidades, los desvíos en los tiempos del proyecto de testing y sus causas, identificar proveedores más confiables, establecer una mejor estrategia para las pruebas de regresión, identificar fortalezas y debilidades en el proyecto de desarrollo, entre muchos otros posibles objetivos.

Procesar Información de Incidentes: el esfuerzo en esta actividad se concentra en ordenar la información y procesarla según las mediciones que se deseen obtener. Los incidentes suelen tener información de gravedad, tipo de incidente, versión del producto, proveedor o desarrollador involucrado, estado de la resolución, entre otros. Lo importante es tomar la muestra de incidentes a analizar, supongamos, para una versión del producto, y categorizarlos para ajustarse a las diferentes métricas definidas. Información común que se desea obtener puede ser: funcionalidades con más y menos errores (según gravedad), proveedores que provocan más defectos, incidentes de gravedad alta que no fueron reparados, cantidad de idas y vueltas entre desarrollo y testing para un incidente de cierta funcionalidad, incidentes de las diferentes etapas de desarrollo (errores en requerimientos, diseño de la solución, en implementación) entre muchos otros posibles factores.

Crear Informe de Análisis de Incidentes: es el trabajo de documentar la etapa de Análisis de Defectos. El informe debe contener los resultados obtenidos del procesamiento de la información del análisis y puede ir acompañado de algunas sugerencias sobre los puntos fuertes y débiles, y cómo atacarlos. También puede contener secciones comentando el proceso seguido para el análisis, medición y construcción de la información, así como también alguna guía acerca de cómo se debe interpretar la información contenida en el documento.

4.4.2. Roles

Un rol se especifica mediante un nombre y una descripción de las funciones que desempeña una persona que lo asume. Puede incluir las aptitudes técnicas deseadas o requeridas para desempeñar cierto grupo de actividades, así como las características más importantes que distinguen al rol. Los roles pueden ser adoptados por varias personas, y a su vez, una persona puede tomar varios roles. Esto sucede a menudo con roles más generales o más experientes que abarcan las actividades de los roles más básicos.

Tester Funcional

Conoce los conceptos básicos de testing de software. Es capaz de ejecutar pruebas, trabajar en conjunto con líderes de testing y testers más experimentados. Posee habilidades de comunicación y es capaz de participar en los procesos de diseño de pruebas aplicando técnicas, y análisis de requerimientos de testing en conjunto con testers más experientes o profesionales.

Tester Funcional Profesional

Cuenta con las habilidades del *Tester Funcional* más experimentadas, además es capaz de analizar requerimientos, construir requerimientos de testing, seguir una metodología o proceso de testing, establecer una estrategia adecuada para un determinado contexto, diseñar pruebas y elaborar documentación e informes de testing. También es capaz de contribuir a la elaboración de un plan de testing, seguirlo y ajustarlo. Posee conocimiento en técnicas de diseño de pruebas y es capaz de discernir cuál aplica para el caso dado, y que beneficios aporta. Debe trabajar en coordinación con líderes de testing y responsables. Apoya a los usuarios en sus pruebas de aceptación. Puede responsabilizarse de proyectos de testing de mediano porte.

Tester Funcional Líder

El tester líder, cuenta con las habilidades del tester profesional y además, es capaz de coordinar y dirigir las actividades de testing, siguiendo determinados procesos, pudiendo aportar en la mejora continua de estos. Dirige al equipo de testing, y es su responsable. Interactúa con los responsables de las demás áreas dando visibilidad sobre la información generada por el testing, aportando conocimiento en su utilización para toma de decisiones. Puede asistir al tester profesional en la elección de las estrategias y guiarlo en el rumbo a seguir. Establece los objetivos del proyecto de testing particular y su interacción con el proyecto de desarrollo. Entrena y promueve la capacitación continua del grupo de testing, además de los actores involucrados en el proyecto cuando sea necesario. Puede responsabilizarse por proyectos de gran porte.

Usuarios Funcionales

Serán los encargados de ejecutar las pruebas de aceptación de los productos. Puede ser en conjunto con los testers o no. Son los clientes internos, y definirán si un producto, o determinado cambio cumple con sus expectativas. También ofician de oráculo[12], al ser expertos en el dominio; poseen el conocimiento del negocio, y muchas veces, son los destinatarios finales del producto[13].

Referentes o Responsables de Desarrollo

Se encargan de la implementación de las soluciones, responden al Líder de proyecto. En relación al testing funcional, es el actor encargado de notificar al tester de un nuevo producto, cambio o liberación para pruebas. Coordinan la reparación de los incidentes. Toman decisiones a partir de las pruebas de los testers o de los usuarios funcionales[14].

Líderes, Responsables de Proyecto o Gestores

Están encargados de la coordinación del proyecto de desarrollo. Trabaja en coordinación con el Tester Líder, elaborando nuevas estrategias a partir de la información aportada por testing. En cuanto al relevo de requerimientos, es un nexo importante entre el tester y los usuarios funcionales, sobre todo al inicio del proyecto. Negocia la propuesta de alcance del plan de testing, y colabora en la priorización de funcionalidades y casos de prueba. El Líder puede estar presente en pruebas de aceptación y debe tomar decisiones de liberación de funcionalidades y productos a partir de la información aportada por testing.

En cuanto a los incidentes, el Líder de proyecto es también un nexo inicial entre desarrolladores y testers, y puede coordinar la reparación de los incidentes según su percepción de la prioridad, sirviéndose de la prioridad sugerida por testers.

Especialista en Infraestructura:

Coordinan y gestionan la instalación y configuración de los ambientes, entornos, herramientas de apoyo, permisos, y todo lo que tenga que ver con infraestructura tecnológica para que se puedan llevar a cabo las pruebas. Generalmente toda organización cuenta con un responsable en esta materia (o es implícitamente ejecutado por integrantes del equipo de desarrollo).

[12] Un oráculo nos brinda información acerca del resultado esperado de una prueba. Es usado para discernir cuando un test fue exitoso o estamos ante un posible incidente.
[13] Este rol se conserva desde el modulo inicial.
[14] Este rol tiene variaciones desde el identificado en el punto de partida.

4.5. FIT Módulo III: Proceso de Testing Funcional Automatizado

Existen ciertos tipos de problemas que no pueden ser atacados solamente con las etapas del proceso del Módulo II del FIT. Algunos de ellos motivan a la incorporación de soluciones orientadas a la automatización de pruebas, por ejemplo si:

- El testing Manual consume mucho tiempo, es repetitivo y tedioso, y es propenso a contener errores.
- Las restricciones de tiempo del proyecto no admiten dedicar esfuerzo a ejecutar pruebas de regresión ante sucesivas liberaciones del producto, o ante pequeños cambios en el software. Finalmente, el producto es liberado con poco testing, o sin testing alguno.
- Es necesario correr pruebas que involucren grandes volúmenes de datos.

Un siguiente análisis posible, podría referirse al estado de la organización, para decidir por ejemplo si se encuentra en condiciones de enfrentar un proyecto de automatización. Los siguientes son algunos indicadores de situaciones en las que sería conveniente implementar automatización de pruebas:

- Cuando no se está en momentos de grandes cambios organizacionales, grandes presiones, o un clima de des-organización general en curso.
- Cuando el estado actual del testing no es satisfactorio.
- Cuando existe un compromiso e interés estratégico y político de la organización en apoyar al testing automatizado y manual.

Si la realidad de la organización no se ajusta a los puntos mencionados, no significa que no se deba introducir testing automatizado, sino que quizás sean mayores los obstáculos a superar. En definitiva, la automatización supone una inversión relativamente superior a la de las pruebas manuales, con un retorno a más largo plazo.

El *roadmap* definido ha transitado por el proceso del Módulo II del FIT, cumpliendo etapas y actividades de testing manual y a continuación lo hará con las etapas y actividades del proceso de testing automatizado del Módulo III del FIT. En este módulo se busca definir el proceso de automatización, con la tecnología a usar, y sus detalles técnicos. El Módulo III del FIT tiene como objetivo brindar soporte a la implementación del testing automatizado y lograr su institucionalización.

4.5.1. Etapas de Proceso de Testing Funcional Automatizado

Definiremos a continuación las etapas del módulo con sus identificadores, mencionando los productos de trabajo que están relacionados. Posteriormente, detallaremos cada una de las etapas involucradas y las actividades que las componen.

El proceso de testing automatizado consta de las siguientes etapas:

- SHE - Seleccionar Herramientas.
- SPR - Seleccionar Pruebas.
- PDA - Preparación de Automatización.
- IPA - Implementación de Prueba Automatizada.
- EPA - Ejecución de Pruebas Automatizadas.
- AED - Análisis de Errores Detectados.
- MTA - Mantenimiento de Testware de Automatización.
- INR - Informar Resultados.

En este módulo, todas las etapas del proceso deberían ejecutarse. Para que el proceso sea flexible, se admite variar la rigurosidad con que se ejecutan las diferentes actividades, cómo se documenta el proceso, y además cuáles de las actividades opcionales serán elegidas para integrar el *roadmap* del proyecto particular.

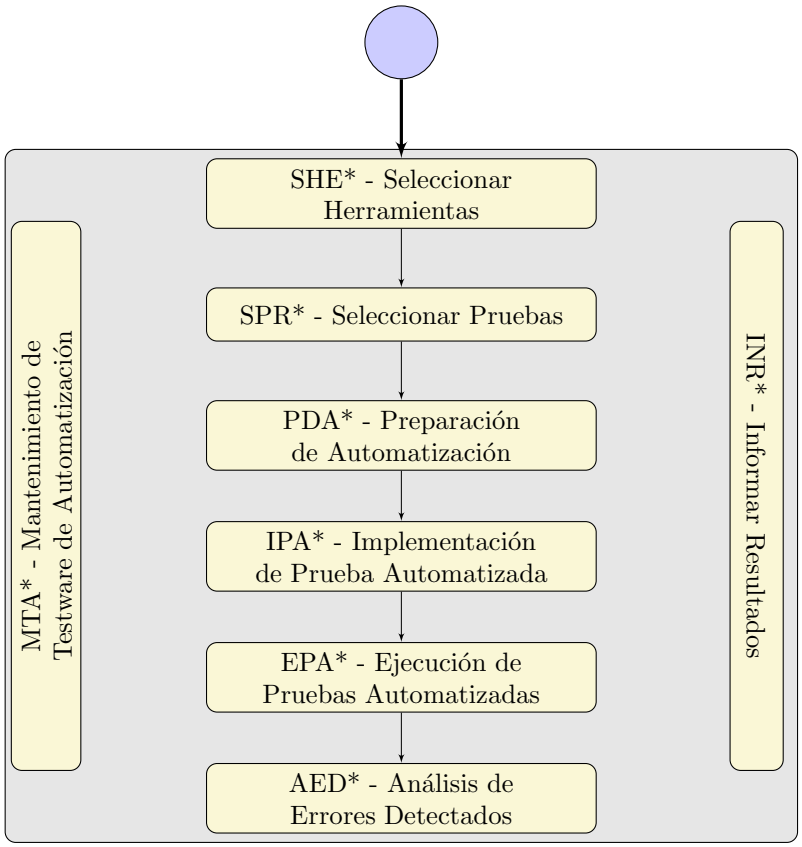

Figura 4.15: FIT Módulo III: Proceso de Testing Funcional Automatizado

El cuadro siguiente muestra en qué etapa se construye o se complementa con más información cada documento.

Identificador	Nombre de la Etapa	Documento Involucrado
SHE	Seleccionar Herramientas	Informe de Herramientas
SPR	Seleccionar Pruebas	Casos de Pruebas Diseñados / Informe de Incidentes / Documento de Misiones y Sesiones / Documento de Diseño de Pruebas Automatizadas
PEJ	Preparación de Ejecución	Documento de Entornos y Ambientes
IPA	Implementación de Pruebas Automatizadas	Documento de Diseño de Pruebas Automatizadas
EPA	Ejecución de Pruebas Automatizadas	Documento de Resultados de Automatización
MTA	Mantenimiento de Testware de Automatización	Documento de Mantenimiento de Automatización

Cuadro 4.13: Documentos del Proceso de Testing Funcional Automatizado.

Seleccionar Herramientas

En la etapa de selección de herramientas se compromete gran parte del éxito del proyecto. Las herramientas son "la punta del iceberg", es decir, si bien no es todo el proceso de automatización que pasa por las herramientas a utilizar, son el elemento más notable. Se busca lograr el conjunto de herramientas más adecuado al contexto. Para este fin seguiremos algunas actividades que den un marco de confianza que evite una decisión apresurada y con pocos argumentos.

Identificador	Nombre de la Actividad	¿Es requerida?
IDR	Identificar Restricciones	SI
INH	Investigar Herramientas	NO
CPC	Construir Prueba de Concepto	NO
EIH	Elaborar Informe de Herramientas	NO
DFH	Definir Herramientas	SI

Cuadro 4.14: Etapa Seleccionar Herramientas

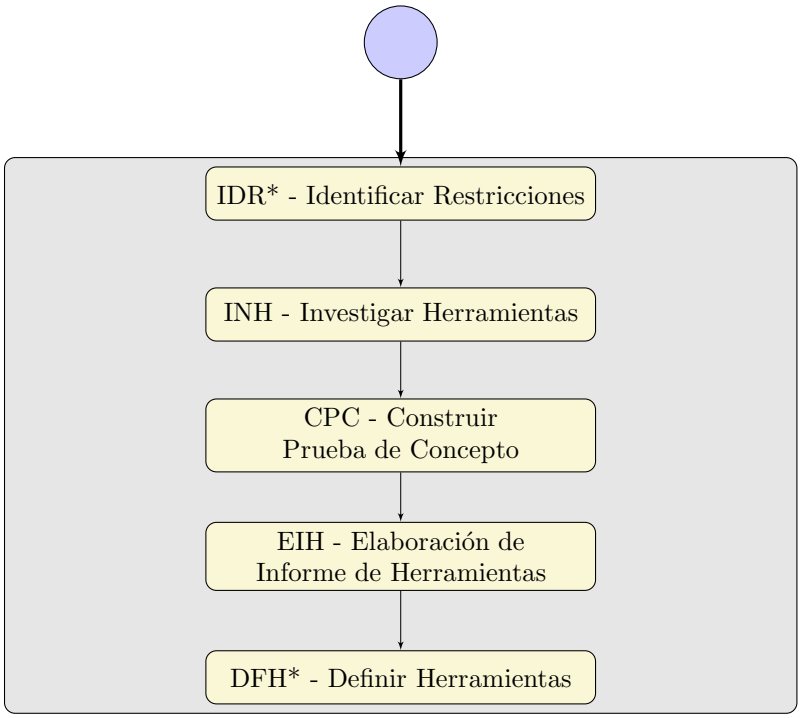

Figura 4.16: Etapa Seleccionar Herramientas

Roles involucrados en la etapa

- *Tester Profesional, Tester Funcional Líder, Tester Automatizador:* investigan y elaboran pruebas de concepto con el fin de evaluar las herramientas y recopilar información para que la decisión sobre las herramientas a utilizar se base en argumentos sólidos, dado que la definición de las herramientas a utilizar puede estar en manos del equipo de testing, o puede ser responsabilidad de otros roles (sobre todo si involucra una inversión económica).

Productos de trabajo involucrados en la etapa

- *Informe de Herramientas:* contiene las conclusiones sobre el proceso de selección de herramientas, con los resultados de la evaluación destacando puntos fuertes y débiles de cada una en relación con el contexto. Además, de ser necesario, se puede incluir información desprendida de cualquier otra actividad del proceso de selección de herramientas, como ser la identificación de las restricciones (las que pueden transformar determinadas características de las herramientas en ventajas o desventajas).

Identificar Restricciones: consiste en la elaboración de una lista de condiciones específicas de la organización, el producto y el estado actual del proyecto. Las restricciones se pueden agrupar en diferentes categorías como veíamos en la sección 2.5.2.

Algunas de las categorías para ordenar restricciones pueden ser:

- *Requisitos de Uso*: las configuraciones de hardware y software que deben estar presentes para poder hacer uso de la herramienta.

- *Plataforma del Producto*: la plataforma y la arquitectura del producto que se va a automatizar, deben ser soportadas por la herramienta seleccionada. Por ejemplo, si la aplicación es web, sobre java, o si es cliente/servidor con una interfaz de escritorio, entre otras posibilidades.

- *Interoperabilidad*: cómo se integra la herramienta con el producto, con otras herramientas, con software y hardware de base.

- *Soporte*: la disponibilidad de recursos para solucionar eventuales problemas, como ser soporte técnico propietario o la disponibilidad de comunidades de usuarios para herramientas libres. Un factor importante a considerar al evaluar las restricciones soporte son la presencia en el mercado local de un representante de la herramienta, o de una comunidad de usuarios activa.

- *Costo y Estrategia*: la restricción económica al uso de la herramienta en cuanto al precio de las licencias de uso; o imposiciones de marco político y estratégico, por ejemplo que indiquen la obligatoriedad del uso de ciertos componentes o herramienta de determinado proveedor comercial, o del mundo del software libre. También el tiempo requerido o disponible para desarrollar un proyecto de automatización, es una limitante estratégica.

- *Curva de Aprendizaje*: el tiempo estimado que le tomaría a un usuario inexperto, o a un equipo en determinado nivel en relación a una herramienta, dominar la tecnología que ofrece cierta herramienta.

Teniendo claras las restricciones, se puede contrastar las prestaciones de las herramientas investigadas y candidatas, tratando de identificar cuáles las satisfacen.

Investigar Herramientas: las herramientas candidatas pueden estar predeterminadas por la organización por temas políticos o estratégicos, o porque se considera que existe mucha experiencia acumulada en su utilización. De todas formas es conveniente aventurarse periódicamente a investigar nuevas soluciones en el mercado comercial, y en el mundo del software libre.

Construir Prueba de Concepto: consiste en crear prototipo tecnológico sencillo con el fin de identificar mayores inconvenientes al interactuar con la herramienta, y así evaluar la posible inclusión en la lista de herramientas candidatas. Se busca identificar las ventajas y desventajas de la herramienta en los siguientes aspectos:

- *Curva de aprendizaje*: la dificultad con la que el automatizador puede interactuar con la herramienta y conseguir un resultado satisfactorio para el proyecto de automatización
- *Funcionalidad de grabación y reproducción*: es deseable que para ciertos escenarios, como ser el de desarrollo web, se pueda contar con la ventaja de poder grabar un esqueleto a la vez que se corre la aplicación. Esto es ventajoso dado que no es necesario comenzar a escribir el *script* desde cero.
- *Configuración*: el tiempo que se dedica a que la herramienta esté funcionalmente disponible para su uso.

Elaborar Informe de Herramientas: se construye un documento listando las herramientas analizadas, destacando las herramientas candidatas, identificando sus ventajas y desventajas. El informe contiene algunos puntos del proceso de selección de herramientas, como las restricciones de la organización, comentario sobre las herramientas, y opiniones, percepciones o juicios informales sobre las herramientas, entre otros puntos.

Definir Herramientas: finalmente, luego del proceso, se busca llegar al conjunto de herramientas a utilizar para la automatización. Es importante que esta decisión sea tomada cuidadosamente, basándose en el análisis de la información generada en el proceso, dado que puede tener un gran impacto en el proyecto el hecho de que sea necesario redefinir el esquema de herramientas de automatización.

Seleccionar Pruebas

La etapa de selección de pruebas consta del análisis del conjunto total de elementos automatizables con el fin de decidir cuáles de ellos serán automatizados, cuáles no, y en qué orden. Se puede optar entre múltiples criterios para seleccionar pruebas, como los vistos en la sección 2.5.1.

El conjunto inicial de casos de pruebas podría ser generado a través de testing diseñado, ser compuesto por descripciones de pruebas motivadas por anotaciones del testing exploratorio, e incluso ser la reproducción de los incidentes encontrados en otras instancias de pruebas. Es importante identificar los objetivos que buscamos de la automatización y priorizarlos. El foco puede ser ahorrar tiempo en tareas repetitivas, optimizar la ejecución con grandes volúmenes de datos, evitar repetir tareas complejas y tediosas o llegar a un conjunto de pruebas que

ataquen el núcleo del producto, entre otros. Tener las prioridades nos ayudará a seleccionar los casos de prueba y decidir en qué orden serán implementadas las pruebas automatizadas y en qué orden es conveniente ejecutarlas.

Esta etapa está compuesta por algunos puntos estrechamente relacionados, en lugar de actividades como las restantes etapas:

- **Contar con el conjunto de casos de prueba:** sean casos de pruebas diseñados, notas de testing exploratorio, reproducciones de incidentes.
- **Identificar el objetivo de la automatización:** crear una lista priorizada de lo que se espera conseguir con la automatización de pruebas.
- **Seleccionar las pruebas según objetivos:** teniendo en cuenta la lista de objetivos de la automatización, seleccionar los casos de prueba que estén mejor orientados a cumplir los objetivos.
- **Decidir orden de implementación y ejecución:** organizar los casos de pruebas seleccionados con el fin de decidir cuáles se implementan primero y en qué orden deberían ser ejecutados. La relación entre los diferentes casos de prueba y la lista priorizada de objetivos nos ayudará con esta organización.

Roles involucrados en la etapa

- *Tester Profesional, Tester Funcional Líder:* analizan las pruebas ejecutadas y sus resultados, notas de ejecución y los incidentes detectados y corregidos para seleccionar un conjunto de pruebas a desarrollar, o identificar los lugares donde va a estar atacando la automatización. Trabajan además con el *Tester Automatizador* para conocer la visión de qué implica automatizar determinada prueba.
- *Tester Automatizador:* aporta la opinión más valiosa acerca de qué trae como consecuencia automatizar determinada prueba. Por ejemplo en temas de complejidad de implementación con las herramientas dadas, mantenibilidad de las pruebas, tiempo que insume y en general qué recurso insume la automatización en sí (información crucial a la hora de contrastarla con el beneficio que reporta, y poder tomar una decisión al respecto de cómo invertir en automatización).

