UNIVERSITAT POLITÈCNICA DE VALÈNCIA

DSIC
DEPARTAMENTO DE SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

DISEÑO DE UN CUADRO DE MANDO INTEGRAL PARA GESTIÓN ÁGIL DE PROYECTOS DE DESARROLLO DE SOFTWARE

Trabajo Fin de Máster

Máster Universitario en
Ingenieria y Tecnología de Sistemas Software

Autor: Alejandra Rojas Ortiz

Tutor: Dr. Patricio Letelier Torres

Valencia, Julio de 2019

Agradecimientos

Este trabajo es el resultado de un esfuerzo incesante durante año y medio que me ha permitido crecer técnicamente en la ingeniería del software y los sistemas de información, profundizando en las metodologías ágiles, lo que me apasiona.

Quisiera hacer mención de las personas que me han apoyado durante este proceso y que han colaborado con mi formación profesional.

En primero lugar quiero agradecer a mi director de tesis, Patricio Letelier, por guiarme y ser paciente conmigo. Me siento muy afortunada de haber tenido la oportunidad de realizar un trabajo de fin de máster en un área que, sin dudas, ha sido mi especialización profesional desde que empecé mi carrera laboral. Gracias Patricio por las enseñanzas y consejos, y por permitirme aprender de su vasta experiencia en el campo de las metodologías ágiles y el software.

A mis padres, mis hermanos y familia les agradezco por brindarme su apoyo incondicional en los momentos más difíciles de este proceso. Esto no sería posible sin ustedes.

Y en último lugar quisiera agradecer a Intellisys D. Corp, por darme la oportunidad de crecer como profesional en este ámbito a través de los años. La experiencia obtenida trabajando con ustedes ha sido invaluable a la hora de realizar este trabajo de fin de máster.

Tabla de Contenido

Capítulo 1. Introducción

1.1 Motivación

Al hacer seguimiento de un proyecto de software es difícil decidir qué métricas serán efectivas y verdaderos indicadores del desarrollo que se está realizando para crear un producto o dar mantenimiento al mismo.

¿Qué debemos medir? ¿Cómo debemos hacerlo? ¿Existen ya herramientas que lo hagan por nosotros? ¿Alcanzaremos la fecha de entrega final? Estas son preguntas que los directores de proyecto se hacen. Y algunos, al encontrar un volumen de información masivo y variado, se dan por vencidos y dejan a un lado los beneficios de un seguimiento cuantitativo de proyectos.

En el ámbito del desarrollo de software los requisitos cambian al mismo tiempo que existen plazos ajustados y exigentes que se deben satisfacer. Esto hace que desarrollar software no sea una tarea mecánica. Los equipos se enfrentan a un gran número de desafíos que pueden volver procesos eficaces en ineficaces si no son ajustados oportunamente.

En el enfoque ágil no existe un estándar que sea supervisado por un organismo principal. Existe una cantidad numerosa de métodos ágiles y mucha información disponible que es interpretada de forma diversa. En cuanto a seguimiento se refiere, no existe un consenso respecto de cuales métricas deberían utilizarse.

Si existiera una guía a su alcance que brinde soluciones de seguimiento a los proyectos se podrían tomar oportunamente algunas acciones correctivas antes de que los problemas pongan en riesgo los objetivos del sprint o del proyecto como un todo.

No se trata de la cantidad de datos sino de la calidad de estos y de la información que nos aportan.

1.2 Objetivos

Con el fin de proporcionar una solución general para el seguimiento ágil de proyectos, en este trabajo de fin de máster se plantean los siguientes objetivos:

1. Analizar los diferentes tipos de gráficas e indicadores que actualmente son ofrecidos en herramientas populares orientadas al seguimiento de proyectos que se desarrollen con metodologías ágiles.
2. Crear ejemplos de las gráficas ofrecidas por las herramientas que simulen diferentes escenarios del desarrollo de software con el propósito de seleccionar las gráficas que tengan una mayor utilidad y versatilidad con respecto a los diferentes tipos de proyectos de software que implementan metodologías ágiles
3. Diseñar un cuadro de mando y dashboard con las gráficas seleccionadas con la finalidad de ofrecer a los equipos que implementan metodologías ágiles populares una solución más global para el seguimiento de proyectos de software.

1.3 Estructura de la memoria

A continuación, se presenta la estructura de este trabajo de fin de máster:

El capítulo 1 presenta la introducción de este trabajo de fin de máster que incluye la motivación del mismo y los objetivos a alcanzar. Siendo el objetivo más importante, el desarrollo de una propuesta de Cuadro de Mando.

En el capítulo 2 resume la historia de los cuadros de mando. Haciendo mención del "The Balanced Scorecard Institute"[1] y el trabajo de Dr. Robert Kaplan sobre el desarrollo de los cuadros de mando y su impacto en las compañías.

En el capítulo 3 se introducen los métodos tradicionales escogidos por la industria para monitorear el desarrollo de proyectos. Haciendo especial énfasis en las guías del "Project Management Institute"[2] para los Project Managers.

El capítulo 4 define de forma breve las siguientes metodologías ágiles: Scrum, Lean, Kanban y Extreme Programming (XP).

En el capítulo 5 se explica el concepto de seguimiento con algunas preguntas. Entre ellas ¿Qué es hacer seguimiento? y ¿Por qué es esencial para la gestión de proyectos? También se mencionan los problemas ocasionados si no se hace seguimiento a un proyecto de forma correcta.

En el capítulo 6 se define qué es una métrica y las propiedades que la hacen apta para incluirla en el monitoreo de proyectos de desarrollo de software.

El capítulo 7 describe de forma breve las siguientes herramientas: JIRA, Axosoft, Targetprocess, Mingle, y TUNE-UP Process.

En el capítulo 8 se analizan y describen las diferentes gráficas ofrecidas por las herramientas escogidas para el seguimiento de proyectos de software.

El capítulo 9 presenta las opciones actuales en las herramientas escogidas para Dashboards a nivel de proyecto y Cuadros de Mando que evalúan el conjunto de proyectos en los que se esté trabajando.

[1] https://www.balancedscorecard.org/
[2] https://www.pmi.org/

En el capítulo 10 se describe la propuesta de cuadro de mando como resultado del análisis hecho de las diferentes gráficas evaluadas en el capítulo 8.

El capítulo 11 concluye este trabajo de fin de máster mediante reflexiones del trabajo realizado y su impacto en los equipos que implementan metodologías ágiles y que hacen uso de herramientas para la gestión de sus proyectos. Asimismo, se hace mención del posible trabajo futuro como resultado de la tesis.

Capítulo 2. Introducción a Cuadros de Mando

Parte de información del capítulo presente ha sido resumida desde la página oficial del "Balanced Scorecard Institute" [18].

Según "The Balanced Scorecard Institute" (BSI), el Cuadro de Mando "The Balanced Scorecard (BSC)", fue desarrollado por el Dr. Robert Kaplan de la Universidad de Harvard como un framework para medir la productividad de una organización, usando un set de medidas de rendimiento más balanceadas. Tradicionalmente las compañías usaban solo medidas financieras de rendimiento a corto plazo como indicador de éxito. El cuadro de mando introdujo medidas estratégicas no financieras a la mezcla a fin de enfocarse mejor en el éxito a largo plazo. El sistema ha evolucionado al pasar de los años y es ahora considerado un sistema integrado y estratégico para administración de proyectos.

En la página oficial del BSI se argumenta que la frase "Cuadro de mando" fue usada en sí a principios de los noventa, pero la raíz de este tipo de enfoque es profunda, e incluye el trabajo pionero de General Electrics en el reporte de medidas de desempeño en los años cincuenta y el trabajo de ingenieros franceses (quienes crearon "Tableau de Bord", literalmente un Dashboard o tablero de medidas de desempeño) a principios del siglo XX.

El BSI resume que este nuevo enfoque de gestión estratégica fue detallado en una serie de artículos y libros de Drs. Kaplan y Norton y construido por Art Schneiderman en "Analog Devices". Reconociendo algunas de las debilidades y ambigüedad de los enfoques de gestión previos, el cuadro de mando provee una prescripción clara sobre qué cosas deben medir las compañías para que puedan balancear su perspectiva financiera.

Más precisamente en las palabras de Kaplan y Norton, el cuadro de mando integral mantiene las medidas financieras tradicionales. Pero las medidas financieras cuentan la historia de los eventos

pasados, una historia adecuada para compañías de la época industrial, para las cuales inversiones en las capacidades a largo plazo y las relaciones con los clientes no eran cruciales para el éxito. Estas medidas financieras son inadecuadas, para guiar y evaluar el viaje que las compañías de la época de la información que deben hacer para crear valor futuro a través de la inversión en clientes, suplidores, empleados, procesos, tecnología e innovación.

Según el BSI los motivos por los cuales muchas organizaciones usan cuadros de mando son los siguientes:

- Comunicar las metas que están tratando de lograr.
- Alinear el trabajo del día a día de sus empleados con su estrategia.
- Priorizar proyectos, productos, y servicios.
- Medir y monitorear el progreso de los objetivos estratégicos.

El BSl en la figura 1 muestra que el sistema permite conectar varios elementos de la compañía. Entre ellos:

- Aspiraciones.
- Propósito.
- Valores fundamentales.
- Áreas de enfoque estratégicas.
- Temas, resultados y objetivos.
- Actividades de mejoramiento continuo.
- Medidas tomadas para el mejoramiento continuo.
- Indicadores de rendimiento.
- Metas con respecto a productividad.
- Iniciativas para alcanzar esas metas.

Esto quiere decir que hay una conexión visible entre:

- Los proyectos y los programas en los que los empleados trabajan.
- Las métricas usadas para dar seguimiento.
- Los objetivos estratégicos que la organización está tratando de alcanzar.
- La visión, misión, y estrategia de la organización.

Figura 1. Conexión entre proyectos y propósitos de la compañía (traducido desde www.balancedscorecard.org)

A través del cuadro de mando, el BSI sugiere que se vea la organización desde cuatro perspectivas, y se desarrollen objetivos, medidas (KPIs), metas, e iniciativas relativas al punto de vista financiero, del cliente y los *Stakeholders*, procesos internos (calidad y eficiencia), y capacidad de organización (capital humano, infraestructura, tecnología, cultura, entre otros).

Las cuatro perspectivas son las siguientes:

1. Crecimiento y aprendizaje de los empleados: la organización debe ser capaz de innovar y refinar eficiencias para poder iniciarse en nuevos mercados e incrementar sus márgenes. La fundación para las demás perspectivas son las capacidades y habilidades de los empleados y los sistemas de información.
2. Procesos internos: procesos que tienen el mayor impacto en la satisfacción del cliente, y en las competencias principales de la empresa. Centrarse en la innovación y el mejoramiento incremental.
3. Satisfacción del cliente: la organización debe hablar con sus clientes para entender sus expectativas, y medir resultados en relación con las mismas. Entre ellas se encuentran el tiempo de respuesta, retención, y adquisición.
4. Financiera: el rendimiento financiero debe ser el resultado lógico de hacer bien lo fundamental. Estos resultados son los más fáciles de medir, pero al mismo tiempo limitan, ya que tienden a enfocarse en los resultados a corto plazo.

2.1 KPI

Un factor importante a tener en cuenta para diseñar un cuadro de mando son los KPI (Key Performance Indicators). A continuación, una breve descripción de su significado y los parámetros que deben cumplir para ser considerados útiles.

Los KPI indican el estado de progreso de un resultado deseado. Estos ayudan a monitorear la implementación y efectividad de las estrategias de una organización, determinan la brecha entre el

rendimiento actual y al que se quiere llegar, y determinar la efectividad de la organización y la eficiencia operacional [31]. Según el "Strategy Management Group" (SMG) para que un KPIs sea útil debe:

- Proveer una manera objetiva de ver si la estrategia escogida está funcionando.
- Ofrecer una comparación que calibre el grado de cambio de rendimiento a medida que va pasando el tiempo.
- Enfocar la atención de los empleados en lo que importa más para ser exitosos.
- Permitir medida de logros, no solamente el trabajo realizado.
- Proveer un lenguaje común de comunicación.
- Ayudar a reducir la incertidumbre intangible.

2.2 Beneficios de la implementación de un Cuadro de Mando

En esta sección, se presentan los beneficios más destacados al implementar un cuadro de mando según el BSI.

Los dos beneficios más destacados son:

- Ofrecer una visión amplia que englobe aspectos que nos permitan detallar el progreso de nuestros objetivos, al igual que el desarrollo óptimo de la empresa.
- Al proporcionar de forma visual la evolución de la empresa, permite planificar estrategias de largo y corto plazo. Dando la información necesaria que es vital para la rápida y consciente toma de decisiones.

Capítulo 3. Gestión Tradicional de Proyectos

El PMI (Project Management Institute) es una organización profesional estadounidense sin ánimos de lucro para la gestión de proyectos. Esta organización provee servicios como el desarrollo de estándares, investigación, educación, publicación, oportunidades de networking locales, conferencias y seminarios de entrenamiento, y ofrecen acreditaciones en administración de proyectos [20].

Para dar seguimiento a lo largo de sus proyectos los directores de proyecto utilizan técnicas que permiten mantener al tanto a todos los interesados sobre el estado del proyecto, las medidas que se han tomado y las proyecciones con respecto al presupuesto, cronograma y alcance.

Este seguimiento consiste en medir, recopilar y compartir la información que sea relativa al desempeño. También se evalúan las mediciones y se observan las tendencias que permiten implementar mejoras en diferentes fases del proceso. Al hacer eso de una manera continua, el equipo está al corriente del estado de todas las áreas, y puede redireccionar esfuerzos a tareas que sean de prioritarias o que de cierta forma estén causando retrasos en el proyecto [19].

Según el PMBOK, el director de proyecto se ocupa de los siguientes aspectos para dar seguimiento:

- Comparar el desempeño real del proyecto con respecto al plan para la dirección del proyecto.
- Evaluar el desempeño para determinar la necesidad de una acción preventiva o correctiva y en su caso recomendar aquellas que se consideran pertinentes.
- Identificar nuevos riesgos y analizar, revisar y monitorear los riesgos existentes del proyecto, para asegurarse de que se identifiquen los riesgos, se informe sobre su estado y se implementen los planes apropiados de respuesta a los riesgos.
- Mantener, durante la ejecución del proyecto, una base de información precisa y oportuna relativa al producto o a los productos del proyecto y a su documentación relacionada.

- Proporcionar la información necesaria para sustentar el informe de estado, la medida del avance y los pronósticos.
- Proporcionar pronósticos que permitan actualizar la información relativa al costo y al cronograma actuales.
- Dar seguimiento a la implementación de los cambios aprobados cuando éstos se producen.
- Informar adecuadamente sobre el avance del proyecto y su estado a la dirección del programa, cuando el proyecto forma parte de un programa global.

Con respecto a las herramientas y técnicas que se utilizan para el seguimiento del proyecto en el PMBOK se presentan las siguientes:

- Juicio de expertos: los expertos pueden interpretar más a fondo la información proporcionada por los procesos de monitoreo y control. El director de proyecto y el equipo son capaces de determinar los pasos a seguir para asegurar el éxito del proyecto. No sólo a nivel de expectativas del cliente, sino también a nivel de rendimiento.
- Técnicas Analíticas: algunas técnicas que se utilizan son análisis de regresión, métodos de clasificación, análisis casual, análisis de causa raíz, métodos de pronóstico, análisis de modos de fallo y efectos, análisis de árbol, análisis de reservas, análisis de tendencias, gestión del valor ganado, y análisis de variación.
- Sistemas de información: estos sistemas proporcionando acceso a herramientas de programación, indicadores de rendimiento, bases de datos, información financiera, y costos y recursos durante la vida del proyecto.
- Reuniones: estas pueden incluir a *Stakeholders*, miembros del equipo, grupos de usuarios, entre otros.

En las próximas secciones se describirán gráficas que tradicionalmente han sido usadas para el seguimiento de proyectos.

3.1 Diagrama Gantt

El diagrama Gantt, es una forma visual de representar el trabajo que ha sido planeado para la duración de un proyecto como se puede ver en el ejemplo de la figura 2. Estos diagramas proporcionan una visión a corto y largo plazo del proyecto.

La idea básica de un diagrama Gantt según Wong [21], es que se muestre el plan del Proyecto, las dependencias entre tareas, y las fechas límites en un solo lugar. Un gráfico básico consiste en tareas y fechas. La estructura de las tareas se encuentra a la izquierda del gráfico y las fechas en la parte de arriba. Cada tarea está listada, una por una, empezando con la primera tarea programada en la parte superior. La fecha de inicio de la tarea es representada por el comienzo de la barra dibujada debajo de la fecha listada correspondiente en la parte de arriba del gráfico. Esta barra empieza al principio y se extiende a la fecha en que se supone será finalizada.

Al incluir dependencias, el gráfico proporciona más información que puede resultar muy beneficiosa.

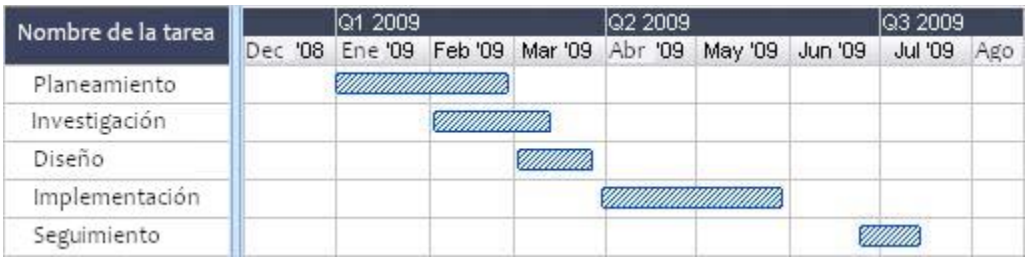

Figura 2. Ejemplo de Diagrama Gantt (traducido desde www.gantt.com)

Sin embargo, ¿tendría lugar un Diagrama de Gantt en proyectos que implementen de métodos ágiles?

Dados la naturaleza cambiante de los proyectos de software a lo largo de su desarrollo, un diagrama de Gantt puede ser poco práctico.

Letelier [32] expone las siguientes debilidades:

- Las tareas técnicas suelen ser planificadas de forma que su duración coincida con los plazos deseados, ya sea que el tiempo de duración se alargue, se acorté o se elimine por completo la tarea.
- La duración de las tareas es específica. Se asignan fechas de inicio y fin sin tener en cuenta qué tan incierta pueda ser su estimación.
- Se asigna tareas a realizar a los participantes en fechas específicas sin considerar posibles cambios o impedimentos que se presenten.
- La relación entre tareas revela la implementación de un modelo "En Cascada". Es decir, no hay iteraciones donde se puedan hacer cambios o se contemple re-trabajo.
- La actualización de un plan que no vaya de acuerdo a lo estimado requiere de gran esfuerzo, pues a medida que no se alcancen hitos el desajuste será en cascada.

Una de las características más relevante de un proyecto de software es la incertidumbre. Establecer fechas puntuales de inicio a fin y asignar todas las tareas de un proyecto por anticipado a sus participantes no es conveniente, pues los cambios en un proyecto de software son frecuentes. Al no tener la posibilidad de implementar dichos cambios en un proyecto, “la elaboración típica de un Diagrama de Gantt, lo hace incompatible con el enfoque ágil" [32].

3.2 Línea de tiempo

Las líneas de tiempo de un proyecto muestran una secuencia tecnológica de los eventos sin mayores detalles, como asignaciones por equipo, y limitación de recursos como se puede observar en la figura 3. Las líneas de tiempo se prestan para conversaciones con S*takeholders* sobre una visión más global del proyecto.

Bunner [25] argumenta que el tamaño y la estructura de una línea de proyecto depende del mismo. Puede ser muy detallada, con cientos de tareas y subtareas, o muy simple, solamente mostrando unos pocos entregables y las fechas límites. Sin importar como se construya, la línea de tiempo de un proyecto debe capturar la siguiente información clave:

- La lista de tareas a completar
- Las fechas en las cuales las tareas deben ser completadas.
- La duración estimada de cada tarea.

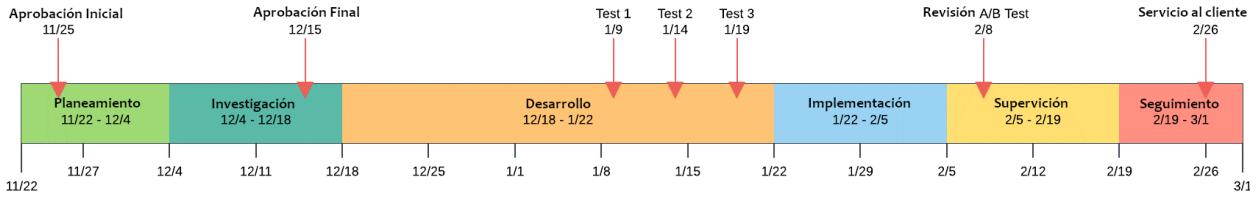

Figura 3. Ejemplo de línea de tiempo (traducido desde lucidchart.zendesk.com)

3.3 Diagramas de flujo

Diagramas de flujo o diagramas de red muestran las tareas de un proyecto en el orden que necesitan ser completadas. Incorporan más detalles que las líneas de tiempo, y se puede hacer seguimiento de las dependencias al igual que de detalles como el número de identificación de una tarea y los recursos que están dedicados a ella. Sin embargo, si se agrega demasiada información al mismo, pueden ser difíciles de leer y no proveer una imagen clara del proyecto.

Su uso es recomendado para proyectos pequeños y simples, con tareas que son secuenciales como el ejemplo mostrado en la figura 4.

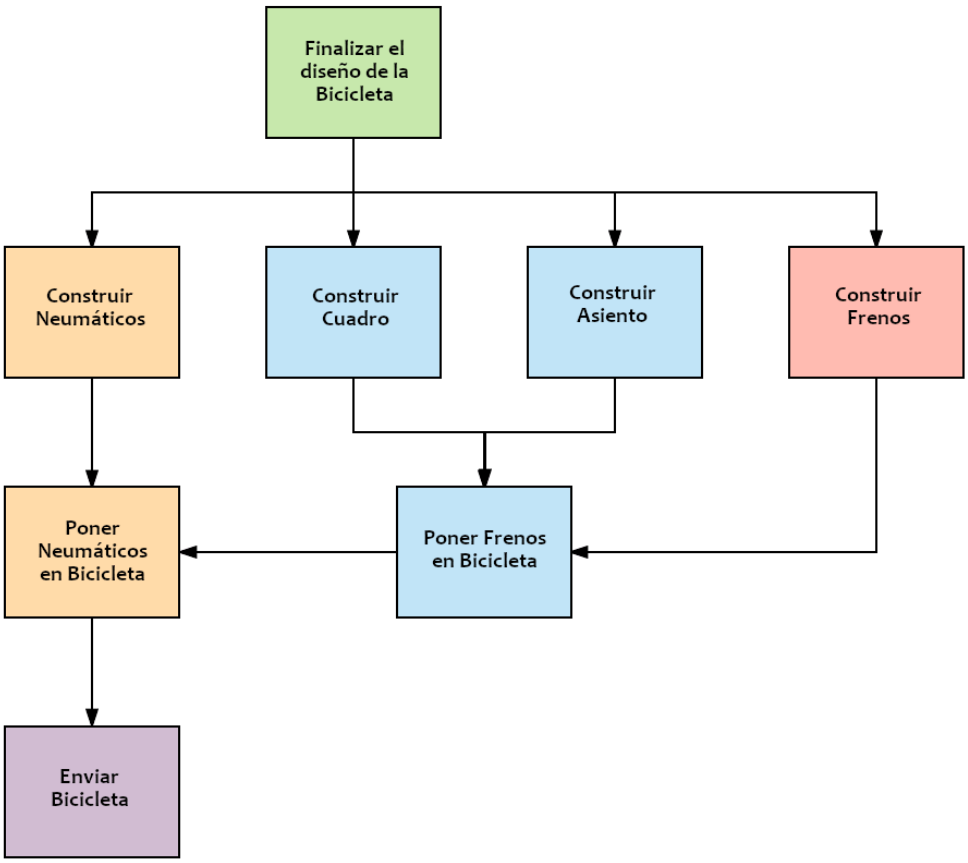

Figura 4. Ejemplo de diagrama de flujo (traducido desde lucidchart.zendesk.com)

3.4 PERT

Parte de información de esta sesión ha sido resumida desde el curso " Agile Planning for Software Products " impartido por Kenny Wong [21].

