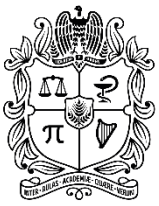


UNIVERSIDAD
NACIONAL
DE COLOMBIA

Factores que afectan la productividad en equipos Scrum analizados con Pensamiento sistémico

Miguel Ángel Maldonado Arango

Universidad Nacional de Colombia
Facultad de Minas, Departamento de Ingeniería de Sistemas
Medellín, Colombia
2017

Factores que afectan la productividad en equipos Scrum analizados con Pensamiento sistémico

Miguel Ángel Maldonado Arango

Trabajo de grado presentado como requisito parcial para optar al título de:

Magister en Ingeniería de Sistemas

Director:

Ph.D. Santiago Arango Aramburo

Universidad Nacional de Colombia

Facultad de Minas, Departamento de Ingeniería de Sistemas

Medellín, Colombia

2017

A mi familia, especialmente a mis padres Irma y Miguel, quienes me dieron todo su apoyo y su esfuerzo para poder continuar. A mi novia que con su amor me llenó de fuerza y compromiso.

Agradecimientos

A mis padres Irma Arango y Miguel Ángel Maldonado que cada día me animan a alcanzar mis metas.

A mis hermanos Erika María Maldonado, Luis Fernando Maldonado y Juan Felipe Maldonado, quienes con su alegría me dan la fuerza para llegar más lejos.

A mi novia Laura Vélez, quien cada día me sorprende más, me arranca sonrisas y me llena de amor para alcanzar lo inalcanzable.

A mi director y amigo PhD. Santiago Arango Aramburo, quien con sus consejos y recomendaciones me enseñó el significado del aprendizaje.

Al grupo Bancolombia que me proporcionó los medios para lograr este aprendizaje y poder compartirlo con todos los interesados.

Resumen

El análisis continuo de la productividad de los equipos de desarrollo ágil de software se hace relevante en esta industria, ya que permite a los directivos y miembros de equipos de desarrollo, evaluar sus resultados y determinar los puntos que requieren atención y mejora. Este trabajo presenta un análisis realizado con Pensamiento sistémico (PS) de los factores de productividad derivados de la aplicación de la cultura, métodos y marcos ágiles de trabajo en equipos de desarrollo de software, campo que ha venido incrementando su uso desde sus inicios. Para esto, se realiza una propuesta de análisis de la productividad basado en el aumento del producido o la disminución de los recursos necesarios para producir. A continuación se explica la cultura de agilidad, que comprende los valores, principios y pilares que adoptan las organizaciones al implementar los métodos ágiles. Atado a la cultura de agilidad se encuentran los marcos de trabajo ágil, de los cuales abordaremos el más usado actualmente: Scrum. Con la aplicación de la cultura y de Scrum como marco de trabajo, se identificaron 25 factores explícitos que afectan la productividad de los equipos de desarrollo ágil de software y a partir de allí se planteó su clasificación en cuatro dimensiones sobre las cuales se realizó el análisis de PS. Como conclusión del análisis se proponen prácticas como: la generación de pequeños incrementos de producto, acelerar los tiempos de realimentación y adecuación del producto, promover la auto-organización y destinar espacios frecuentes de mejora continua para mejorar la productividad, apalancando el sistema donde sea más conveniente. Dicho modelo se validó con encuestas realizadas a cinco equipos de desarrollo ágil de software en la industria financiera colombiana, logrando demostrar que las 17 prácticas propuestas son adecuadas para percibir mejoras en la productividad de los equipos que desarrollan software.

Palabras clave: Productividad, desarrollo ágil de software, marcos de trabajo ágil, cultura de agilidad, pensamiento sistémico.

Abstract

The continuous analysis of the productivity of agile software development teams becomes relevant in this industry, as it allows managers and members of development teams to evaluate their results and determine the points that need attention and improvement. This work presents a systemic thinking (PS) analysis of productivity factors derived from the application of culture, methods and agile frameworks in software development teams, a field that has been increasing its use since its inception. For this, a proposal of analysis of the productivity based on the increase of the produced or the reduction of the necessary resources to produce is realized. The culture of agility is explained below, which includes the values, principles and pillars adopted by organizations when implementing agile methods. Tied to the culture of agility are the agile frameworks, of which we will tackle the most used at the moment: Scrum. With the application of culture and Scrum as a framework, we identified 25 explicit factors that affect the productivity of agile software development teams and from there, it was proposed to classify them into four dimensions on which the analysis of PS. As a conclusion of the analysis, practices such as: the generation of small product increments, accelerate product feedback and adaptation times, promote self-organization and allocate frequent spaces for continuous improvement to improve productivity, leveraging the system wherever it is most convenient. This model was validated by surveys of five agile software development teams in the Colombian financial industry, demonstrating that the 17 practices proposed are adequate to perceive improvements in the productivity of software development teams

Keywords: Productivity, agile software development, agile frameworks, agility culture, systemic thinking.