Productos de trabajo involucrados en la etapa

- *Documento de Diseño de Pruebas Automatizadas:* presenta las pruebas para las cuales se tiene la intención de implementar pruebas automatizadas. Los casos de pruebas pueden estar identificados y así hacerlos corresponder con su implementación, lo cual es un extra de trazabilidad que puede facilitar las tareas de mantenimiento de los tests automatizados. Puede incluir diseño detallado de los pasos y puntos de verificación de los *scripts*, para

un mantenimiento más riguroso. Además pueden organizarse los *scripts* por criterio de funcionalidad cubierta, complejidad, tipo de prueba, *suites* a las que pertenecen, entre otros.

Preparación de Automatización

Para dar paso a la automatización de pruebas, es necesario transitar por las actividades de preparación. Esta etapa incluye la instalación y configuración de las herramientas que sirven a la automatización, así como también los ambientes en donde estará instalado el producto, junto con sus fuentes de datos. Por último, como actividad prioritaria se destaca el diseño de la prueba automatizada, la agrupación lógica de los archivos de la automatización y el orden en que se deben ejecutar las pruebas.

Identificador	Nombre de la Actividad	¿Es requerida?
PEP	Preparar Entorno y Producto	SI
DPA	Diseñar Prueba Automatizada y Organizar Ejecución	SI
PFA	Preparar Framework de Automatización	SI

Cuadro 4.15: Etapa Preparación de Automatización

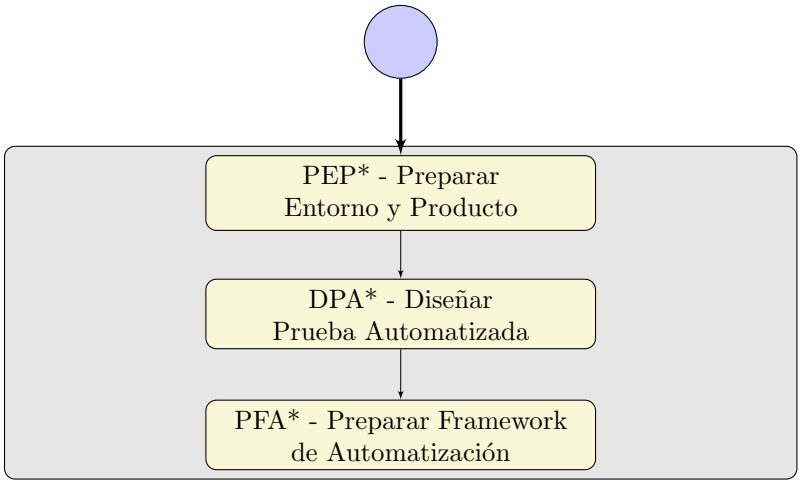

Figura 4.17: Etapa Preparación de Automatización

Roles involucrados en la etapa

- *Tester Profesional, Tester Funcional Líder, Tester Automatizador:* colaboran en la definición de las pruebas automatizadas y en cómo serán ejecutadas. Es conveniente que el *Tester Automatizador* cuente con esta información como entrada a su trabajo, dado que de lo contrario tendrá que invertir tiempo investigándola y definiéndola. El *Tester Automatizador* por su parte construye y acondiciona las herramientas de automatización necesarias, contando con el apoyo de algún *Especialista en Infraestructura*.

- *Especialista en Infraestructura:* instala las versiones del producto y las herramientas necesarias para la construcción y ejecución de las pruebas automatizadas. Puede trabajar en equipo con el *Tester Automatizador*, dado que por su perfil más técnico en programación, tiene conocimientos de desarrollo, instalación y configuración de herramientas y ambientes.

Productos de trabajo involucrados en la etapa

- *Documento de Entornos y Ambientes:* como en el módulo de Testing Funcional Manual, este documento detalla los ambientes de testing, incorporando además la información sobre las herramientas a utilizar y cómo encontrarlas en el entorno preparado (en términos generales, se puede decir que es el mismo documento presentado en el módulo II, que además anexa esta información).

Preparar Entorno y Producto: dependiendo de la naturaleza de las pruebas automatizadas que se intenten lograr, será la severidad con que esta actividad debe ser ejecutada. Las pruebas orientadas a datos son extremadamente sensibles a los cambios en los entornos. Es conveniente, siempre que sea posible, independizarse de los datos almacenados en las bases de datos y que la prueba cuente con un mecanismo para generarse sus propios datos de entrada. Esto puede resultar costoso de implementar y en tiempo de ejecución, pero muchas veces es la única alternativa posible para poder ejecutar pruebas automatizadas, sobre todo en entornos inestables o que son accedidos por múltiples actores. La preparación de los datos y del producto para la ejecución de la automatización se puede hacer mediante módulos especiales que ingresen lo datos necesarios, ya sea mediante la interacción con el sistema o levantando respaldos en bases de datos. Cuando el volumen de la prueba lo amerita, la preparación de los datos y el producto puede ser manual; por ejemplo, si el ingreso de datos es muy complejo de automatizar, pero generándolos manualmente es posible poner a ejecutar una prueba que ahorra mucho tiempo de testing manual, entonces, la preparación manual de los datos y el producto se vuelve conveniente.

Diseñar prueba automatizada y organizar ejecución: el objetivo es documentar la organización lógica de los *scripts*, y las particularidades de organizar las pruebas en conjuntos relacionados. Los *scripts* de prueba pueden tener una contraparte en mayor nivel que indique los puntos más relevantes

que toca el *script* lo que favorece el mantenimiento de los *scripts*, visto en la sección 4.5.1. Llamaremos *"suite"* al conjunto de *scritps* agrupados bajo algún criterio. Una *suite* suele encargarse de un ciclo funcional, de una serie de pasos, por ejemplo, imitando el comportamiento de un usuario en la aplicación desde que ingresa, tiene interacción, ingresa datos, obtiene resultados y sale de la aplicación. Esta forma de estructurar las pruebas automatizadas favorece la reutilización de *scritps*. Es conveniente que los *scritps* se encarguen de tareas puntuales y manejables para poder integrar más de una *suite*. De contar con una complejidad superior, puede ser necesario organizar las *suites* en áreas de pruebas funcionalmente relacionadas. Un área de prueba se compone de *suites*, y estas se componen de *scritps*. Áreas de pruebas y *suites* son agrupaciones lógicas de elementos semánticamente relacionados, sólo que de diferente jerarquía.

Preparar Framework de Automatización: consta de la instalación de las herramientas necesarias para implementar y ejecutar las pruebas automatizadas, y del diseño de la arquitectura de las pruebas tanto en la organización del *testware* como en la definición de la información a extraer de las pruebas. Por ejemplo, se tratan temas de en qué forma se van a almacenar los archivos generados por la automatización, cuál va a ser la jerarquía de los directorio, qué información se registrará en un *script*, y cómo se relacionarán las áreas de pruebas, *suites* y *scripts* entre sí y en la jerarquía de directorios propuesta.

Un framework mínimo de automatización cuenta con:

- *Herramientas de desarrollo:* los entornos en los cuales se implementaran las pruebas. Puede ser que la herramienta cuente con su entorno y lenguaje propios, o que se acuda a lenguajes de programación populares.
- *Herramientas de ejecución:* son capaces de ejecutar una prueba automatizada escrita en determinado lenguaje sobre un sistema. Por ejemplo existen motores de ejecución sobre plataforma web así como también para aplicaciones de escritorio. Las herramientas de ejecución pueden ser para uso general, como específicas de un producto particular. Este es el caso de las aplicaciones que se construyen con la intención de que sean testeables de forma automática, y con este fin se provee de herramientas y procedimientos auxiliares que permiten una interacción desatendida que puede ser invocada desde un código programático.
- *Organización del testware:* es la forma en que se va a organizar el software de automatización generado. En qué lugar va el código de la ejecución, de dónde se toman los resultados esperados y en dónde y cómo se dejan los resultados de las ejecuciones de pruebas.

Implementación de Prueba Automatizada

Esta etapa tiene un gran componente técnico. El *tester automatizador* con apoyo del resto del equipo comenzará las actividades de construcción de las pruebas

automatizadas.

Identificador	Nombre de la Actividad	¿Es requerida?
IPS	Implementación de Scripts	SI
VFS	Verificación de Scripts	SI

Cuadro 4.16: Etapa Implementación de Pruebas Automatizadas

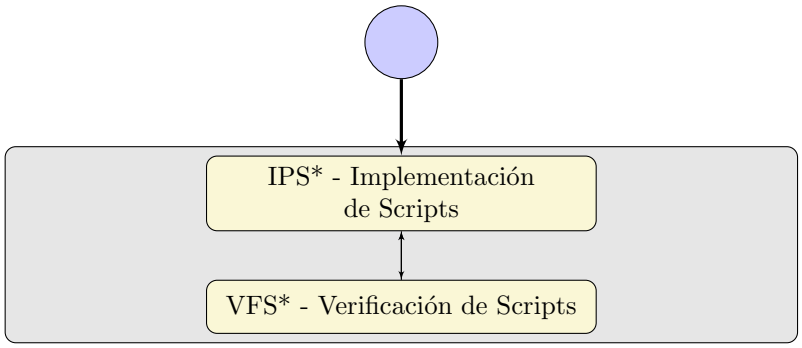

Figura 4.18: Etapa Implementación de Prueba Automatizada

Roles involucrados en la etapa

- *Tester Automatizador:* genera el código de las pruebas automatizadas en el conjunto de herramientas seleccionadas, siguiendo la especificación dada en el diseño de las pruebas automatizadas. Es positivo que pueda contar con el apoyo del *tester funcional* para definir temas relativos a las pruebas, y con la con la colaboración de integrantes del equipo de desarrollo para apoyar en la parte técnica de implementación.

Productos de trabajo involucrados en la etapa

- *Documento de Diseño de Pruebas Automatizadas:* es utilizado por el *Tester Automatizador* como guía sobre cuáles son las pruebas que debe implementar. Puede ser un documento simple, que contenga una lista de las pruebas deseables, hasta uno que especifique incluso una prioridad u orden de implementación.

Implementación de Scripts: en la actividad de implementación, se construyen los *scripts* de la prueba en el lenguaje de la herramienta de automatización seleccionada o en el lenguaje de desarrollo correspondiente. Las principales tareas para el desarrollo de *scripts*:

- **Generar Esqueleto de Script:** consta del uso de una herramienta de grabación que capture los gestos efectuados en un producto y que sea capaz de traducirlo a un lenguaje de programación de *scripts*. Las herramientas comúnmente cuentan con esta funcionalidad, conocida en la jerga como *"Rec And Play"*, o grabar y ejecutar. Otra de las formas de generar un esqueleto, es a través de la reutilización de *scripts*, o partes de *scripts* que ataquen problemas similares.

- **Depuración de Scripts:** luego degenerado el esqueleto de un *script*, es necesario ajustarlo para sacar el mejor provecho. El objetivo es efectuar el mantenimiento necesario para que el esqueleto se ajuste al nuevo contexto de uso. Los *scripts* generados por las herramientas de grabación son un volcado de la ejecución de acciones sobre el software. Por lo tanto es importante identificar los lugares donde se manifiestan las entradas y salidas del sistema, identificando las variables, datos, valores constantes; para conseguir un *script* parametrizado, y que se integre correctamente con el resto del conjunto de elementos de la prueba automatizada. Por ejemplo, el *script* resultante podría luego ser ejecutado para varios datos diferentes, accediendo una fuente de datos externa, aprovechando las ventajas de lo que se conoce comúnmente como *Data-Driven Scripts*[15].

- **Programación de Scripts:** implementación de las pruebas, partiendo de *scripts* base grabados, o desde un *script* vacío. Esta es una actividad plena de desarrollo. Se supone que los temas de testing están en su mayoría resueltos. Eventualmente esta tarea podría ser ejecutada por un desarrollador, si estuviese completamente especificada. El rol del *tester automatizador*, que se describe en la sección 4.5.2 tiene un valor agregado, ya que podrá tomar decisiones relativas al testing y hacer ajustes al enfoque, lo cual, no estaría a cargo de un desarrollador.

Verificación de Scripts: las pruebas automatizadas son en definitiva software. Por lo tanto, deben ser verificadas. El enfoque para verificar los *scripts* es variado, y aplican las técnicas de pruebas unitarias y de integración[16] como las que ejecutan los desarrolladores a sus módulos. El comportamiento esperado de

[15] *Data-Driven Scripts* se podría traducir como *"pruebas automatizadas orientadas a datos"*. [FG99h]

[16] Según [IEE91], testing unitario es el testing individual de unidades de hardware o software, o grupos de unidades relacionadas. Una unidad es un elemento separable que puede ser verificado. Suelen ser módulos o piezas de software relacionadas. El test de integración es el testing de la combinación de diferentes componentes, tanto software como hardware, para evaluar y verificar cómo interactúan. Ver en el documento referenciado *"unit testing"*, *"integration testing"* y *"unit"*.

un *script*, o conjunto de *scripts*, es que ejecute y no falle, a menos que encuentre un incidente. Como veremos más adelante, los *scripts* son muy sensibles a los cambios en la aplicación y en los datos, por lo tanto en las tareas de construcción y mantenimiento, es necesario correr varias veces los *scripts* para asegurarnos de que estos no nos traen falsos positivos, es decir, fallan por otros factores que no sean *bugs* o errores en el producto.

Ejecución de Pruebas Automatizadas

Esta es la etapa donde se ejecutan las pruebas automatizadas. La ejecución de las pruebas debe planificarse, dado que no siempre es conveniente correr todo el conjunto de pruebas automatizado. Los resultados de las pruebas tienen que ser recolectados y analizados, para ser debidamente informados. La etapa de ejecución de pruebas es un punto crítico del proceso, dado que su resultado es fuente de información para otras etapas y actividades, por ejemplo, la ejecución puede revelar nuevos incidentes, así como también la necesidad de que las pruebas mismas necesiten mantenimiento.

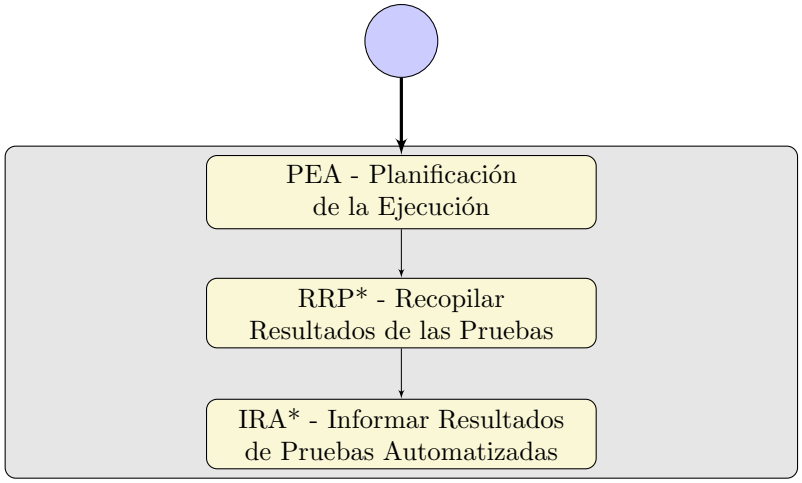

Figura 4.19: Etapa Ejecución de Pruebas Automatizadas

Roles involucrados en la etapa

- *Tester Profesional, Tester Funcional Líder:* planifican y ejecutan las pruebas automatizadas. Luego deben analizar la información. Los testers no automatizadores deben ser capaces de correr las pruebas, por eso deben trabajar en colaboración de quien las implementó. Analizan la información de los resultados de las pruebas, lo que puede modificar reporte de incidentes, consultas o ajustes a las pruebas.
- *Tester Automatizador:* asiste en la obtención de los resultados de las

pruebas para su correcto análisis. Colabora en la ejecución de las pruebas con el resto del equipo de testing.

Productos de trabajo involucrados en la etapa

- *Documento de Resultados de Automatización:* contiene las conclusiones obtenidas del análisis de los resultados de las pruebas. Puede contener los incidentes encontrados, la cantidad de pruebas ejecutadas, las funcionalidades cubiertas, el tiempo que demora la ejecución de las pruebas, comentarios acerca de la mantenibilidad, entre otros.

Identificador	Nombre de la Actividad	¿Es requerida?
PEA	Planificación de la Ejecución	SI
RRP	Recopilar Resultados de las Pruebas	SI
IRA	Informar Resultados de las Pruebas Automatizadas	SI

Cuadro 4.17: Etapa Ejecución de Pruebas Automatizadas

Planificación de la Ejecución: acarrea la decisión de cuáles pruebas se van ejecutar, y en qué momento. Por ejemplo una estrategia puede ser ejecutar las pruebas del núcleo del producto tras las liberaciones diarias, y ejecutar pruebas anexas en ciertos momentos y a demanda, involucrando el resto de las funcionalidades. Así como también priorizar y planificar las pruebas más costosas en tiempo y recursos. La actividad involucra el posible agendado de la ejecución, por ejemplo, que corran durante la noche las pruebas más costosas en tiempo y proceso.

Recopilar Resultados de las Pruebas: en esta actividad se consultan los resultados de las pruebas automatizadas y las posibles trazas almacenadas. La información puede estar en bases de datos, archivos de logs y puede ser generadas por los motores de ejecución de pruebas, o monitores instalados en diferentes partes del sistema (más común en pruebas relacionadas con mediciones de rendimiento). El caudal de datos obtenido se procesa convenientemente para su posterior manejo y estudio.

Informar Resultados de las Pruebas Automatizadas: generamos en esta actividad un Documento de Resultados de Automatización. El documento es un resumen de lo acontecido en la ejecución de la automatización de pruebas. Esta es una herramienta importante para identificar errores en el producto, o en las mismas pruebas automatizadas. La construcción y detalle de este

informe será acorde a las necesidades del producto. Por ejemplo, si las pruebas automatizadas intentan verificar que el producto no ha sufrido una regresión en su calidad, antes de una liberación, es importante detallar qué puntos fallaron o no, y qué pruebas han podido ser ejecutadas. Si las pruebas son referentes a aspectos de rendimiento, es común que este material sea enriquecido con gráficas y análisis de los datos, con el fin de acotar los posibles errores encontrados.

Análisis de Errores Detectados

Esta etapa no está subdividida en actividades. Consiste en la revisión de los resultados de las pruebas con el fin de identificar si han encontrado errores o los *scripts* requieren ajustes. La automatización no es capaz de discernir entre estas posibilidades, por lo cual requiere de la intervención humana.

Puede suceder que el volumen de casos de pruebas que fallan sea muy grande, en ese caso, conviene evaluar cuidadosamente cómo invertir el tiempo en analizarlos. Una estrategia posible puede ser tomar un núcleo representativo de los casos de prueba que fallan y tratar de identificar el motivo, para luego de reparados esos incidentes, volver a ejecutar las pruebas automatizadas. Aunque las pruebas sean automatizadas, pueden requerir muchos recursos para su ejecución repetitiva y demorar en completar su ejecución, por lo tanto, es importante revisar con cuidado cómo se invertirá el esfuerzo de análisis y re-ejecución de pruebas.

Escenarios de falla de casos de prueba

Al analizar los resultados de la ejecución pueden darse los siguientes casos:

- **El caso de prueba fue exitoso:** no falla la ejecución de la prueba automatizada. En este escenario sabemos que el software no presenta incidentes para el escenario ejercitado por el caso de prueba.
- **El caso de prueba falló:**
  - **Por causa de errores en el producto:** el producto presenta una regresión, y se registran los incidentes asociados. En una nueva versión se volverá a correr la prueba automatizada para verificar que el caso de prueba se ejecute exitosamente.
  - **Por motivo de errores en las pruebas automatizadas:** El producto tiene el comportamiento correcto y se deben ajustar las pruebas. En este punto es donde aplican las tareas de mantenimiento de las pruebas. Los motivos por los cuales puede suceder esto es porque el producto cambió su especificación y la prueba automatizada no refleja este cambio, o porque los datos utilizados por las pruebas no se condicen con el estado actual de la aplicación, o por un error en las pruebas debido a mantenimientos erróneos.

Mantenimiento de Testware de Automatización

La etapa de mantenimiento es transversal al resto de las etapas del Proceso de Testing Automatizado. No es necesaria ejecutarla rigurosamente, pero en ciertos contextos se vuelve imprescindible y permanente. Es común que el mantenimiento sea necesario en proyectos donde las pruebas automatizadas se ejecuten repetidamente. Suele suceder que cambios en el producto impactan en los test automatizados implicando que estos fallen, no por la presencia de *bugs*, sino porque no están acordes a los cambios; o que cambios en el proyecto motiven el ajuste de diferentes elementos del testware de automatización o su arquitectura.

Identificador	Nombre de la Actividad	¿Es requerida?
REC	Revisar Etapa de Automatización en Curso	NO
ATA	Ajuste de Testware de Automatización	SI
EAM	Elaborar Informe de Ajuste de Mantenimiento	NO

Cuadro 4.18: Etapa Mantenimiento de Testware de Automatización

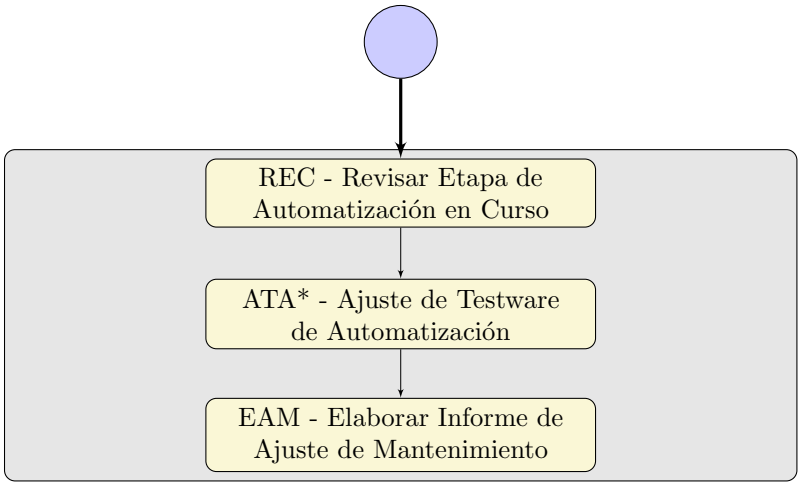

Figura 4.20: Etapa Mantenimiento de Testware de Automatización

Roles involucrados en la etapa

- *Tester Profesional, Tester Funcional Líder:* identifican las fallas en las pruebas y si su procedencia es relativa a un incidente o a un ajuste necesario en la implementación de la automatización. Le indican al *Tester Automatizador* el conjunto de pruebas que necesitan mantenimiento.