Wong define PERT (Program Evaluation and Review Technique) como una representación visual del Proyecto con nodos y aristas. Las aristas son los bordes que conectan los nodos. En un gráfico PERT los nodos representan hitos. Cada uno de estos nodos representa algo que está sucediendo, usualmente el desarrollo de una parte del producto, o algún evento. Las tareas están representadas por las aristas. La dirección de las flechas representa el orden en que las tareas deben ser completadas.

Una secuencia de tareas es representada por un camino a lo largo de las aristas siguiendo las flechas. Cada arista es denotada con la tarea y el tiempo correspondiente estimado. Cuando hay múltiples caminos independientes que llegan a un nodo, es una indicación de que las tareas pueden ser llevadas a cabo en paralelo. Según Wong, esto significa que los caminos paralelos pueden ser completados al mismo tiempo si existe la cantidad de recursos suficientes, ya que no hay dependencias entre las tareas a través de los caminos. Alternativamente, cuando hay múltiples caminos que llegan a un mismo nodo, esto indica que los caminos deben sincronizarse. Es decir, todos los caminos deben ser completados antes de poder proceder al siguiente camino que va fuera de ese nodo.

El camino crítico es representado de forma horizontal a través del gráfico. Todos los otros caminos pueden ser dibujados arriba o debajo del camino crítico.

El camino crítico calcula el mínimo de tiempo que tomaría completar el proyecto. Generalmente, se toma más tiempo completar el camino crítico que cualquier otro. Lo que permite cierta flexibilidad a la hora de programar tareas que no son parte de él.

En la figura 5 se puede observar un ejemplo de una gráfica PERT con un camino crítico y múltiples caminos más que han de ser completados para, en este caso, finalizar la pasta. El camino crítico en este caso es:

1. Empezar Pasta.
2. Preparar ingredientes.
3. Mezclar ingredientes.
4. Formar las albóndigas.
5. Hornear las albóndigas.
6. Agregar albóndigas al plato.
7. Finalizar pasta.

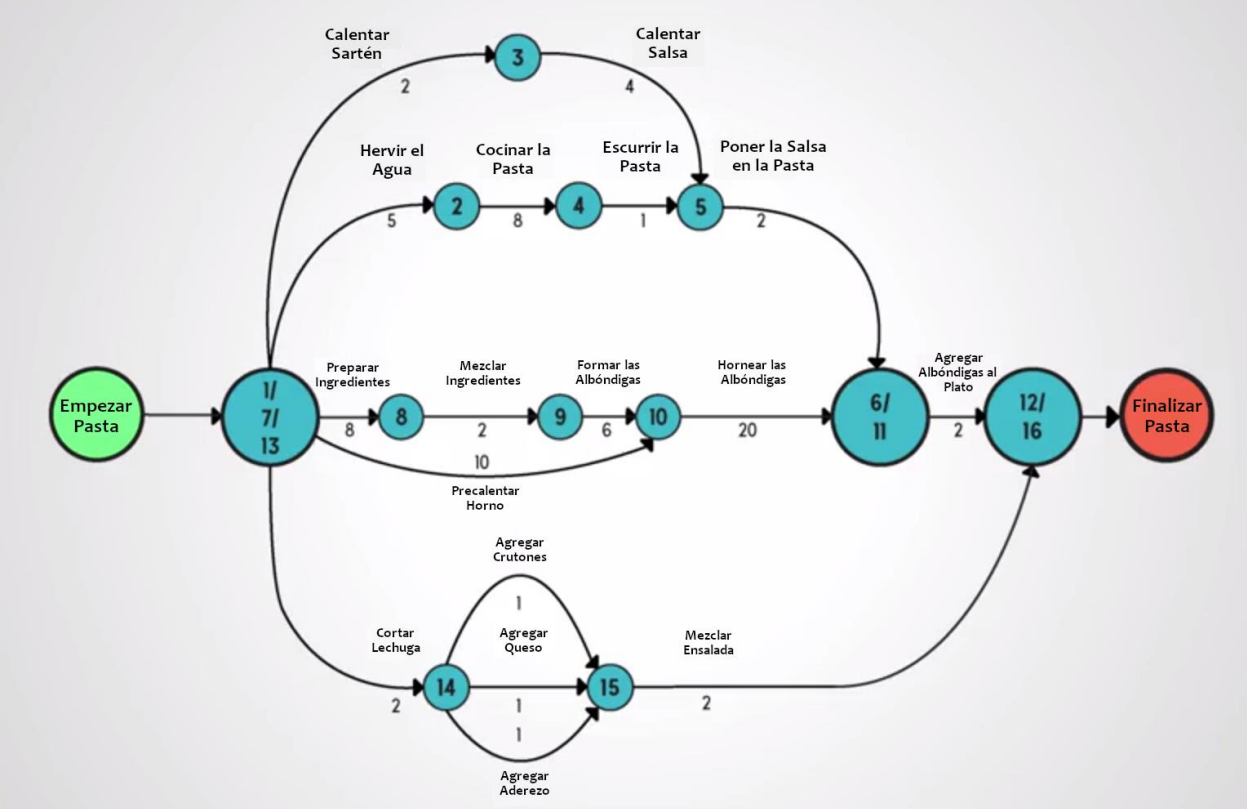

Figura 5. Ejemplo de PERT (traducido desde "Agile Planning for Software Products")

3.5 CPM

La gráfica del método de camino crítico, más conocida como gráfico CPM, es una forma visual de organizar las dependencias entre tareas.

Según Wong [21] se establece que primero se deben definir las tareas. Después se añaden estimaciones para cada una de estas tareas. Luego se empiezan a organizar las tareas dependiendo de sus dependencias. Las flechas representan la dependencia de tareas. Una tarea debe terminarse antes de empezar la siguiente. Las gráficas CPM tienen caminos, que se pueden definir como una secuencia que sigue las flechas que van de una tarea a otra. Las tareas que están en caminos diferentes pueden ser completadas independientemente.

Wong concluye que, a partir del gráfico se establece el camino crítico, es decir, el camino de mayor duración entre dos puntos lógicos. Con esta información se determinan los estimados de tiempo más cortos y más largos que tendrán lugar en el proyecto. Determina que tareas son críticas, y que tareas pueden ser aplazadas sin agregar tiempo adicional al proyecto. Además, el gráfico muestra las tareas que pueden ser llevadas a cabo en paralelo.

En la figura 6, se puede observar que en paralelo se puede cocinar la pasta, calentar la salsa, hacer las albóndigas y preparar la ensalada. Sin embargo, no se puede agregar las albóndigas al plato si primero no se ha mezclado la salsa con la pasta.

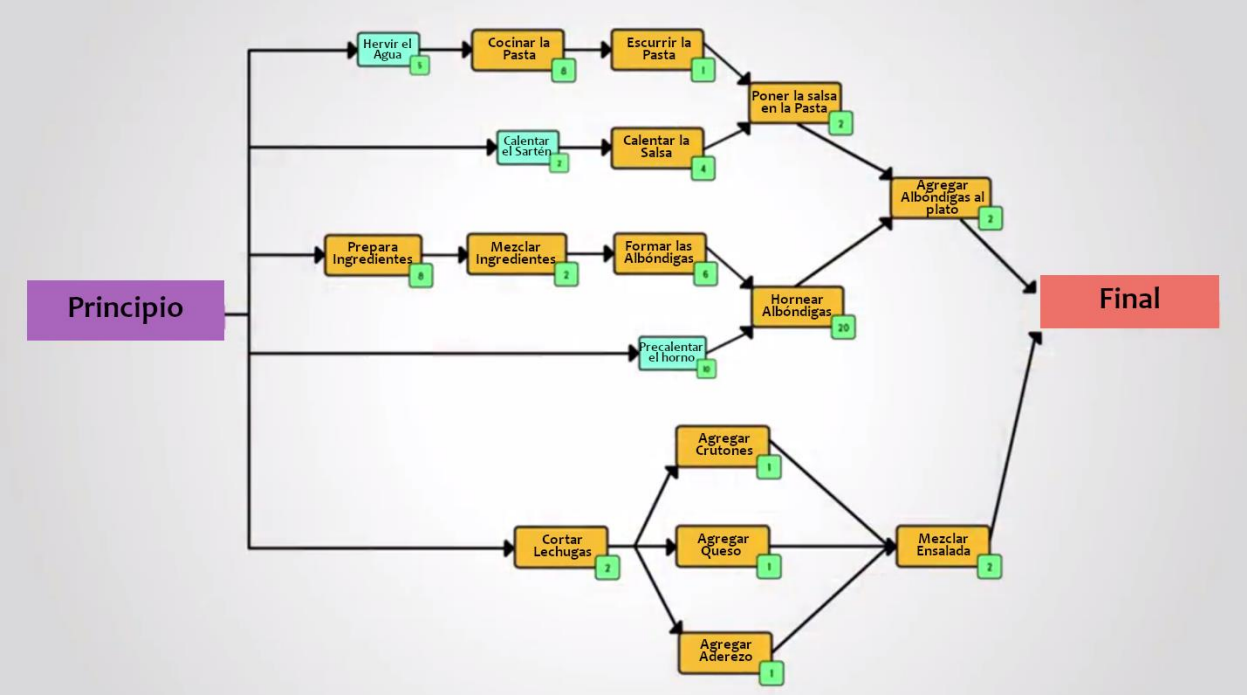

Figura 6. Ejemplo de CPM (traducido de "Agile Planning for Software Products")

3.6 Seguimiento tradicional

Generalmente el seguimiento tradicional va de la mano del modelo de procesos *Waterfall*. En esta modelo las fases de requisitos, diseño, implementación, verificación y mantenimiento son desarrolladas secuencialmente, es decir, cada fase debe completarse antes de empezar la siguiente y ninguna fase debe repetirse.

Thomas [36] argumenta que el seguimiento tradicional se enfoca en observar y medir el rendimiento real del proyecto según el plan establecido y "arreglar" este rendimiento que varía de lo planeado. Esta variación es considerada como negativa, pues metodologías tradicionales no es bienvenida la introducción de cambios en un proyecto ya iniciado ni se considera hacer correcciones de fallos en fases ya terminadas.

La desventaja más palpable de este tipo de seguimiento es que "se puede perder fácilmente el foco respecto del producto o servicio que se espera como resultado del proyecto al intentar cuadrar las actividades que deben realizarse, dentro de las restricciones de plazos y costos" [33].

Las herramientas de seguimiento de proyectos utilizadas por equipos deben servir como apoyo para el desarrollo del proyecto. Si para el seguimiento de un proyecto de software se utiliza cualquiera de los diagramas descritos en las sesiones anteriores, se generaría un esfuerzo extra al tener que actualizar los diagramas en su totalidad cada vez que se produzca un cambio. De esta forma la herramienta se convierte en una carga para el quipo en vez de una ayuda.

3.7 Cuadro de Mando Tradicional

La herramienta ProjectManager[3] ofrece seguimiento para proyectos de software que utilicen metodologías ágiles o el modelo de procesos *Waterfall.* Para esta sesión se hará enfoque en el Cuadro de Mando que se ofrece para proyectos de software que implementen el modelo de procesos *Waterfall.*

ProjectManager [34] expresa que el Diagrama de Gantt es la herramienta más importante para gestión de proyectos que emplean el modelo de procesos *Waterfall*, pero para más detalle se construye un Cuadro de Mando con widgets que monitorea la cantidad de trabajo, los costos y la diferencia entre lo planificado y el progreso actual.

El cuadro de mando de la figura 7 ha sido inspirado por los widgets de ProjectManager.

[3] https://www.projectmanager.com/

Figura 7. Cuadro de Mando seguimiento tradicional

Como se puede observar en la figura 7, la información de los widgets se enfoca en la finalización de tareas, en el tiempo empleado en el proyecto actual versus el planificado y el porcentaje del proyecto finalizado.

En el widget Resumen se muestra claramente que el Proyecto A está retrasado con respecto a su fecha de finalización y en el widget de Gráfico de Entregables muestra que la mayoría de sus tareas están atrasadas. A parte de esta información, el Cuadro de Mando no ofrece más detalle. Esta poca información y falta de detalle no permite que el equipo entienda la razón del retraso y la acumulación de tareas no finalizadas.

Capítulo 4. Gestión Ágil y Metodologías

La gestión ágil es un enfoque incremental e iterativo, que provee la habilidad de responder al cambio intentando evitar que el proyecto fracase. Consiste en: entregar los elementos que constituyen un proyecto de forma incremental, colaboración del equipo, planificación y aprendizaje continuo, etc. Las metodologías ágiles se basan en el "Manifiesto Ágil" [32], un grupo de principios que sirven como guía para un mejor desarrollo de software. Estos principios son los siguientes:

- Los individuos e interacciones son más importantes que los procesos y herramientas.
- El software en funcionamiento es más importante que documentación extensa.
- La colaboración con el cliente es más importante que la negociación de contrato.
- La respuesta al cambio es más importante que seguir un plan.

Esto no implica que los apartados de la derecha no sean importantes o que no se necesiten, sino que los de la izquierda agregan más valor al cliente, proyecto, y equipo, al final del día.

La gestión ágil, es adoptada como una mentalidad, ya que no todos los métodos y prácticas se pueden aplicar de manera exitosa para diferentes proyectos. Según Sutherland [22], ser ágil es aplicar las prácticas y métodos que se alineen con los principios del manifiesto y se adecuen a la situación actual de un proyecto.

En las siguientes secciones del capítulo se describirán de resumidamente las cuatro metodologías ágiles más comúnmente usadas por equipos de desarrollo de software.

4.1 SCRUM

Scrum es un framework para la gestión de proyectos ágiles basado en un proceso iterativo e incremental. Scrum ha ganado mucha popularidad en la comunidad de desarrollo de software ágil gracias a su simplicidad y adaptación al cambio.

En esta metodología, el "Product Owner" trabaja muy de cerca con el equipo para identificar y dar prioridad a las funcionalidades del sistema en forma de un "Product Backlog". El "Product Backlog" consiste en características, correcciones de errores, requisitos no funcionales, y otros items que se necesitan hacer para poder entregar de una manera exitosa un sistema de software funcional.

Los equipos se comprometen a realizar un número específico de items dentro den un sprint. El "Product Backlog" es analizado, organizado y priorizado continuamente por el "Product Owner" según las necesidades del proyecto.

4.2 LEAN Development

Es una metodología ágil desarrollada por Mary y Tom Poppendieck [26]. Está basada principalmente en los principios y prácticas del movimiento Lean, y las prácticas de compañías como Toyota. Esta metodología enfoca al equipo en entregar valor al usuario, y en la eficiencia del "Flujo de Valor", los mecanismos que hacen posible la entrega del mismo.

Mary y Tom Poppendieck [26] definen los principios principales de Lean como los siguientes:

- Eliminar desperdicio.
- Ampliar el aprendizaje.
- Decidir lo más tarde posible.

- Entregar resultados lo más pronto posible.
- Empoderar al equipo.
- Construir integridad.
- Tener una visión global.

La metodología Lean elimina desperdicios a través de prácticas como, seleccionar sólo las características que agreguen más valor al sistema, dar prioridad a las mismas, y entregándolas en pequeños lotes. Enfatiza la velocidad y eficiencia del flujo de trabajo de desarrollo, se basa en feedback rápido y confiable entre programadores y clientes. Lean se enfoca en la autoridad y habilidad de los individuos y equipos pequeños en tomar decisiones, ya que investigaciones demuestran que es más rápido y eficiente que el flujo de control jerárquico. Lean también se concentra en la eficiencia del uso de los recursos de un equipo, tratando de asegurar que todos sean tan productivos como sea posible. Se concentra en trabajo concurrente y generar la menor cantidad de dependencias en el flujo de trabajo de un equipo.

4.3 KANBAN

El método Kanban es utilizado por organizaciones para manejar la creación de proyectos con un énfasis en entrega continua mientras no se sobrecargue al equipo de desarrollo. Al igual que Scrum, Kanban es un proceso diseñado para ayudar a los equipos a trabajar juntos de una manera más eficiente.

Según Colb y Scotcher [27], existen cinco pasos claves para implementar Kanban:

- Visualización del flujo de trabajo: representación visual del progreso de las unidades de trabajo desde su estado inicial (“to-do”) hasta su final (“done”).
- Limitar el trabajo en progreso: Kanban limita el número de unidades de trabajo que se desarrollan al mismo tiempo para asegurar eficiencia.
- Administrar el flujo de trabajo: el objetivo es que el paso de las unidades de trabajo entre estados sea rápido y fluido.
- Hacer el proceso explícito: debe haber un control al final de cada paso con reglas claras para mover las unidades de trabajo al siguiente.
- Mejorar la colaboración: una vez el flujo de trabajo está establecido, surgen ideas de cómo mejorarlo.

4.4 Extreme Programming (XP)

XP fue originalmente descrita por Kent Beck, y ha surgido como unas de las metodologías ágiles más controversiales y populares. XP es un enfoque disciplinado en producir software de alta calidad de una forma rápida y continua. Además, promueve la participación del cliente, *feedback* rápido, pruebas continuas, planificación continua, y trabajo en equipo para desarrollar software funcional en intervalos frecuentes.

Según Wong [28], XP se basa en cinco valores, comunicación, simplicidad, *feedback*, respeto y valentía. Y adicionalmente se apoya en las siguientes prácticas:

- Juego de planificación.
- Pequeños lanzamientos.
- Pruebas de aceptación definidas por el cliente.

- Diseño simple.
- Programación en parejas.
- Desarrollo guiado por pruebas.
- Refactorización de código.
- Integración continua.
- Propiedad colectiva del código.
- Estándares de programación.
- Metáfora.
- Ritmo sostenible.

En XP, el cliente trabaja muy de cerca con el equipo de desarrollo para definir y darle prioridad a las unidades de trabajo. El equipo de desarrollo estima, planea, y entrega en iteraciones las unidades de trabajo con mayor prioridad de forma que el software haya sido probado y esté funcionando adecuadamente.

4.5 Seguimiento Ágil

Las metodologías ágiles presentan más oportunidades de hacer seguimiento e intervenir en un proyecto que las metodologías tradicionales.

Según Steve Thomas [36], el seguimiento ágil se centra en lo siguiente:

- Entrega frecuente: la mejor evidencia de que un proyecto va por buen camino es que el software funcione y sea desplegado en producción.

- Co-localización: el equipo y *stakeholders* deben estar en la misma área para trabajar juntos diariamente.
- Daily meeting: reunión corta y diaria para saber si el progreso es el esperado y exponer posibles impedimentos.
- Revisiones: reunión en la que se hace una demostración del producto y una retrospectiva donde se reflexiona sobre la efectividad del equipo,

Estás guías generales pueden ser implementadas por cualquier proyecto de software que implemente metodologías ágiles. Sin embargo, no existe un consenso con respecto a qué tipo de diagramas y métricas se deben generar para el seguimiento de proyectos.

Esta falta de estandarización facilita que se generen diversas interpretaciones sobre cómo se debería hacer seguimiento a un proyecto software que use metodologías ágiles.

Capítulo 5. ¿Qué significa hacer seguimiento?

Según Wong [3], hacer seguimiento se refiere a revisar y evaluar el producto y el proceso de desarrollo de software. Es esencial para obtener un producto correcto, hecho bien y gestionado adecuadamente. Hacer seguimiento es importante porque permite que todos los involucrados sepan el estado del proyecto en cualquier momento.

Las métricas son usadas en desarrollo de software para medir varios aspectos de un producto, proceso, o proyecto. Las métricas proveen datos cuantificables y los resultados para lograr esto. En realidad, son el medio para el seguimiento. Sin embargo, es importante recordar que no todos los aspectos importantes de un proyecto son cuantificables, y no todo lo que se puede cuantificar es necesariamente importante para el proyecto.

Tanto la acción de hacer seguimiento como las métricas ayudan a los directores de proyectos y equipos de desarrollo a obtener *feedback*. *Feedback* es información o críticas que pueden ser usadas para identificar mejoras en un producto o un proceso, o para afirmar que el proyecto va por el camino correcto. De hecho, el *feedback* puede ser usado para mejorar o verificar que no solo el producto o el proceso, sino también la comunicación u otras prácticas usadas en el proyecto. El *Feedback* puede ser cualitativo o cuantitativo. Cálculos, resultados de estudios de usuario, sugerencias del equipo o los clientes, y las reacciones de los clientes, son todos diferentes formas de *feedback*. El feedback es de gran importancia para las metodologías ágiles ya que están comprometidas a mejorar constantemente los proyectos y productos tan frecuente como sea posible [3].

En las siguientes secciones del capítulo se exponen otros aspectos a considerar de la monitorización de los proyectos de desarrollo de software y cómo influyen en el equipo, producto y cliente final.

5.1 ¿Por qué es esencial hacer seguimiento al progreso de un proyecto?

Según Wong [3], para asegurar el éxito de un proyecto, se debe dar seguimiento, analizar y revisar activamente el progreso.

Un equipo de programadores se puede encontrar trabajando en un proyecto donde su fecha límite se acerca y aún no está completada la mayor parte del proyecto. Para lograr finalizarlo a tiempo, se implementan "estrategias" como programar sin cesar y agregar nuevos miembros al equipo. Tras numerosas horas extras, el equipo llega al quiebre, no finaliza el producto o lo entrega con múltiples fallos. Ante este resultado, un cliente no estará satisfecho y naturalmente culpará al director de proyecto. ¿Cómo suceden estas situaciones si al inicio todo iba tan bien?

El simple hecho de que el equipo esté ocupado no garantiza que se esté progresando de forma real. Wong argumenta que, al hacer seguimiento, el progreso se pueden introducir cambios y adaptaciones para poder cumplir fechas límite. Uno de los objetivos de hacer seguimiento a la productividad de los programadores es visualizar en qué aspectos cada integrante del equipo puede mejorar.

Una de las métricas de hacer seguimiento que se usa en Scrum, es la velocidad. Esta métrica es una de las formas con las que fácilmente se puede identificar cuando algo no va bien y aplicar los cambios correspondientes para solucionar el problema. Esto no solo mantendrá el proyecto por buen camino y mejorará su calidad cuando ya sea entregado al cliente, sino también subirá la moral del equipo proporcionándoles transparencia en cuanto al estado del proyecto.

La transparencia significa que todos los miembros del equipo conocen el estado del proyecto, y no solo los directores de proyectos. Hacer seguimiento al proyecto puede ser una forma de unificar al equipo de desarrollo y el director de proyecto al ver esta tarea de una forma positiva y al enfocar a todos los integrantes en una meta común.

La idea central de hacer seguimiento el progreso es entregar el producto a tiempo y según las especificaciones dadas. Además de que ayuda a desarrollar un mejor software y tener clientes más satisfechos.

5.2 Problemas de Seguimiento

Existen diferentes formas de hacer seguimiento al progreso y productividad de un proyecto. Sin embargo, también existen problemas relacionados con el seguimiento de proyectos como, saltarse las métricas y utilizar unas que no sean efectivas o adecuadas para proyecto. Wong [3] detalla los problemas a continuación:

5.2.1 No utilizar métricas

Muchos proyectos de software evaden medir o utilizar métricas en sí. Existe muchas razones que los llevan a hacer esto. No hacerlo puede ser problemático, pues no hay forma de identificar problemas que pudieron haber sido identificados a tiempo, y solucionados para evitar el fracaso de un proyecto.

- Falta de tiempo: los programadores tienen un tiempo limitado para programar, así que cualquiera tarea que no esté relacionada con esto, como son las métricas, se ve como un impedimento para llevar el proyecto acabo. Este problema en particular es complicado en un enfoque ágil. Las metodologías ágiles se concentran en desarrollo rápido y eficiente, de modo que usar métricas que no sean automáticas puede hacer pensar al equipo ágil que se está perdiendo el tiempo.
- Falta de estándar en la industria: existen muy pocos estándares en la industria porque sus estudios no son concluyentes o no pueden ser generalizados. Algunos estudios sugieren que las métricas funcionan mejor en ciertas situaciones o solamente en ciertas organizaciones.

Pero la cantidad de métricas y la falta de estándares hacen difícil la selección de las métricas adecuadas para un proyecto.

- Falta de conocimiento en la dirección de proyectos de desarrollo: es el problema más frecuente asociado a no usar métricas o no interpretarlas correctamente.

5.2.2 ¿Qué estás midiendo?

Wong [3] expone que otro problema muy común es el uso de métricas ineficaces. Argumentando que generalmente se calculan métricas que no terminan siendo útiles para el proyecto o que no informan de forma efectiva acerca de su estado. El mal uso indica que hay una falta de entendimiento de por qué ciertas cosas deben ser medidas.