Contenido

Lista de figuras

Pág.

Introducción: La productividad de los equipos de desarrollo de software

Las personas no necesitan mucho software, las personas necesitan soluciones.

En esta tesis se desarrolla el concepto de productividad, lo cual conviene iniciar explicándolo en términos de producto de software. Esto con el fin de comprender el análisis posterior de los factores identificados y de las políticas propuestas para aumentar la productividad. En este capítulo, se describe la perspectiva desde la cual se abordará la productividad de los equipos de desarrollo ágil de software y cómo esta es determinada por factores emocionales y de entorno. Posteriormente, se plantearán los objetivos que se desarrollarán en este trabajo y que permitirán tener una base de prácticas que favorecen el incremento de la productividad de los equipos de desarrollo en las organizaciones.

La productividad de equipos de desarrollo de software

Según la Real Academia de la Lengua Española, la productividad es: “1. F. Cualidad de productivo. 2. F. Capacidad o grado de producción por unidad de trabajo, superficie de tierra cultivada, equipo industrial, etc. 3. F. Econ. Relación entre lo producido y los medios empleados, tales como mano de obra, materiales, energía, etc. EJ: la productividad de la cadena de montaje es de doce televisores por operario hora” (RAE, 2014). Dada la tercera definición y llevándolo a términos de software, lo adecuado es conocer el número de unidades de software que se producen (tamaño del software), respecto a los medios empleados para obtenerlo, tales como: unidades de tiempo, cantidad de personas o equipos de desarrollo, sin embargo, a pesar de que podemos identificar fácilmente estos medios, no es igual de evidente identificar de forma exacta el tamaño del software producido. A pesar que existen diferentes estándares paramétricos para medir el tamaño

del software como puntos de función (IFPUG, 2000) y COCOMO II, no son exactos y esto se debe a que existe variabilidad en la forma en cómo diferentes equipos de desarrollo resuelven el mismo problema, por lo tanto se tendrían valores de parámetros diferentes para una misma solución. A esto se suman factores emocionales, los cuales también influyen en cuánto producto se desarrolla y con qué calidad, para lo cual estos estándares no tienen una aproximación adecuada (Chulani, Boehm, & Steece, 1999) . Esto hace que medir la productividad no sea algo necesariamente estándar. Como resultado, expresiones como estimación del esfuerzo para desarrollar un producto de software a menudo difieran en su resultado entre diferentes equipos (Chulani et al., 1999).

Otros métodos para medir la cantidad de producto de software producido como líneas de código (Nguyen, Deeds-Rubin, Tan, & Boehm, 2007) o cantidad de funcionalidades, no son lo suficientemente estándares para considerarse en una medición apropiada. Por ejemplo, las líneas de código difieren en los estilos de programación, lenguaje de programación y patrones, además de no tener en cuenta todas las actividades en el ciclo de vida del desarrollo. La medición por funcionalidades tiene una gran deficiencia ligada al mismo criterio de no tener un producto estándar como lo es el software. Estas funcionalidades pueden ser triviales o complejas o que agregan mucho o poco valor para su usuario final.

Adicional, se deben considerar los equipos de desarrollo como un sistema complejo, conformado por personas que trabajan con su conocimiento, experiencia y habilidades grupales e individuales, motivación, entre otros. Esto agrega complejidad y hace cuestionable la idea de estandarizar la productividad, por lo tanto, se debe ampliar la visión a factores más perceptivos para analizar la productividad de los equipos (de O. Melo, S. Cruzes, Kon, & Conradi, 2013).

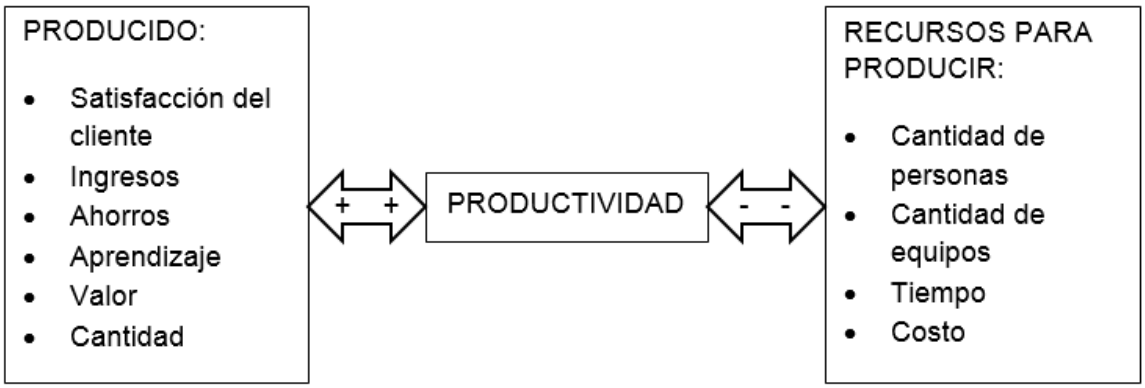
Análisis y medición de la productividad de los equipos de desarrollo de software