- *Tester Automatizador:* implementa los ajustes necesarios para que las pruebas que deben ejecutarse puedan correr. Completa el Documento de Mantenimiento de Automatización para registrar los cambios.

Productos de trabajo involucrados en la etapa

- *Documento de Mantenimiento de Automatización:* detalla qué pruebas sufrieron cambios y por qué motivo; por ejemplo por ser obsoletas, por contener errores ante una nueva realidad o por no ser ejecutables debido a un incidente presente.

Revisar Etapa de Automatización en Curso: consiste en identificar la etapa en curso y extraer sus particularidades para decidir cuáles son las tareas de mantenimiento correspondientes. En la etapa de diseño de la automatización, el mantenimiento es enfocado a los documentos. En etapas próximas a la implementación y la ejecución de las pruebas, se trata del ajuste de los *scripts* y archivos específicos del desarrollo. Los *scripts* de automatización de pruebas suelen ser muy sensibles aciertos cambios del producto, el entorno de los datos y la estabilidad general del sistema. El trabajo de mantenimiento es menor si se dedica esfuerzo relativo a planificar cómo serán mantenidos los *scripts* en el momento del diseño y la implementación de las pruebas, como se menciona en la sección 4.5.1. Los factores que dañan la mantenibilidad, son similares a los que amenazan a los módulos de desarrollo. Ciertos aspectos dificultan las tareas de mantenimiento, como ser por ejemplo, un sistema inestable, un entorno de ambiente cambiante, cambios importantes en requerimientos y definiciones, entre otros.

Ajuste de Testware de Automatización: es la actividad de impactar los cambios necesarios a los elementos involucrados en cada etapa. Puede corresponder a ajustar documentos de diseño, archivos de *scripts*, e incluso puede ser necesario hace una re-ingeniería de la arquitectura de la solución de automatización, aunque es la situación menos deseada.

Elaborar Informe de Ajustes de Mantenimiento: esta actividad se trata de documentar los cambios, para contar con un registro de mantenimiento. La existencia de buena información de los ajustes facilita las probables futuras tareas de mantenimiento. El registro de mantenimiento suele contener la fecha y el responsable del cambio, la etapa involucrada y los elementos afectados.

Informar Resultados

Esta etapa no tiene diferencia con la del módulo anterior en sus actividades. Difiere solamente en qué tipo de información es extraída acorde a las actividades de automatización. Véanse en la sección 4.4.1 las actividades que involucra esta etapa.

4.5.2. Roles

Como se describe en la sección 4.4.2, un rol consta de un nombre y una descripción de las funciones que desempeña la persona que lo asume.

Tester Automatizador

A los roles del nivel anterior, se incorpora la figura del *Tester Automatizador*. Hasta esta incorporación, tienen lugar las etapas de testing funcional manual, luego se requiere un conjunto de habilidades técnicas diferentes para automatizar, más vinculadas con desarrollo de software. Es importante que la persona en este rol colabore con el resto del equipo de testers en la definición de los casos de prueba, y que estos colaboren en la selección, así como también tenga buena comunicación con el equipo de desarrollo quienes le serán de mucha ayuda para sortear posibles inconvenientes técnicos a la hora de automatizar.

4.6. Relevancia y profundidad de la documentación

Con el fin de registrar lo ocurrido en las diferentes actividades, se elaboran documentos para apoyar las tareas, brindar información y facilitar el seguimiento y control de las diferentes actividades y etapas. La relevancia y la necesidad de construcción de un documento debe evaluarse según las características del proyecto. No construir ni mantener un documento, puede permitir invertir tiempo en otras actividades, pero va en demérito de las actividades de seguimiento y de eventuales revisiones posteriores, como apoyo a futuros proyectos o cambios dentro del proyecto actual. Determinados proyectos pueden no requerir estrictamente contar con un documento particular, e incluso puede configurarse como un obstáculo al desarrollo del proyecto.

Algunos documentos pueden ser presentados en versiones resumidas, o con menos formalidad, por ejemplo omitir secciones que apuntan a que el documento se presente y explique a sí mismo, como ser la introducción y propósito, yendo directamente al punto central.

Parte importante de la flexibilidad de la propuesta metodológica se basa en saber qué documentos aportan al desarrollo del proceso de testing. A la hora de elaborar un documento, se evalúa primero quién lo solicita, cuál es su aporte,

y qué esfuerzo requiere su construcción para diferentes niveles de rigurosidad. Un documento no requerido por la organización, que insume mucho tiempo en formalismos y que además rara vez será consultado; es un firme candidato a ser descartada su mantenimiento o incluso elaboración.

4.7. Resumen

Con la intención de generar un marco de trabajo adaptable a las necesidades de ASSE, se elabora una propuesta metodológica organizada en módulos, cuyo nombre es *"Framework Instanciable de Testing"*. Se definen tres módulos. Un módulo inicial se utilizará como punto de partida identificando el estado de la organización, un módulo que contiene un proceso de testing funcional manual y otro módulo que está compuesto por un proceso de testing funcional manual automatizado. Cada proceso cuenta con etapas, y las etapas con actividades. La metodología debe ser personalizada para cada contexto de uso, siendo posible seleccionar qué etapas y actividades transitar (dentro de cierto espectro) y en qué profundidad. A la configuración de etapas y actividades para un proyecto particular, se le da el nombre de *"roadmap"*.

Capítulo 5

Proyecto piloto

"Son vanas y están plagadas de errores las ciencias que no han nacido del experimento, madre de toda certidumbre."

Leonardo da Vinci

5.1. Introducción

Una de las primeras actividades del proyecto de grado, fue la identificación de las realidades de la organización en cuanto al testing y desarrollo de sistemas informáticos. Con esta información como situación inicial, se investiga y documenta el estado del arte en testing funcional. Se construye luego un marco de trabajo, suficientemente flexible, cuya intención es cubrir lo máximo posible de las diferentes realidades de los proyectos de ASSE. La etapa posterior, es poner en práctica la forma de trabajo en proyectos reales de la organización, observando el proceso y extrayendo conclusiones. En este capitulo se presenta esta última etapa, mostrando la experiencia y los resultados del trabajo de campo.

Uno de los puntos relevantes del proyecto de grado es la implantación de herramientas que permitan asistir al testing, logrando enfocar las pruebas manuales a puntos donde es necesaria más actividad cognitiva.

5.2. Descripción de la Metodología adoptada

El marco de metodológico propuesto organiza el trabajo en tres módulos principales. El primer módulo corresponde a la identificación inicial de las particularidades del proyecto y la forma de trabajo, rescatando fortalezas e identificando debilidades con respecto al testing funcional y elementos relacionados. La etapa inicial se nutre de un análisis al estilo que se presenta

en el capitulo 3, siendo este un ejercicio de informe de análisis y diagnóstico inicial para la situación de ASSE. Los módulos II y III contienen definiciones del proceso de testing funcional manual y testing funcional automatizado, respectivamente. Cada proceso cuenta con sus etapas y actividades, que aplicarán a cada proyecto particular. Decimos que el marco de trabajo es *instanciable*, ya que es posible seleccionar para un proyecto específico, qué etapas y actividades se transitan de cada proceso propuesto en la metodología y la rigurosidad con que se van a ejecutar.

5.3. Propuesta de proyecto piloto

Como parte del *proyecto de grado*, es importante aplicar la metodología en un contexto de uso real. Con tal motivo, se han investigado tres posibles sistemas de ASSE, candidatos a participar en el proyecto piloto. Estos son *Escritorio Clínico*, *OpenSIH* y *Médico de Referencia*. En el cuadro 5.1 se describen las características percibidas de cada producto en el análisis de la información obtenida.

La intención del proyecto piloto, es que sea una experiencia aislada (dentro de lo posible) de la forma de trabajo actual de ASSE, para que de esta manera, sea pueda evaluar su funcionamiento y viabilidad. Por este motivo, es preferible que el proyecto piloto no esté bajo condiciones reales de criticidad, presión o riesgo.

Para que el proyecto piloto tenga éxito, fue necesario contar con el apoyo y consentimiento de la dirección, dado que insumimos recursos materiales y sobre todo trabajo de personas que han colaborado para poder llevar adelante la experiencia exitosamente.

5.4. Proyectos Candidatos

Luego de analizar comparativamente los proyectos (véase el cuadro 5.1), el software que mejor cumple con las características necesarias para un proyecto piloto es OpenSIH, dado que es un sistema estable de mediano porte, que no cuenta con una alta tasa de liberaciones y se puede automatizar a través de la construcción de un framework de herramientas libres. De los restantes sistemas, se elige Escritorio Clínico, lo cual nos brinda la oportunidad de atacar un sistema de mediano porte, con una funcionalidad compleja, implementado en GeneXus[1], usando la herramienta GXtest[2] (ver análisis de la conveniencia del uso de GXtest en este tipo de escenarios en la sección 5.8). El tamaño de los formularios de ingresos es similar al de OpenSIH pero la aplicación cuenta con un bajo nivel de documentación, lo que pone a prueba la adaptabilidad de la propuesta metodológica.

La experiencia simula una reestructura del área de Testing y por lo tanto es necesario contar con el apoyo de la dirección, para que asigne recursos y declare al proyecto como "de interés" para lograr sortear posibles dificultades externas.

[1] GeneXus es un generador de código muy popular de la empresa uruguaya Artech, para más información www.Genexus.com

[2] GXtest es una herramienta de gestión y automatización de pruebas funcionales para aplicaciones web desarrolladas en GeneXus. Pertenece a la empresa uruguaya *Abstracta*, por más información, visitar http://www.abstracta.com.uy/es/gxtest-overview-testing-tool-GeneXus.html, accedido por última vez en mayo de 2011.

	Escritorio Clínico	OpenSIH	Médico de Referencia
Porte del Software	Mediano	Pequeño	Pequeño
Nivel de Documentación	Bajo	Alto	Alto
Estabilidad de las funcionalidades	Media	Alta	Alta
Tamaño de los Formularios	Entre 50 y 100 campos	Entre 50 y 100 campos	Entre 5 y 20 campos
Frecuencia de Cambios	Alta	Moderada	Moderada
Plataforma de Desarrollo	GeneXus 9.0 - Java sobre MySQL	Java - Ajax - JBOSS con MySQL	GeneXus 9.0 - Java sobre Oracle
Herramienta de automatización	GXtest	Selenium Webdriver	GXtest

Cuadro 5.1: Sistemas candidatos a incluirse en el Proyecto Piloto.

Algunos datos complementarios:

- **Escritorio Clínico:** tiene una alta frecuencia de cambios[3] y liberaciones a producción. Es común que se solicite el testing del producto para su liberación inmediata, en el mismo día que termina el desarrollo.

- **Comentarios sobre OpenSIH:** es un sistema estable y en desarrollo. Cuenta con una moderada frecuencia de cambios[4] y está siendo instalado en todos los CTI de ASSE.

- **Médico de Referencia:** para cada usuario del sistema de salud, se define el médico de referencia, quién puede acceder a la historia clínica. Contiene un Web Service *Médico de Referencia*.

Al inicio del proyecto piloto en ASSE, Fernando Pereira y Carlos Gutierrez tienen en curso una automatización de pruebas sobre los WebServices Personas y Autenticación LDAP. En este punto se brindó apoyo en diseño de pruebas para completar la cobertura de la prueba en curso, y además ensayar la aplicación de técnicas de diseño de pruebas para un caso práctico.

[3]Una frecuencia de cambios Alta implica que en la forma de trabajo actual es dificultoso aplicar un proceso de testing riguroso

[4]Una frecuencia de cambios Moderada, deja espacio para estimar esfuerzo y completar las etapas requeridas en un proyecto de testing

5.5. Proceso de Testing aplicado a Escritorio Clínico

En esta sección se describen particularidades del sistema Escritorio clínico y el resultado de aplicar la propuesta metodológicas.

5.5.1. Introducción a la aplicación

Escritorio Clínico se trata de una aplicación Web desarrollada en GeneXus. Se compone de varios módulos, uno de ellos es la *consulta de agenda*, donde un usuario del servicio de salud tiene agendado (mediante otro sistema, el SGA) una cita con un médico.

El *usuario del sistema* es un médico que atiende consultas en determinado *centro de salud*. Ingresa con un nombre de usuario y va a la funcionalidad *consulta de agenda*, donde podrá informarse de los *pacientes* que tiene para atender en el día. Además, puede ingresar a chequear la información de cada uno de esos pacientes.

Los pacientes se clasifican como niño, adolescente y adulto. Para cada uno de estos, al entrar a ver la información, el sistema muestra un menú con diferentes opciones.

5.5.2. Tipo de aplicación

Arquitectura y acceso a datos

Es una aplicación web desarrollada en GeneXus generando para Java, que utiliza una base de datos MySQL[5]. La base de datos de Escritorio Clínico está centralizada en ASSE.

Complejidad del dominio

El dominio del sistema es complejo y la solución desarrollada lo refleja. Involucra manejo de historias clínicas, información que es muy sensible y a la vez perteneciente a un contexto complejo, partiendo por ejemplo, desde la dificultad del vocabulario manejado.

5.5.3. Contexto de desarrollo

[5] Es un manejador de base datos *opensource* muy popular, para más información se puede consultar documentación en http://dev.mysql.com/doc/

Estado de la aplicación

Si bien el producto es estable y se puede testear, se percibe una alta tasa de errores y oportunidades de mejora. Por ejemplo, un inconveniente de usabilidad recurrente es que todas las consultas cargan datos patronímicos (datos que definen al paciente), y es necesario ingresarlos manualmente en cada consulta, es decir, no se contextualizan los datos a lo largo de un *flujo funcional*.

No es claro qué tan estable se mantendrá el sistema en cuestiones de diseño de la interfaz gráfica, dado que para el tipo de paciente "niño", los formularios se muestran de una forma y para "adultos" y "adolescentes" se muestran de otra. Es de suponer por lo tanto que sufrirán cambios; lo cuál tiene impacto directo en la automatización y puede elevar el costo de mantenimiento al punto de la re-implementación de los casos de prueba, incluso usando una herramienta eficiente como GXtest (por contar con un manejo particular de las pruebas, que favorece el mantenimiento en aplicaciones generadas con GeneXus).

Estado de los datos

Para el testing del sistema, se trabaja regularmente sobre el mismo conjunto de datos y los que son estables en el ambiente de pruebas. En producción, los datos no están sincronizados con la base de persona (datos del paciente, a través del *Web Service "persona"*), debido a que la calidad de los datos no es confiable y esta información causa muchos errores en la aplicación. Los datos son extraídos la primera vez que se da de alta una consulta para un paciente y luego se trabaja con la base de datos centralizada de Escritorio Clínico.

Disponibilidad de apoyo de desarrollo

La interacción con los desarrolladores es directa y muy buena. El equipo está instalado en la institución y se muestra sumamente colaborativo. Se ha consultado a los desarrolladores por temas técnicos y algunos detalles del dominio, logrando colaboración inmediata.

5.5.4. Proceso de testing

Estrategia adoptada

En este proyecto se ha adoptado por una estrategia híbrida de testing exploratorio y testing planificado para el testing manual, y pruebas automatizadas para apoyar las pruebas de regresión.

Se emplea testing planificado para verificar la información obtenida en el panel *Programa del Adulto*, el cual se accede desde el menú *Paciente*, luego *Consulta Espontanea del Paciente* e ingresando la cédula del paciente. Otra opción para acceder es desde el menú *Paciente*, luego *Ver Agenda del Técnico*, elegir la agenda del día y luego seleccionar el paciente; esta última opción involucra una carga previa de datos en otro sistema (SGA) para lograr visualizar el paciente en la agenda.

La forma de llegar al panel *Programa del Adulto* a través de *Consulta Espontanea del Paciente* es más recomendable para las tareas de automatización, dado que no hay interacción con otro sistema.

La documentación generada tiende a ser la mínima imprescindible suponiendo un contexto ágil y orientado a resultados, en el cual la información auditable no es la prioridad; de los documentos se genera solamente los necesarios, haciendo hincapié en el el punto central.

Herramientas definidas

La automatización será llevada a cabo con la herramienta GXtest, para evaluar su desempeño para sistemas de este tipo en el contexto de la organización. GXtest permite grabar una ejecución y generar un modelo gráfico sobre el cual se ejecutarán los casos de pruebas. También resuelve temas de planificación de ejecución de pruebas y mantenimiento (ver en la sección 5.8).

Complejidad del testing funcional

Desde el punto de vista de la verificación, el sistema es testeable (ver testeabilidad en la sección 2.6), al contar con muchos lugares de los cuales el testing se puede servir, dado que existen herramientas construidas y utilizadas por los propios desarrolladores para sus pruebas y depuración.

La funcionalidad de *Alta de Agenda* del paciente se hace a través del sistema SGA, lo que implica la interacción con un sistema externo para crear los escenarios de testing relacionados.

En cuanto a las pruebas, se diseñan casos de prueba para un formulario complejo (de los que cambian su interfaz según el tipo del paciente) y se automatiza un subconjunto de los casos.

Complejidad de automatización

El formulario de ingreso está subdivido en bloques que agrupan campos con una lógica de relación entre los datos. Por ejemplo, *Datos Patronímicos* y *Antecedentes Socioeconómicos del Usuario*. Esto facilita la división lógica del diseño de las pruebas y su posterior automatización, siendo posible atacar diseño, automatización y ejecución de pruebas de manera incremental; comenzando por los datos patronímicos y luego el resto.

Esta estrategia permite que en cada momento se pueda enfocar en un grupo de campos en particular y favorece la modularización tanto del diseño de las pruebas manuales cómo de la implementación de los casos automatizados.

Para la automatización, se dividen los casos según la agrupación lógica de los campos, logrando casos genéricos y modulares, que permiten ingresar información a los bloques de forma independiente, permitiendo así ejecutar pruebas para los bloques en conjunto o por separado, siempre teniendo en cuenta las pre-condiciones que pueden tener cada caso, como ser el ingreso de los datos obligatorios.

Un ejemplo de caso de prueba de un formulario completo:

- Ejecuta un caso de prueba de ingreso de Datos Obligatorios
- Ejecuta un caso de prueba del Bloque Patronímico
- Ejecuta un caso de prueba del Bloque Socioeconómico

Recursos de Testing

El proyecto es llevado adelante trabajando con la colaboración de Carlos Gutierrez, integrante del área de testing de ASSE, quien contaba con conocimiento del sistema, y además mostró interés en el uso de la herramienta GXtest.

Al finalizar las etapas de testing manual, aún no se contaba con la licencia de la herramienta GXtest, por temas administrativos ajenos al *proyecto de grado*, lo que dificultó el desarrollo del proceso. Finalmente se obtuvo la licencia para poder utilizar la herramienta y continuar la experiencia.

Se genera para el proyecto, la documentación necesaria para apoyar el proceso en el supuesto del contexto ágil ya mencionado.

5.5.5. Conclusiones

La experiencia de trabajo en conjunto con Carlos Gutierrez y el grupo de desarrollo aplicando la metodología propuesta es muy positiva. El proceso de testing pudo ser aplicado, y se logró automatizar pruebas mediante el uso de GXtest.

Las tareas de análisis, diseño y planificación insumen la mayor parte del tiempo. El desarrollo de las pruebas automatizadas se hace rápidamente con la herramienta GXtest.

El equipo de desarrollo se integra a la herramienta de gestión propuesta para el proyecto piloto, *Redmine*, que se configura para que modele las necesidades del proyecto, favoreciendo la interacción entre ambos equipos.

5.6. Proceso de Testing aplicado a OpenSIH

El marco de trabajo diseñado, también se aplica en otro sistema de ASSE, OpenSIH. Esta sección describe algunas características de OpenSIH y las conclusiones extraídas acerca del uso de la propuesta metodológica.

5.6.1. Introducción a la aplicación

El Sistema de Ingreso Hospitalario, OpenSIH, es una herramienta web para el ingreso de *intervenciones quirúrgicas de las unidades ejecutoras por parte de los médicos cirujanos, anestesistas y demás integrantes del equipo medico.* Es de destacar que entre otras funcionalidades, se encarga del cálculo del puntaje VAC (variable anestésico quirúrgico), lo que está directamente relacionado con los honorarios del personal actuante, y por lo tanto es muy sensible a eventuales errores.

La funcionalidad principal de OpenSIH es la de ingreso de la descripción de la intervención quirúrgica, la cual consta de un formulario extenso, implicando que cada caso de prueba ejecutado manualmente insume mucho tiempo. Por lo tanto, se invertirá en automatizar casos de prueba de ingresos, dado que representan una buena relación de costo y beneficio, frente a la ejecución manual de *pruebas de regresión.*

Al explorar el sistema OpenSIH, se percibe como una herramienta estable y en condiciones de recibir pruebas de sistema [6].

[6] Se ensayan pruebas exploratorias en la versión 2.6, previamente instalada en un ambiente de testing disponible en ASSE

5.6.2. Tipo de aplicación

Arquitectura y acceso a datos

Es una aplicación web desarrollada en Java que utiliza AJAX, corriendo en un servidor JBoss[7] a la que se accede mediante un navegador, preferentemente Mozilla Firefox[8]. Los datos se encuentran en una base de datos MySQL administrada por ASSE.

Complejidad del dominio

Si bien el dominio es complejo y la tarea de análisis de los desarrolladores fue laboriosa, la solución ataca una problemática compleja de manera muy elegante (la interfaz es intuitiva, y controla el ingreso de datos, se asiste con el ingreso de códigos de diagnósticos y procedimientos, entre otros). Resulta por lo tanto fácil conocer el dominio del sistema a través del uso de la herramienta y el conocimiento acumulado en los testers de ASSE.