Wong presenta como ejemplo la métrica conocida como líneas de código, o LOC. Esta métrica hace seguimiento al número de líneas de código escritas para una parte del software. Este número es usado para generar datos relacionados al proyecto. Los datos pueden incluir información, como el precio de una línea de código o el tiempo que toma escribir una línea de código para el proyecto.

Wong explica que esta métrica es problemática y no fiable. Debido a que diferentes partes del proyecto, o la variación entre un proyecto y otro puede ser drástica, llevando a números de líneas muy diferentes. Los lenguajes de programación también pueden influenciar en el número de líneas que se necesitan para desarrollar una misma funcionalidad.

Esto hace no recomendable usar las líneas de código como un medio de valoración del proyecto porque las circunstancias impiden que los números sean significativos o comparables, incluso para el mismo lenguaje de programación.

Las métricas también son susceptibles al mal uso. Por ejemplo, muchas medidas pueden cambiar el comportamiento de los programadores. Una vez que se sabe cuál es la medida que se está tomando,

es parte de la naturaleza humana querer maximizar el número. Esto puede llevar a los desarrolladores a escribir más líneas de código de las que escribiría si no supieran acerca de la métrica. Enfocarse en escribir más puede llevar a un descenso de la productividad, dado a que la meta cambia, ahora es crear más cantidad de líneas que entregar el producto. Por esta razón es importante entender por qué se ha decido escoger una métrica en sí.

"No todo lo que puede ser contado cuenta, y no todo lo que cuenta puede ser contado" – William Bruce Camenron [3]. Esta frase permite resumir porque es importante en un proyecto ser consciente de no cuantificar absolutamente todo. Las métricas se vuelven inefectivas cuando son usadas para cuantificar cosas que no necesitan serlo.

Capítulo 6. Métricas

A lo largo de este trabajo de fin de máster se utiliza el término métrica en varios capítulos, pues la elección de las métricas correctas es fundamental para el seguimiento de proyectos de software. En este capítulo se definirá el concepto de métrica y sus propiedades. Parte de información de este capítulo ha sido resumida desde el curso "Reviews & Metrics for Software Improvements" impartido por Kenny Wong [3].

Una *métrica* es la combinación de dos o más medidas que proveen un resultado significativo. Por ejemplo, kilómetros por hora (km/h) es un ejemplo de una métrica. Kilómetros y horas son dos medidas que cuando son combinadas son significativas. No todas las métricas son significativas, ya que casi todo puede ser medido.

Un *indicador* es una medida que atrae la atención de la persona que está tomando las medidas. El valor del indicador nos informa si los números indican algo bueno o malo. Por ejemplo, la temperatura normal de una persona está entre los 36.5 y los 37 grados, esto es un indicador que las persona está saludable. Si su temperatura está por arriba de este rango quiere decir que tiene fiebre.

6.1 Propiedades deseables de una Métrica

Wong expone las siguientes propiedades para una métrica:

- Simple y Computable: entre más compleja una métrica es, es más probable que la persona que la está usando cometa un error. Al ser una métrica compleja, puede tomar más tiempo darle seguimiento y arreglarla.
- Intuitivamente persuasiva: la métrica tiene que tener sentido para poder usarla, incluso para alguien fuera del equipo.

- Objetiva: la métrica no depende de la opinión de una persona. Debe estar basada en datos empíricos para que se pueda aplicar a un gran número de problemas.
- Consistente en el uso de unidades y dimensiones: la métrica debe tener una definición consistente y estable de lo que se está midiendo. Esto hace que la métrica sea significativa, más fácil de interpretar, y comparar.
- Independiente del lenguaje de programación: la métrica debe ser fiable usando cualquier lenguaje de programación.
- Proveer un mecanismo efectivo para mejorar la calidad de software: la métrica debe ser un medio efectivo para identificar si se están haciendo mejoras o si la calidad del producto está decayendo.

Además, Wong expone como ejemplo de una buena métrica el número de bugs encontrados en un producto por semana.

- Simple y Computable: es un cálculo simple, ya que todo lo que se tiene que hacer es dar seguimiento al número de bugs reportados en una semana mediante un reporte de bugs o problemas logueados y luego sumarlos al final de cada semana.
- Intuitivamente persuasiva: los defectos son claramente negativos y el descenso de los mismos crea un mejor producto.
- Consistente: no depende del lenguaje de programación, y asumiendo que los desarrolladores tienen el mismo tipo de habilidad sin importar el lenguaje, el número de defectos debería permanecer aproximadamente consistente independiente del lenguaje de programación.
- Objetiva: los defectos no están basados en la opinión de una persona.
- Provee un mecanismo efectivo para mejorar la calidad de software: Dar seguimiento a los defectos cada semana durante la duración del proyecto debería permitir ver si los defectos

aumentan o disminuyen. Al tenerlos presente, el equipo está consciente de que arreglar los defectos generará un mejor producto.

Es importante recordar que las métricas no son siempre absolutas; pueden ser relativas solo a un proyecto en específico. Por ejemplo, la velocidad. Los puntos que son estimados pueden variar por proyecto. Usar una métrica relativa está bien mientras se tenga en cuenta que dos métricas basadas en medidas un poco diferentes no son cien por ciento comparables. La velocidad de un proyecto A no puede ser comparada con la velocidad de un proyecto B.

Capítulo 7. Estudio de Herramientas

En este capítulo se estudian cinco herramientas y las gráficas que ofrecen para realizar seguimiento a proyectos. Nuestro propósito es escoger las gráficas adecuadas para diseñar un Cuadro de Mando que englobe proyectos que usen las metodologías ágiles.

Las cinco herramientas elegidas para hacer el estudio de las gráficas fundamentales para un cuadro de mando fueron elegidas por su aceptación en el mercado e integración de gráficas por defecto generadas automáticamente. A continuación, se presente una breve descripción de cada una de ellas.

7.1 JIRA[4]

JIRA es una herramienta ampliamente usada para la gestión ágil de proyectos. JIRA tiene características como historias de usuario, y seguimiento del progreso de las tareas asignadas a cada historia de usuario. Además de almacenar estos elementos en un "backlog", JIRA también se usa para el seguimiento de "bugs".

JIRA ofrece varias pantallas, entre estas un tablero de tareas. Esto podría conceptualizarse como una lista de tareas en progreso. Es configurable y uno de los escenarios más simples consiste en tres columnas, "Por Hacer", "En progreso", y "Finalizado". La lista puede ser ordenada de acuerdo al miembro del equipo que la esté visualizando. Los miembros del equipo pueden arrastrar y dejar las tareas que se les han sido asignadas en cualquiera de las tres de esta forma indicando los estados actuales de las tareas.

[4] https://www.atlassian.com/software/jira

JIRA también tiene un tablero de planificación, donde se ofrece una visión más general sobre las historias de usuario y sus respectivas tareas por sprint. Este tablero permite asignar las tareas a los miembros encargados de realizarlas.

Las historias de usuario incluyen ID, descripción, etiquetas, a quien está asignada, su estado, el sprint al que pertenecen, estimación de puntos, y estimación de horas.

7.2 AXOSOFT[5]

AXOSOFT es un software de gestión ágil de proyectos que proporciona gráficos de velocidad e indicadores de capacidad que muestran la relación entre estimaciones y cantidad de trabajo. Esta herramienta posee flujos de trabajo predefinidos. Los usuarios pueden usar Scrum, Kanban, u otro framework personalizado.

Al igual que JIRA, AXOSOFT, permite a sus usuarios arrastrar y dejar tareas en un tablero kanban.

AXOSOFT cuenta con un modo "daily", donde se comunica el progreso y los obstáculos encontrados. La herramienta informa el estado de las tareas, su terminación, y quien lo ha hecho. Asimismo, dispone de notificaciones de email, filtros, vistas y wikis.

Este software permite al usuario generar reportes desde cualquier vista del "backlog". En estos reportes se puede visualizar Burndown Charts, velocidades, cantidad de trabajo, y proyecciones.

AXOSOFT cuenta con emails automatizados e hilos de conversaciones dentro de la aplicación.

Por último, permite integraciones con otros softwares como Bugsnag, GitHub, Salesforce, TestLodge, Slack, Visual Studio, Usersnap, and Zendesk.

[5] https://www.axosoft.com/

7.3 TARGET PROCESS[6]

Esta herramienta ofrece gestión ágil del manejo de vida de aplicaciones, manejo ágil de portafolio de proyectos, y elimina la necesidad de transferir y estandarizar datos entre diferentes soluciones de gestión de proyectos.

Ofrece tableros con una visión general y del avance de los proyectos. Targerprocess permite la personalización de tableros, creación de filtros, personalización de las propiedades y selección de vistas.

Esta herramienta permite al usuario el uso de diferentes métodos ágiles.

Targetprocess dispone de reportes gráficos en todos los niveles, y de métricas como tendencias de tiempo de ciclo, esfuerzo, "bugs" y el tiempo invertido en ellos.

7.4 MINGLE[7]

Es una herramienta para el manejo ágil de proyectos que puede modificarse de acuerdo al equipo, el tipo de trabajo, y el proyecto.

Mingle proporciona fórums, wiki, integración de código fuente, y tablero de cartas. Además, presenta gráficas de flujo acumulativo, cálculos personalizados, fuentes de información, entre otros.

Conjuntamente muestra la información a tiempo real para los equipos que están distribuidos geográficamente, y genera reportes teniendo en cuenta todos los proyectos que se estén llevando a cabo.

[6] https://www.targetprocess.com/
[7] https://www.thoughtworks.com/mingle/

Mingle ofrece seguimiento de "bugs" y una plataforma de pruebas.

7.5 TUNE-UP Process[8]

TUNE-UP Process es un software de manejo de proyectos que permite abordar la implantación de prácticas ágiles, basado en los métodos ágiles más populares (Kanban, Lean Development, Scrum y Extreme Programming). [29]

Al igual que las herramientas mencionadas anteriormente permite la generación de historias de usuario con estimaciones, y asignación a miembros del equipo. Conjuntamente anuncios a nivel de aplicación seleccionando los interesados. Generación de reportes y gráficas a nivel de sprints y de portafolio de proyectos.

También ofrece supervisión a nivel ejecutivo, pero dando autonomía a los miembros del equipo al actualizar el estado de la tarea en la que se encuentren trabajando de momento.

Adicionalmente incluye TDRE (Test-Driven Requirement Engineering), un enfoque para gestión de requisitos basado en pruebas de aceptación integradas a la planificación, seguimiento y gestión de los productos que se encuentran en desarrollo y mantenimiento.

[8] http://tuneupprocess.com/

Capítulo 8. Gráficas para seguimiento

En este capítulo se presentan las gráficas mediante una plantilla compuesta por su descripción general, sus elementos y a que herramientas pertenece. Además, para su análisis se exponen ejemplos y su interpretación con respecto a las tendencias que se observan.

Se resume también la información general de todas las gráficas evaluadas en las siguientes tres tablas:

- Gráficas ofrecidas por las herramientas (Tabla 1).
- Información requerida para crear las gráficas (Tabla 2).
- Objetivo de las gráficas (Tabla 3).

8.1 Gráficas ofrecidas por las herramientas

En la tabla 1 se puede apreciar de forma resumida las gráficas que ofrecen cada una de las herramientas que fueron escogidas para evaluar.

A continuación de la tabla 1 se explicará cada una de las gráficas. Para presentar el análisis de cada gráfica se ha hecho uso de una plantilla. Esta plantilla incluye:

- Descripción general: en esta sección se hace una descripción general de la gráfica.
- Elementos de la gráfica: se presentan y describen los elementos que componen la gráfica.
- Herramienta: en esta sección se menciona la herramienta que ofrece la gráfica.
- Ejemplo: se muestra un ejemplo de la gráfica.
- Interpretación: se presenta un análisis de los datos que muestra el ejemplo de la gráfica.

GRÁFICA	HERRAMIENTA				
	TargetProcess	TUNE-UP PROCESS	JIRA	Mingle	AXOSOFT
Estado de Proyectos/Sprints		✓			
Control de Proceso en Proyectos/Sprints	✓				
BurndownChart	✓	✓	✓		✓
BurnUPChart			✓	✓	
UT Finalizadas vs No Finalizadas		✓			
Sprint Report			✓		
Entregables				✓	
Edad Promedio			✓		
Esfuerzo por Actividad		✓		✓	
Tiempo Promedio de Procesamiento por Estado	✓				
Esfuerzo por Tipo de Actividad		✓			
Esfuerzo Asociado a Re-trabajo		✓			
Primera Estimación versus Esfuerzo Real		✓	✓		✓
KOs en Pruebas de Aceptación		✓			
Pruebas de Aceptación según Estado		✓			
Tiempo de Resolución de Problemas por Proyecto			✓		
Flujo Acumulado	✓	✓	✓		✓
Capacidad del Equipo por Tipo de Trabajo		✓			
Capacidad del Equipo por Tipo de UT		✓			
Gráfica de Número de UT por Tipo		✓			
Gráfica de Speedometer					✓

Tabla 1. Gráficas ofrecidas por las herramientas

8.2 Gráfica Estado de Proyectos/Sprints

Descripción General

La gráfica de Estado de Proyectos/Sprints muestra el progreso de Proyectos o de Sprints. La figura 8 muestra un ejemplo de esta gráfica, representando cada Proyecto/Sprint con un punto. La línea diagonal trazada desde el punto (0,0) al punto (100,100) que divide el conjunto de puntos nos indica que los Proyectos/Sprints que se encuentren sobre la línea de referencia tendrían un progreso adecuado, y aquellos que están por debajo de la línea estarían retrasados.

Elementos de la gráfica

- Eje X: porcentaje de tiempo transcurrido del Proyecto/Sprint.
- Eje Y: el porcentaje de progreso del Proyecto/Sprint calculado como 100*(Esfuerzo Estimado – Esfuerzo Invertido) / Esfuerzo Estimado.
- Información de cada punto: Esto incluye nombre del Proyecto/Sprint, tiempo estimado de ejecución, tiempo registrado, tiempo restante, fecha de inicio, fecha de fin, duración, porcentaje de tiempo transcurrido, porcentaje de progreso de trabajo, días restantes, y anomalías, en este caso:
    - UT que no han sido estimadas.
    - UT que han sobrepasado su estimación.

Si se da alguna de estas anomalías, el nombre del Proyecto/Sprint se muestra en rojo.

Herramienta

- TUNE-UP Process.

Ejemplo

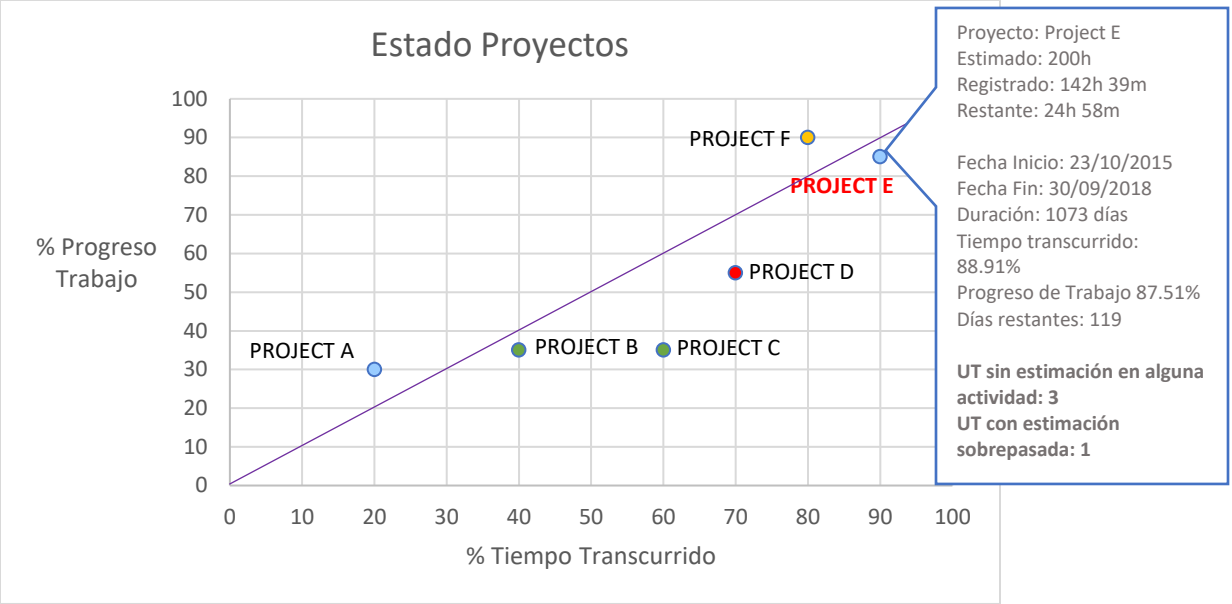

Figura 8. Gráfica Estado Proyectos.

Interpretación

La gráfica proporciona un estado de progreso de todos los Proyectos/Sprint de una forma sencilla. En la figura 8, se puede observar que el Proyecto E se encuentra un poco por debajo de la línea trazada. A primera vista, el proyecto solo parece estar un poco retrasado. No obstante, en el cuadro de leyenda se indica que aún hay 3 UT sin estimar, y 1 con estimación sobrepasada. Estas dos posibles anomalías podrían invalidar la interpretación de la posición del punto. Por lo cual, antes de interpretar el estado de un Proyecto/Sprint se debe asegurar que no existen dichas anomalías en los datos.

8.3 Gráfica Control de Proceso en Proyectos/Sprints

Descripción General

La gráfica de Control de Proceso en Proyectos/Sprints muestra la distribución de los *cycle time* por UT que ya estén completadas dentro de un rango de tiempo determinado por el usuario [2].

El *Cycle Time* es la diferencia entre el momento que se empieza a desarrollar la UT y el momento en que es finalizada.

Existen tres líneas que dividen el gráfico en tres áreas. Estas líneas se calculan estadísticamente, y dependen de la distribución general de las UT:

- Línea de media: representa el *cycle time* clasificado como "bueno" de días que toma completar una UT.
- Línea de límite de advertencia: representa el *cycle time* clasificado como "sospechoso" de días que toma completar una UT.
- Línea de límite de control: representa el *cycle time* clasificado como "malo" de días que toma completar una UT.

Elementos de la gráfica

- Eje X: días por fecha del Proyecto/Sprint.
- Eje Y: tiempo de ciclo en días de las UT.
- Información de cada punto: esto incluye, descripción, tareas, errores, casos de prueba, fuente, seguimiento, flujo, y relaciones de la UT.

Herramienta

- Targetprocess.

Ejemplo

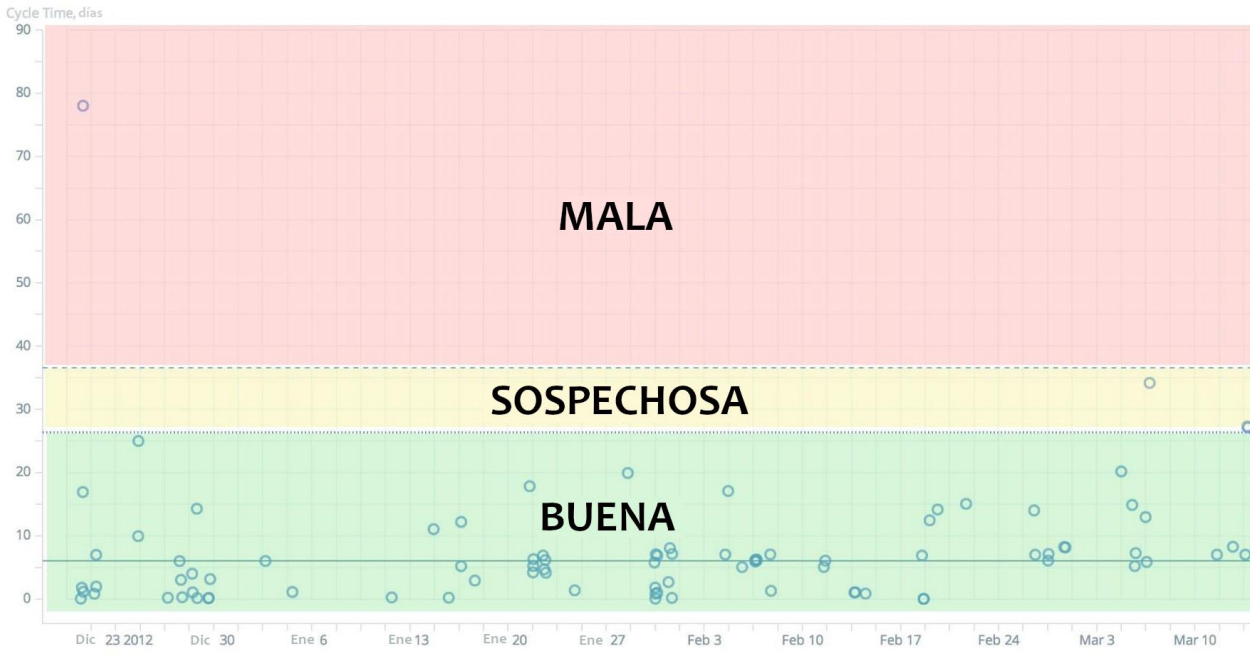

Figura 9. Gráfica de Control de Proceso

Interpretación

La gráfica proporciona de forma inmediata el estado de las unidades de trabajo dentro de las divisiones establecidas como “buena”, “sospechosa”, y “mala”. En la figura 9 se puede observar que la mayoría de las UT se encuentran dentro de la división “buena”, estableciendo que el desarrollo de las UT ha sido estable.

Sin embargo, existen algunas UT dentro de las otras dos divisiones. En la división “mala” se encuentra una UT que ha tomado alrededor de 80 días en completarse.

8.4 Gráfica Burndown para Proyectos/Sprints

Descripción General

La gráfica Burndown para Proyectos/Sprints muestra la evolución del trabajo restante. En el extremo derecho de la gráfica está la fecha final, en la cual el equipo debe completar las UT [1].

La gráfica de Burndown permite visualizar:

- Puntos u horas completados por el equipo.
- Puntos u horas que quedan por completas.
- Visualizar si la tendencia es coherente con la fecha estimada.

En el caso en que la gráfica se realice con puntos, la línea que atraviesa la gráfica indica la tendencia esperada en cuanto a la evolución del trabajo restante desde el total de puntos estimados al comienzo hasta cero puntos restantes al final del sprint o el proyecto.

En el caso en que la gráfica se realice con horas, la línea que atraviesa la gráfica indica la tendencia esperada en cuanto a la evolución del trabajo restante desde el total de horas estimadas al comienzo hasta ceros horas restantes al final del sprint o el proyecto.

Para una correcta interpretación de la gráfica, todas las UT deben estar estimadas. En el caso de que una UT supere su estimación debe inmediatamente reestimarse.

Elementos de la gráfica

- Eje X: tiempo.
- Eje Y: esfuerzo total en puntos u horas ideales.

Herramienta

- Las gráficas Burndown están presentes en casi todas las herramientas al ser gráficas básicas y fundamentales de seguimiento de proyectos que implementan metodologías ágiles.

Ejemplo 1

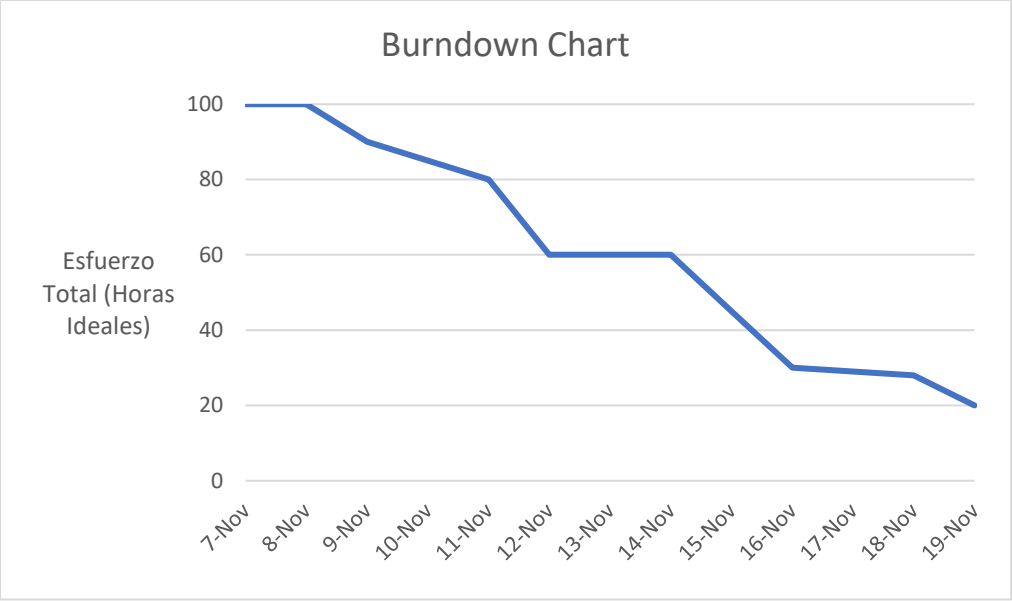

Figura 11. Gráfica Burndown 1

Interpretación

La gráfica proporciona una visualización del progreso del trabajo en base al esfuerzo restante. La figura 11, es un típico ejemplo de una gráfica real de Burndown. En los días 7 y 8 disminuyó el trabajo restante, al igual que del día 12 al 14. Sin embargo, en los demás días se observa un progreso, ya sea leve o considerable.