Frecuentemente la literatura reporta la introducción de diferentes factores que apoyan la percepción de la productividad de una forma más subjetiva (de O. Melo et al., 2013; Melo, Cruzes, Kon, & Conradi, 2011). Resulta evidente que no es conveniente estandarizar la medición de productividad, es por esto que se propone para este trabajo una mirada diferente. Según la definición 3 de la RAE respecto a productividad, esta es una relación

entre lo producido y los recursos usados para producir, esto significa, que en la ecuación, cuando lo producido aumenta, la productividad aumenta y cuando los recursos usados para producir aumentan, la productividad disminuirá.

A partir de esta definición, se identificaron las variables que espera aumenten en clientes y/o usuarios con la entrega del producto de software y a su vez se identificaron las variables que son consideradas recursos para su producción y se relacionan en la **Figura 1-1**: Variables de crecimiento y decrecimiento de la productividad

Figura 1-1: Variables de crecimiento y decrecimiento de la productividad



Fuente: Elaboración propia

En este trabajo se realiza una interpretación de crecimiento en la productividad, siempre que una de las variables de PRODUCIDO aumente y de decremento de la productividad siempre que una de las variable de RECURSOS PARA PRODUCIR, aumente. Este análisis se hace considerando una visión holística del problema, mostrando como el pensamiento sistémico (Senge, 1990), ayuda a visualizar aspectos que permiten mejorar la productividad, considerando marcos ágiles de desarrollo como se expone a continuación.

La productividad de equipos de desarrollo ágil de software como problema de investigación

El uso de métodos o marcos de trabajo ágil para el desarrollo de software se ha venido incrementado respecto a métodos tradicionales como “cascada” (Dyba & Dingsoyr, 2009; Lo Giudice & Angel, 2013; Maurer & Martel, 2002; Pikkarainen, Haikara, Salo,

Abrahamsson, & Still, 2008; Sverrisdottir, Ingason, & Jonasson, 2014; Vlietland & van Vliet, 2015). Son amplios los marcos y metodologías que se pueden usar bajo la ideología de agilidad, entre los cuales se pueden encontrar algunos altamente usados: Scrum (Deemer, n.d.; Dingsøyr, Nerur, Balijepally, & Moe, 2012; Dyba & Dingsoyr, 2009; Schwaber & Sutherland, 2016), XP (Beck, 1999; Maurer & Martel, 2002), Lean software development (Poppendieck & Llc, 1991) , KANBAN (Kniberg & Skarin, 2010; Lei, Ganjeizadeh, Jayachandran, & Ozcan, 2015; Sugimori, Kusunoki, Cho, & Uchikawa, 1977).

La productividad en el desarrollo de software ha sido ampliamente analizada (Purna Sudhakar, Farooq, & Patnaik, 2011; Rasch & Tosi, 1992; Shull et al., 2010) y se ha mostrado que los métodos ágiles incrementan la productividad respecto a métodos como cascada, para esto se han realizado estudios de variables como: la composición del equipo y asignación, las dependencias externas y la rotación de personal (Melo et al., 2011). Sin embargo, la afectación a la productividad de un equipo de desarrollo podría darse también por otros factores (Balijepally, Mahapatra, Nerur, & Price, 2009; Petersen, 2011; Rasch & Tosi, 1992). Se propone estudiar algunos de estos factores, para lo cual se plantea la siguiente pregunta:

¿Cuáles y cómo influyen algunos factores identificados en las prácticas o métodos ágiles, en la productividad de los equipos de desarrollo de software?

El análisis permite a las organizaciones, directivos y equipos, identificar factores para incrementar la productividad en el desarrollo de software de un equipo ágil por el efecto de incrementar aquellos comportamientos que mejoran la productividad o disminuir aquellos que la afectan negativamente. Este análisis es realizado a través de PS.

Este trabajo se realiza para cubrir la necesidad de las organizaciones de mantener un ritmo constante de aumento de la productividad de equipos de desarrollo, conociendo los factores que influyen en esta e identificando las causas que pueden ayudar a su mejora continua. Conociendo los factores influyentes en la productividad, complementando algunos trabajos realizados en el tema, es importante conocer los ciclos de realimentación y balance que la afectan y concluir con las acciones que pueden mejorarla.