5.6.3. Contexto de desarrollo

Estado de la aplicación

El sistema se somete a pruebas exploratorias para recorrer las funcionalidades y aprender del dominio. En la exploración se percibe que la aplicación es estable y se puede testear sin manifestar interrupciones del servicio.

Estado de los datos

El producto se comunica correctamente con las fuentes de datos, y el conjunto de datos con el que se trabaja es controlado y con referencias de uso por parte de los testers de ASSE. Además, el sistema tiene validaciones al ingreso de los datos, y asiste en la selección de información al ingreso.

Disponibilidad de apoyo de desarrollo

Los integrantes del equipo de desarrollo de OpenSIH muestran actitud colaborativa y se encuentran físicamente en la institución (y pertenecen a ella). En esta etapa no fue necesario contar con su asistencia, dado que los temas técnicos programáticos se pudieron resolver desde testing y las dudas del dominio fueron en su mayoría resueltas o manejadas por Fernando Pereira.

[7] JBoss es un servidor de aplicaciones basado en Java, para más información se puede consultar http://www.jboss.org, accedido por última vez en setiembre de 2011

[8] Mozilla Firefox es un navegador web opensource, para más información visitar http://www.mozilla.org, accedido por última vez en setiembre de 2011

5.6.4. Proceso de testing

Estrategia adoptada

Para el testing de OpenSIH se emplea una estrategia mayoritariamente de testing planificado. Dado que se cuenta con una versión del producto instalada, también se aplica testing exploratorio para conocer las funcionalidades básicas del sistema y su interacción con el usuario. Se generan los documentos de las etapas abordadas a modo de ejercicio, para dejar constancia de la aplicación del proceso en la modalidad de *proceso auditable*. El objetivo principal es lograr automatización a modo de pruebas de regresión de la parte más costosa para el testing manual, que es el ingreso de descripción operatoria.

Herramientas definidas

Dada la naturaleza del proyecto se opta por herramientas en plataforma libre. El marco de trabajo es construido sobre herramientas gratuitas. Las herramientas definidas son Selenium Webdriver [9] como motor de interacción con la aplicación web, Java como lenguaje de programación y TestNG [10] como motor de ejecución de pruebas automatizadas.

Complejidad del testing funcional

La herramienta cuenta básicamente con un gran formulario, intuitivo y de fácil interacción. La complejidad fundamental radica en la gran extensión de campos a ingresar para cada alta de descripción operatoria. Consecuentemente cada caso de prueba que involucre un ciclo funcional entero de ingreso de descripción operatoria insume mucho tiempo, haciendo que el testing manual en sí requiera una gran inversión en tiempo, consuma muchos recursos del área de testing y se torne tedioso.

Complejidad de automatización

La herramienta está desarrollada en plataforma web, por lo tanto es accesible por la herramienta Selenium WebDriver. La presencia de AJAX[11] dificulta levemente la automatización, pero la popularidad de las herramientas utilizadas para el testing y la forma de implementación de OpenSIH, facilita encontrar en la web muchos ejemplos de soluciones a problemas de similares características. La complejidad mayor se basa en el diseño de un marco de trabajo para la automatización, dado que se trabaja con un conjunto de herramientas libres que hay

[9] Selenium WebDriver es una herramienta para interactuar programáticamente con diferentes navegadores. En la web del proyecto http://seleniumhq.org/projects/webdriver/ se puede obtener más información.

[10] TestNG es un *framework* http://testng.org/doc/index.html de planificación y ejecución de pruebas unitarias que puede ser utilizado como motor de ejecución de pruebas automatizadas.

[11] La sigla AJAX significa JavaScript Asincrónico y XML (Asynchronous JavaScript And XML) y es una conjunto de técnicas de programación para crear aplicaciones web interactivas.

que orquestar para que funcionen como una sola solución de automatización de pruebas.

Con este motivo se definió:

1. **Obtención de parámetros de configuración de los tests en archivos de propiedades:** aquí se almacenan datos relativos al acceso a la aplicación en el entorno actual, los cuales se comparten por muchos tests y tienden a cambiar menos frecuentemente.

2. **Aplicación de patrones de automatización para favorecer la mantenibilidad de los tests:** se basa en el uso del patrón Page Object, que a grandes rasgos recomienda el manejo de sectores de una web lógicamente relacionados por medio de una clase cuyas propiedades y métodos modelan la interacción con los diferentes conceptos del sitio web, encapsulando así toda la lógica de interacción con la web y favoreciendo las tareas de mantenimiento de los tests automatizados.

3. **La forma de acceder a los datos almacenados en disco:** se define una forma de acceder a archivos que se puedan acceder con las herramientas libres disponibles en la organización. Se incluye para este fin acceso a archivos CSV[12] para consumir datos en forma tabular.

4. **Uso de proveedores de datos (data providers) para las pruebas:** se implementa además que los datos accedidos desde archivos (en primera instancia CSV) puedan cargarse en memoria y se modele su uso mediante Data Types, y que estos datos sean suministrados a las pruebas automatizadas para implementar pruebas orientadas a datos, que significa poder utilizar una misma implementación con un conjunto de datos, de manera que una prueba automatizada pueda representar potencialmente una gran cantidad de casos de prueba.

5. **Interacción entre herramienta de programación, ejecución web y motor de testing:** se define la forma en que los tests serán jerarquizados, utilizando la herramienta TestNG como motor de ejecución de pruebas automatizadas integrado con Java[13] como lenguaje de programación, e interactuando con la librería para Java de Selenium Webdriver, quien se encarga de comunicarse con el navegador desde la interfaz de usuario.

[12] *CSV* significa *valores separados por coma* (de *comma separated values*) es un formato de archivo simple para representar datos tabulados en texto plano, donde las filas se separan por saltos de línea y las columnas por coma o punto y coma.

[13] Java es un lenguaje de programación orientado a objetos de libre distribución cuyo sitio oficial en español es http://java.com/es

Recursos de Testing

Se trabajó en conjunto con un integrante del equipo de estudiantes del proyecto de grado y un integrante de ASSE. Se pudo avanzar rápidamente en la definición de las pruebas y planificación dado que Fernando Pereira cuenta con amplio conocimiento sobre el sistema a verificar.

Se generó documentación formal de todo el proceso aplicado.

Luego de que estuvo el conjunto de herramientas orquestado, fue posible avanzar velozmente con la implementación de la automatización, sorteando rápidamente los obstáculos.

5.6.5. Conclusiones

El trabajo conjunto aplicando los procesos de testing dio resultados muy positivos. Si bien se generó gran documentación, no implicó una sobrecarga extra de trabajo, en parte porque el integrante de ASSE tenía mucho conocimiento de la aplicación y en parte porque se pudo bajar a tierra rápidamente siguiendo las actividades propuestas en el proceso.

El proceso de automatización también fue seguido exitosamente, siendo las actividades de *construcción del framework de automatización* y *programación de las pruebas* las que insumieron la mayor parte del tiempo. Se trabajó de forma colaborativa, y se acumuló mucho conocimiento acerca de las herramientas y la forma de trabajo, luego utilizado para ajustar los procesos propuestos. Durante la automatización se impactaron nuevas versiones y se pudo actualizar rápidamente el conjunto de pruebas dado que la solución de automatización fue diseñada para que sea mantenible.

La gestión e interacción con los equipos de desarrollo se lleva a cabo en la herramienta propuesta Redmine[14], la cual modela el sistema de gestión de incidentes, de manera aislada a las existentes en la organización, con el fin de no alterar el foco en la evaluación de la forma de trabajo. Los desarrolladores se integran paulatinamente al sistema, el cual es configurado para modelar las necesidades de todos los equipos y procesos involucrados.

[14]Redmine es una herramienta web *opensource* para gestión y manejo de proyectos se puede consultar el sitio http://www.redmine.org/ (accedido en última instancia en agosto de 2011) para obtener más información.

5.7. Cuestionario para los Integrantes de ASSE

Luego de la aplicación práctica en los proyectos piloto, se genera un breve cuestionario a los integrantes del área de testing de ASSE, con el fin de conocer sus impresiones acerca de la metodología y el desarrollo de los proyectos piloto en general. Esas recomendaciones han sido tomadas en cuenta para mejorar aspectos del presente informe.

Las preguntas planteadas son las siguientes:

1. ¿La información de cada etapa/actividades del FIT permite llevar a cabo su aplicación?

2. ¿Los casos de pruebas automatizados cubrieron el alcance estipulado?

3. ¿El framework de la programación de pruebas automatizadas de aplicaciones web no-GeneXus es útil?

4. ¿La documentación generada en los proyectos piloto es de utilidad según el contexto?

5. ¿Se lograron los objetivos planteados para los proyectos piloto?

6. ¿La herramienta de colaboración para la gestión de incidente es útil?

7. Sugerencias/Comentarios generales de la metodología.

5.7.1. Respuestas de Carlos Gutiérrez

Carlos Gutiérrez participó en la experiencia piloto sobre la aplicación Escritorio Clínico, aplicando la metodología en conjunto con los estudiantes, utilizando GXtest como herramienta de automatización de pruebas. A continuación se presentas sus repuestas al cuestionario planteado.

1. ¿La información de cada etapa/actividades del FIT permite llevar a cabo su aplicación?

   - Sí, la permite llevar a cabo sin problemas

2. ¿Los casos de pruebas automatizados cubrieron el alcance estipulado?

   - En el caso de la aplicación de Escritorio Clínico hubiese sido deseable dedicar mas tiempo a la automatización con GXtest, pero dado que se tuvieron problemas para la validación de las licencias y por otra parte los estudiantes no podían utilizarlo fuera de ASSE, no fue posible. De todas formas se aplicaron los criterios de automatización estipulados en lo que se hizo, lo cual va a permitir automatizar con la herramienta sin problema en un futuro.

3. ¿El framework de la programación de pruebas automatizadas de aplicaciones web no-Genexus es útil?

   - Si bien no era mi tema, en los momentos que tenía problemas con la licencia de GXtest intenté utlizarlo para la aplicación de Escritorio Clínico y lo poco que pude probar me pareció muy útil. Estaba muy bien modularizado, muy claro y fácil de utilizar, sin dudas que en la parte de automatización los logros por el lado de este framework fueron mucho más útiles que los obtenidos con GXtest. Vale decir que GXtest ya lo conocíamos, en cambio de Selenium teníamos solamente alguna experiencia mas que nada con el uso del IDE, y no teníamos ninguna experiencia con Webdriver.

4. ¿La documentación generada en los proyectos piloto es de utilidad según el contexto?

   - La documentación generada me parece lo mas útil que ha dejado el proyecto.

5. ¿Se lograron los objetivos planteados para los proyectos piloto?

   - Sí, se lograron los objetivos.

6. ¿La herramienta de colaboración para la gestión de incidente es útil?

   - Si, me pareció muy útil y muy apropiada para el Área de Testing para una mejor documentación de su trabajo y como forma de que el trabajo del área sea visible para los encargados y los involucrados.

7. Sugerencias/Comentarios generales de la metodología.

   - Me hubiese gustado una mayor preparación de los estudiantes en GXtest para desarrollar el proyecto, se necesitaba un conocimiento similar al que si tienen en el framework para aplicaciones no GeneXus, pienso que es el gran debe del proyecto piloto.

   - Personalmente fue un gusto participar del proyecto de grado, en cuanto a metodología de testing y del framework para aplicaciones no GeneXus aprendí bastante y me fue de suma utilidad.

5.7.2. Respuestas de Fernando Pereira

Fernando Pereira fue parte de la experiencia piloto sobre la aplicación OpenSIH. Utilizando la metodología propuesta, se construye además una plataforma basada en herramientas libres para la automatización de las pruebas. Las siguientes, son las respuestas obtenidas para el cuestionario planteado.

1. ¿La información de cada etapa/actividades del FIT permite llevar a cabo su aplicación?

   - Sí.

2. ¿Los casos de pruebas automatizados cubrieron el alcance estipulado?

   - Sí, totalmente en lo que respecta Java + Webdriver + Testng para no GeneXus. En cuanto a GXtest me gustaría se tomara la opinión de Carlos.

3. ¿El framework de la programación de pruebas automatizadas de aplicaciones web no-Genexus es útil?

   - Sí, últimamente también por nuestra cuenta probamos grabar con Selenium IDE y ejecutarlas en Java haciendo uso de el framework. La grabación de este modo permitiría acelerar una aproximación a la automatización por medio de usuarios no tan avanzados.

4. ¿La documentación generada en los proyectos piloto es de utilidad según el contexto?

   - Sí, totalmente en lo que respecata Java+ Webdriver+ Testng para no GeneXus. En cuanto a GXtest me gustaría se tomara la opinión de Carlos.

5. ¿Se lograron los objetivos planteados para los proyectos piloto?

   - En lo que respecta a OpenSIH sí, por Escritorio Clínico la respuesta corresponde a Carlos.

6. ¿La herramienta de colaboración para la gestión de incidente es útil?

   - Sí, totalmente.

7. Sugerencias/Comentarios generales de la metodología.

   - Dado que la metodología plantea casi la totalidad de las disciplinas/tareas de Testing y al aplicarla se elegiría un roadmap (con algunas obligatorias) pienso se debería disponer de una guía/método sobre cómo determinar las etapas que se saltean, ya sea por falta de personal, documentación y tiempos.

5.8. Herramientas de Automatización de Pruebas

Las herramientas estudiadas como motor de automatización son: Selenium Webdriver, y GXtest. Para los sistemas desarrollados en GeneXus (Escritorio Clínico y Médico de Referencia) GXtest es ventajoso[15] y para el resto, que corre en plataforma web (como OpenSIH) se puede utilizar Selenium Webdriver.

La decisión de la herramienta a utilizar está directamente relacionada con el Software elegido y desde el punto de vista de testing funcional de caja negra.

[15] Es válido destacar que aunque se use GeneXus, siempre que se genere para plataforma Web, se podrán utilizar herramientas como Selenium Webdriver, la ventaja aquí viene de las prestaciones de GXtest como herramienta completa en sus funcionalidades y su natural integración con GeneXus.

GXtest

Las razón de por qué GXtest es muy ventajoso como herramienta de automatización para aplicaciones desarrolladas en GeneXus, pasa mayoritariamente por la gran mantenibilidad de los casos de prueba que provee. GxTest actualiza los casos de prueba luego de un re-ordenamiento de campos de GeneXus, no así en Selenium, teniendo que cambiar todos los casos implementados, o teniendo que construir pruebas enfocadas a la mantenibilidad, aumentando así el costo de mantenimiento, automatización y ejecución de las pruebas.Lo segundo a tener en cuenta es la facilidad de automatizar y reutilizar los casos de prueba ya automatizados.

GXtest brinda una interfaz gráfica que permite a un usuario no experto, automatizar casos de pruebas de manera simple, incluso se puede delegar la construcción de pruebas para ciclos funcionales básicos a Usuarios Finales expertos en el dominio, con el debido entrenamiento, el cuál no es costoso (dado que la curva de aprendizaje de la herramienta es "media"[16].

Selenium Webdriver

Cuando la plataforma es web (en nuestro contexto, tampoco desarrollado en GeneXus, como es el caso de OpenSIH), una opción libre y muy popular, con antecedentes de uso en la organización es Selenium, en este caso Selenium Webdriver. Al momento de estudio, GXTest no aplica para automatización de otras aplicaciones que no sean generadas por GeneXus. Selenium Webdriver permite una gran variedad de escenarios de automatización, es OpenSource, y soporta java. La gran popularidad mencionada, trae consigo la existencia de muchos recursos en foros, comunidades de usuarios, e infinidad de sitios web. En el contexto del proyecto, la herramienta sera utilizada como motor de interacción con el navegador, las tareas de planificación y diseño de tests mantenibles se implementan por fuera de Selenium.

En el contexto de uso (y sin contar la dificultad de armar el framework de pruebas automatizadas), la complejidad de la herramienta se deriva al manejo del lenguaje de programación seleccionado (en este caso Java) y al desafío de identificar los diferentes elementos interactivos en el formulario (como ser los que usan AJAX y tecnologías interactivas de interfaz de usuario).

[16] Aquí el termino "media", significa que Usuarios Funcionales, que en general no tienen conocimiento en programación, pueden aprender a utilizar la herramienta para grabar y ejecutar sus casos de prueba, sencillos en implementación, pero posiblemente muy provechosos por el lado del ciclo funcional representado.

Comparación de herramientas

El siguiente cuadro, 5.2, muestra una comparación de GXtest contra el framework implementado con un conjunto de herramientas libres.

	GXtest	Java + TestNG + Selenium
Curva de aprendizaje	Media	Alta
Costo económico	U$S 10.000 a enero de 2012	Libre
Soporte	Disponible, brindado por la empresa	Material disponible en foros, requiere investigación
Lenguaje de scripting	Se puede utilizar JavaScript o Procedimientos GeneXus	Requiere conocimientos de Programación en Java
Plataformas soportadas	Web. Limitado a desarrollo en GeneXus	Web
Integración con Herramientas de Desarrollo	Compatible con GeneXus	Entornos para Java
Grabación de Esqueleto	Sí, muy amigable e intuitiva.	Posible mediante el uso de Selenium IDE.
Gestión de casos de prueba	Integrado en la herramienta	No integrado.
Trazabilidad con requerimientos	No integrado.	No Integrado.
Trazabilidad con incidentes	No Integrado.	No integrado.
Mantenibilidad de los casos de prueba	Integrado en la herramienta. Es sencillo efectuar tareas de mantenimiento versión a versión.	Es posible diseñar una solución mantenible. Existen patrones de diseño para este fin.
Posibilidad de pruebas orientadas a datos	Integrado	Sí, mediante uso de dataproviders de TestNG.
Diseño de casos	Herramienta de modelado incluida.	No integrado.
Ejecución en diferentes navegadores	Internet Explorer, Firefox	Internet Explorer, Firefox, Chrome.
Ejecución distribuida	Integrado	No integrado.
Planificación de la ejecución	Integrado	No integrado.

Cuadro 5.2: Comparación de herramientas de automatización de pruebas

En el cuadro 5.2, todas las funcionalidades no integradas en las herramientas siempre pueden ser implementadas manualmente, modeladas a través de otra herramienta, o en el caso del framework de automatización, este puede ser extendido.

5.9. Conclusiones

En ambos proyectos el trabajo fue satisfactorio. Se pudo aplicar exitosamente la metodología, generando las etapas y actividades propuestas, para cada proceso de testing (manual y automatizado). Se genera documentación en herramientas gratuitas existentes en la organización que son testigo de etapas y actividades y que sirven de ejemplo para otros proyectos de testing.

Los procesos fueron seguidos en dos proyectos, conformando un equipo de testing de cinco integrantes, los dos testers de ASSE y los tres integrantes del proyecto de grado, además de contar con la colaboración de Ariel Sabiguero y Marcelo Lemos para la parte de infraestructura, instalación y configuración de herramientas.

Accesoriamente se implanta el uso de la herramienta Redmine, cuya flexibilidad permite entre otras cosas modelar el flujo de trabajo y ciclo de vida de los incidentes, permitiendo la paulatina integración de los desarrolladores a la forma de trabajo y generando un entorno aislado de la forma de trabajo habitual, para evitar interferencias.

Se logra excelente interacción y colaboración con los desarrolladores, equipos internos y externos a ASSE.

La dirección apoyó el desarrollo del proyecto, y los integrantes del proyecto de grado pudimos entrar a trabajar en la organización a la par del resto de los equipos como meros integrantes del área de testing.

Capítulo 6

Conclusiones

Este capítulo resume comentarios acerca del diagnóstico inicial, muestra una síntesis de las conclusiones sobre el proyecto piloto y presenta alternativas de trabajo a futuro. El capítulo cierra con conclusiones globales acerca de todo el trabajo del proyecto de grado.

6.1. Conclusiones del diagnóstico actual

El análisis del diagnóstico muestra que la realidad de ASSE frente al desarrollo de software y testing, carece de proceso o etapas claramente definidas. Por el lado de desarrollo, esto se da porque cuenta con una estructura compleja de proveedores y desarrollo interno la cual no responde a un proceso único gestionado por ASSE, y ademas porque los contextos son cambiantes y complejos. El área de testing tampoco tiene procesos claramente definidos, pero además tiene el agravante de estar a cargo de un personal muy reducido (al momento, un integrante), que no puede abarcar todo el volumen de trabajo de testing necesario para todos los productos. Para lidiar con estos temas, se recomienda contar con consultorías en diferentes áreas, como ser gestión de proyectos, ingeniería de software y testing de software en particular.

6.2. Conclusiones generales del proyecto piloto

El trabajo en el proyecto piloto fue satisfactorio, permitiendo ver la metodología propuesta en funcionamiento para dos productos de ASSE. Para ambos sistemas se siguen las etapas y actividades relevantes para cada contexto y se genera la información pertinente. Se logra verificar de forma manual, y generar pruebas automatizadas en dos herramientas de diferentes características, apuntando a la mantenibilidad de las pruebas.

En cada proyecto se trabaja en conjunto con los integrantes del área de testing de ASSE y en relación con los diferentes desarrolladores, todos quienes colaboran excelentemente con la experiencia. Se contó con el apoyo de la dirección en todo momento y con otros integrantes de ASSE, apoyando con la infraestructura y configuración de herramientas de apoyo (Ariel Sabiguero y Marcelo Lemos).

6.3. Trabajo a futuro

Como continuación de este trabajo, se podía extender el estado del arte a otros tipos de testing, generando nuevos módulos con nuevos procesos. Temas a ser abordados podrían ser Testing de Seguridad, Testing de Rendimiento y Performance, Testing de Usabilidad, Gestión del Testing y Mejora del proceso en sí. Siempre con la premisa de tratar de lograr propuestas que se puedan adaptar al contexto de la organización y a muchos otros.