Cabe agregar que las UT pueden decrementar su estimación inicial, reflejando en la gráfica un avance menor del real.

Ejemplo 2

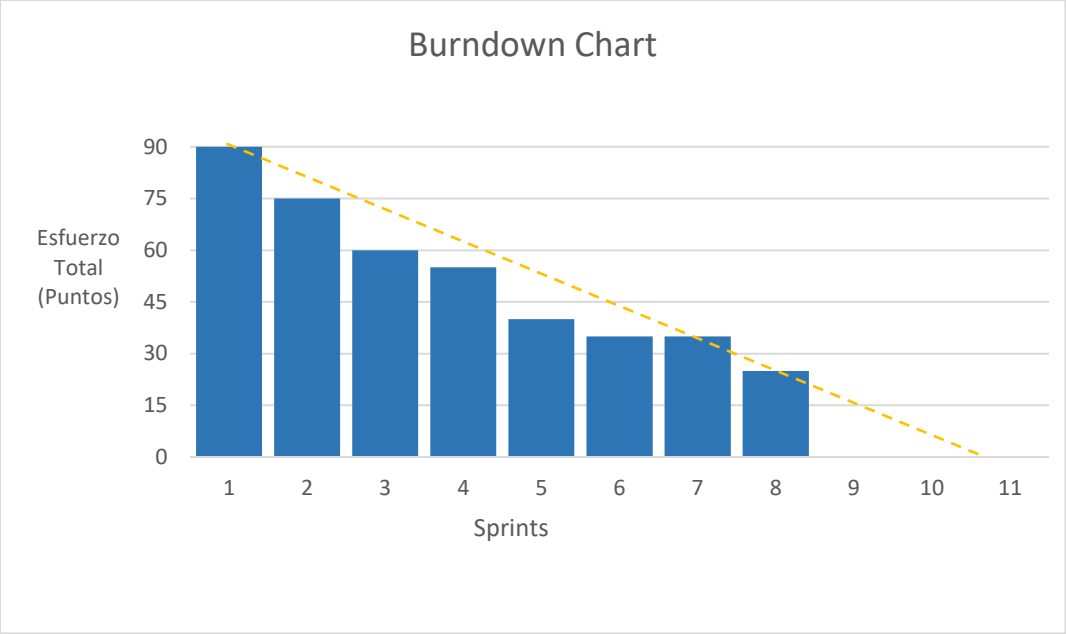

Figura 12. Gráfica Burndown 2

Interpretación

A diferencia de la figura 11 que muestra la gráfica de Burndown con el esfuerzo medido en horas ideales, la figura 12 muestra la misma gráfica con el esfuerzo medido en puntos y el tiempo dividido en Sprints de la misma duración. Como se puede observar de un sprint a otro el salto es de bloques, ya que solo se resta del total de puntos los que componen una UT finalizada.

La línea de predicción empieza en el centro del bloque de puntos del sprint 1 hasta que pasa por el centro del bloque del último sprint finalizado, en este caso el 8. Esto indica que se completarán los puntos restantes en el sprint 11, siempre y cuando la tendencia se mantenga.

Ejemplo 3

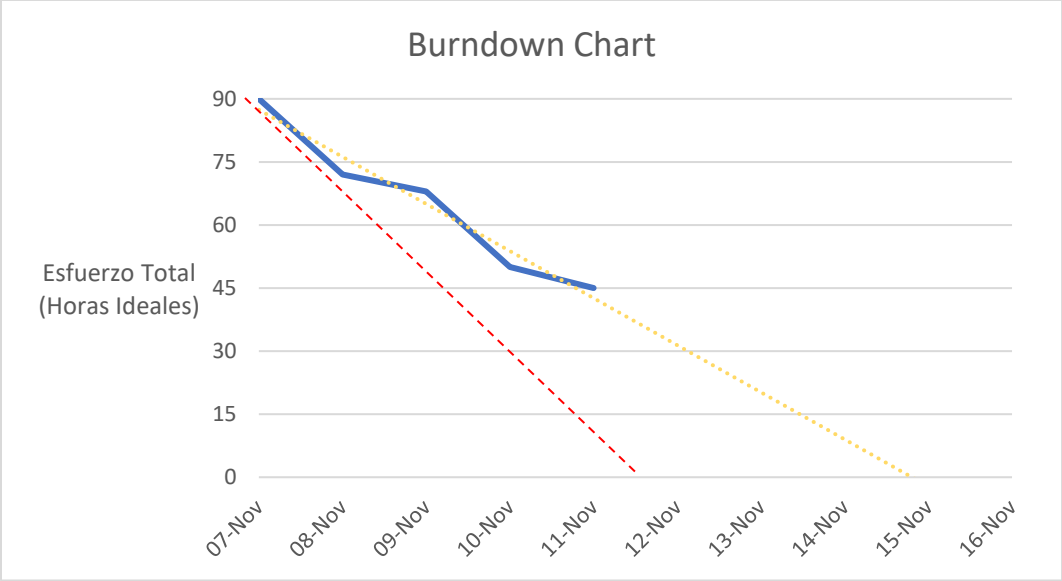

Figura 12. Gráfica Burndown 3

Interpretación

La línea de predicción amarilla muestra que el día 15 se finalizarían las UT de acuerdo con la tendencia actual. La línea de predicción roja apunta a la fecha establecida de finalización de las UT. Como se puede observar, el equipo está bastante alejado del trayecto que debería llevar para cumplir con la terminación de todas las UT.

8.5 Gráfica BurnUp

Descripción General

La gráfica BurnUp provee una representación visual del tiempo invertido en UT. Además, proporciona información sobre el progreso del sprint o proyecto, al igual que advertencias que facilitan la identificación de problemas, así como la desviación de la trayectoria del proyecto o sprint [4].

La distancia entre la línea verde y la línea de predicción naranja que se muestran en la figura 14, es la cantidad de horas que faltan por invertir. Cuando el sprint o el proyecto hayan finalizado, estas líneas se unirán [4].

Elementos de la gráfica

- Eje X: Tiempo.
- Eje Y: Horas invertidas.

Herramienta

- La gráfica BurnUp se encuentra presente en JIRA y MINGLE.

Ejemplo

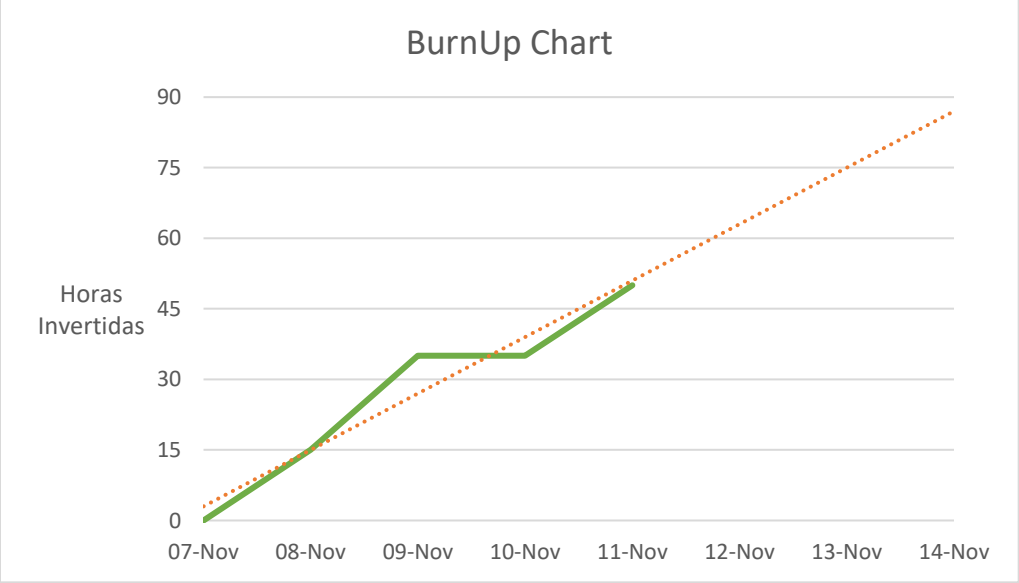

Figura 14. Gráfica BurnUp

Interpretación

La gráfica muestra que la cantidad de horas invertidas en programación del día 7 al 9 es progresiva, mientras que del 9 al 10 no se invierte ninguna. Sin embargo, al haber invertido una cantidad de horas

más alta del día 8 al 9 y luego del día 10 al 11, la línea verde que representa las horas invertidas se junta con la línea de predicción, indicando que al día 11 el sprint o proyecto se encuentra donde debería.

La información que se aprecia en las gráficas Burndown y Burnup es complementaria, por lo que resulta natural y útil presentar la información de ambas gráficas en una sola.

8.6 Gráfica UT Finalizadas vs. UT no Finalizadas

Descripción General

La gráfica de UT Finalizadas vs. UT no Finalizadas muestra la proporción gradual del trabajo no terminado. La cantidad de horas o puntos que han invertido durante el sprint o proyecto se muestra en color azul, y la cantidad de horas o puntos estimados no finalizados en color naranja.

Elementos de la gráfica

- Eje X: Tiempo.
- Eje Y: Esfuerzo total en horas ideales o puntos.

Herramienta

- TUNE-UP Process.

Ejemplo

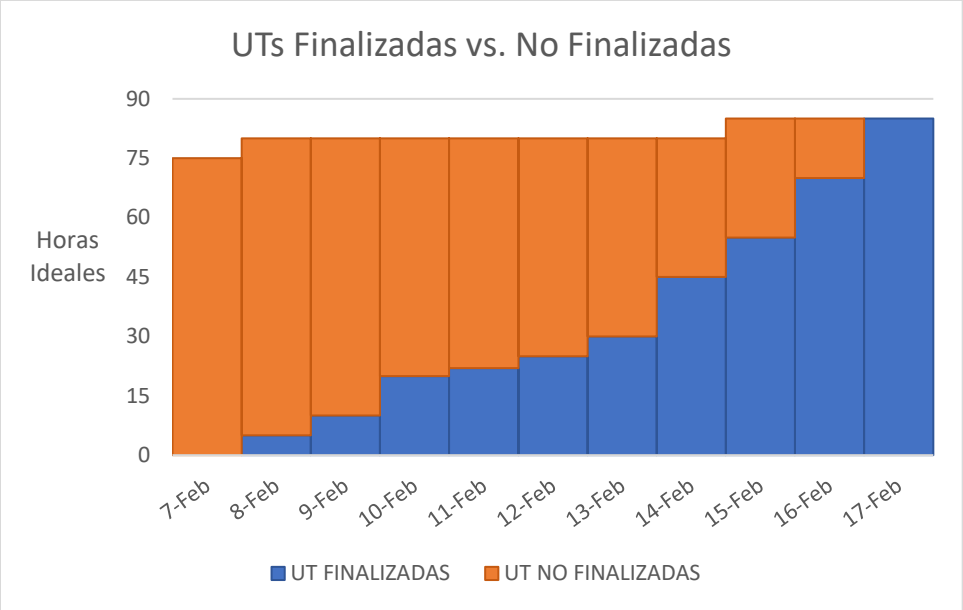

Figura 15. Gráfica UT Finalizadas vs. No Finalizadas

Interpretación

La gráfica de UT Finalizadas vs. No Finalizadas muestra la evolución del Sprint o Proyecto contrastando la proporción de horas ideales de UT no Finalizadas con la proporción de horas ideales que se han invertido en UT Finalizadas.

En la gráfica de la figura 15 se puede observar que la evolución del trabajo terminado es constante de una forma ascendente, ya que las barras que representan las UT Finalizadas van creciendo conforme pasan los días, mientras que las barras que representan las UT no Finalizadas van disminuyendo. Esto resalta lo importante que es la proporción del trabajo terminado respecto del total.

En el día 8 y el día 15 se muestra un aumento en las barras que representan las UT no Finalizadas. Esto indica que nuevas UT se han añadido al Sprint o Proyecto.

El comportamiento esperado de una gráfica de UT finalizadas vs. UT no finalizadas es que tempranamente se terminen UT a buen ritmo. Un buen ritmo quiere decir que, transcurrido un tercio del tiempo se ha completado un tercio del trabajo.

Esta gráfica también complemente a la gráfica Burndown pues muestra las UT terminadas, mientras que en la de Burndown solo se visualiza el tiempo restante, no la proporción del trabajo terminado.

8.7 Gráfica de Entregables

Descripción General

La gráfica de entregables muestra el estado de cada una de las partes que componen cada UT. Se asignan diferentes colores a la cantidad de puntos de cada UT dependiendo de su estado.

- Azul: para los puntos que han sido completados.
- Naranja claro: para los puntos que están en progreso.
- Naranja: para los puntos que aún no se han completado.

Elementos de la gráfica

- Eje X: UT.
- Eje Y: Esfuerzo total en puntos.

Herramienta

- Mingle.

Ejemplo

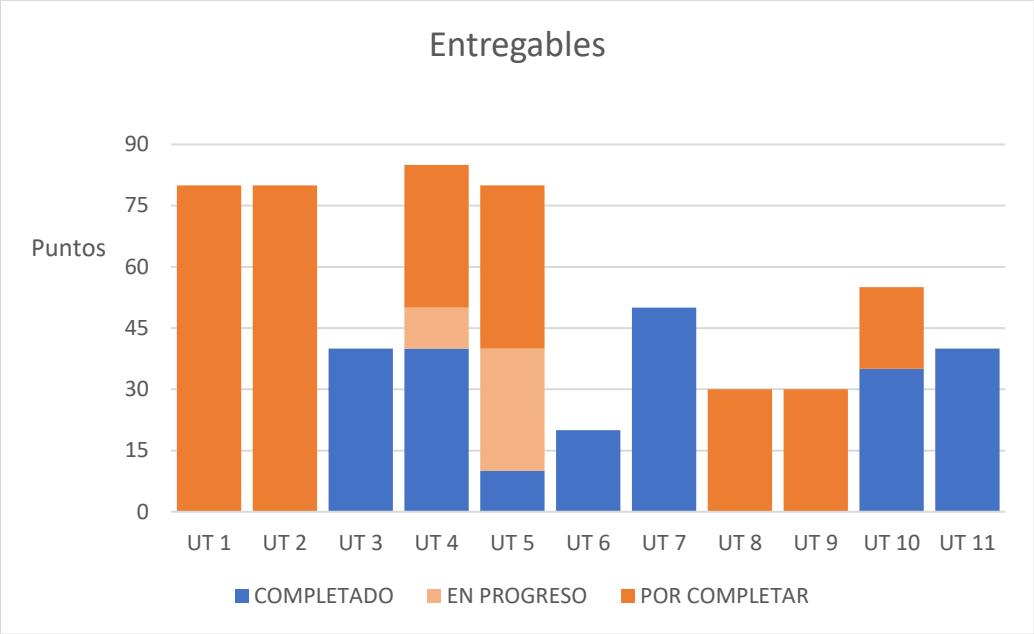

Figura 17. Gráfica de Entregables

Interpretación

La gráfica de Entregables de la figura 17 no es recomendable pues proporciona una visión confusa del estado de una UT. ¿Cómo determina el equipo que ha terminado solo cierta cantidad de puntos de una UT, o la cantidad de puntos en los que está trabajando y los que quedas\n por completar? ¿De qué forma se trabaja con las UT? ¿Acaso esto nos indica que varias personas pueden trabajar en una UT al mismo tiempo y no ser dependientes del esfuerzo previamente realizado? ¿Es esta representación fiel al desarrollo de la UT?

La información que representa este gráfico no es fácil de interpretar y su concepto de que una UT está terminada por completo es confuso. Por esta razón, su uso no es recomendable para el seguimiento de proyectos ágiles.

8.8 Gráfica de Edad Promedio de UT

Descripción General

La gráfica de Edad promedio muestra la "edad" que tiene una UT desde su fecha de creación. La gráfica muestra la edad de una UT dentro del "backlog" en una fecha específica durante los últimos 30 días. [6]

Elementos de la gráfica

- Eje X: Fecha.
- Eje Y: Días.

Herramienta

- Jira.

Ejemplo

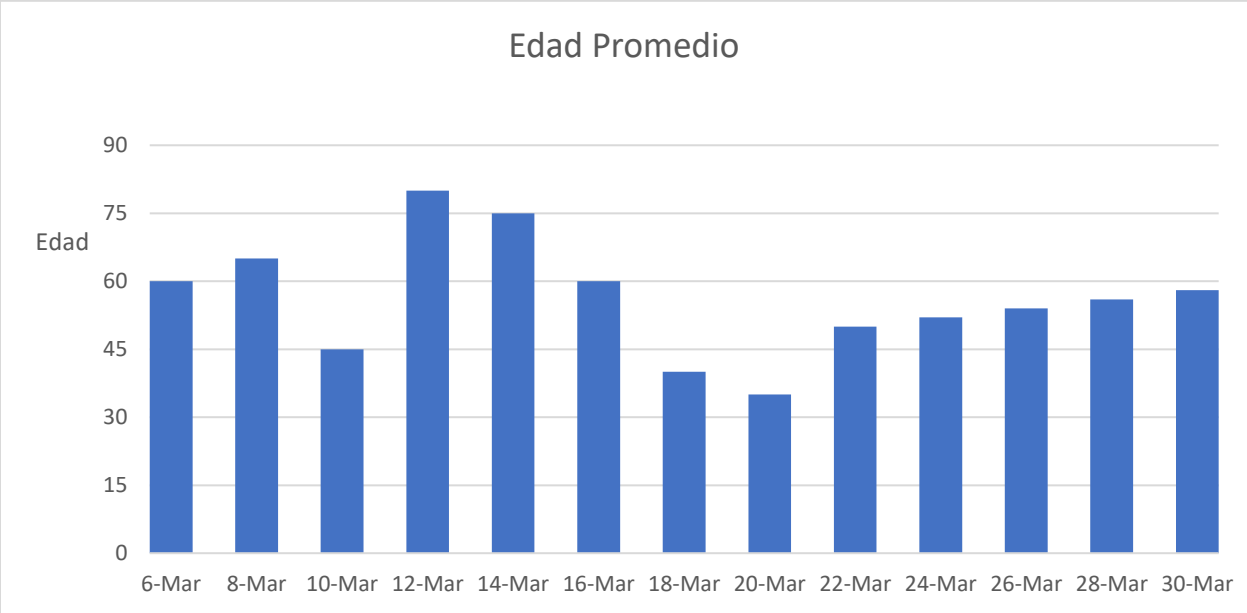

Figura 18. Gráfica de Edad Promedio

Interpretación

La gráfica de edad promedio de la figura 18 proporciona información útil para el equipo, especialmente para el "Product Owner". Cuando una UT pasa mucho tiempo sin avanzar o cambiar de estado, el equipo debe preguntarse porqué. Por ejemplo, las prioridades en el proyecto pueden haber cambiado y las UT que tienen más tiempo sin resolverse han pasado a ser irrelevantes. También podría ser que el equipo no tiene la capacidad para cumplir el plazo de la realización de UT.

La gráfica mostrada en la figura 18, puede motivar la revisión constante del "Backlog" para evitar postergación de UT, desestimar o eliminar UT que ya no son necesarias u obsoletas. El "Product Owner" debe estar constantemente reorganizando y depurando el "backlog" del proyecto de manera eficiente. Y asegurarse de que las UT estén formuladas claramente y que sean relevantes para el desarrollo actual del proyecto.

Sin embargo, hay que tener en cuenta que se trata de valores promedios, y estos pueden verse alterados por valores extremos.

8.9 Gráfica de Esfuerzo por Actividad

Descripción General

La gráfica de Esfuerzo por Actividad muestra la cantidad de horas invertidas en las actividades del flujo de trabajo por las que ha pasado una UT para llegar a completarse, durante un Sprint o Proyecto.

Para diferenciar en la gráfica las UT completadas de las que están en progreso, se dibujan líneas diagonales sobre la barra que corresponde a una UT completada.

El color marrón representa la actividad de programación, el color amarillo la actividad de pruebas, y el color naranja la actividad de diseño.

Elementos de la gráfica

- Eje X: UT.
- Eje Y: Horas Invertidas.

Herramienta

- TUNE-UP Process y Mingle.

Ejemplo

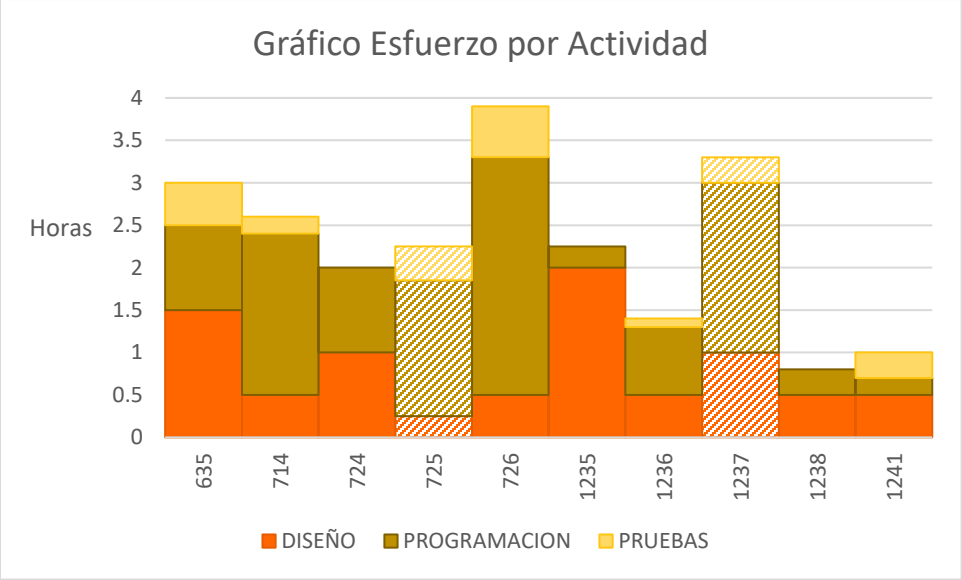

Figura 19. Gráfica de Esfuerzo por Actividad

Interpretación

La gráfica de Esfuerzo por Actividad en la figura 19 muestra dos UT, 725 y 1237, finalizadas y las demás aún en progreso. En la UT 635 se ha invertido 1.5 hora en la actividad de diseño, una hora en la actividad de programación y 0.5 horas en la actividad de testeo. El tiempo que se invierte ya sea en pruebas, diseño o programación, es producto de la complejidad de la UT. En la UT 724 se puede observar que sólo se ha invertido tiempo de programación y diseño, lo que podría ser un indicador de que aún no se

ha terminado de desarrollar la UT, o podría ser que no hay una persona del equipo que esté disponible para realizar las pruebas.

8.10 Gráfica de Tiempo Promedio de Procesamiento por Estado

Descripción General

La gráfica de Tiempo Promedio de Procesamiento por Estado muestra el promedio de días que pasa una UT en un estado durante un Sprint o Proyecto. Por ejemplo, esta gráfica podría mostrar los siguientes siete estados:

- Abierto: Tiempo que pasa desde que una UT es creada hasta que es finalizada.
- Planificado: Tiempo que duran las UT planificadas esperando a que sean desarrolladas.
- Re-abierto: Tiempo que pasa una UT re-abierta.
- En desarrollo: Tiempo que pasa una UT en desarrollo.
- Programado: Tiempo en el que ya una UT está programada y está a la espera de ir a Testeo.
- En testeo: Tiempo en el que una UT está siendo testeada.
- Testeado: Tiempo en el que ya una UT ha sido testeada y está a la espera de ir a producción.

Elementos de la gráfica

- Eje X: Estados.
- Eje Y: Días.

Herramienta

- TargetProcess.

Ejemplo

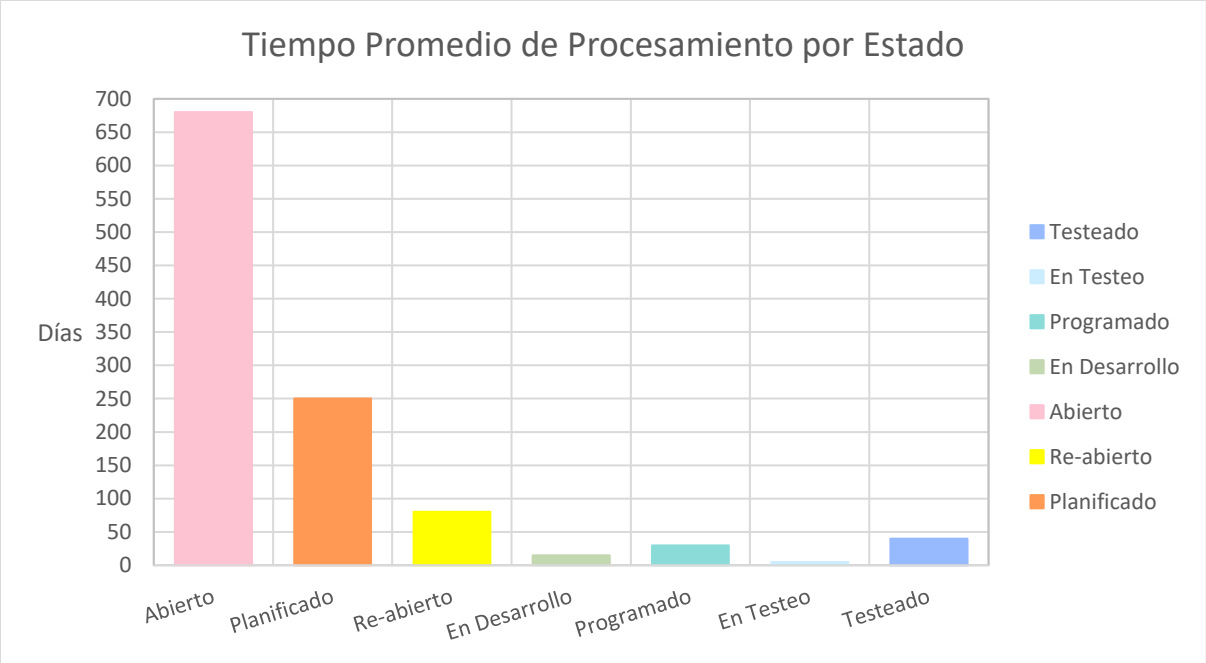

Figura 20. Gráfica Tiempo Promedio de Procesamiento por Estado

Interpretación

En la gráfica que se muestra en la figura 20, se observa que las actividades de desarrollo y testeo son realizadas bastante rápido ya que el promedio de tiempo que las UT pasan en los estados de "En desarrollo" y "En testeo" es relativamente bajo. Sin embargo, las UT pasan más tiempo en los estados de "Planificado", "Programado", y "Testeado" [7]. Esto puede ser un indicador de que no hay suficientes personas en el equipo para que no se acumulen las UT en estados que son simplemente de espera.

Cabe resaltar que los promedios no siempre son un buen indicador pues el tamaño y complejidad de las UT normalmente es diferente y el tiempo empleado en una UT puede variar significantemente.

8.11 Gráfica de Esfuerzo por Tipo de Actividad

Descripción General

En la gráfica de Esfuerzo por Tipo de Actividad se muestra la proporción de horas invertidas en el tipo de actividad por el que pasa una UT.

La gráfica incluye las horas invertidas en seguimiento, comunicación por mensajes y reuniones. Estas horas deben ser registrados con disciplina para reflejar de forma precisa el tiempo empleado en cada tipo de Actividad.

Los tipos de actividad que se visualizan en la gráfica están diferenciados por colores:

- Seguimiento: color morado.
- Responder mensajes: color azul.
- Reuniones: color azul claro.

Elementos de la gráfica

- Eje X: UT.
- Eje Y: Horas Invertidas.

Herramienta

- TUNE-UP Process.

Ejemplo

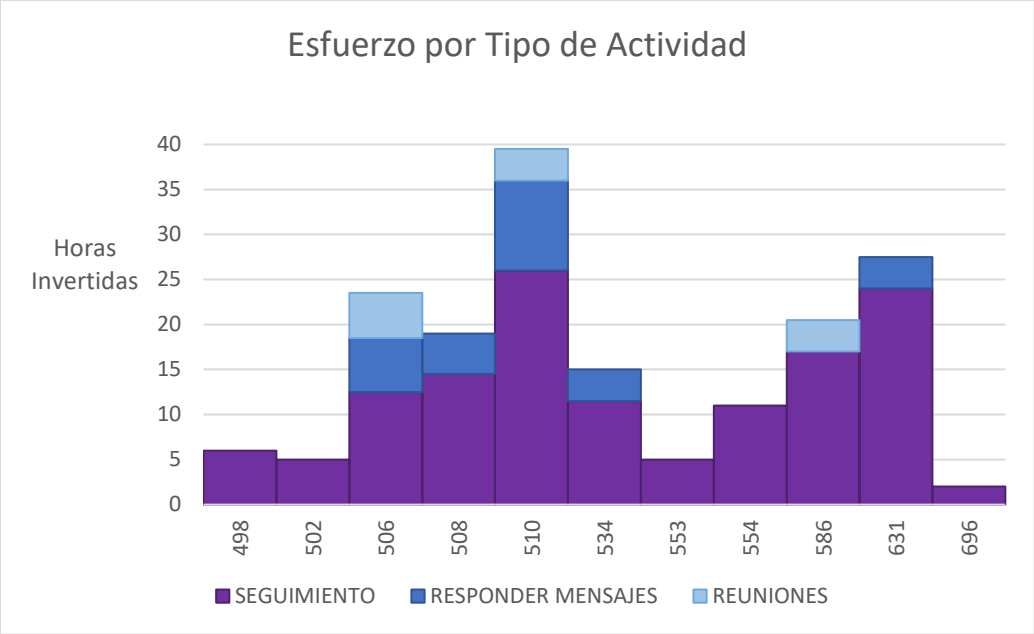

Figura 21. Gráfica de Esfuerzo por Tipo de Actividad

Interpretación

En la gráfica que se observa en la figura 21, se puede apreciar la cantidad de horas invertidas según el tipo de actividad. Por ejemplo, la UT 510 tiene un total de 25 horas invertidas en seguimiento, 10 horas invertidas en responder mensajes relacionados a ella, y 7 horas en reuniones. Se puede asumir que es una UT bastante compleja que demandas mucho esfuerzo.

Las UT no solo consisten en ser desarrolladas y probadas. También existen fases de seguimiento y reajuste que deben ser consideradas al planificar y llevar a cabo un proyecto. Al tener la información proporcionada por la figura 21 a mano, el equipo tiene una idea más clara de todo lo que conlleva realizar una UT en un Proyecto o Sprint.

8.12 Gráfica de Esfuerzo Asociado a Re-trabajo

Descripción General

La gráfica de Esfuerzo Asociado a Re-trabajo muestra la proporción de horas invertidas en Re-trabajo por cada UT durante un Sprint o Proyecto. Las situaciones de re-trabajo que se pueden presentar son las siguientes:

- Una UT se podría re-abrir sin aún no se ha entregado.
- Una UT de corrección de fallo o mejora de otras UT ya implementadas.
- Repetir una actividad de la UT mientras la UT no está terminada.

La gráfica de la figura 22 presenta la situación donde se ha repetido la actividad de "Programación" de la UT. Es decir, el tiempo invertido en programar después de que la UT hubiera pasado a pruebas por primera vez.

En la gráfica se diferencian dos actividades por colores:

- Programación: color azul oscuro.
- Re-trabajo: color rojo.

Elementos de la gráfica

- Eje X: UT.
- Eje Y: Horas Invertidas.

Herramienta

- TUNE-UP Process.

Ejemplo

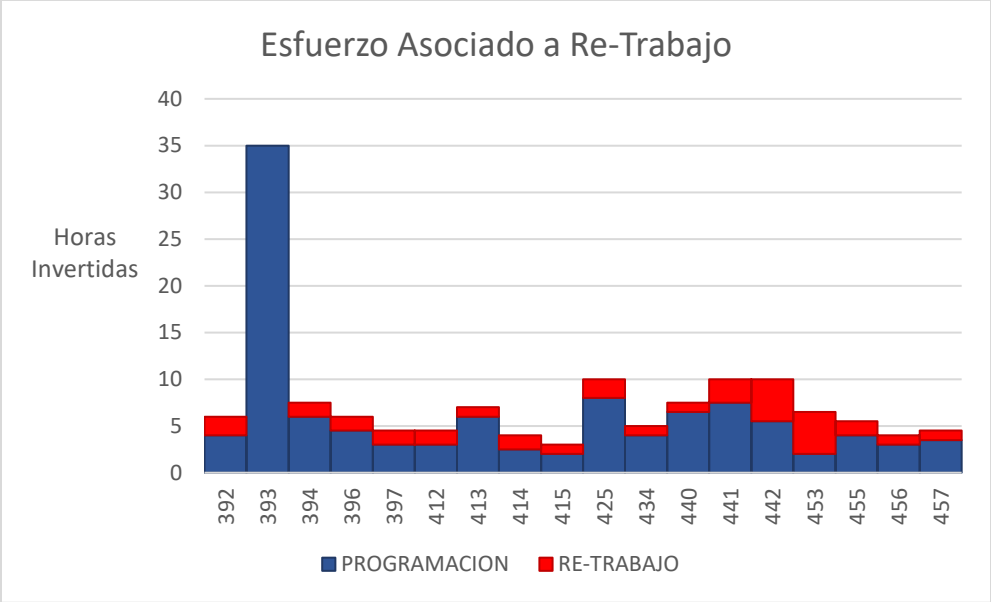

Figura 22. Gráfica de Esfuerzo Asociado a Re-trabajo

Interpretación

La gráfica de Esfuerzo Asociado a Re-trabajo que se muestra en la figura 22, permite visualizar que UT han tenido re-trabajo y cuantas horas se han invertido en ello.

En la UT 453, se puede observar que las horas invertidas en re-trabajo duplican a las primeras horas invertidas en programación. Esto puede ser producto de diversas situaciones que pueden ser tanto negativas como positivas, entre ellas:

- Los requisitos de la UT no estaban claros o completos, lo que genera un va y viene entre el desarrollo de la UT y su testeo.
- Introducción de cambios por parte del cliente que originalmente no estaban contemplados para la UT.

- El equipo está trabajando incrementalmente en la UT, es decir, se va definiendo a medida que va tomando forma y se va probando hasta que la UT esté terminada.

Dependiendo del caso de la UT, podrían existir diferentes niveles aceptables de re-trabajo.

8.13 Gráfica de Primera Estimación versus Esfuerzo Real

Descripción General

La gráfica de Primera Estimación versus Esfuerzo Real permite visualizar al equipo la comparación entre las horas que se estimaron inicialmente para una UT y las horas que se invirtieron en dicha UT para ser completada durante un Sprint o Proyecto.

La información presentada debería distinguir entre los tiempos dedicados a las diferentes actividades que hacen parte de una UT. Sin embargo, el enfoque más simple y efectivo es centrarse en la actividad de "Programación", pues es normal que las UT se reestimen más adelante en el proyecto donde el equipo tiene más experiencia con el desarrollo que antes de iniciar el proyecto.

Para diferenciar la Primera Estimación del Esfuerzo real, se usan los colores amarillo y azul respectivamente.

Elementos de la gráfica

- Eje X: UT.
- Eje Y: Horas.

Herramienta

- TUNE-UP Process, Jira, y Axosoft

Ejemplo

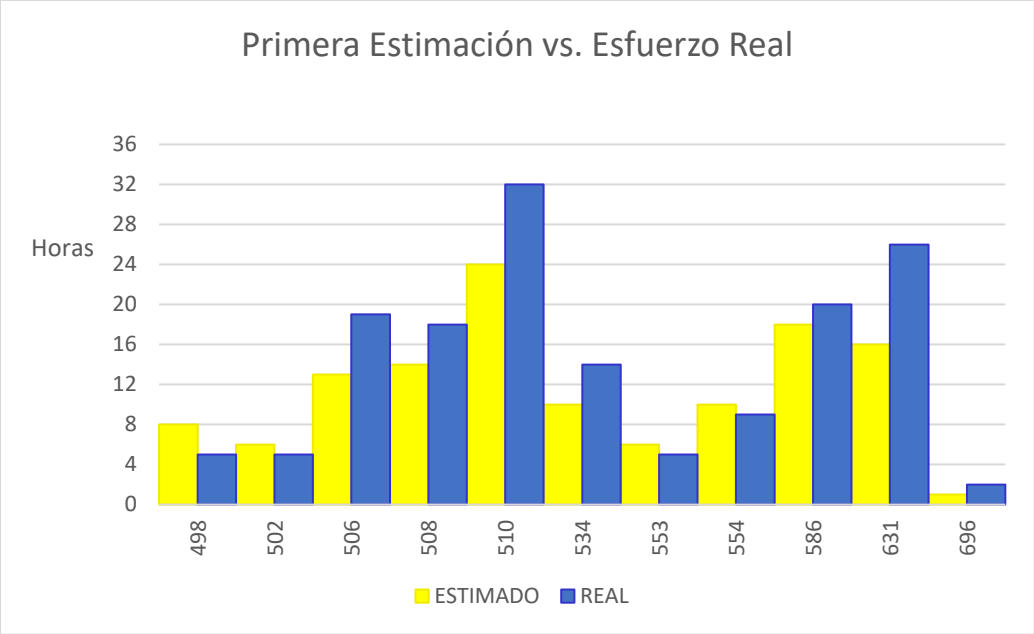

Figura 23. Gráfica de Primera Estimación vs Estimación Real

Interpretación

La gráfica de Primera Estimación vs Estimación Real que se muestra en la figura 23, proporciona al equipo la diferencia que existe entre las horas que se estimaron inicialmente para una UT y las que se invirtieron para terminarla. Esto permite que el equipo reflexione sobre las estimaciones que se hacen y provee una mejor referencia para estimaciones futuras, ya sea en futuros sprints o proyectos.

El desarrollo de software puede resultar impredecible en muchas ocasiones. El desarrollo de una UT puede estar muy claro, sin embargo, siempre existe la posibilidad de que no se consideren todos los escenarios con los que el desarrollo de una UT se puede encontrar.

8.14 Gráfica de KOs y OKs en Pruebas de Aceptación

Descripción General

La gráfica de KOs y OKs en Pruebas de Aceptación muestra que está sucediendo a nivel de pruebas en cada UT durante un Sprint o Proyecto. Un KO indica la detección de algún fallo o defecto en la prueba. Un OK indica el éxito en la prueba. Esta gráfica alerta al equipo si el desarrollo no ha ido como esperado y hay más trabajo a realizar a nivel de UT.

Elementos de la gráfica

- Eje X: UT.
- Eje Y: Número de KOs y OKs.

Herramienta

- TUNE-UP Process.

Ejemplo

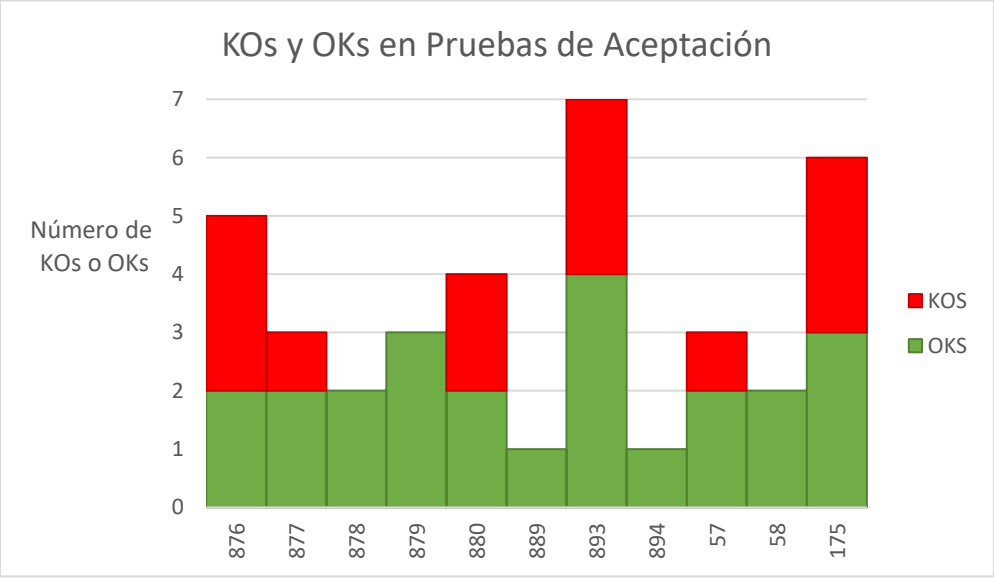

Figura 24. Gráfica de KOs y OKs en Pruebas de Aceptación

Interpretación

En la gráfica de KOs y OKs en Pruebas de Aceptación que se muestra en la figura 24 se puede apreciar que hay varias UT que no tienen ningún número de KOs en pruebas de aceptación y varios OKs. Esto quiere decir que su desarrollo ha sido exitoso por tanto el estado de sus pruebas de aceptación es OK.

Las UT 877 t 57 sólo han tenido un KO respectivamente en sus pruebas de aceptación. Este escenario es bastante común en el desarrollo de software.

Las UT 876, 893 y 175 han tenido tres KOs cada una. Esto podría ser el resultado del fallo en múltiples pruebas de aceptación, o múltiples fallos en la misma prueba de aceptación. Si el caso es el último, podría tratarse de un fallo de comunicación entre el tester y el desarrollador. Donde se están indicando errores que no logran ser corregidos.

8.15 Gráfica de Pruebas de Aceptación según Estado

Descripción General

La gráfica de Pruebas de Aceptación según Estado muestra en qué estado se encuentran las pruebas de aceptación de las UT de un Sprint o Proyecto. En TUNE-UP Process las pruebas de aceptación tienen estados de acuerdo con su grado de especificación.

Los estados de las pruebas de aceptación son los siguientes:

- Identificada: La prueba de aceptación solo tiene un nombre.
- Definida: se ha especificado su cuerpo en término de Condiciones-Pasos-Resultado.
- Diseñada: se han especificado instanciaciones de la prueba con datos de prueba concretos.

Elementos de la gráfica

- Eje X: UT.
- Eje Y: Número de pruebas de aceptación.

Herramienta

- TUNE-UP Process.

Ejemplo

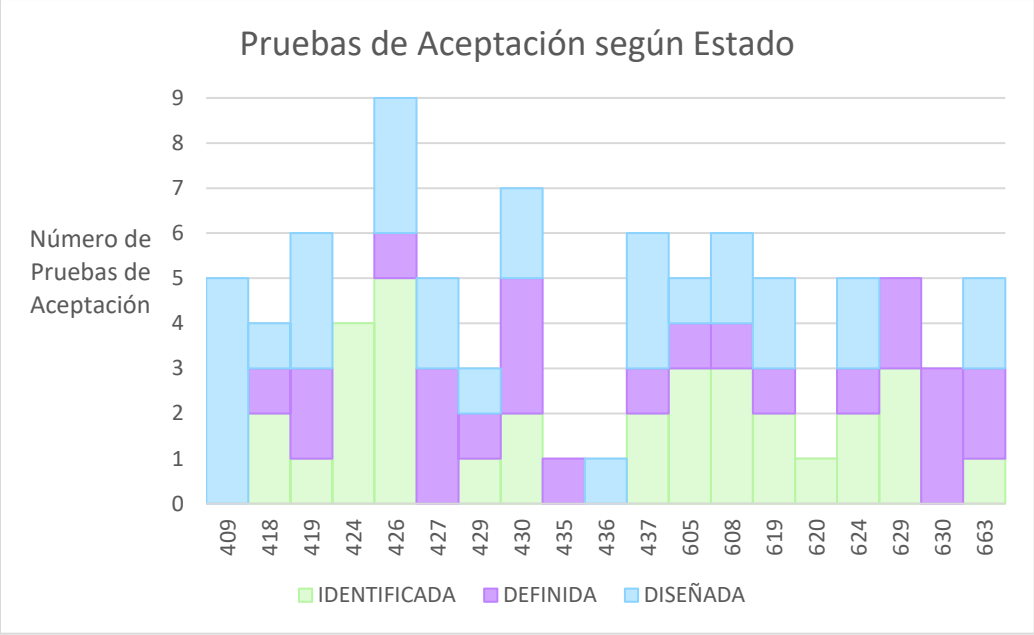

Figura 24. Gráfica de Pruebas de Aceptación según Estado

Interpretación

La gráfica de Pruebas de Aceptación según su Estado presentada en la figura 24 muestra los estados de las pruebas de aceptación de una UT. Por ejemplo, la UT 409 tiene cinco pruebas de aceptación y todas ellas ya se encuentran en el estado de "Diseñadas".

La UT 437 tiene seis pruebas de aceptación, pero cada una de ellas en distintos estados. Dos de ellas solo han sido “Identificadas”, una ya está “Definida”, y las tres restantes ya están “Diseñadas”.

El progreso en cuanto al estado de las pruebas de aceptación de una UT es un indicador sustancial del progreso del trabajo de la UT.

8.16 Gráfica de Lead Time Promedio de Fallos

Descripción General

La gráfica de Lead Time Promedio de Fallos muestra el tiempo promedio en días que se toma resolver un problema. Esta toma como referencia para el promedio los ítems resueltos en una fecha específica.

El “Lead Time” es la diferencia entre el momento en el que es creada una UT y el momento en que es finalizada.

Elementos de la gráfica

- Eje X: Fechas.
- Eje Y: Días.

Herramienta

- Jira.

Ejemplo

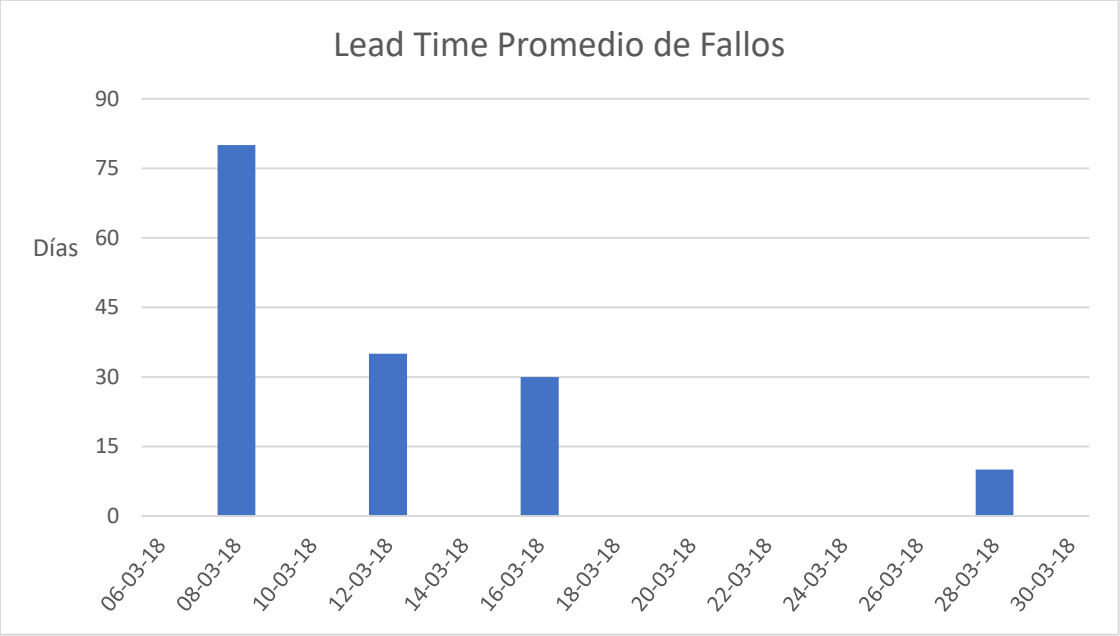

Figura 25. Gráfica de Lead Time Promedio de Fallos

Interpretación

La gráfica de Lead Time Promedio de Fallos en la figura 25, muestra la tendencia en los últimos 30 días de tiempo de resolución de problemas.