Por el lado de la aplicación práctica, seria recomendable extender las sugerencias al área de Ingeniería de Software, con el fin de atacar temas más generales que además influyen directamente con el área de estudio. En el capítulo 3, se recomienda entre otras cosas, contar con consultorías en el área de Ingeniería de Software, para tratar temas de gestión de proyectos en la modalidad adoptada por ASSE (desarrollo interno y además muchos proyectos tercerizados), para trabajar aspectos de Ingeniería de Requerimientos, entre otras sugerencias.

6.4. Conclusiones generales del trabajo

Si bien el testing de software es una disciplina relativamente reciente con respecto al desarrollo de software, el material existente es abundante y en ocasiones reciente. El estado del arte tiene una extensión quizás excesiva para lo que se espera de una investigación para el proyecto de grado, sobre todo porque se invirtió tiempo en revisar y filtrar mucho material. Por temas de extensión y prioridades no se pudo abarcar tanto en temas de actualidad como se pretendía (lo que implicaría dejar por fuera los temas más básicos e importantes).

El diseño de la metodología propuesta insumió gran parte del tiempo del proyecto antes de ser aplicada. Queda la incertidumbre de si hubiese sido posible diseñarla de manera incremental, y así poder ajustarla simultáneamente con su aplicación.

Trabajar con los integrantes de ASSE y en relación con otros actores fue muy satisfactorio y conlleva un aprendizaje conjunto. De la buena relación generada, tenemos la convicción de que también fue provechoso y satisfactorio para ellos, como se expresa en los comentarios de la sección 5.7 del capítulo 5. Como complemento se podría haber ahondado más en temas de capacitación técnica en testing, más allá de los casos puntuales que derivan de la colaboración en el

proyecto piloto.

Como conclusión general, los estudiantes integrantes del proyecto de grado, estamos conformes con el trabajo logrado y sobre todo con la haber podido transitar esta experiencia junto a todos los involucrados.

Apéndice A

Sistema de Seguimiento de Incidentes

De la ejecución de las pruebas, surge mucha de la información a partir de la cual los responsables o gestores de proyecto toman la decisión sobre el futuro de los desarrollos. Los incidentes detectados, debidamente registrados, son una fuente fundamental de información. De su análisis se pueden extraer muchas conclusiones interesantes acerca del estado de las funcionalidades, la confiabilidad de nuestros proveedores, y la evolución de los productos, entre otras.

Es importante contar con una herramienta que oficie de sistema de gestión de incidentes, en la cual se pueda registrar, mantener y extraer la información necesaria.

Dentro de las características deseables de un sistema de gestión de incidentes, destacamos:

Accesibilidad global: el sistema tiene que poder ser consultado desde diferentes puntos de la organización. No puede estar concentrado en un sólo recurso físico al cual sea difícil de acceder simultáneamente. Un ejemplo de un sistema poco accesible, seria una planilla electrónica donde se registran los incidentes, la cual es compartida y mantenida por los testers. Soluciones más adecuadas son las que involucran arquitecturas cliente-servidor.

Centralización de información: implica que los datos no están distribuidos en diferentes fuentes. Centralizar la información del registro de incidentes evita el re-trabajo de actualizar el registro desde los posibles orígenes que proveen información.

Modelado del flujo de trabajo y manejo de perfiles: hay sistemas que permiten o sugieren, cierta forma de trabajo para registrar y seguir los cambios,

según el perfil del usuario del sistema. Adoptar el flujo de trabajo propuesto por la herramienta, o personalizarla, puede colaborar a la organización del trabajo y la comunicación.

Modelado del ciclo de vida del incidente: junto con el flujo de trabajo, el incidente pasa por diferentes estados. Es deseable que el sistema cuente con al menos los estados utilizados en la organización para el ciclo de vida de los incidentes. Si la herramienta no provee un conjunto de estados, al menos que sea posible definirlos, junto con su ciclo de vida. Cuando el ciclo de vida además, modela qué rol puede acceder a ciertos estados y cambiarlo, estamos ante el modelado del flujo de trabajo mencionado anteriormente.

Disponibilidad y Confiabilidad: el sistema tiene que estar a disposición al menos todo el tiempo que esté funcionando la organización. Además, la persistencia y manejo de la información debe ser confiable, dado que los datos registrados son muy valiosos para las actividades de testing y desarrollo.

Control de acceso basado en roles: es posible configurar la herramienta de tal forma que ciertos perfiles puedan acceder a la información de diferente forma. Se puede restringir quienes pueden consultar, modificar y eliminar la información registrada.

Posibilidad de extensión y adaptación: finalmente, además de que las características deseables de personalización y adaptación de la herramienta, muchos sistemas de seguimiento de incidentes proveen mecanismos para extender sus funcionalidades según las necesidades de cada proyecto y organización. Por ejemplo, supongamos que un proyecto necesita que todos los reportes de incidentes cuenten con una captura de pantalla que muestre el momento en el que se manifiesta; entonces, una herramienta extensible provee un mecanismo de incluir ese detalle en la descripción del incidente.

Además de los puntos mencionados, también se deben tomar en cuenta las restricciones de la organización a la hora de incorporar una herramienta de seguimiento de incidentes; similares a los mencionados en la sección 2.5.2 sobre la elección de una herramienta de automatización. Existen muchas soluciones comerciales y gratuitas para este fin. Un buen ejemplo es la herramienta Mantis[1], ya utilizada por la organización, la cual es una muy buena alternativa libre para este tipo de sistemas.

[1]Manti es una herramienta de seguimiento de incidentes *opensource*, ver el sitio http://www.mantisbt.org/ (accedido por última vez en marzo de 2011)

Apéndice B

Ejemplos de Instrumentación de la Metodología

El propósito de este apartado es ilustrar ejemplos de cómo personalizar la propuesta metodológica para diferentes realidades supuestas. Para definir y clasificar los diferentes contextos, se establece un conjunto de criterios. Cada criterio admite un conjunto de valores posibles.

Los diferentes contextos de estudio se clasificarán a través de los criterios definidos y sus valores. Esta clasificación es un supuesto arbitrario, y a modo de ejemplo, de un escenario real para una organización.

Se representará un posible escenario de instanciación de FIT para cada contexto, especificando la estrategia a adoptar y el resumen del supuesto *roadmap* para esa realidad. Además, para los escenarios particulares de "Testing Ágil" y "Testing Independiente con Interrelación con Equipos (tester como articulador)", se complementa la instanciación con una representación gráfica de etapas y actividades involucradas.

Los criterios y sus valores son totalmente arbitrarios, así como la valoración de cada contexto, los cuales no responden a ninguna fuente oficial, sino que deriva de la percepción y la experiencia de los autores. Especialistas en la materia podrían proponer muchas otras alternativas.

B.1. Criterios para clasificar los contextos

B.1.1. Frecuencia de liberaciones

Este criterio se refiere al ritmo de las liberaciones de productos o cambios en el mismo.

- **Alta:** en el entorno de liberaciones diarias.
- **Media:** corresponde a liberaciones en intervalos de algunas semanas, por ejemplo, de una a cuatro semanas.
- **Baja:** son proyectos que tienen entregas más a largo plazo, como ser de uno a seis meses.

B.1.2. Proceso seguido por desarrollo

En este punto se presta atención a la naturaleza del proceso de desarrollo seguido en la organización.

- **Varios:** la organización cuenta con un equipo de desarrollo heterogéneo, o con varios equipos de desarrollo, cada uno siguiendo su propia metodología.
- **Ágil:** se sigue una metodología ágil de desarrollo con iteraciones que duran de tres a seis semanas generalmente.
- **Cascada:** el proceso abordado cuenta con etapas definidas, que se suceden unas a otras.

B.1.3. Necesidad de evidencia de pruebas

La rigidez de los demás procesos de la organización, pueden marcar la exigencia de que todos los procesos sean auditables. Esto puede afectar el proceso de testing, sobre todo en lo que respecta a documentación que deje constancia de la ejecución de las pruebas y sus resultados.

- **Alta:** todos los procesos y sus resultados deben estar documentados.
- **Media:** es recomendable dejar constancia de la ejecución de las pruebas y sus resultados. Puede servir para resolver diferencias.
- **Ninguna:** no es necesario documentar ninguna parte del proceso. La decisión de documentar las pruebas pasa por la conveniencia detectada por el tester y en todo lo que aporte a su labor, dado que no hay una exigencia por parte de la organización al respecto.

B.1.4. Complejidad del SUT

Apunta a la complejidad percibida del producto de software a testear (o *SUT*, correspondiente a la sigla en inglés *"Software Under Test"*). Un software se puede considerar complejo porque su funcionalidad responde a un comportamiento intrincado, difícil de entender y aprender; o porque su funcionamiento es oscuro al usuario y es difícil generar un modelo mental de su operativa; o involucra cálculos muy complejos o maneja grandes volúmenes de datos, o tiene una arquitectura muy compleja, entre otros aspectos.

- **Alta:** sistema muy complejo. Es necesaria la colaboración de muchos actores para lograr un conocimiento del producto.
- **Media:** el software es complejo, pero es posible aprender a utilizarlo sin mucha asistencia externa a testing.
- **Baja:** el software es fácil de entender y su correspondencia con el dominio es directa.

B.1.5. Criticidad del SUT

Responde al aspecto de qué tan crítico es el producto o la funcionalidad visto desde la perspectiva del negocio, esto es, un software crítico que falla, genera una gran daño al negocio.

- **Alta:** el sistema tiene un gran impacto al negocio. Su falla implica un gran riesgo.
- **Media:** corresponde a productos que tienen relación directa con su contexto, pero son tolerantes a algunas fallas en su dominio.
- **Baja:** este tipo de producto está enfocado a cubrir la funcionalidad, pero la presencia de fallas no es crítica para el negocio. Suele corresponderse con sistemas o funcionalidades secundarias o poco usadas.

B.1.6. Conocimiento Previo del Dominio

Refiere a la experiencia previa que existe en los testers acerca del producto. Esto influye directamente en la estrategia a adoptar y el esfuerzo a invertir en tareas análisis de requerimientos y ejecución de pruebas.

- **Bajo:** el producto y sus funcionalidades son de total desconocimiento para el equipo de testing, y existen pocos antecedentes de pruebas similares.
- **Medio:** se conocen las funcionalidades del producto y los nuevos desarrollos representan desafíos en aprendizaje para el testing, pero conservan muchos puntos en común con el conocimiento existente en el área de testing.

- **Alto:** el sistema es ampliamente manejado por el equipo de testing, los cambios y funcionalidades nuevas no presentan una gran diferencia con las existentes.

B.1.7. Complejidad del Dominio

El criterio de complejidad del dominio expresa la dificultad de adquirir el conocimiento de cómo se comporta el negocio. Además de estar relacionado con la dificultad de la realidad abordada por el producto, también influye la disponibilidad de expertos en el dominio que puedan asistir en esta tarea.

- **Alta:** el dominio es muy complejo, maneja terminología muy extensa y particular, y/o es poco probable contar con la colaboración de un experto en el dominio.
- **Media:** si bien el dominio del problema puede ser complejo, es posible acceder a la asistencia de los expertos en el negocio, de tal manera que se pueda aprender del producto en contraste con la realidad que maneja.
- **Baja:** el negocio manejado por el sistema es sencillo, los términos involucrados son fácilmente manejados por el área de testing y la relación del dominio con el uso del sistema es directa.

B.1.8. Plazos establecidos externamente

Los intervalos de tiempo marcados por la organización afectan directamente la estrategia y la planificación de actividades del proyecto de testing.

- **Cortos y rígidos:** la organización establece plazos muy exigentes para la concreción de los proyectos. Estos plazos, además, no pueden ser negociados y modificados.
- **Cortos y flexibles:** los intervalos de tiempo son acotados, pero el no cumplimiento de los hitos generalmente deriva en un ajuste de los planes y modificación de plazos.
- **Flexibles:** los plazos establecidos y no alcanzados no tienen gran impacto en el negocio. Las metas no alcanzadas se descartan, o se re-planifican con nuevos plazos.

B.2. Testing Ágil

El testing ágil se da en organizaciones que adoptan una metodología ágil de desarrollo. Consecuentemente, en este contexto la documentación de requerimientos es escasa, y sólo se genera aquella que es necesaria. Se cuenta con liberaciones muy frecuentes y las actividades están orientadas a resultados. Una práctica de testing rígida y extremadamente documentada no tendrá éxito en esta realidad.

Suponemos además que la infraestructura, configuración de ambientes y conjunto de herramientas a utilizar son provistas por la organización y encargada a otros equipos.

Supuestos según criterios definidos:

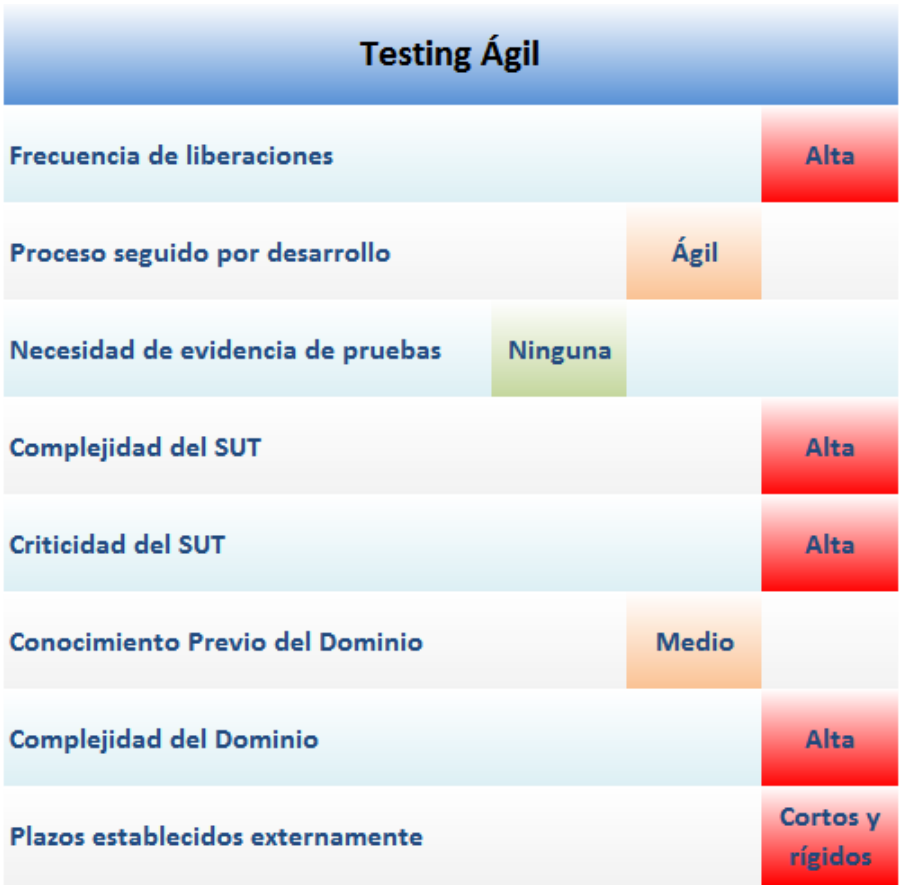

Figura B.1: Contexto de organización que adopta una metodología Ágil.

B.2.1. Estrategia

La estrategia a adoptar, será ejecutar testing exploratorio para la mayoría de las funcionalidades, y testing planificado para las funcionalidades más críticas y que atentan al negocio. La documentación generada es la mínima imprescindible que permita apoyar el proceso, más que dejar constancia de su aplicación (aunque

siempre es positivo contar con ella). El foco tiene que estar en la automatización de pruebas, sobre todo para las tareas engorrosas y pruebas de regresión del producto. Las herramientas de comunicación giran entorno al sistema de gestión de incidentes, e interacciones cara a cara (preferibles antes que otros medios de comunicación, como ser correo electrónico o chat).

B.2.2. Instanciación de FIT

Se enfoca el proceso a la automatización. Se prescinde de las etapas de verificación de documentos, y la etapa de análisis de defectos se utiliza para identificar debilidades en los componentes del sistema y mejorar la estrategia de pruebas de regresión. Los procesos de testing manual y automatizado generan la documentación mínima imprescindible. El testing manual se encarga de las pruebas de humo y de generar información que es insumo para el testing automatizado. De ser posible se puede aplicar testing temprano, y enfocar las actividades de testing para proveer información lo antes posible ante nuevo software funcionando, apto para ser verificado. La comunicación es importante, las etapas de informar resultados además pasan a tener un componente fuerte de interacción humana y reporte oral, antes que escrito.

B.2.3. Representación gráfica del roadmap

Etapas del Módulo II y III en el roadmap

Como se puede ver en los siguientes diagramas, casi todas las etapas y actividades de los procesos son transitadas, de hecho, en el módulo II solamente se deja por fuera la etapa de Verificación de Documentos, debido a que se supone que en la organización no hay documentos formales del proceso de desarrollo. Para el módulo III se toman todas las etapas.

La interrogante válida que surge para este caso, sería: ¿Por qué en un contexto ágil se toma la decisión de abordar tantas etapas y actividades de los procesos?, dado que es razonable pensar que implica una sobrecarga de trabajo que obstaculizaría el logro de los objetivos. El motivo por el cuál se considera esta instrumentación como la adecuada, es que por más que se transiten muchas actividades y etapas; no se generan documentos formales y extensos para documentar el proceso, sino que se vuelca el esfuerzo en las actividades de comunicación y el registro de información que corresponde sólo a la generación de material imprescindible para las pruebas. Como comentario de ejemplo, se podría decir que en este contexto no se contará con plantillas de documentos formales conteniendo control de versión y autores, introducción, puntos centrales y conclusiones.

Etapas del Proceso de Testing Funcional Manual

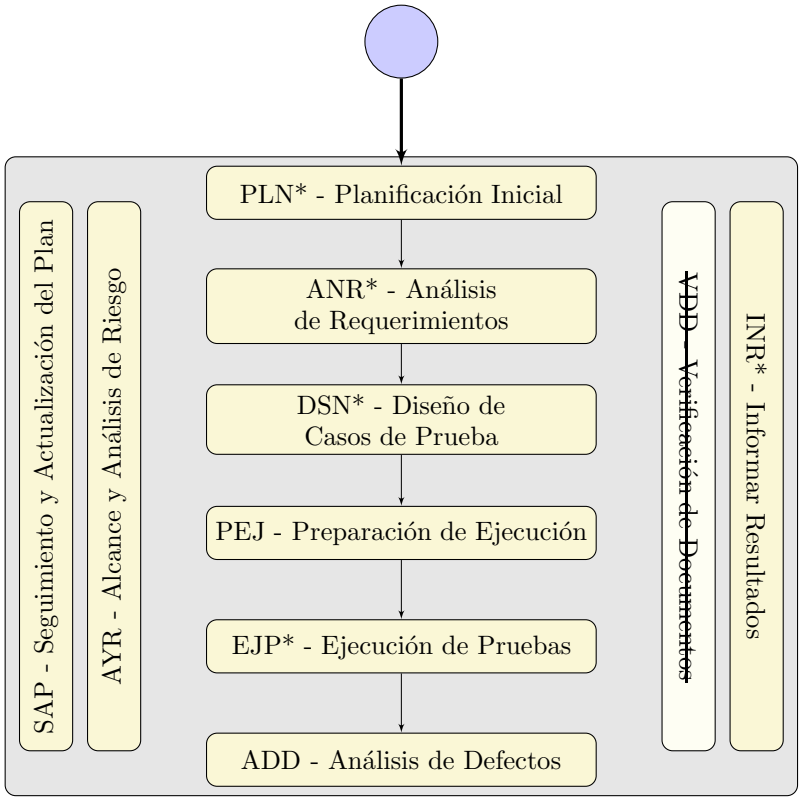

Figura B.2: FIT Módulo II: Proceso de Testing Funcional Manual para Testing Ágil

.

Actividades de Planificación Inicial

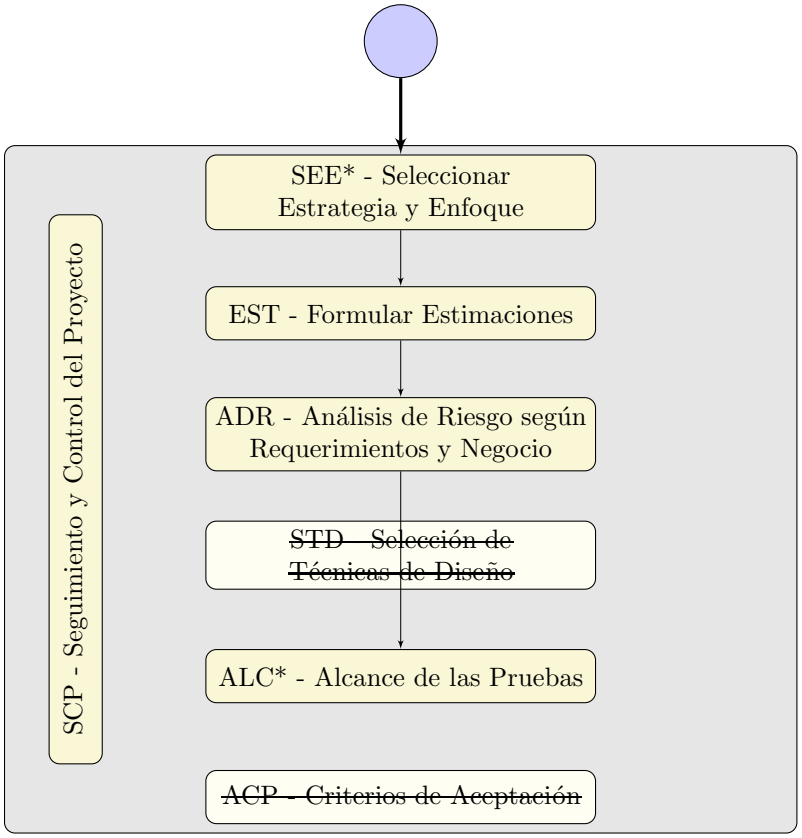

Figura B.3: Etapa Planificación Inicial para Testing Ágil

.

Actividades de Análisis de Requerimientos

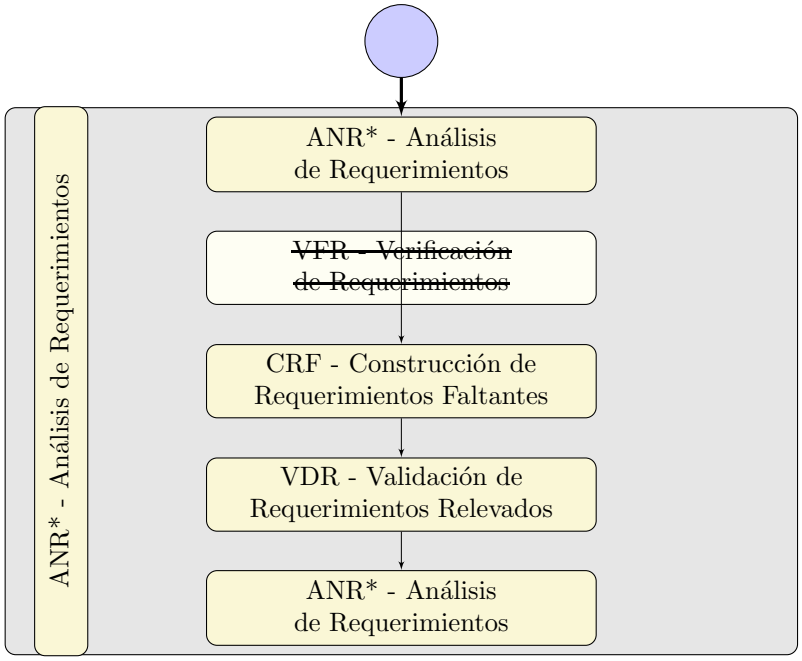

Figura B.4: Etapa Análisis de Requerimientos para Testing Ágil

Actividades de Diseño de Pruebas

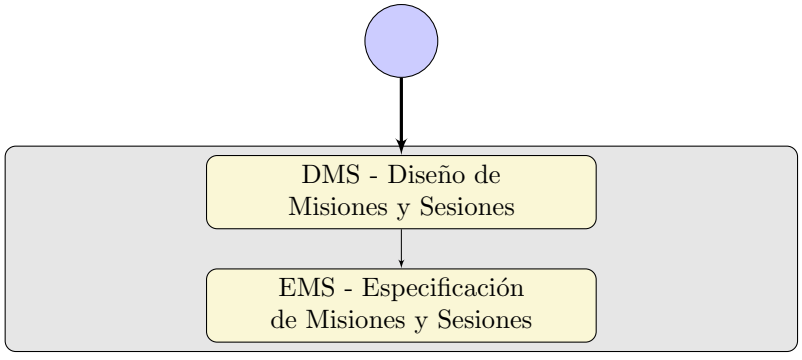

Figura B.5: Etapa Diseño de Pruebas del Testing Exploratorio para Testing Ágil

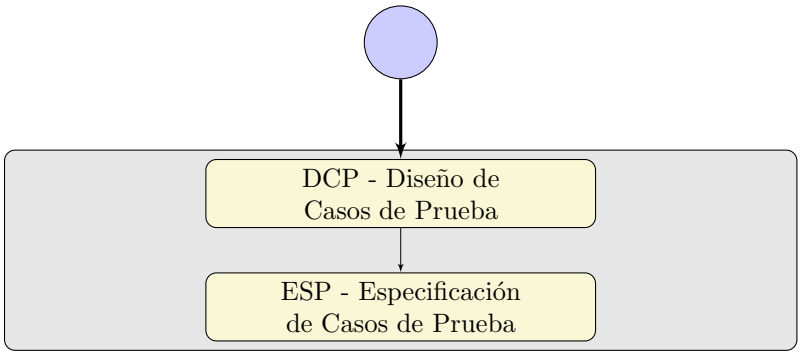

Figura B.6: Etapa Diseño de Pruebas del Testing Planificado para Testing Ágil

Actividades de Preparación de Ejecución

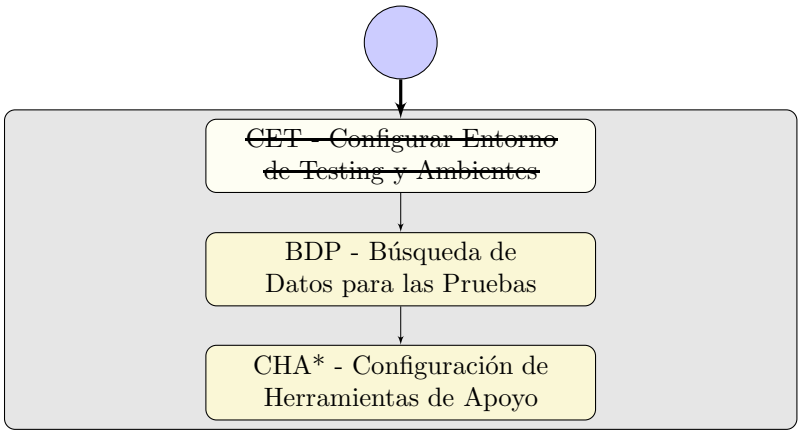

Figura B.7: Etapa Preparación de Ejecución para Testing Ágil

Actividades de Ejecución de Pruebas

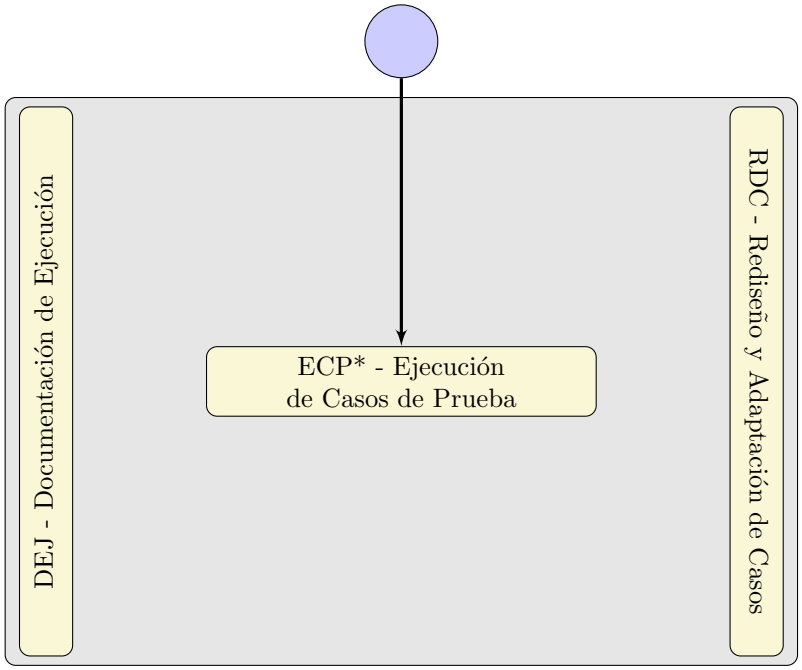

Figura B.8: Etapa Ejecución de Pruebas para Testing Ágil

Actividades de Alcance y Análisis de Riesgo

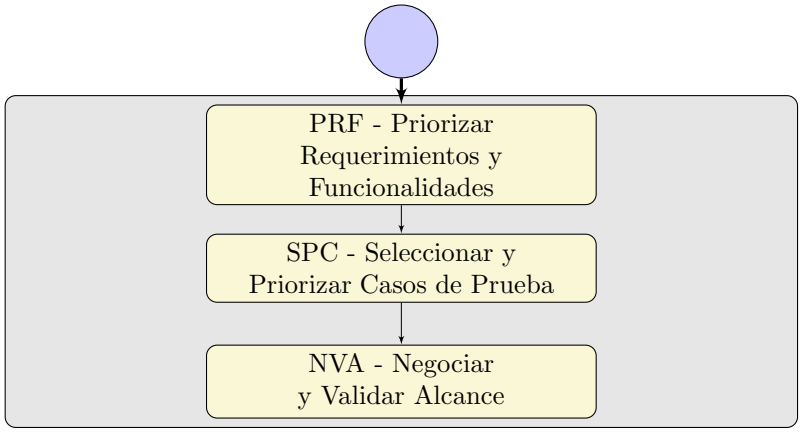

Figura B.9: Etapa Alcance y Análisis de Riesgo para Testing Ágil

Actividades de Informar Resultados

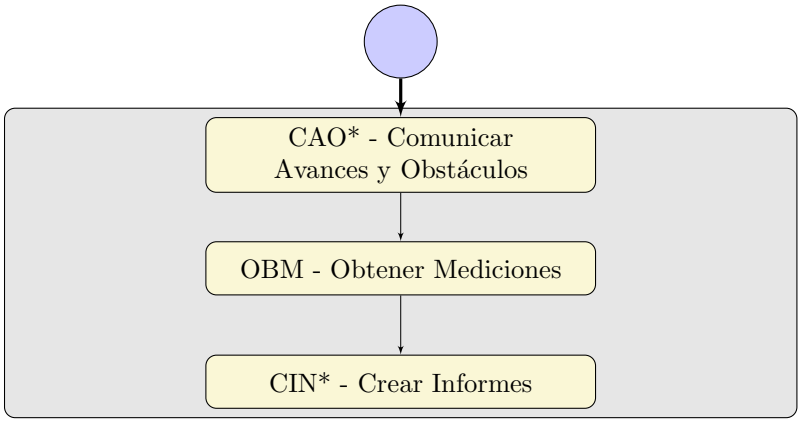

Figura B.10: Etapa Informar Resultados para Testing Ágil

Actividades de Seguimiento y Actualización del Plan

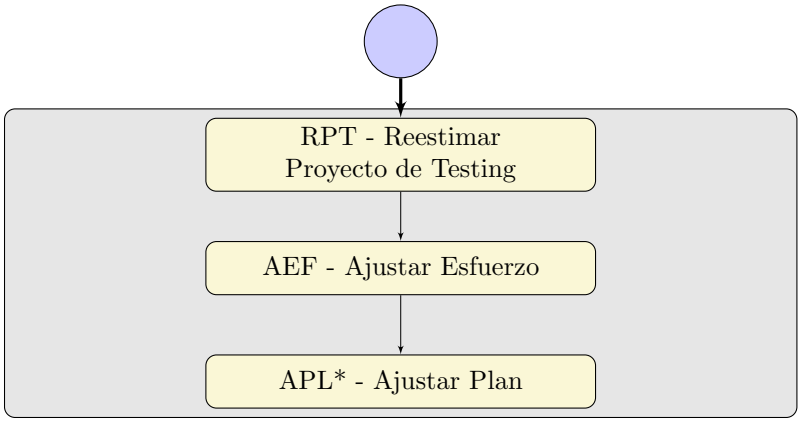

Figura B.11: Etapa Seguir y Actualizar el Plan para Testing Ágil

Actividades de Análisis de Defectos

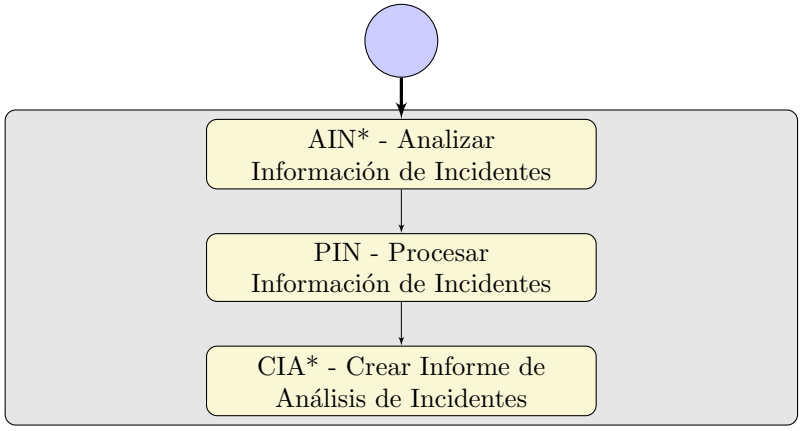

Figura B.12: Etapa Análisis de Defectos para Testing Ágil

.

Etapas del Proceso de Testing Funcional Automatizado

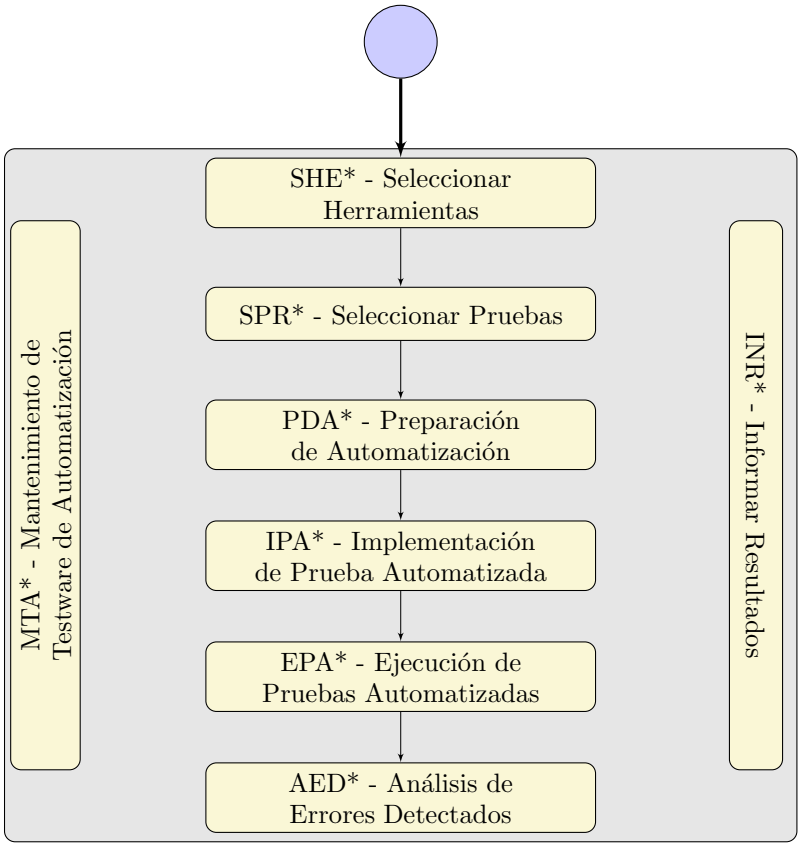

Figura B.13: FIT Módulo III: Proceso de Testing Funcional Automatizado para Testing Ágil

.

Actividades de Seleccionar Herramientas

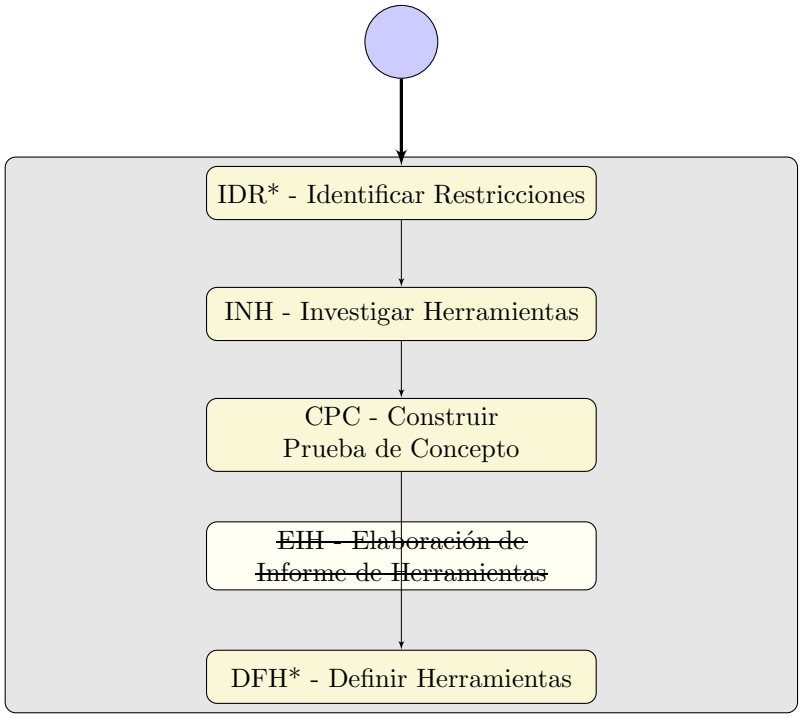

Figura B.14: Etapa Seleccionar Herramientas para Testing Ágil

.

Actividades de Preparación de Automatización

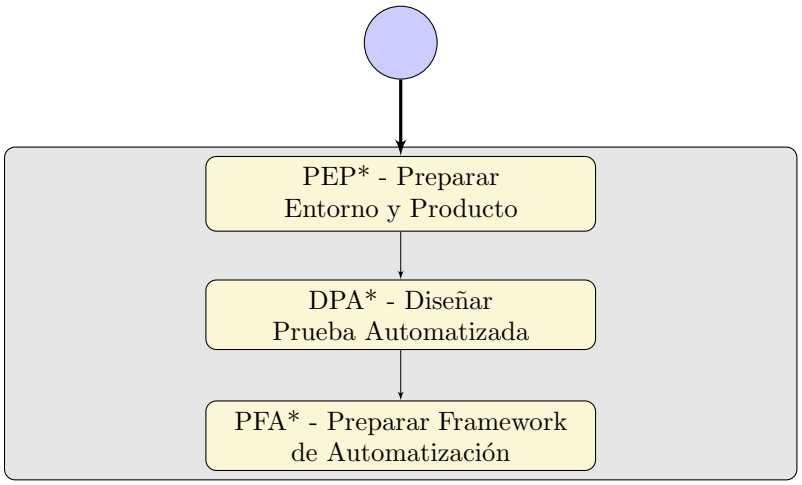

Figura B.15: Etapa Preparación de Automatización para Testing Ágil

Actividades de Implementación de Prueba Automatizada

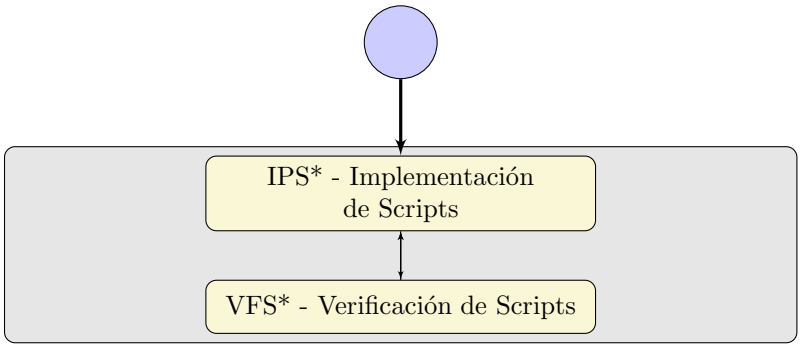

Figura B.16: Etapa Implementación de Prueba Automatizada para Testing Ágil

Actividades de Ejecución de Pruebas Automatizadas

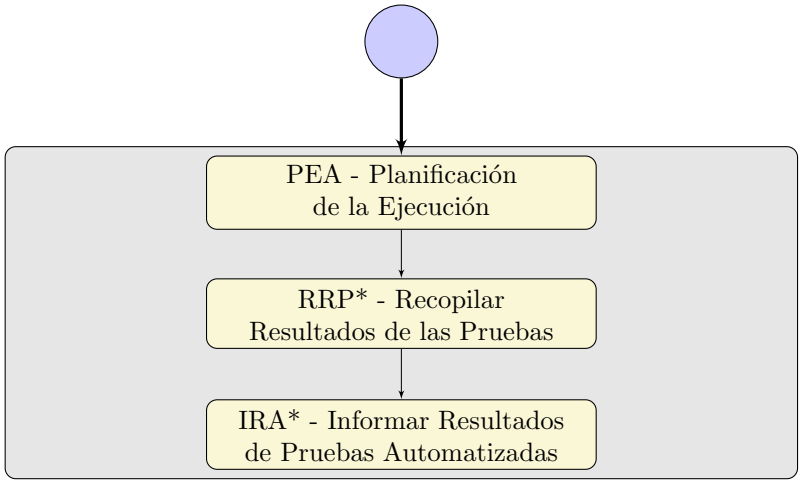

Figura B.17: Etapa Ejecución de Pruebas Automatizadas para Testing Ágil

Actividades de Mantenimiento de Testware de Automatización

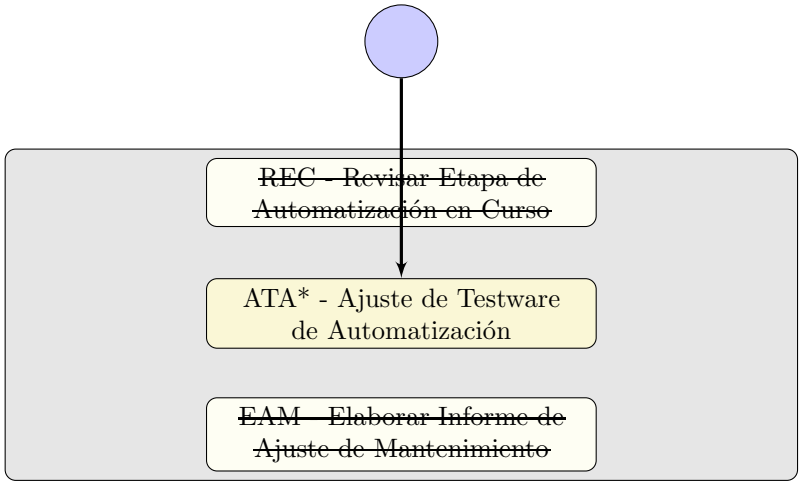

Figura B.18: Etapa Mantenimiento de Testware de Automatización para Testing Ágil

B.3. Testing Independiente con Interrelación con Equipos

En este contexto, se cuenta con una realidad muy dispar de procesos de desarrollo y actores involucrados. El tester suele tomar un rol articulador, al necesitar la visión de varios actores (a quienes también puede proveer información) cuyos canales de comunicación parecen caóticos.

Supuestos según criterios definidos:

Testing Independiente con Interrelación con Equipos (tester como articulador)	
Frecuencia de liberaciones	Alta
Proceso seguido por desarrollo	Varios
Necesidad de evidencia de pruebas	Media
Complejidad del SUT	Alta
Criticidad del SUT	Alta
Conocimiento Previo del Dominio	Medio
Complejidad del Dominio	Alta
Plazos establecidos externamente	Cortos y rígidos

Figura B.19: Contexto de organización con Testing Independiente integrado.