En el día 8 de marzo, se muestra un promedio de 80 días. Esto quiere decir que en esa fecha se finalizaron una cantidad de problemas cuyo tiempo promedio de resolución fue 80 días. Por ejemplo, en dicho día se resolvieron 3 problemas; el primero empezó a tratarse hace 50 días, el segundo hace 120 días, y el tercero hace 70 días. Al sumar los tres se obtiene un total de 240 días que al dividirse entre tres daría un promedio de 80 días.

Para un proyecto de mantenimiento continuo, los datos que proporciona la gráfica son de gran valor, pues a la hora de tener un fallo crítico ya se tiene un promedio de cuánto tiempo le podría tomar al equipo arreglarlo.

El histórico de estos datos ayuda a velar por el cumplimiento del SLA (Acuerdo de nivel de servicio). El SLA es el contrato donde un servicio se define formalmente. El SLA puede especificar los niveles de disponibilidad, utilidad, rendimiento, operación, u otros atributos del servicio. En cada nivel se puede especificar el "mínimo" o "lo esperado" [8].

Si el promedio del tiempo de respuesta establecido en el SLA no es el mismo que el real, entonces el equipo debe evaluar los datos de la gráfica y hacer ajustes en los procesos de respuesta para poder cumplir con el SLA.

8.17 Gráfica de Flujo Acumulado

Descripción General

La gráfica de Flujo Acumulado muestra el WIP ("work in progress") día a día en cada actividad/estado de la UT en un Sprint o Proyecto. Cada franja de la gráfica representa un estado. Si las franjas se extienden verticalmente a través del tiempo el WIP aumenta y esto puede indicar la existencia de un cuello de botella.

Para generar la gráfica de Flujo acumulado cada día se hacen *snapshots,* esto quiere decir que se hace una traza diaria de la cantidad de UT que se encuentran en cada actividad/estado.

Las actividades/estados pueden ser configurables dependiendo de la herramienta que se use. En la figura 26 hay siete actividades: por completar, programación, análisis, diseño, integración pruebas aceptación, y completado. Cabe resaltar que la franja correspondiente a completado es la más interesante para el seguimiento del proyecto.

Elementos de la gráfica

- Eje X: Fechas.

- Eje Y: Número de UT.

Herramienta

- La gráfica de Flujo Acumulado se encuentra en todas las herramientas estudiadas.

Ejemplo

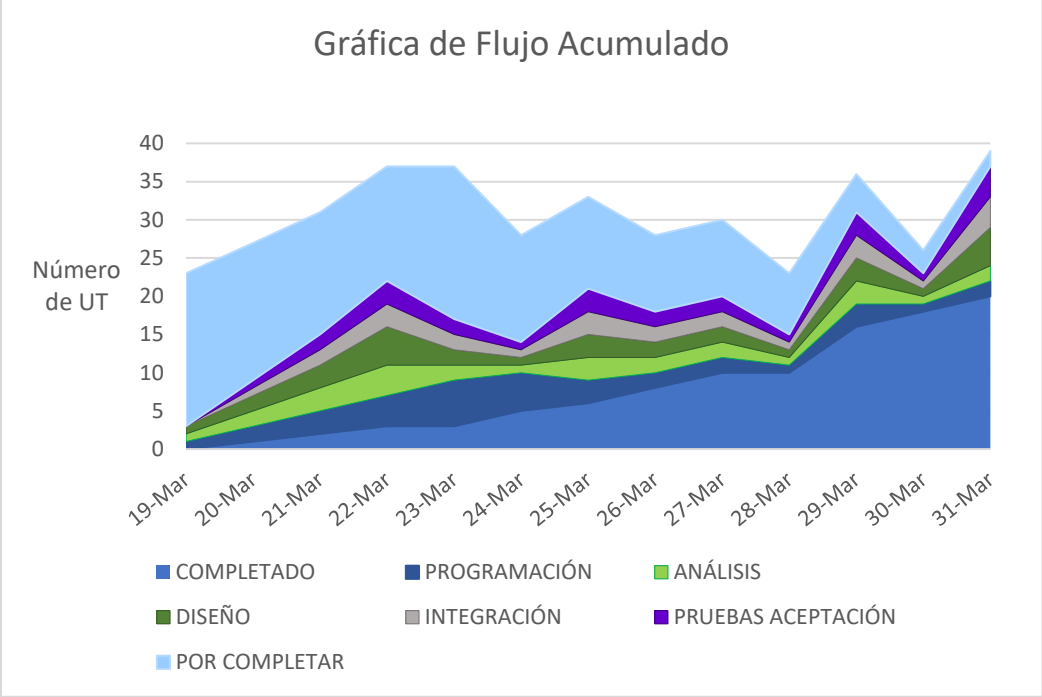

Figura 26. Gráfica de Flujo Acumulado

Interpretación

En la gráfica de Flujo Acumulado de la figura 26, se puede observar que a medida que van pasando los días el número de UT en ser completadas va aumentando proporcionalmente. Esto es un indicador de que el trabajo va fluyendo normalmente.

Si se observa el número de UT que están en el estado "Por Completar" el día 23 de marzo se puede apreciar un pico. Este pico refleja que una o varias UT fueron introducidas fuera de la fecha de la fecha

de inicio. Sin embargo, la mayoría de las UT llegan al estado de Completado en la fecha final que muestra la gráfica.

La gráfica de Flujo Acumulado es muy útil para distinguir entre proyectos/sprints que tienen un mejor o peor flujo de trabajo. En el caso de la figura 26 el flujo de trabajo es ideal, pues el número de UT por completar va disminuyendo mientras que el número de UT completadas aumenta.

8.18 Gráfica de Capacidad del Equipo por Tipo de Trabajo

Descripción General

La gráfica de Capacidad del Equipo por Tipo de Trabajo permite visualizar la capacidad del equipo por tipo de trabajo durante los sprints de un producto. Las franjas dentro de las barras de la gráfica indican el tiempo invertido en tipos de trabajo.

Los tipos de trabajo son:

- Desarrollo de UT.
- Desarrollo de UT no comprometidas (cambio de servidor, tareas que no están relacionadas con cambios para el producto).
- Preparación de UT para Sprints posteriores

Elementos de la gráfica

- Eje X: Sprints.
- Eje Y: Horas Invertidas.

Herramienta

- TUNE-UP Process

Ejemplo

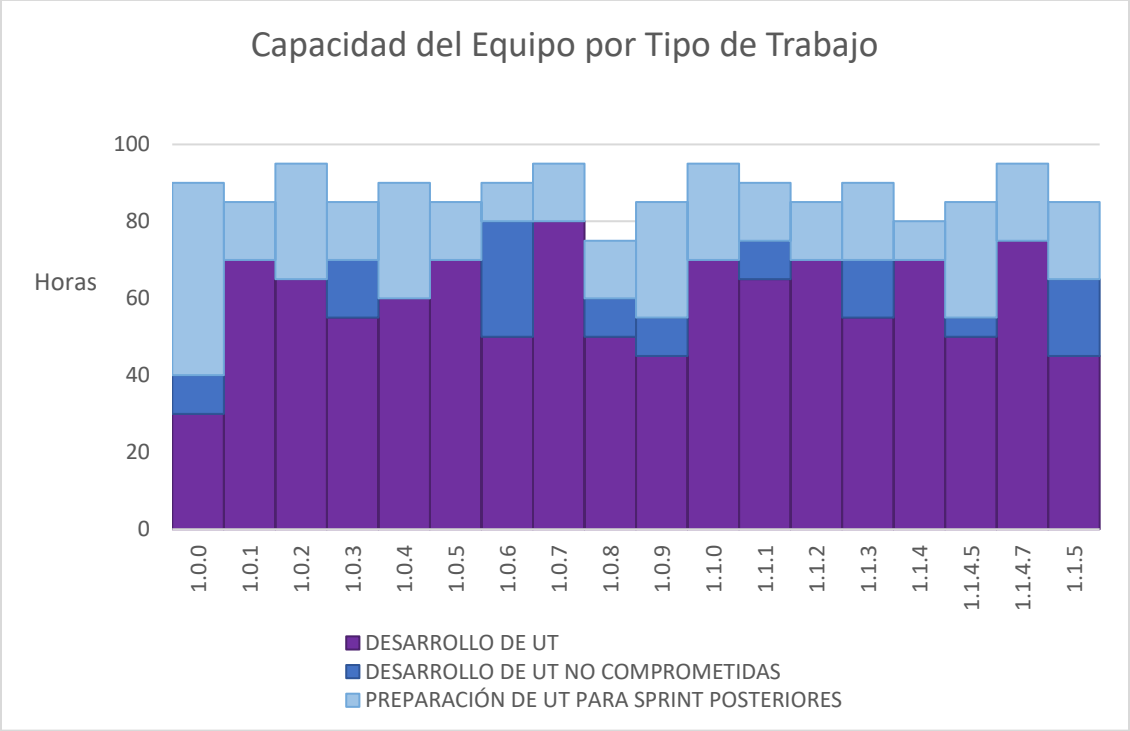

Figura 27. Gráfica de Capacidad del Equipo por Tipo de Trabajo

Interpretación

En la gráfica de Capacidad del Equipo por Tipo de Trabajo que se muestra en la figura 27 se puede observar que en el sprint 1.1.1 el equipo se dedicó en la mayor parte del sprint a desarrollar UT y una pequeña parte a realizar UT que no estaban comprometidas.

El Sprint 1.0.0, que se encuentra al principio de la gráfica, muestra que la mayor cantidad de tiempo se invirtió en la preparación de UT para sprints futuros. Esta situación es normal si el equipo se encuentra al principio de un proyecto o de un gran cambio dentro del mismo.

Esta gráfica es clave para estudiar la capacidad del equipo. Se podría alternativamente elaborar usando puntos.

8.19 Gráfica de Capacidad del Equipo por Tipo de UT en Sprints

Descripción General

La gráfica de Capacidad del Equipo por Tipo de UT permite visualizar la capacidad del equipo por tipo de UT durante los sprints de un producto. Las franjas dentro de las barras de la gráfica indican el tiempo invertido en tipos de UT.

Este tiempo está asociado a la actividad de "programación", pues en desarrollo de software lo más efectivo para el seguimiento de proyectos es contrastar horas de programación invertidas en una UT con las estimaciones en horas ideales de esa misma UT.

Los tipos de UT son:

- Nuevo requisito.
- Corrección de fallos.
- Mejoras.

Elementos de la gráfica

- Eje X: Sprints.
- Eje Y: Horas Invertidas.

Herramienta

- TUNE-UP Process

Ejemplo

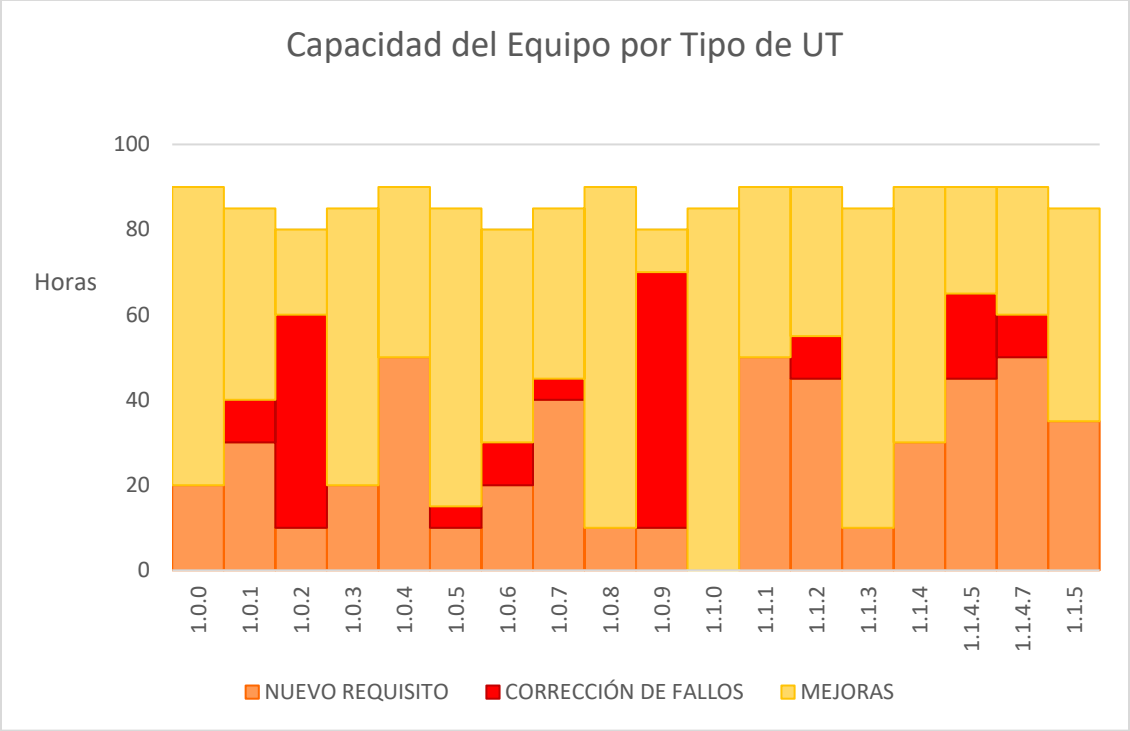

Figura 28. Gráfica de Capacidad del Equipo por Tipo de UT

Interpretación

En la gráfica de Capacidad del Equipo por Tipo de UT que se muestra en la figura 28 se puede observar que la mayoría del tiempo se ha invertido en mejores. Esto puede ser una indicación de que la base del producto está desarrollada y ahora el equipo se está dedicando a mejorarlo.

En el sprint 1.0.9 se observa que el equipo invirtió más de la mitad del tiempo en corregir fallos, probablemente procedentes del desarrollo de nuevos requisitos de los sprints anteriores.

8.20 Gráfica de Número de UT por Tipo en Sprints

Descripción General

La gráfica de Número de UT por Tipo permite visualizar el número de UT que se realizan dependiendo del tipo durante los sprints de un producto. Las franjas dentro de las barras de la gráfica indican el número de los diferentes tipos de UT.

Los tipos de UT son:

- Nuevo requisito.
- Corrección de fallos.
- Mejoras.

Elementos de la gráfica

- Eje X: Sprints.
- Eje Y: Número de UT.

Herramienta

- TUNE-UP Process

Ejemplo

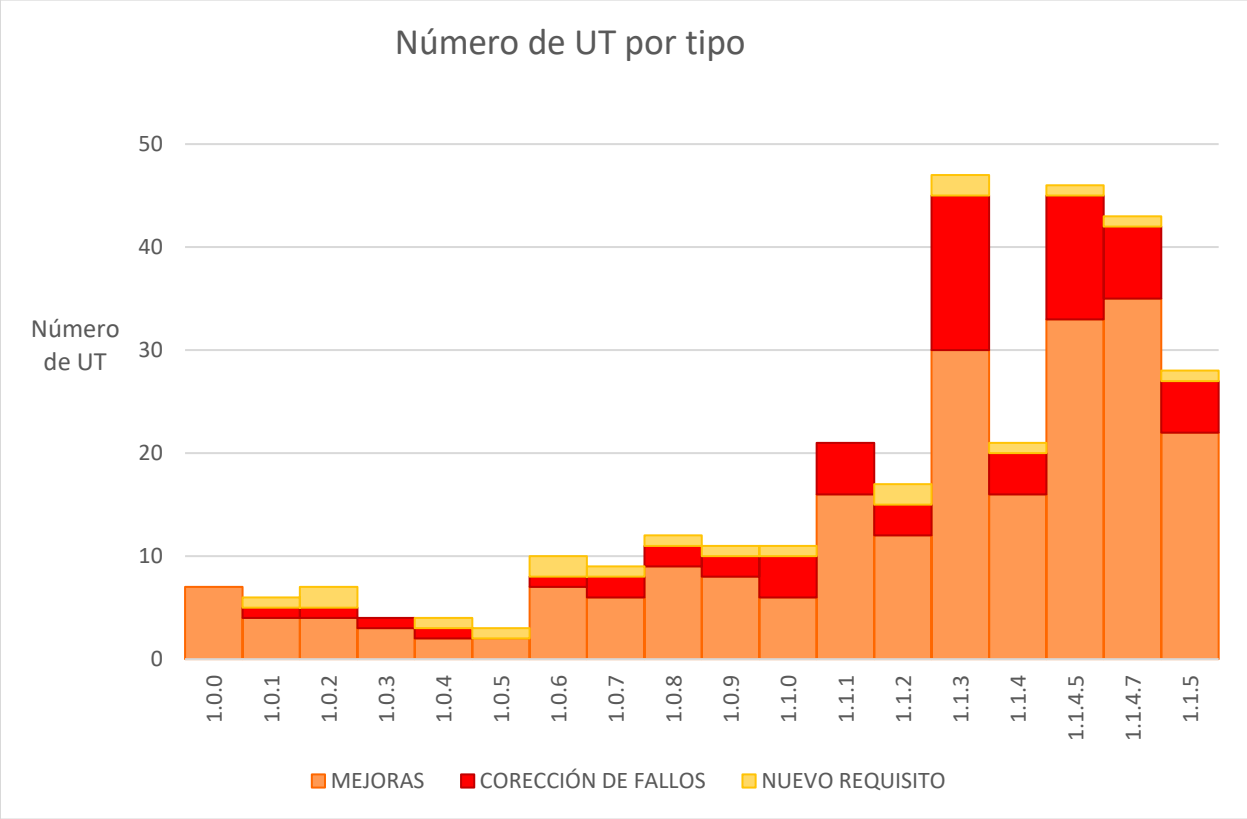

Figura 29. Gráfica de Número de UT por Tipo

Interpretación

En la gráfica de Número de UT por Tipo que se muestra en la figura 29 se puede observar al igual que en la figura 28, que el equipo ha trabajado mayormente en UT del tipo "mejoras".

Sin embargo, se debe tener en cuenta que generalmente no se invierte la misma cantidad de tiempo en todas las UT y los sprints pueden tener distinta duración. Por ejemplo, en el Sprint 1.1.2 se realizaron dos UT del tipo "nuevo requisito" que al comparar con el tiempo (35 horas invertidas) que se registró en la figura 28 se ve muy diferente.

8.21 Gráfica de Speedometer

Descripción General

En la gráfica de Speedometer se muestra si la velocidad del equipo es la adecuada para alcanzar el trabajo comprometido para el proyecto con relación a la fecha de finalización establecida.

La velocidad se basa en las horas que invierte el equipo para completar las UT, es decir, desde que se empieza la actividad de "programación" de las UT hasta que son publicada en producción.

Al tener un total fijo de horas a completar se divide entre los días hasta finalizar el proyecto. Por ejemplo, si se han estimado un total de 5000 horas para un proyecto que dura 180 días tendremos una velocidad de 27.78 horas/día. Si al pasar los primeros 100 días solo se han completado 2000 horas, la velocidad del equipo sería de 20 horas/día. Al estar por debajo de 27.78 horas/días quiere decir que el equipo no terminará a tiempo si continúa así.

Elementos de la gráfica

- Área roja: indica que el proyecto no será terminado a tiempo.
- Área amarilla: indica que el proyecto no será terminado a tiempo, pero con un poco más de esfuerzo sí.
- Área verde: indica que el proyecto será terminado a tiempo, o incluso antes

Herramienta

- Axosoft.

Ejemplo

Figura 30. Gráfica de Speedometer traducida desde https://blog.axosoft.com

Interpretación

En la gráfica de Speedometer que se muestra en la figura 30 se puede observar a simple vista que el equipo está a tiempo de completar el proyecto para la fecha establecida.

Si el Speedometer muestra la velocidad de un equipo en el área roja, esto quiere decir que el proyecto no será terminado a tiempo. Esta información puntual sobre la velocidad y el día permite que el equipo dé prioridad al desarrollo de UT fundamentales (UT que agregan el mayor valor al proyecto) y se asegure de que las UT fundamentales sean publicadas en producción antes de la fecha límite.

8.22 Información requerida para crear las gráficas

Esta tabla indica la información que el equipo necesita recolectar para poder generar una gráfica determinada.

Los * indican que se puede usar cualquiera de los datos, siendo obligatorio escoger alguno de ellos.

Mientras que los campos marcados con ✓ son obligatorios.

GRÁFICA	INFORMACIÓN NECESARIA							
	PUNTOS	HORAS IDEALES RESTANTES	HORAS IDEALES ESTIMADAS	HORAS INVERTIDAS	DIAS	PRUEBAS ACEPTACIÓN	KANBAN	SPRINT/ PROYECTO
Estado de Proyectos/Sprints	*	*	*	*				✓
Control de Proceso en Proyectos/Sprints				✓				✓
Burndown	*	*	*	✓			✓	✓
BurnUP				✓			✓	✓
UT Finalizadas vs No Finalizadas	*		*					✓
Entregables	✓						✓	✓
Edad Promedio					✓			
Esfuerzo por Actividad				✓			✓	✓
Edad Promedio de UT en cada Estado					✓		✓	✓
Esfuerzo por Tipo de Actividad				✓			✓	✓
Esfuerzo Asociado a Re-trabajo				✓				
Primera Estimación versus Esfuerzo Real			✓	✓				
Kos y OKs en Pruebas de Aceptación						✓		
Pruebas de Aceptación según Estado						✓		
Lead Time promedio de Fallos					✓			✓
Diagrama de Flujo Acumulado							✓	
Capacidad del Equipo por Tipo de Trabajo				✓				✓
Capacidad del Equipo por Tipo de UT				✓				✓
Número de UT por Tipo								✓
Speedometer		✓	✓	✓				✓

Tabla 2. Información requerida para crear las gráficas

8.23 Objetivo final de cada gráfica

La siguiente tabla muestra de forma resumida el objetivo de cada una de las gráficas estudiadas.

GRÁFICA	OBJETIVO
Estado de Proyectos/Sprints	Proporcionar al equipo una visión del estado positivo o negativo de los proyectos/sprints respecto al esfuerzo realizado.
Control de Proceso en Proyectos/Sprints	Controlar el estado de las UT, al clasificarlas en "buena", "sospechosa", y "mala" respecto a su cycle time.
Burndown	Mostrar diariamente la cantidad de esfuerzo restante.
BurnUP	Mostrar las horas invertidas por el equipo durante sprints/proyecto.
UT Finalizadas vs No Finalizadas	Comparar la proporción de trabajo finalizado y el no finalizado.
Entregables	Mostrar el estado de las UT respecto a las partes en que haya sido dividida.
Edad Promedio	Mostrar el número promedio de días desde que una UT fue creada hasta que se introduce en un sprint.
Esfuerzo por Actividad	Mostrar la cantidad de horas invertidas en cada actividad para cada UT.
Edad Promedio de Procesamiento por Estado	Mostrar el tiempo promedio de días que una UT pasa en un estado durante un sprint/proyecto.
Esfuerzo por Tipo de Actividad	Mostrar la proporción de horas invertidas en el tipo de actividad por el que pasa una UT.
Esfuerzo Asociado a Re-trabajo	Mostrar la proporción de re-trabajo por UT durante un sprint/proyecto.
Primera Estimación versus Esfuerzo Real	Comparar las horas estimadas inicialmente para una UT con las horas que se invirtieron en ella para su finalización durante un sprint/proyecto.
Kos y OKs en Pruebas de Aceptación	Mostrar la cantidad de fallos detectados en cada UT
Pruebas de Aceptación según Estado	Mostrar los estados de las pruebas de aceptación de una UT. Los estados son: identificada, definida, diseñada.
Lead Time promedio de Fallos	Mostrar el lead time promedio de los fallos corregidos en cada día.
Flujo Acumulado	Mostrar el ritmo de trabajo, si se produce cuellos de botella, y si no se trabaja.
Capacidad del Equipo por Tipo de Trabajo	Mostrar la capacidad media del equipo por tipo de trabajo por semana.
Capacidad del Equipo por Tipo de UT	Mostrar la capacidad media del equipo por tipo de UT por semana.
Número de UT por Tipo	Mostrar el número de UT que se realizan durante el sprint/proyecto de un producto dependiendo de su tipo.
Speedometer	Alertar al equipo si su velocidad actual permite que el proyecto finalice en la fecha establecida.

Tabla 3. Objetivo Final de la gráfica

Capítulo 9. Ejemplos de Dashboards y Cuadros de Mando

En este capítulo se introduce el concepto de Dashboard y se muestran ejemplos del mismo y ejemplos de Cuadros de Mando que se encuentran disponibles en las herramientas que fueron evaluadas.

Moreira define Dashboard como una forma de suministrar información donde se muestran las métricas claves que ayudan al equipo a entender hacia dónde va el proyecto. Moreira [16] recomienda una cantidad de tres a seis métricas que deben ser priorizadas según el valor que le aporten al cliente, la velocidad de entrega y la satisfacción.