B.3.1. Estrategia

Mayoritariamente testing exploratorio, diseño de pruebas para asuntos muy críticos o por negociación con el contratante. La documentación generada es la imprescindible para apoyar el proceso, pero en condiciones normales, se genera bastante documentación, dado que el tester tiene acceso al núcleo de los requerimientos y acumula mucha información, sobre todo para verificar realidades complejas. La automatización puede ser abordada como estrategia de mitigación a tareas que insumen mucho tiempo, o puede ser interés de la organización. Tiene una política de informe de resultados frecuente para los demás responsables.

B.3.2. Instanciación de FIT

El proceso de testing es "descontracturado". Se toman todas las etapas de análisis y de planificación, pero se genera la documentación necesaria y/o exigida por la organización contratante. Las etapas de análisis marcan el perfil articulador del tester en este contexto, consiguiendo que los diferentes actores trabajen en conjunto con el fin común de aportar a la mejora de la calidad (desarrolladores, gerentes, usuarios finales, interesados, entre otros). En este contexto se puede automatizar pruebas dependiendo de la disponibilidad de recursos y el interés del contratante.

B.3.3. Representación gráfica

Etapas del Módulo II y III en el roadmap

Para este contexto, el proceso de testing prescinde solamente de la etapa de Análisis de Defectos, suponiendo que no hay una estrategia prevista en la organización para dedicar un tiempo a investigar la información residente en el sistema de seguimiento de incidentes. El proceso de testing automatizado cuenta con todas las etapas definidas.

Se asume que la organización no es extremadamente formal con sus procedimientos y cuenta con muchos proyectos de desarrollo. La rigurosidad de las actividades de testing al nivel de exigencia y forma de trabajo de la organización, en incluso del proyecto en particular. Las actividades que generan documentación más extensa son las relacionadas con comunicación, ejecución de pruebas y automatización de pruebas.

Etapas del Proceso de Testing Funcional Manual

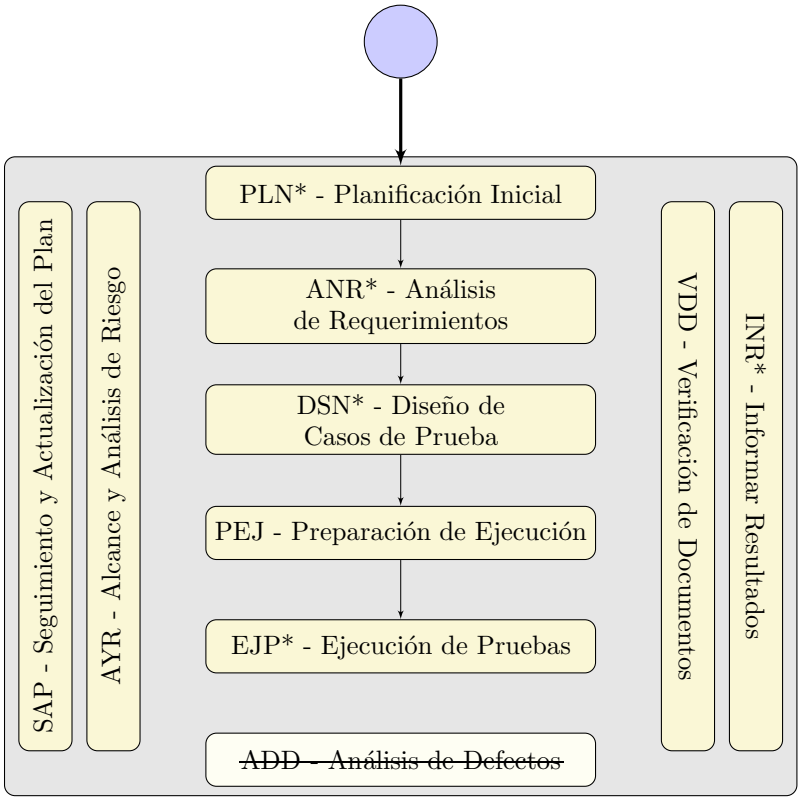

Figura B.20: FIT Módulo II: Proceso de Testing Funcional Manual para Testing Independiente

.

Actividades de Planificación Inicial

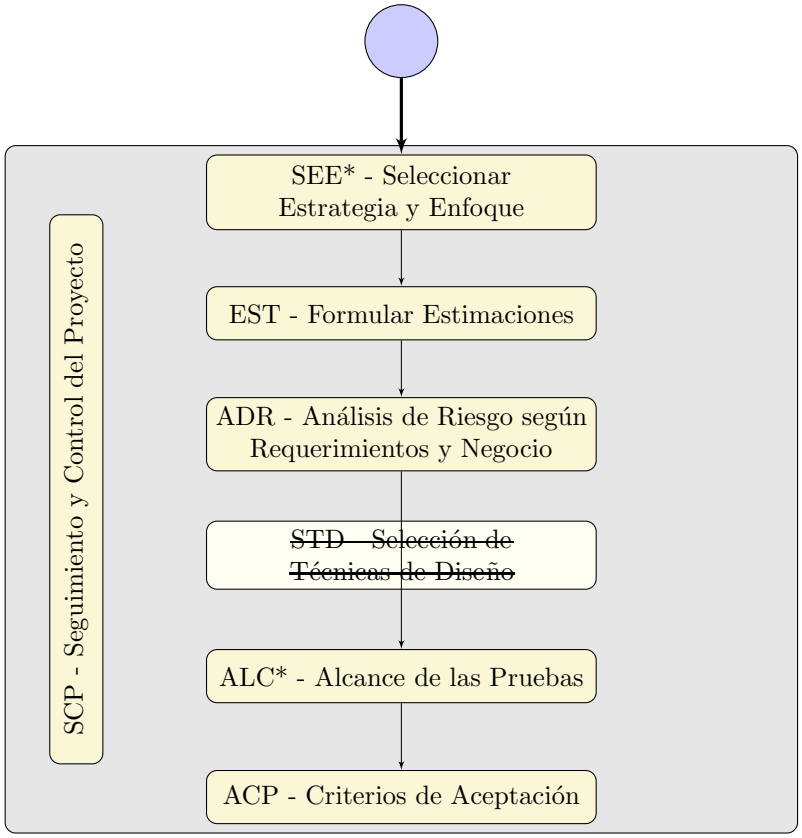

Figura B.21: Etapa Planificación Inicial para Testing Independiente

.

Actividades de Análisis de Requerimiento

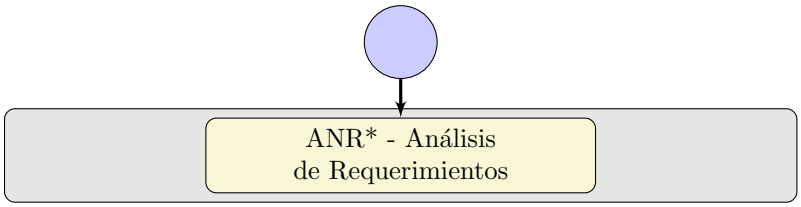

Figura B.22: Etapa Análisis de Requerimientos para Testing Independiente

Actividades de Diseño de Pruebas

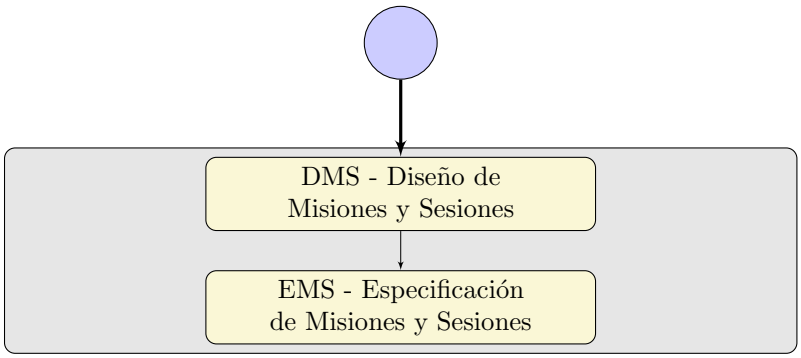

Figura B.23: Etapa Diseño de Pruebas del Testing Exploratorio para Testing Independiente

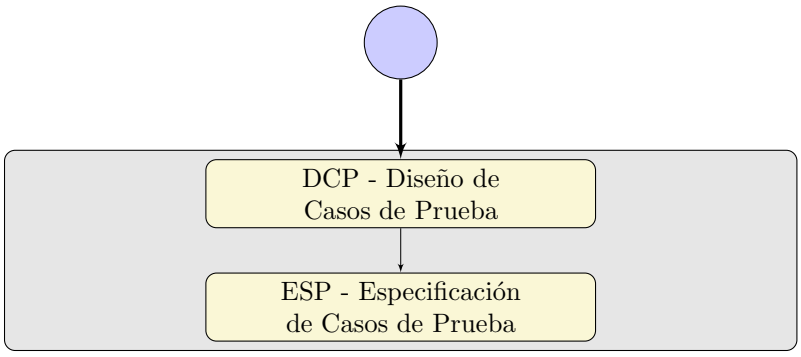

Figura B.24: Etapa Diseño de Pruebas del Testing Planificado para Testing Independiente

Actividades de Preparación de Ejecución

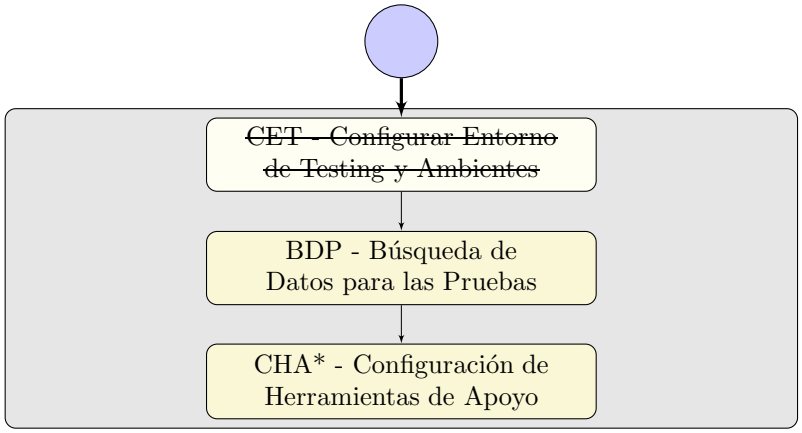

Figura B.25: Etapa Preparación de Ejecución para Testing Independiente

Actividades de Ejecución de Pruebas

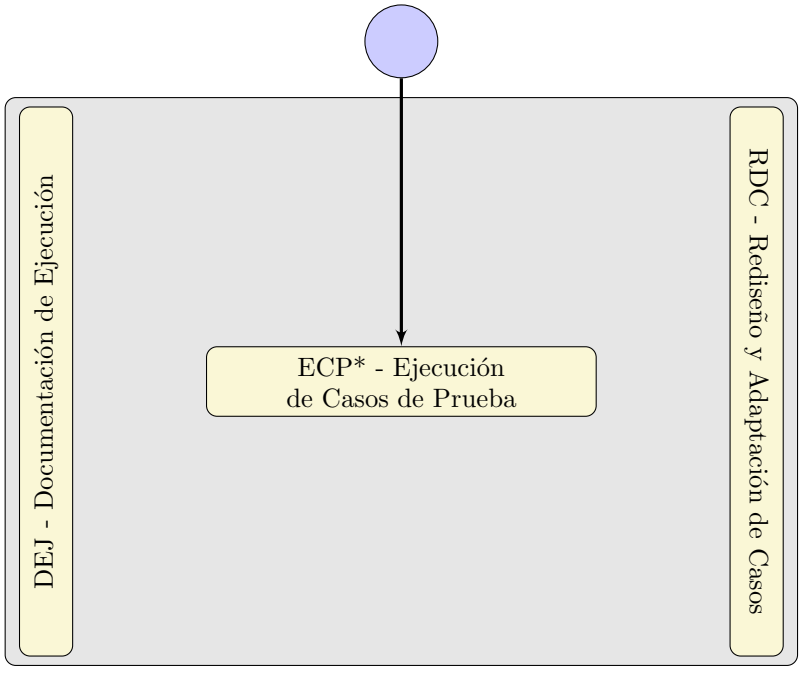

Figura B.26: Etapa Ejecución de Pruebas para Testing Independiente

Actividades de Alcance y Análisis de Riesgo

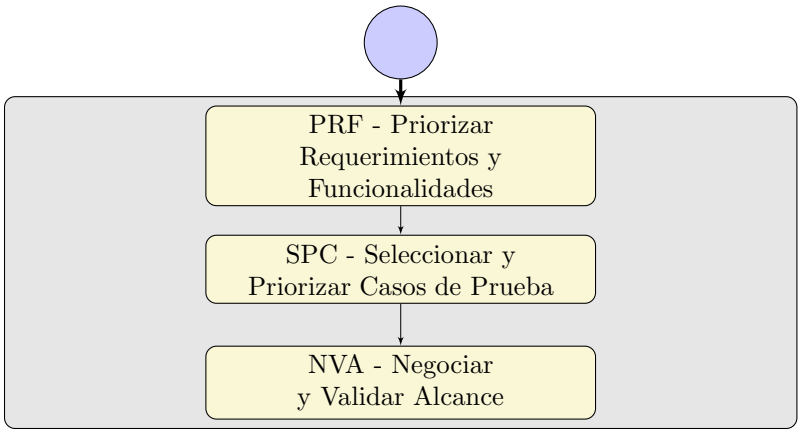

Figura B.27: Etapa Alcance y Análisis de Riesgo para Testing Independiente

Actividades de Informar Resultados

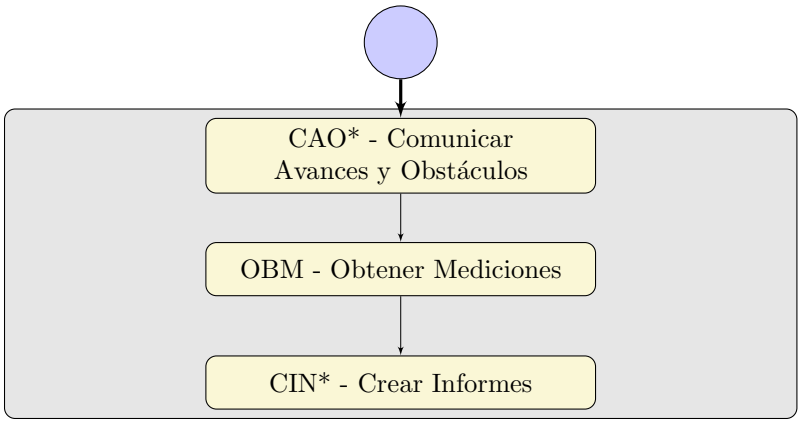

Figura B.28: Etapa Informar Resultados para Testing Independiente

Actividades de Seguimiento y Actualización del Plan

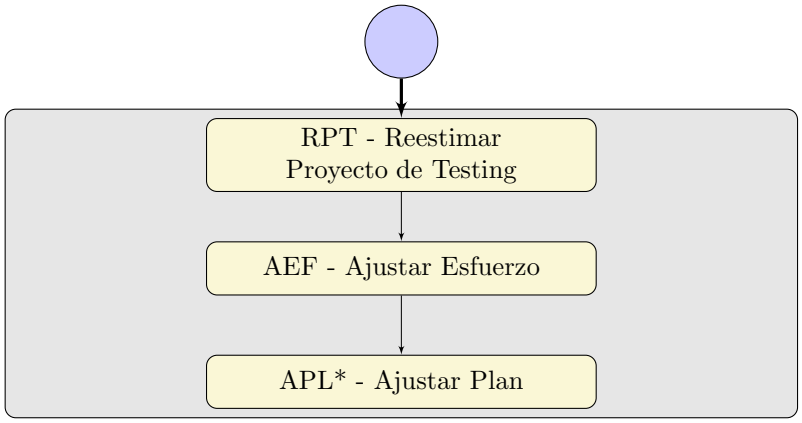

Figura B.29: Etapa Seguir y Actualizar el Plan para Testing Independiente

Actividades de Verificación de Documentos

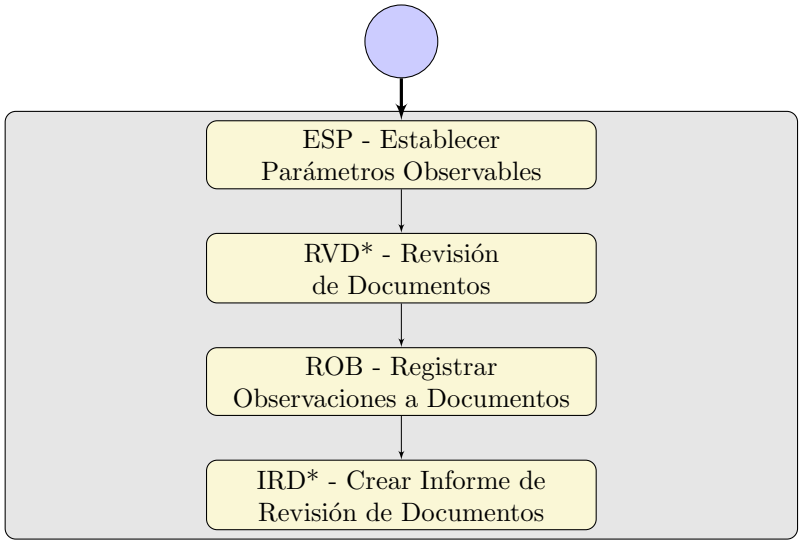

Figura B.30: Etapa Verificación de Documentos para Testing Independiente

.

Etapas del Proceso de Testing Funcional Automatizado

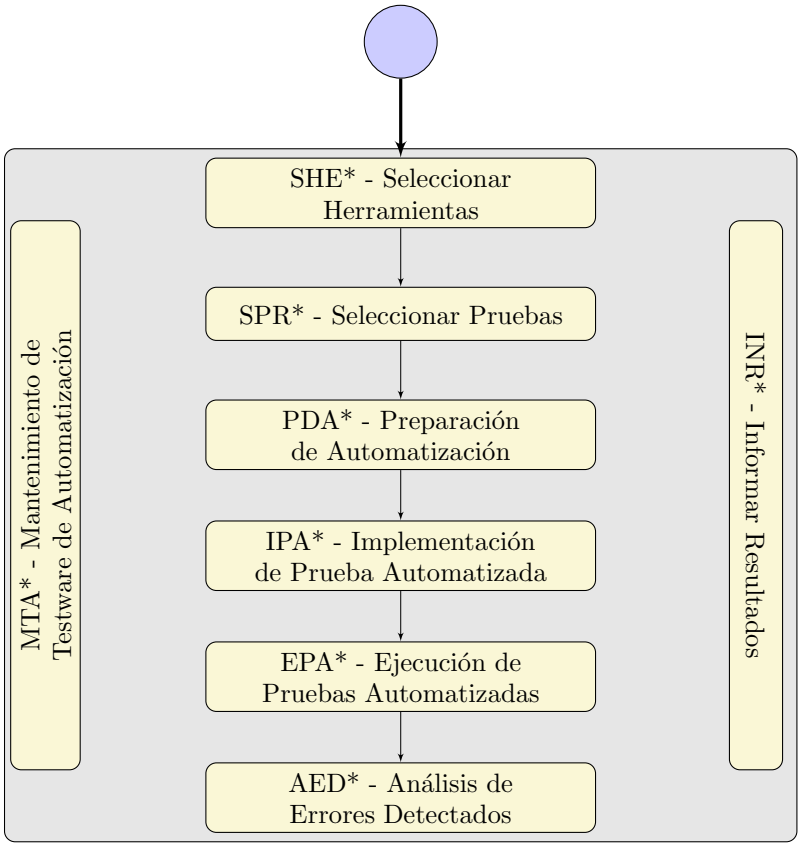

Figura B.31: FIT Módulo III: Proceso de Testing Funcional Automatizado para Testing Independiente

.

Actividades de Seleccionar Herramientas

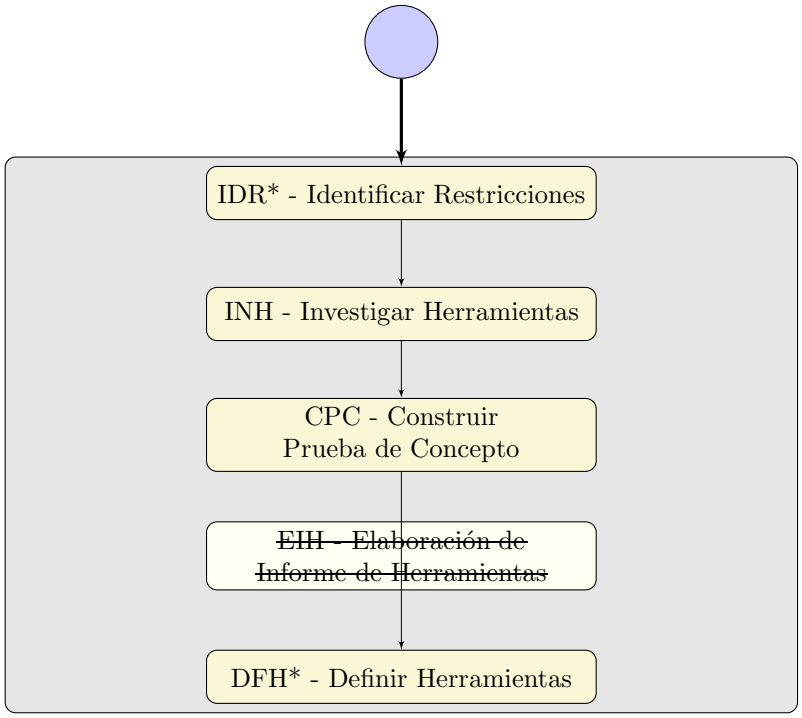

Figura B.32: Etapa Seleccionar Herramientas para Testing Ágil

.