Según Moreira [16], un dashboard provee al menos dos beneficios:

- Ver las métricas en un solo lugar y verlas lado a lado.
- Se puede correlacionarlas para asegurarse que no se está optimizando para alguna métrica en particular inapropiadamente.

Es muy importante que las métricas ayuden al equipo a tomar decisiones, y sirvan para determinar si la forma de trabajar se debe adaptar o mantener.

Para fines de este Trabajo de Fin de Máster la información mostrada en un Dashboard hace referencia al estado de un proyecto y mientras que la información mostrada en un Cuadro de Mando hace referencia al estado de todos los proyectos en una compañía. De las herramientas que se han escogido para evaluar, solo dos de ellas presentan un Dashboard ya construido para un proyecto y una un Cuadro de Mando.

En las siguientes secciones se muestra lo que ofrece cada una de las herramientas de seguimiento de proyectos escogidas.

9.1 Axosoft

Esta herramienta ofrece un Dashboard de sprint por defecto, como se muestra en la figura 31, pero no ofrece ningún Cuadro de Mando a nivel general.

El Dashboard contiene los siguientes elementos:

- Burndown Chart.
- Gráfico Circular (Pie Chart).
- Gráfica de Cycle Time.
- Speedometer.
- Velocidad Actual.
- Velocidad Requerida para terminar el proyecto.
- Fecha final del proyecto.

Figura 31. Dashboard Ejemplo (traducido desde www.axosoft.com)

9.2 TUNE-UP PROCESS

Esta herramienta ofrece un Dashboard generando automáticamente las gráficas de acuerdo a la información que haya sido recopilada por el equipo durante un Sprint/Proyecto, permitiendo que el equipo tenga una visualización instantánea sobre información referente a el esfuerzo invertido y el estado de las UT. En la figura 32 se puede observar un ejemplo.

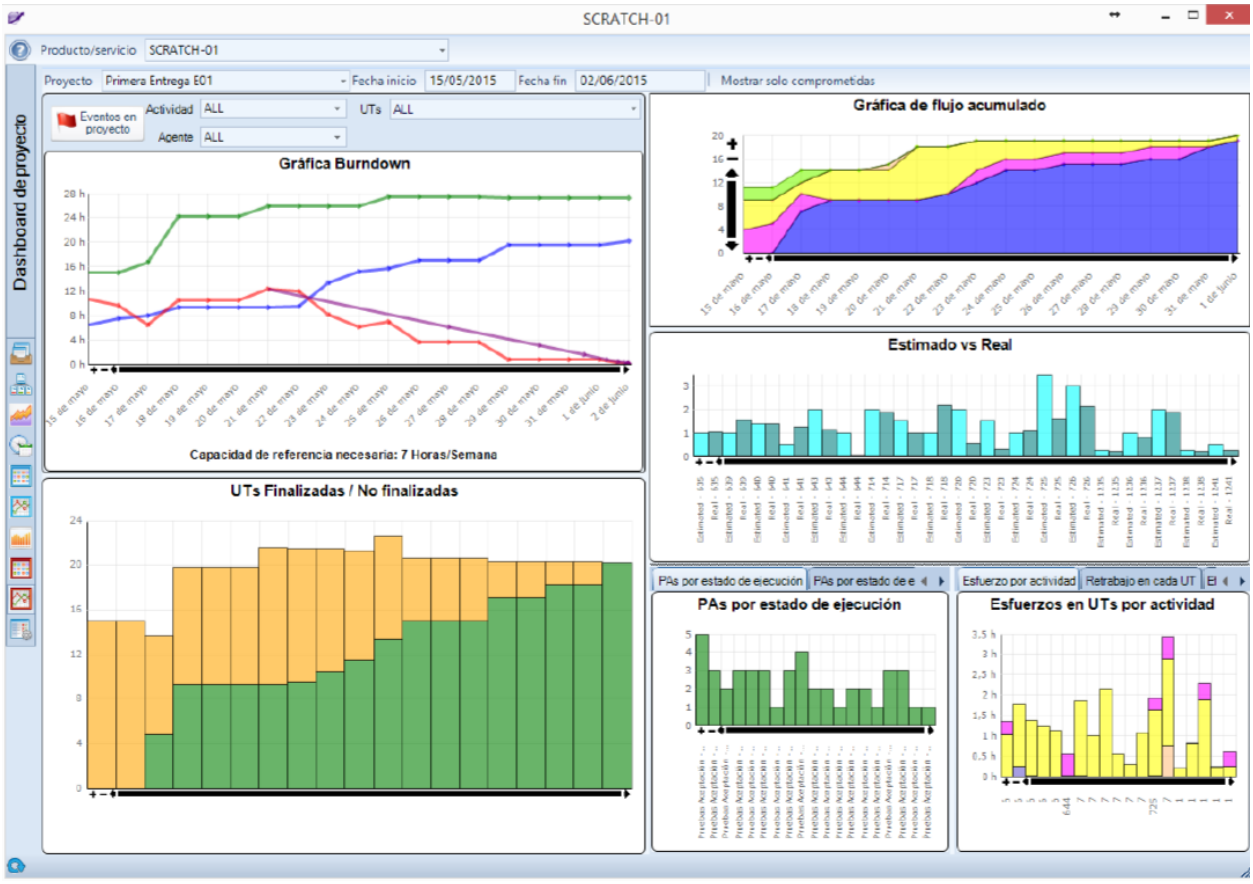

Figura 32. Dashboard TUNE-UP PROCESS

TUNE-UP PROCESS también ofrece un Cuadro de Mando. Este proporciona una visualización más general de las líneas de producto y proyectos que lleve el equipo o los equipos.

Según el manual de la herramienta *Help-Dashboard-Global,* cada una de las columnas que se muestran en figura 33 describe la siguiente información:

- Estado: se mide haciendo la diferencia entre el % Progreso (el porcentaje de tiempo registrado respecto del total de tiempo estimado) y el % Tiempo (el porcentaje de tiempo que ha pasado respecto del tiempo total de duración). Se utiliza un código de colores para indicar los siguientes rangos:
    - Muy bien: Verde claro. Diferencia > 20
    - Bien: Verde normal. -5 <= Diferencia <= 20
    - Regular: Amarillo. Diferencia < -5 y Diferencia >= -20
    - Mal: Naranja. Diferencia < -20 y Diferencia >= -40
    - Muy Mal: Rojo. Diferencia < -40
- Producto/Servicio y Sprint/Proyecto: indica los nombres de estos elementos.
- Flujo Acumulado: si la fecha de inicio y fin ha sido establecida para el sprint o proyecto se muestra el diagrama de flujo correspondiente a ese tiempo. De lo contrario, sólo se mostrará la información de las últimas dos semanas.
- Estimado: Tiempo estimado del Sprint o Proyecto.
- Registrado: Tiempo registrado en el Sprint o Proyecto.
- Restante: Tiempo restante del Sprint o Proyecto. Obtenido como la diferencia entre el tiempo estimado de cada actividad y su correspondiente tiempo registrado.
- Días restantes: Días que quedan desde la fecha actual hasta la fecha de término del Sprint o Proyecto.
- Inicio: Fecha de inicio del Sprint o Proyecto.
- Fin: Fecha de término del Sprint o Proyecto.
- Duración: Días del Sprint o Proyecto.
- % Tiempo: Porcentaje de tiempo transcurrido desde el inicio del Sprint o Proyecto respecto de la duración del Sprint o Proyecto.

- % Progreso: Porcentaje de tiempo registrado respecto del tiempo estimado del conjunto de actividades estimables en cada una de las UTs del Sprint o Proyecto.
- Advertencias: aquí se indican posibles anomalías que afectan al Sprint o Proyecto en cuanto a la interpretación de su estado.

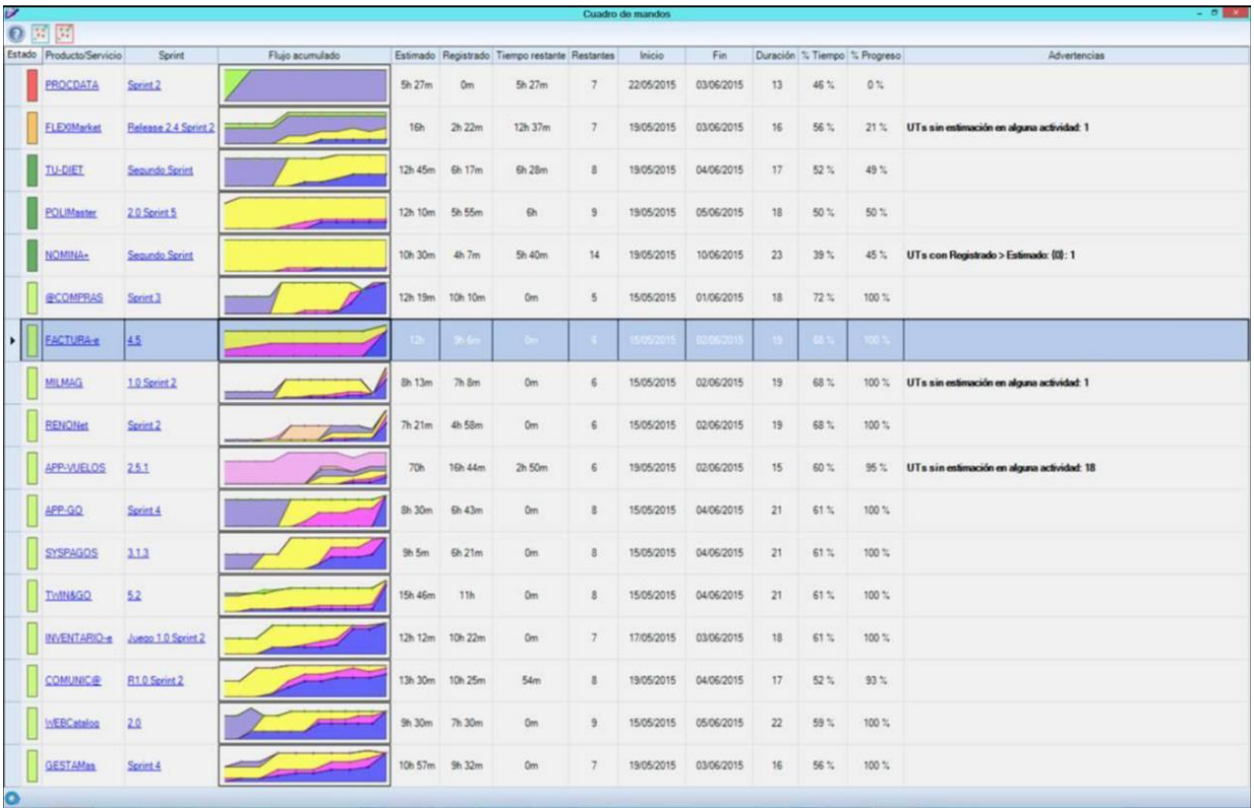

Figura 33. Cuadro de Mando TUNE-UP PROCESS

9.3 JIRA

Esta herramienta ofrece gadgets y plugins para visualizar las gráficas que estén disponibles y con ellas construir un Dashboard.

- Los gadgets permiten agregar las gráficas propias de JIRA tal como el usuario decida.

- Los plugins pueden tener un set de gráficas y datos ya estructurados o darle la posibilidad al usuario de costumizarlo. Estos plugins son desarrollados por compañías externas a JIRA.

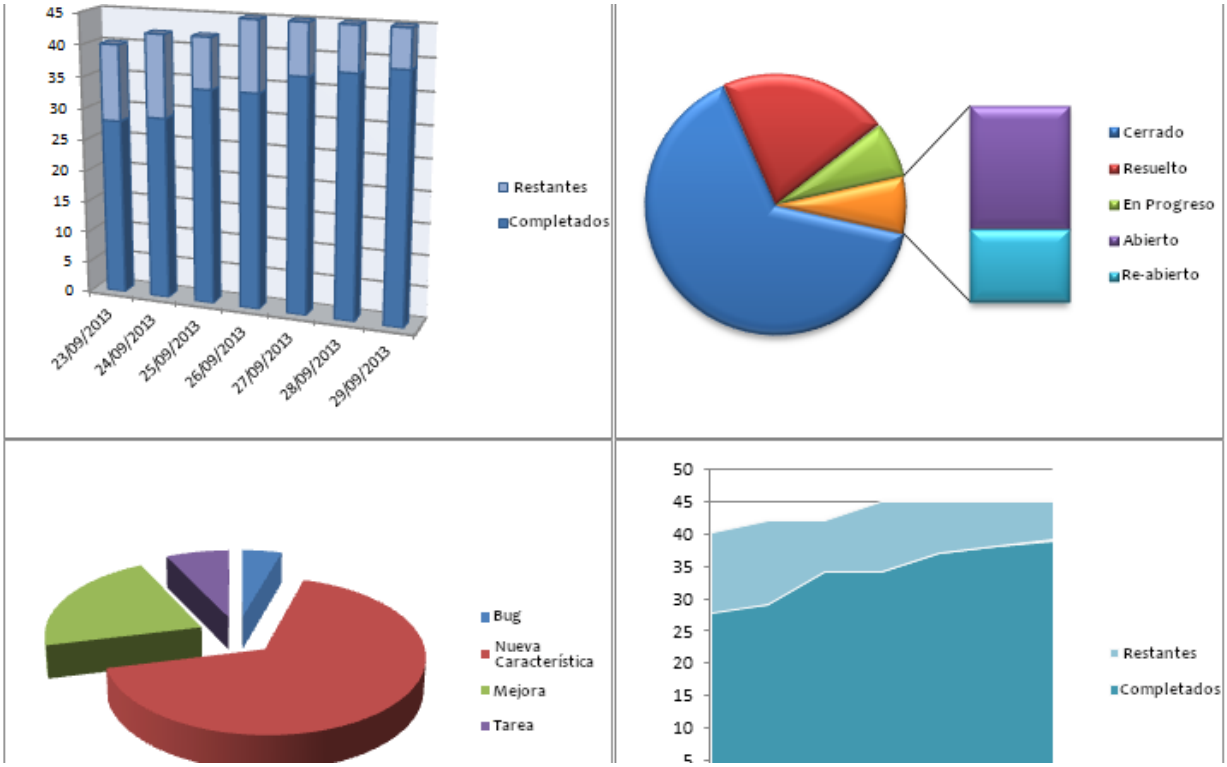

Figura 34. Dashboard plugin Intelligent Reports (traducida desde https://marketplace.atlassian.com)

Un ejemplo de estos plugins, es el plugin de "Intelligent Reports", que se muestra en la figura 34.

La construcción de un Cuadro de mando también es posible a través de plugins y customización del usuario.

9.4 MINGLE

Esta herramienta permite que el equipo construya un Dashboard o Cuadro de Mando de una forma customizable a través de configuraciones básicas y programación.

En la figura 35 se puede ver un ejemplo de un Dashboard.

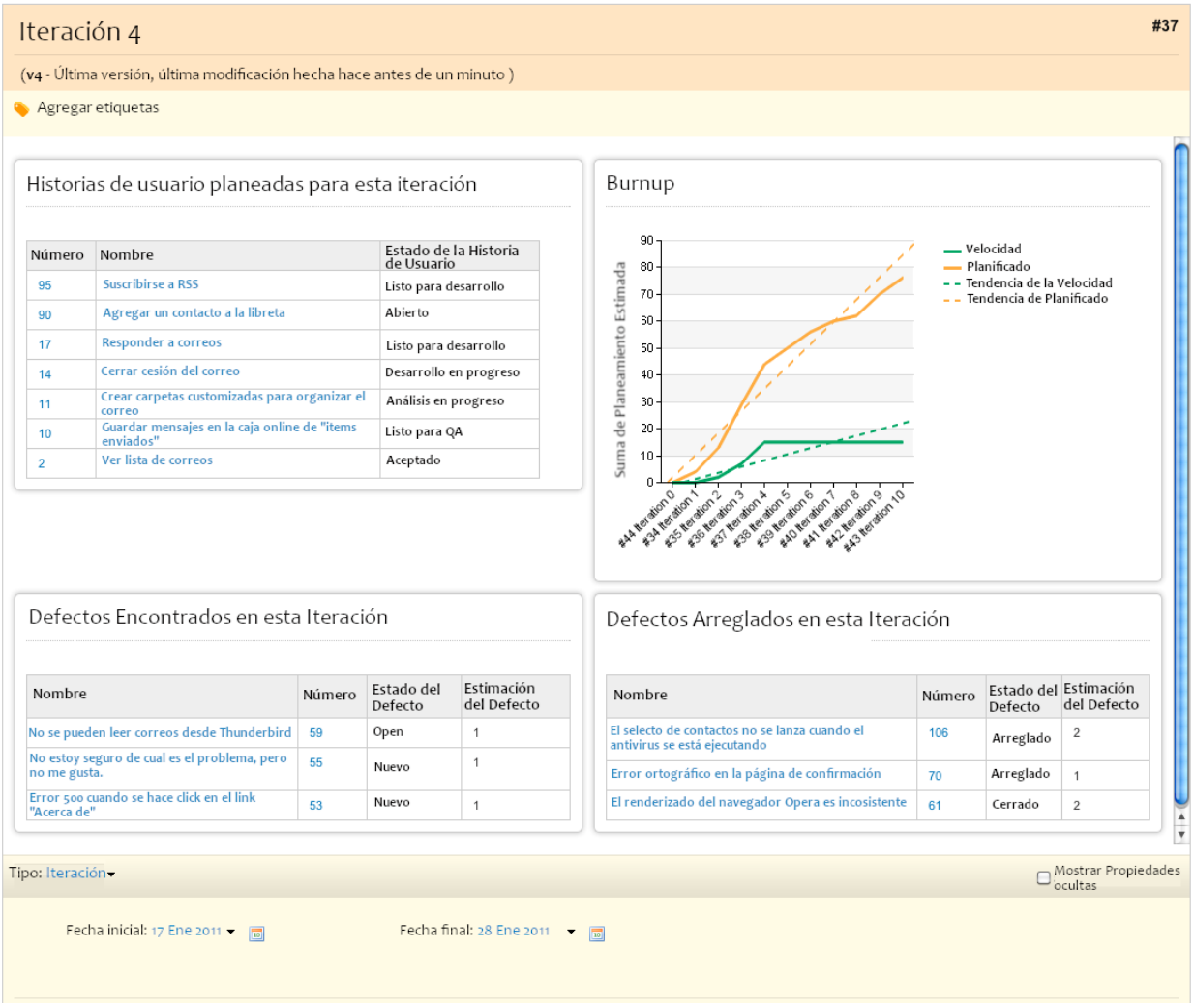

Figura 35. Ejemplo Dashboard Mingle (traducido desde www.thoughtworks.com)

9.5 TargetProcess

Esta herramienta permite al usuario construir un Dashboard o Cuadro de Mando a partir de widgets, vistas y gráficas propias de la herramienta. Además, da acceso al mismo en diferentes niveles: privado, público y customizado.

TargeProcess [14] ofrece en su página web ejemplos basados en el objetivo del Dashboard. Entre ellos:

- **Iteración/Relase:** este dashboard tendría información crítica relacionada con la finalización del Sprint (Burndown chart, impedimentos, comentarios recientes, bugs).
- **QA:** este dashboard mostaría toda la información referente a la calidad en proyectos (nuevos bugs, bugs urgentes abiertos, bugs no asignados, tendencias).
- **Equipo:** este dashboard se centraría en cómo le está yendo al equipo.
- **Proyecto:** este dashboard se centraría en cómo yendo el proyecto.
- **Soporte:** este dashboard mostraría información relacionada con actividades de soporte (nuevas peticiones, tendencias SLA, peticiones abiertas urgentes).

La figura 36 muestra un ejemplo de Dashboard ya creado.

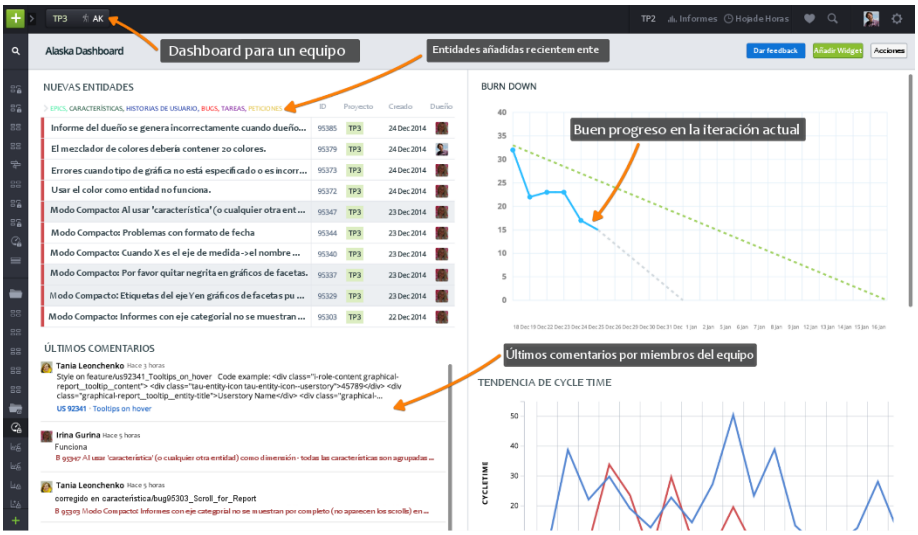

Figura 36. Dashboard Equipo (traducido desde www.targetprocess.com)

Capítulo 10. Propuesta de Cuadro de Mando y Dashboard

En este capítulo se presenta una Propuesta de Cuadro de Mando y Dashboard resultado del análisis de las gráficas evaluadas por las herramientas escogidas.

En muchas ocasiones los equipos son interrogados sobre cuándo empezarán a trabajar en una característica en específico cuando aún no han tenido la oportunidad de terminar la anterior. Otras veces les piden acelerar el desarrollo de nuevas características, esperando que también se finalice el trabajo ya planeado, llevando esto finalmente a trabajar tiempo extra.

Las gráficas permiten al equipo visualizar irregularidades. El análisis de estas posibilita la toma de acciones de forma proactiva para resolver problemas que puedan surgir. Las gráficas deben permitir al equipo tomar mejores decisiones.

Cabe resaltar que los *stakeholders* usualmente no están familiarizados con los conceptos de puntos u horas ideales, por lo que gráficas que contengan estos datos resultan solo útiles para el equipo. La cantidad de unidades y el tiempo transcurrido son conceptos más simples y generales para los *stakeholders*. [38]

El acceso a la información que proporciona el Cuadro de Mando debe permitir un flujo de información constante y disponible sin necesidad de interrumpir al equipo para obtenerla. Por esta razón gráficas que sin duda aportan información útil para el equipo a nivel de proyecto, pero no proporcionan información concreta y concisa del portafolio de proyectos no fueron incluidas en el diseño del Cuadro de Mando.

El fin de presentar esta Propuesta de Cuadro de Mando es ofrecer a los equipos de desarrollo de software una solución general para hacer seguimiento adecuado a sus proyectos, independiente de la metodología ágil que implementen y el tipo de proyecto a desarrollar.

El Cuadro de mando mostraría los siguientes elementos:

- Proyecto: indica el nombre del proyecto.
- Sprint: indica el nombre del sprint actual del proyecto. Si el proyecto usa Kanban, el valor de este elemento será Kanban.
- Flujo Acumulado: si la fecha de inicio y fin ha sido establecida para el sprint o proyecto se muestra la gráfica de flujo acumulado correspondiente a ese tiempo. De lo contrario, sólo se mostrará la información de las últimas tres semanas.
- Inicio: Fecha de inicio del Sprint o Proyecto.
- Fin: Fecha de término del Sprint o Proyecto.
- % Tiempo: Porcentaje de tiempo transcurrido desde el inicio del Sprint o Proyecto respecto de la duración del Sprint o Proyecto.
- % Progreso: Porcentaje de tiempo registrado respecto del tiempo estimado del conjunto de actividades estimables en cada una de las UTs del Sprint o Proyecto.
- Speedometer en caso de tener fechas establecidas:
    - Rojo: Porcentaje de tiempo es 6 % mayor que el porcentaje de progreso.
    - Amarillo: Porcentaje de tiempo es igual o 5 % mayor al porcentaje de progreso.
    - Verde: Porcentaje de tiempo es menor o igual al porcentaje de progreso.

Los porcentajes se definen de la siguiente forma:

- Porcentaje de tiempo: Porcentaje de tiempo transcurrido desde el inicio del Sprint o Proyecto respecto de la duración del Sprint o Proyecto.
- Porcentaje de Progreso: Porcentaje de tiempo registrado respecto del tiempo estimado del conjunto de actividades estimables en cada una de las UTs del Sprint o Proyecto.

Los elementos que fueron escogidos resumen de forma concisa la información más relevante a cerca de un proyecto/sprint. Datos como el nombre, sprint y fechas son imprescindibles y proveen el contexto de cada proyecto/sprint. El porcentaje de tiempo y progreso son indicadores claros y simples del avance del proyecto, mientras que la gráfica escogida transmite el estado actual del proyecto/sprint.

La gráfica de flujo acumulado es una herramienta indispensable para el seguimiento de proyectos de software, pues independiente de la metodología que se utilice, todos los proyectos/sprints se componen de UT. La gráfica permite visualizar de forma concisa y coherente el WIP (*Work in progress*) mediante la distancia vertical entre las bandas. En la gráfica de flujo acumulado cada franja muestra la cantidad de UT que se contabilizan día a día en cada actividad. Esto es útil para reasignar recursos cuando una actividad presenta cuellos de botella.

Por último, el Speedometer indica al equipo si el proyecto/sprint está riesgo de no completarse a tiempo. Esta información permite que el equipo realice los ajustes necesarios para finalizar el proyecto o completar las UT más importantes para el producto.

A continuación, la representación gráfica de la propuesta de Cuadro de Mandos.

Proyecto	Sprint	Flujo Acumulado	Fecha Inicio	Fecha Fin	% Tiempo	% Progreso	Speedometer
Proyecto A	Sprint 1		20-05-19	30-05-19	83	0	
Proyecto B	Sprint 5		16-05-19	01-06-19	58	40	
Proyecto C	Sprint 3		16-05-19	01-06-19	58	58	
Proyecto D	Sprint 10		15-05-19	15-06-19	33	40	
Proyecto E	Kanban		04-05-19	25-05-19			

Figura 37. Propuesta de Cuadro de Mando

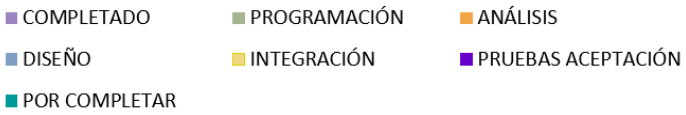

Figura 38. Leyenda gráfica de Flujo Acumulado

En el cuadro de mando de la figura 37 se muestran cinco proyectos evaluados a nivel de Sprint con respecto al día 25-05-19:

- El proyecto A muestra que no se ha completado ninguna UT del sprint 1, y como consecuencia el total de las UT asignadas al sprint 1 no se completarán.
- El proyecto B muestra que se han completado algunas UT, sin embargo, el speedometer refleja que con el ritmo actual no se completarán las UT asignadas al sprint 5.
- El proyecto C muestra en la gráfica de flujo acumulado que a medida que ha transcurrido el sprint 3 la franja turquesa, que indica las UT por completar, ha ido reduciéndose mientras que la franja lila, que indica las UT completadas, ha ido creciendo en grosor. Lo que revela

que el proyecto tiene un buen ritmo. Sin embargo, este ritmo puede no ser suficiente para completar todas las UT al finalizar el sprint.

- El proyecto D, al igual que el proyecto C muestra un buen ritmo, con la diferencia de que el sprint no está pronto a finalizar, por lo que permitiría completar las UT restantes.
- El proyecto E, a diferencia de los proyectos anteriores no tiene sprints, por lo que se evalúa tres semanas anteriores a la fecha mencionada. El día 04-05-19 se tienen en el "backlog" 20 UT, y a medida que transcurre el tiempo estas UT se finalizan y se agregan otras seis UT, tres de ellas ya completas y tres restantes en el "backlog".

Los equipos internamente pueden hacer uso de un sin número de gráficas para el seguimiento y la mejora de sus proyectos. Sin embargo, si se pretende establecer un flujo de comunicación con niveles superiores, se debe hablar el mismo lenguaje que los *stakeholders* para transmitir de manera transparente si un proyecto está bien encaminado o no.

10.1 Propuesta de Dashboard

Al principio de este capítulo se hace mención de la existencia de gráficas que aportan información útil al equipo, pero no para los *stakeholders* debido a la interpretación de conceptos de medición provenientes de metodologías ágiles como puntos y horas ideales.

Agilist Andre [37] expone que los equipos que implementan metodologías ágiles se caracterizan por tener una mentalidad donde:

- Las decisiones son tomadas por el equipo que realiza el trabajo.
- Los directores de proyecto fomentan un espacio donde el equipo puede hacer su mejor trabajo.

- La meta es entregar trabajo finalizado que aporte valor al proyecto o sprint.
- Los programadores no son tratados como recursos, sino como individuos con capacidad de realizar su trabajo y de aprender.

Los equipos ágiles tienen autonomía sobre la toma de decisiones preventivas o correctivas durante un proyecto o sprint. Por esta razón es de suma importancia que tengan herramientas que les permitan hacer el mejor trabajo posible. Dentro de las herramientas se encuentra el Dashboard cuyas gráficas resumen información que es de gran valor para el equipo.

El Dashboard mostraría los siguientes elementos:

- Proyecto: indica el nombre del proyecto.
- Sprint: indica el nombre del sprint actual del proyecto. Si el proyecto usa Kanban, el valor de este elemento será Kanban.
- Gráfica Burndown: muestra diariamente la cantidad de esfuerzo restante.
- Gráfica Esfuerzo por Actividad: muestra la cantidad de horas invertidas en cada actividad para cada UT.
- Gráfica Primera Estimación vs. Esfuerzo Real: compara las horas estimadas inicialmente para una UT con las horas que se invirtieron en ella para su finalización durante un sprint/proyecto.
- Gráfica KOs y OKs en Pruebas de Aceptación: muestra la cantidad de fallos detectados en cada UT.

Al igual que para el Cuadro de Mando, la información sobre el nombre del Proyecto y Sprint es esencial. Mientras que las 4 gráficas escogidas muestran información relacionadas con el esfuerzo por UT y actividad, estimación y pruebas de aceptación.

La gráfica de Burndown permite que el equipo evalúe su esfuerzo diariamente. Al tener en el eje horizontal la fecha puede observar progresivamente la cantidad de esfuerzo que le queda por completar para finalizar el sprint o el proyecto.

La gráfica de Esfuerzo por Actividad permite que el equipo estudie el tiempo dedicado en diferentes actividades y analice si este tiempo ha aportado valor al proyecto o sprint. En el caso de la gráfica presentada en la figura 39 las actividades incluidas son: Diseño, Programación y Pruebas. Esto puede ser modificado según las necesidades del equipo.

La gráfica de Primera Estimación vs. Esfuerzo Real permite que el equipo examine su precisión con respecto a las estimaciones y tenga para el futuro un punto de referencia más preciso y real como producto de los desarrollos previamente implementados.

La gráfica de KOs y OKs en Pruebas de Aceptación permite que el equipo controle de forma proactiva la cantidad de fallos que puede presentar una UT durante su desarrollo y tome las medidas necesarias para que las UT estén libres de errores.

A continuación se presenta la propuesta de Dashboard para la duración de un sprint o proyecto.

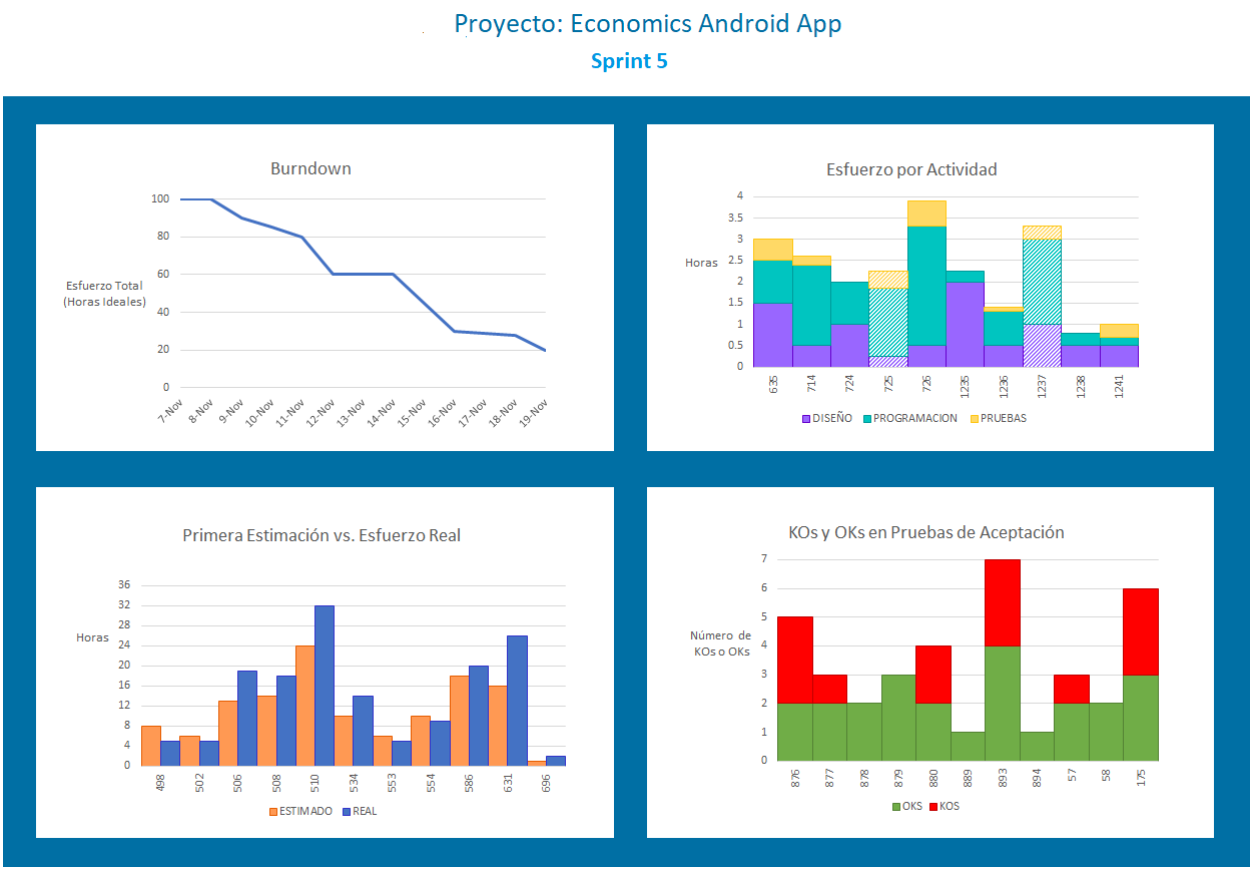

Figura 39. Propuesta de Dashboard

Los equipos que implementan metodologías ágiles normalmente tienen un evento llamado retrospectiva. En este evento se evalúa el desempeño del equipo durante el sprint o proyecto. Esta tarea resulta más sencilla cuando se tienen datos visuales del progreso del proyecto en el tiempo evaluado. El equipo puede apoyarse en el Dashboard y el Cuadro de Mando para la toma de decisiones con respecto al desarrollo futuro del proyecto.

Capítulo 11. Conclusiones y Trabajo Futuro

Estudios demuestran que la adopción de un cuadro de mando ayuda a la organización a eliminar tiempo y esfuerzo malgastado [30].

La recopilación de información relevante a métricas es una parte fundamental del mundo del software. Las personas que se desempeñan en este ambiente dependen y se apoyan en las métricas para construir Cuadros de Mando o Dashboards que les permiten hacer seguimiento a sus proyectos de software. Sin embargo, la elección de los datos en los que se debe confiar no es una tarea sencilla. Se puede elegir una métrica correcta y llegar a una conclusión errónea o simplemente elegir una métrica incorrecta, que no aporte información valiosa para el proyecto.

Durante el análisis de los tipos de gráficas e indicadores pude corroborar que la oferta de estas gráficas en las herramientas varía significativamente. Las dos gráficas más frecuentes son el Burndown y el Flujo acumulado. Sin embargo, información tan esencial como la que puede proporcionar la gráfica de Esfuerzo por Actividad se encuentra solo en TUNE-UP Process. A partir de esto pude concluir que cada herramienta ofrece gráficas que, a su criterio, encuentra más útil para el equipo o que son más ampliamente utilizadas, como es el caso del Burndown. Esto a su vez, es resultado de la falta de estandarización y consenso con respecto a qué tipo de diagramas y métricas se deben generar para el seguimiento de proyectos que usan metodologías ágiles.

La creación de ejemplos de las gráficas analizadas me permitió determinar la utilidad de las mismas en el seguimiento de proyectos. Por ejemplo, la gráfica de Entregables no es útil para el seguimiento de proyectos, pues no presenta de forma clara el estado de las UT. En esta gráfica los puntos de una UT se reparten entre los que han sido completados, los que están en progreso y los que aún quedan por completar. Pero ¿cómo puede un programador determinar cuántos puntos quedan por hacer de una UT? Las gráficas deben ofrecer datos precisos y simples que ayuden al equipo, no que lo confundan.

Otro ejemplo son las gráficas de Edad Promedio y Tiempo Promedio de Procesamiento por Estado, si bien no confunden al equipo como la gráfica de Entregables, sus datos pueden verse afectados por valores extremos que distorsionan la información y como resultado no ofrecen datos precisos.

A excepción de las gráficas mencionadas anteriormente, las demás gráficas ofrecen información útil. Sin embargo, en algunos casos, como es el de la gráfica de Tiempo de Resolución de Problemas por Proyecto, su utilidad solo se aplica a escenarios específicos.

La mayoría de las herramientas que fueron evaluadas ofrecen múltiples opciones de Dashboards que al final del día pueden confundir al equipo y no ofrecerle información valiosa. No obstante, TUNE-UP ofrece un cuadro de mando concreto que se centra en la entrega de un producto finalizado. Este enfoque permite que el equipo tome decisiones que redirijan el esfuerzo empleado en la dirección correcta.

Como resultado del análisis realizado y los escenarios imaginarios construidos, el diseño del cuadro de mando y dashboard propuestos se basan en métricas que están alineadas con la entrega constante de las unidades de trabajo de un proyecto, tiempo empleado en el desarrollo del proyecto, y su tiempo de finalización si tienen uno. El valor está principalmente enfocado en los incrementos que se entregan progresivamente y van formando el producto. Esto ayuda a tomar decisiones que establecen la entrega del producto como el foco principal del esfuerzo que emplea el equipo.

Mi experiencia profesional como Scrum Master en el campo de desarrollo de proyectos de software me ha enseñado que la recolección de datos por parte de los programadores no es una tarea fácil, pues en muchos casos sienten que pierden su tiempo y expresan su disgusto a la hora de hacerlo. Por esta razón ha sido importante para mí elegir gráficas que faciliten el trabajo de todo el equipo y agreguen valor a la gestión del proyecto.

En el tiempo que trabajaba como directora de proyectos que implementaban metodologías ágiles, hacía uso de las gráficas Burndown. Al final de cada sprint, durante la retrospectiva junto al equipo evaluábamos la gráfica y discutíamos sobre su rendimiento. En múltiples ocasiones el equipo era capaz de reconocer o resaltar situaciones especiales, fueran negativas o positivas, pues su efecto se podía ver claramente en la gráfica. Al tener esta información de forma visual el equipo tomaba ciertas medidas para mejorar o tomaba nota de prácticas que debía mantener. La gráfica de Burndown también era enviada a los *stakeholders,* pero en ningún momento el equipo recibió feedback sobre estas, pues el cliente no estaba familiarizado con la gráfica ni con el concepto de puntos.

Durante el máster las asignaturas de Modelado de Proceso de Negocio y Organizacional, Ingeniería del Software Avanzada, Ingeniería del Software Experimental, y Emprendimiento en productos software fueron de gran ayuda por su contenido para desarrollar este Trabajo de Fin de Máster.

El desarrollo de este Trabajo de Fin de Máster ha sido una experiencia enriquecedora de muchas formas. Ha puesto a prueba mis conocimientos sobre las metodologías ágiles y la gestión de proyectos. A pesar de tener experiencia gestionando equipos de software que hacían uso de las metodologías Scrum y Kanban, el seguimiento de proyectos basado en métricas era algo secundario pues solo hacía uso de la utilidad de las gráficas Burndown. Con la información que he recopilado durante esta experiencia he aprendido que las gráficas son herramientas altamente beneficiosas de las que haré más uso en el futuro.

11.1 Trabajo Futuro

Uno de los pilares en común que tienen las metodologías ágiles, es la mejora continua del equipo y del producto. Con respecto al diseño de Cuadro de Mando propuesto, estas pueden ser las posibles mejoras:

- Configurar el diseño de Cuadro de Mando y el Dashboard en una herramienta usada en equipos reales que implementen metodologías ágiles.
- Tras la puesta en práctica, evaluar si las gráficas y métricas seleccionadas a raíz de este trabajo de fin de máster son las adecuadas y útiles para diferentes tipos de proyectos que implementen diferentes metodologías ágiles.
- También así, como resultado de la puesta en práctica, reajustar el diseño del Dashboard y Cuandro de Mando en casos muy específicos donde otras métricas agreguen más valor al seguimiento dado el tipo de proyecto y de metodología implementada.

Referencias

1. Rose, Doug. Leading Agile Teams. 2015.
2. “Process Control Chart.” Targetprocess - Visual Management Software, www.targetprocess.com/guide/track-measure-progress/lean-kanban-flow-efficiency/process-control-chart/. Fecha de acceso: 25 Ene 2019.
3. Wong, Kenny. "Reviews & Metrics for Software Improvements". University of Alberta. 2016.
4. “Burnup Chart.” Advanced Searching - Atlassian Documentation, confluence.atlassian.com/jirasoftwarecloud/burnup-chart-945124716.html. Fecha de acceso: 25 Ene 2019.
5. “Sprint Report.” Advanced Searching - Atlassian Documentation, confluence.atlassian.com/jirasoftwarecloud/sprint-report-777002722.html. Fecha de acceso: 25 Ene 2019.
6. “Average Age Report.” Advanced Searching - Atlassian Documentation, confluence.atlassian.com/jira064/average-age-report-720416061.html. Fecha de acceso: 25 Dec 2018.
7. “Historical Visual Reports for Processing Time and Custom Cycle Time.” Targetprocess - Visual Management Software, www.targetprocess.com/guide/reports/visual-reports/visual-reports-cumulative-burnup-burndown-examples-use-cases/historical-visual-reports-processing-time-custom-cycle-time/#average-processing-time-per-state. Fecha de acceso: 25 Ene 2019.
8. Kakadia; Deepak, Polehn; Donna L., Kotecha; Lalit R., Macias; John F., Lam; Maria. Service level agreement (SLA) based provisioning and management. US 20150156082 A1, United States Patent and Trademark Office, 4 de Junio 2015. USPTO Patent Full-Text and Image Database, http://patft.uspto.gov/netacgi/nph-

Parser?Sect1=PTO1&Sect2=HITOFF&d=PALL&p=1&u=%2Fnetahtml%2FPTO%2Fsrchnum.htm&r=1&f=G&l=50&s1=9191282.PN.&OS=PN/9191282&RS=PN/9191282

9. "Speedometer." Dev Blog by Axosoft, Dev Blog by Axosoft, blog.axosoft.com/tag/speedometer/. Fecha de acceso: 25 Ene 2019.
10. "Product Tour - Agile Software | Axosoft." Axosoft.com, www.axosoft.com/tour#release-confidently. Fecha de acceso: 14 Mar 2019.
11. Letelier, P. (2017) TUNE-UP Process: Help-Vista-Dashboard-Sprint. Valencia, ES: TUNE-UP Process.
12. Letelier, P. (2017) TUNE-UP Process: Help-Dashboard-Global. Valencia, ES: TUNE-UP Process
13. Technology, Clarios. "Intelligent Reports." Atlassian Marketplace, marketplace.atlassian.com/apps/1211118/intelligent-reports?hosting=server&tab=overview. Fecha de acceso: 10 Mar 2019.
14. "Using Cards for Reporting." ThoughtWorks, www.thoughtworks.com/mingle/docs/tip_metrics_on_cards.html. Fecha de acceso: 25 Ene 2019.
15. "Use Cases and Examples of Dashboard Configuration." Targetprocess - Visual Management Software, www.targetprocess.com/guide/boards/dashboards/dashboard-use-cases/. Fecha de acceso: 25 Ene 2019.
16. Moreira, Mario E. The Agile Enterprise: Building and Running Agile Organizations. Apress L. P, 2017.
17. Balanced Scorecard Institute, www.balancedscorecard.org/.
18. "What Is the Balanced Scorecard?" Balanced Scorecard Institute, www.balancedscorecard.org/BSC-Basics/About-the-Balanced-Scorecard. Fecha de acceso: 8 Abr 2019.

19. PMBOK® Guide – Fifth Edition (2013).
20. “Project Management Institute.” Wikipedia, Wikimedia Foundation, 17 Dic. 2018, en.wikipedia.org/wiki/Project_Management_Institute. Fecha de acceso: 17 Dic 2019.
21. Wong, Kenny. "Agile Planning for Software Products". University of Alberta. 2016
22. Sutherland, Jeff, et al. “Manifesto for Agile Software Development.” History: The Agile Manifesto, agilemanifesto.org/. Fecha de acceso: 23 Ene 2019.
23. “What Is a Gantt Chart?” Gantt, www.gantt.com/. Fecha de acceso: 23 Ene 2019.
24. “Gantt Chart Template: Project Management Timeline.” Lucidchart, lucidchart.zendesk.com/hc/en-us/articles/360001139163-Gantt-Chart-Template-Project-management-timeline. Fecha de acceso: 23 Ene 2019.
25. Bunner, Angela. “Project Timelines: Why They're So Important.” Clarizen, 5 Oct. 2016, www.clarizen.com/project-timelines-why-theyre-so-important/. Fecha de acceso: 23 Ene 2019.
26. Poppendieck, Mary, et al. Lean Software Development: An Agile Toolkit. 2003.
27. Cole, Rob, and Edward Scotcher. Brilliant Agile Project Management: a Practical Guide to Using Agile, Scrum and Kanban. 2015.
28. Wong, Kenny. "Software Processes and Agile Practices". University of Alberta. 2016.
29. Letelier, P. (2017) TUNE-UP Process: Guia-Inicio-Rapido-Cliente. Valencia, ES: TUNE-UP Process.
30. Alsadeq, I., Fatehy, T., & Othman, O. (2009). PMI® and BSC marriage! Where can PMI standards meet balanced scorecard? Paper presented at PMI® Global Congress 2009—EMEA, Amsterdam, North Holland, The Netherlands. Newtown Square, PA: Project Management Institute.

31. “What Is a Key Performance Indicator (KPI)?” Kpi.org, kpi.org/KPI-Basics. Fecha de acceso: 14 Mar 2019.
32. Letelier, Patricio. “Diagramas De Gantt Para Planificación y Seguimiento Ágil. ¿Es Esto Un Oxímoron?”, 20 May. 2018, agilismoatwork.blogspot.com/2018/05/. Fecha de acceso: 7 Abr 2019.
33. Letelier, Patricio. “Modelos De Proceso Para Desarrollo Ágil.” Agility at Work, 30 Oct. 2014, agilismoatwork.blogspot.com/2014/10/modelos-de-proceso-para-desarrollo-agil.html. Fecha de acceso: 10 Abr 2019.
34. “Waterfall Methodology - Tools and Strategies.” ProjectManager.com, www.projectmanager.com/software/use-cases/waterfall-methodology. Fecha de acceso: 11 Abr 2019.
35. Letelier, Patricio. “Actividad: Tablero Kanban + Concepto De WIP + Diagramas De Flujo Acumulado.” Agility at Work, 1 Dec. 2012, agilismoatwork.blogspot.com/2012/12/actividad-tablero-kanban-concepto-de.html. Fecha de acceso: 22 Abr 2019.
36. Thomas, Steven. “Agile Project Monitoring and Control.” It's a Delivery Thing, 13 July 2015, itsadeliverything.com/agile-project-monitoring-and-control. Fecha de acceso: 24 Abr 2019.
37. “What Makes an Agile Team, Agile; and a Waterfall Team, Not?” Agilist Andre, 20 Dec. 2018, agilistandre.com/agile-team-waterfall-team/.Fecha de acceso: 28 Jun 2019.
38. Vacanti, Daniel. Actionable Agile Metrics For Predictability: An Introduction. 2015