Actividades de Preparación de Automatización

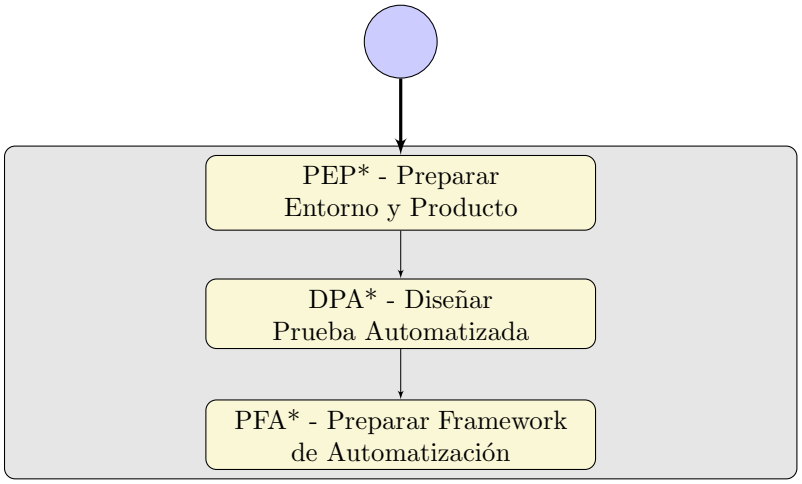

Figura B.33: Etapa Preparación de Automatización para Testing Independiente

Actividades de Implementación de Prueba Automatizada

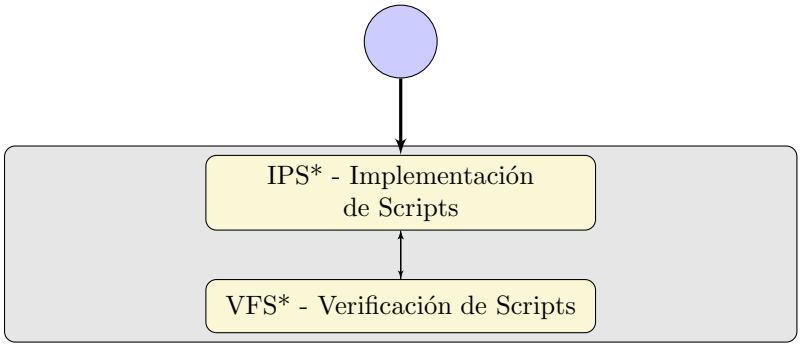

Figura B.34: Etapa Implementación de Prueba Automatizada para Testing Independiente

Actividades de Ejecución de Pruebas Automatizadas

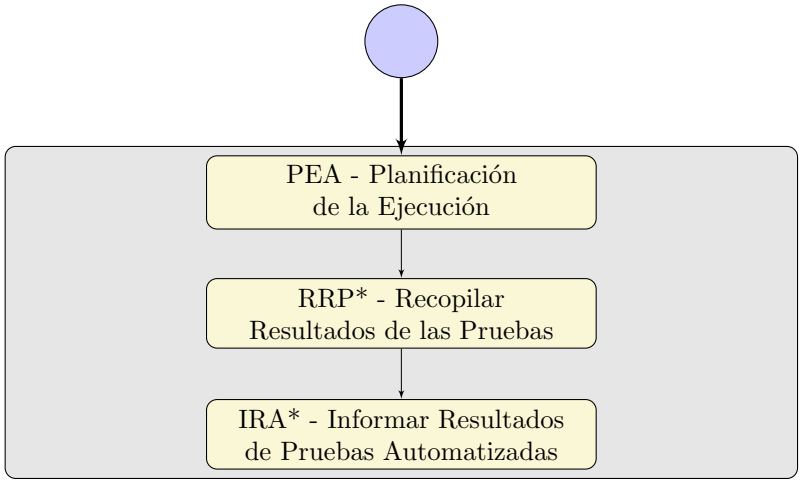

Figura B.35: Etapa Ejecución de Pruebas Automatizadas para Testing Independiente

Actividades de Mantenimiento de Testware de Automatización

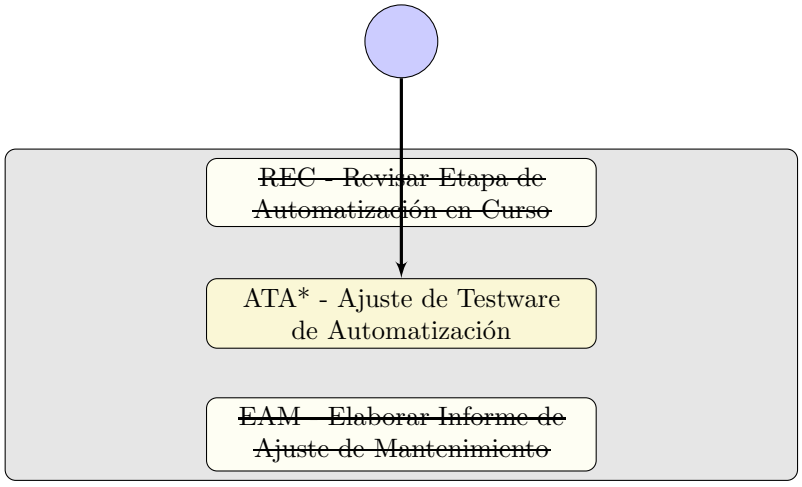

Figura B.36: Etapa Mantenimiento de Testware de Automatización para Testing Independiente

B.4. Desarrollo en Cascada con proceso rígido auditable

Proceso de desarrollo con en etapas definidas, iterativo e incremental, el cual es seguido rigurosamente. Las pautas organizacionales marcan que se debe dejar constancia de los procesos llevados adelante y sus resultados. Los roles y las responsabilidades están definidas, por lo tanto se sabe que se espera de cada equipo. En estos contextos el testing tiene el desafío de no transformarse en cuello de botella al ser muchas veces incorporado en etapas tardías del desarrollo del producto.

Supuestos según criterios definidos:

Desarrollo en Cascada iterativo incremental con proceso rígido auditable	
Frecuencia de liberaciones	Media
Proceso seguido por desarrollo	Cascada
Necesidad de evidencia de pruebas	Alta
Complejidad del SUT	Alta
Criticidad del SUT	Alta
Conocimiento Previo del Dominio	Medio
Complejidad del Dominio	Alta
Plazos establecidos externamente	Cortos y Flexibles

Figura B.37: Contexto de organización que usa desarrollo en Cascada.

B.4.1. Estrategia

La estrategia abordada por el equipo de testing tendrá que basarse en testing temprano con verificación de documentos y validación de requerimientos. El testing planificado tendrá lugar para la mayoría de las funcionalidades a verificar y se preferirán pruebas de humo con testing exploratorio. La documentación generada será detallada, y la automatización será efectuada sólo en partes muy estratégicas del sistema.

B.4.2. Instanciación de FIT

Todas las etapas y actividades son ejecutadas y documentas, salvo análisis de defectos que es opcional. Las etapas de información de resultados y comunicación son especialmente importantes. Las etapas que definen el transcurso del proyecto deben contar con información validada por los responsables..

B.5. Testing Independiente a modo de Testing Factory

Este escenario corresponde a las tareas de testing encargadas a un equipo hermético que trabaja a modo de *Testing Factory*(fábrica de testing). En este contexto la responsabilidad del testing es delegada al equipo y el alcance de las pruebas es estrictamente negociado entre la organización y el equipo. La documentación generada en el proceso tiene dos formas: documentación interna para apoyar el proceso de testing del equipo y documentación visible a la organización comunicando avances, obstáculos y resultados de las pruebas (u otros documentos exigidos por la organización).

Supuestos según criterios definidos:

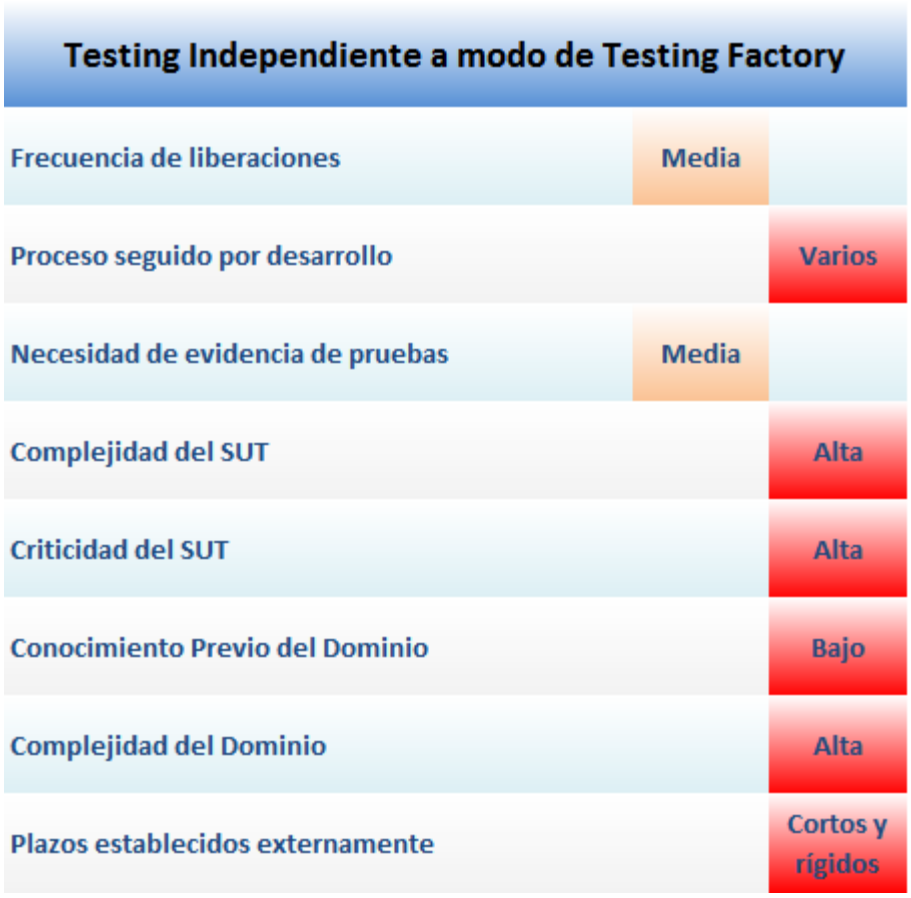

Figura B.38: Contexto de organización interactuando con una Testing Factory.

B.5.1. Estrategia

En este contexto la estrategia es similar a la de un proceso rígido. Por ejemplo, testing exploratorio para las pruebas de humo, testing planificado con diseño de casos de prueba para la mayoría de las funcionalidades. La automatización se lleva adelante sólo si es contratada por la organización o si esta reporta una relación costo beneficio atractiva para la empresa tercerizada. La documentación interna difiere de la externa en la forma de comunicar y su contenido. La documentación externa, no suele revelar los detalles de la metodología seguida, sino proveer información acerca de los avances, resultados y obstáculos de las pruebas.

B.5.2. Instanciación de FIT

El proceso seguido puede ser rígido o flexible dependiendo de la organización interna del equipo. Generalmente en esta modalidad, se define un proceso de cómo llevar adelante el testing, que suele contar con documentación interna y que se comunica con la organización contratante. Las etapas avanzadas, como ser las de *post morten*, se llevan adelante sólo si la organización tercerizadora las cree de importancia y se negocian con el equipo de la testing factory. En este contexto podríamos suponer que las etapas se siguen rigurosamente, y que se genera la documentación pertinente con cierta formalidad. Las etapas de información de resultados son importantes y tienen un componente interno y externo, dependiendo si el destinatario es el mismo equipo, autoridades de la empresa tercerizada, o el contratante.

B.6. Testing en Organización preocupada por la Mejora continua

Es una organización que generalmente cumple con sus metas en tiempo y forma. Se estimulan tareas que analicen los procesos seguidos para aprender de ellos y mejorarlos. La preocupación por la calidad abarca al producto y la organización misma.

Supuestos según criterios definidos:

Testing en Organización preocupada por la Mejora Continua	
Frecuencia de liberaciones	Media
Proceso seguido por desarrollo	Cascada
Necesidad de evidencia de pruebas	Alta
Complejidad del SUT	Alta
Criticidad del SUT	Alta
Conocimiento Previo del Dominio	Medio
Complejidad del Dominio	Alta
Plazos establecidos externamente	Cortos y rígidos

Figura B.39: Contexto de organización preocupada por la Mejora Continua.

B.6.1. Estrategia

La estrategia es apuntar a la automatización y sacar provecho de la información acumulada en los sistemas de gestión de incidentes. El testing manual predominante adopta testing exploratorio. Por fuera del proceso definido, se pueden automatizar pruebas no funcionales.

B.6.2. Instanciación de FIT

Además de las etapas comunes, se pueden agregar las actividades avanzadas de *post mortem*, como ser análisis de defectos. También se apunta al testing temprano y validación de documentos. La documentación generada no es excesiva, sino que es la necesaria para apoyar al proceso. Las etapas de comunicación son muy importantes, y el foco principal está en la automatización de pruebas. Todas las etapas del proceso son ejecutadas, y se apunta a que las pruebas sean mantenibles. Además se puede incluir actividades en las que se promueve el buen uso de los conceptos de testing y la preocupación por la calidad.

Glosario

A

Aplicación Usado como sinónimo de *Producto de Software*.

Automatización Acción de *automatizar*.

Automatizar Tareas entorno al objetivo de generar pruebas implementadas de tal manera que se puedan ejecutar de forma desatendida.

C

Caso de Prueba Unidad mínima de verificación que puede ensayarse sobre un software.

Ciclo de Prueba Período de tiempo en el cual se practican tareas de testing sobre una versión de determinado producto, sin que esta sufra cambios.

Ciclo Funcional Conjunto de funcionalidades relacionadas que corresponden a flujo de uso del sistema.

F

Flujo Funcional Utilizado como sinónimo de *Ciclo Funcional*.

Framework *(como herramienta de software)* Herramienta, o conjunto de herramientas de software que tienen el propósito de asistir o llevar adelante determinada tarea. Por ejemplo **framework** de testing unitario, corresponde a la herramienta que ejecuta, planifica y clasifica las pruebas unitarias. En el capítulo 5 este término es referido de esta forma.
(como propuesta metodológica) Se refiere a la metodología o forma de trabajo con su conjunto de etapa y actividades propuestas para los diferentes procesos. En el capítulo 4 se verá el término usado de esta forma y como sinónimo de *Marco de Trabajo*.

Funcionalidad Característica de funcionamiento del producto de software existente o que se desarrollará.

I

Instancia Elemento particular perteneciente a una clase. El capítulo 4 es usado para referirse a una aplicación particular de la metodología en un contexto dado.

Instanciable Que se pueden generar instancias del elemento referido. En el capítulo 4 se dice que el *framework* es instanciable, porque se pueden generar instancias de él para un contexto particular.

Instanciación Acción de generar instancias del elemento referido. En el capítulo 4 se refiere como instanciación a la acción de contextualizar la metodología propuesta para una realidad concreta.

M

Mantenibilidad Es la capacidad de un elemento de ser *mantenible*.

Mantenible Que se le pueden realizar ajustes y mejoras sin que la mayor parte del esfuerzo sea invertido en la preparación del objeto para dichos ajustes.

P

Producto Usado como sinónimo de *Producto de Software*.

Proyecto de Grado Se refiere al trabajo particular llevado adelante por los estudiantes autores de este documento en la materia proyecto de grado de la carrera de Ingeniería en Computación de la Facultad de Ingeniería, Universidad de la República.

Proyecto Piloto Experiencia práctica de aplicación de la metodología propuesta, en sistemas de software de ASSE.

Prueba Usado como sinónimo de *Caso de Prueba*, o para referirse a todas las actividades de verificación de determinado *sistema*.

Prueba Automatizada Implementación de un caso de prueba en forma programática. Es un producto de trabajo de la *Automatización*.

Pruebas de Regresión Tareas de verificación que tienen la intención de detectar si el producto no ha sufrido una regresión en su calidad debido a cierto mantenimiento.

R

Requerimiento Descripción de una funcionalidad de determinado sistema.

S

Script Código de un caso de prueba implementado en algún lenguaje de programación o lenguaje propio de alguna herramienta.

Sistema Usado como sinónimo de *Producto de Software*.

Software Under Test Sistema de software que es el objetivo de la verificación.

SUT Sigla que significa *Software Under Test*.

T

Tercerización Acción de *tercerizar* determinadas tareas de una organización.

Tercerizado/da Organización que presta servicios al *tercerizador*.

Tercerizador Organización que delega actividades a otra que no la integra.

Tercerizar Significa que una organización decide que ciertas actividades las desarrolle otra ajena. Generalmente implica una relación comercial, pero no es excluyente.

Test Usado como sinónimo de *Caso de Prueba*.

Testeabilidad Testeabilidad es la capacidad que tiene un software de ser probado, en un determinado contexto.

Testeable. Se dice de un software que acorde algún criterio puede ser verificado en determinado contexto.

Tester Persona encargada de ejecutar tareas relativas al *testing*.

Testing Actividades involucradas con la verificación de un software.

Testware Producto de trabajo de las actividades relacionadas con la generación de *Pruebas Automatizadas*.

W

Web Service Es un componente de software que brinda servicios en la web a través de una interfaz siguiendo ciertos estándares y protocolos de comunicación.

Bibliografía

[20005] *Software Engineering - Software product Quality Requirements and Evaluation (SQuaRE) - Guide to SQuaRE*. Number ISO/IEC 25000. ISO/IEC, Geneva, Switzerland, 2005.

[Aca01] Spanish Royal Academy. *Diccionario de la Lengua Espanola (Spanish Edition) (2 volumes)*. Espasa Calpe Mexicana, S.A., 2001.

[Bac99] James Bach. Heuristic risk-based testing. *Software Testing and Quality Engineering*, Noviembre 1999.

[Bac00] Jonathan Bach. Session-based test management. *Software Testing and Quality Engineering*, Noviembre 2000.

[Bac02] James Bach. Exploratory testing explained, Abril 2002. The Test Practitioner, disponible en http://www.satisfice.com/articles/et-article.pdf, accedido Diciembre de 2010.

[Bac11] James Bach. James bach's blog, Mayo 2011.

[Bol09] Michael Bolton. Testability, Julio 2009. Accedido en diciembre de 2010.

[CES10] CES, 12 2010.

[FG99a] Mark Fewster and Dorothy Graham. *Software Test Automation*. Addison-Wesley Professional, 1999.

[FG99b] Mark Fewster and Dorothy Graham. *Software Test Automation*, chapter 9, pages 229–232. Addison-Wesley Professional, 1999.

[FG99c] Mark Fewster and Dorothy Graham. *Software Test Automation*, chapter 10, pages 248–280. Addison-Wesley Professional, 1999.

[FG99d] Mark Fewster and Dorothy Graham. *Software Test Automation*, chapter 1, pages 7–9. Addison-Wesley Professional, 1999.

[FG99e] Mark Fewster and Dorothy Graham. *Software Test Automation*, chapter 1, pages 22–25. Addison-Wesley Professional, 1999.

[FG99f] Mark Fewster and Dorothy Graham. *Software Test Automation*, chapter 9, page 243. Addison-Wesley Professional, 1999.

[FG99g] Mark Fewster and Dorothy Graham. *Software Test Automation*, chapter 1, pages 4–5. Addison-Wesley Professional, 1999.

[FG99h] Mark Fewster and Dorothy Graham. *Software Test Automation*, chapter 3, pages 83–89. Addison-Wesley Professional, 1999.

[Fou11] TMMi Foundation. Test maturity model integration, 2011.

[Her93] Paul Herzlich. The w-model. EuroSTAR, 1993.

[IAB97] IABG. V-modell, Agosto 1997.

[IEE91] IEEE. *IEEE Standard Glossary of Software Engineering Terminology/IEEE Std 610.12-1990*. Institute of Electrical & Electronics Enginee, 1991.

[IEE06] IEEE. *IEEE Standard Dictionary of Measures of the Software Aspects of Dependability*. Number IEEE Std 982.1$^{\text{TM}}$-2005. Mayo 2006.

[JAWea11] J. A. W. James A. Whittaker et al. Google testing blog, Mayo 2011.

[Kne08] Ralf Kneuper. *CMMI: Improving Software and Systems Development Processes Using Capability Maturity Model Integration (CMMI-DEV)*. Rocky Nook, 2008.

[Mil95] J. Voas & K. Miller. Software testability: the new verification. *IEEE Soft*, 12:17 – 28, Mayo 1995.

[Mye04a] Glenford J. Myers. *The Art of Software Testing, Second Edition*, chapter 2, page 6. Wiley, 2004.

[Mye04b] Glenford J. Myers. *The Art of Software Testing, Second Edition*. Wiley, 2004.

[Pet09] Bret Pettichord. Design for testability, Octubre 2009. Accedido en diciembre de 2010.

[Pér06] Beatriz Pérez. Proceso de testing funcional independiente. Master's thesis, Universidad de la República - Facultad de Ingeniería, 2006.

[Sog09] Sogeti. *TPI Next - Business Driven Test Process Improvement*. UTN Publishers, 2009.

[vdABR+08] Leo van der Aalst, Rob Baarda, Ewald Roodenrijs, Johan Vink, and Ben Visser. *TMap Next, Business Driven Test Managament*. UTN Publishers, 2008.

[Whi09a] James A. Whittaker. *Exploratory Software Testing: Tips, Tricks, Tours, and Techniques to Guide Test Design*, chapter Appendix C, page 183. Addison-Wesley Professional, 2009.

[Whi09b] James A. Whittaker. *Exploratory Software Testing: Tips, Tricks, Tours, and Techniques to Guide Test Design*, chapter Appendix C, pages 183–184. Addison-Wesley Professional, 2009.

[Whi09c] James A. Whittaker. *Exploratory Software Testing: Tips, Tricks, Tours, and Techniques to Guide Test Design*. Addison-Wesley Professional, 2009.

[Zap10] Carlos Mario et al. Zapata. Una representación gráfica del testing ágil. *Avances en Sistemas e Informática*, 7:18–25, 2010.