Objetivos

Objetivo General

Analizar con PS los factores que afectan la productividad de los equipos de desarrollo ágil de software y proponer prácticas que mejoren su desempeño.

Objetivos específicos

- Realizar una descripción de la cultura de agilidad y del marco de trabajo ágil Scrum para identificar y seleccionar algunos factores que afectan la productividad en equipos de desarrollo ágil de software.
- Realizar el análisis con PS de cómo los factores identificados afectan la productividad de los equipos de desarrollo ágil de software.
- Plantear y validar algunas prácticas que pueden promover el aumento de productividad en los equipos de desarrollo ágil de software según el análisis realizado

Para desarrollar este trabajo primero se realizó una revisión de la literatura, posteriormente se identificaron los factores relevantes que afectan la productividad de los equipos y se analiza esta afectación con PS. Por último, se realiza una propuesta de prácticas que permiten incrementar la productividad según el modelo realizado y dichas prácticas se validaron con cinco equipos de desarrollo ágil de software que se desempeñan actualmente en la industria financiera Colombiana.

1. La Cultura de agilidad

“Culture eats strategy for breakfast”

Peter Druker.

En este capítulo se explicará el significado de desarrollo ágil de software, más allá de lo que representan los marcos de trabajo ágil como Scrum y XP. Se hablará de la evolución de métodos tradicionales de desarrollo a una cultura de trabajo ágil, explicando el manifiesto desde la perspectiva general encontrada en la literatura y por experiencias propias en equipos de desarrollo. Por último, se explica con un buen nivel de detalle los doce principios atados al manifiesto y cómo estos podrían ser factores de productividad analizados posteriormente.

1.1 De cascada a ágiles

El uso de métodos o marcos de trabajo ágiles para el desarrollo de software está en crecimiento respecto a métodos tradicionales como “cascada” (Dyba & Dingsoyr, 2009; Lo Giudice & Angel, 2013; Maurer & Martel, 2002; Pikkarainen et al., 2008; Sverrisdottir et al., 2014; Vlietland & van Vliet, 2015), esto se debe especialmente a que los métodos ágiles aceptan los cambios como parte importante del proceso y no se basa en planificaciones de largo plazo (Dyba & Dingsoyr, 2009). Esta forma de actuar permite reaccionar rápidamente ante situaciones no planeadas, contrario a los métodos tradicionales, donde se requiere tener baja incertidumbre del resultado del producto final y un alto conocimiento en la tecnología en la cual se desarrolla (Alaimo, 2013).

El modelo secuencial de desarrollo de software o cascada, fue adoptado desde la ingeniería civil en los años 70 (Royce, 1970). Este método está cargado de dificultades para resolver los problemas de los clientes, especialmente por el entorno altamente

cambiante y la falta de adaptación (Alaimo, 2013): Proyectos que no terminan a tiempo, altos costos en la implementación y productos que no resuelven la necesidad como se esperaba son algunos de ellos.

Algunas características de los modelos en cascada son:

1. Se requiere que cada etapa finalice por completo para continuar con la siguiente. (Especificación, análisis y diseño, construcción, integración, pruebas, instalación, mantenimiento)
2. Dado que se requiere finalización de cada etapa para continuar, se convierte en un método muy rígido para aceptar los cambios.
3. Es un modelo basado en la predictibilidad, con estimaciones de largo plazo.
4. Es un modelo donde el control lo tienen los gerentes, además los expertos son quienes dirigen los desarrollos.
   (Dyba & Dingsoyr, 2009)

Dados los problemas identificados en estas metodologías secuenciales, surgen varios movimientos en los años 90 buscando una solución a los mismos. Posteriormente en 2001, se crea el manifiesto ágil, el cual reúne los valores y principios que permitirían a los desarrolladores descubrir nuevas y mejores formas de desarrollar software. En 2001, un grupo de 17 profesionales en desarrollo de software se reunieron en las montañas Wasatch de Utah, Estados Unidos, para dar nacimiento al manifiesto ágil. Este manifiesto marcaría un hito en los principios de los métodos ágiles para la gestión de proyectos de software (Dingsøyr et al., 2012)

1.2 Manifiesto ágil

El manifiesto ágil es el más grande representante de la cultura de agilidad que adoptan las organizaciones al iniciar con desarrollo ágil de software. Este manifiesto representa los principios y valores que las organizaciones incluirán en su cultura organizacional, pues no basta con la adopción de los marcos de trabajo si los hábitos no son los adecuados.

El manifiesto ágil propone cuatro valores y doce principios descritos y explicados a continuación y que se identifican como promotores de la productividad de los equipos, algunos de forma directa según las variables de crecimiento de productividad propuestos

en la **Figura 1-1**: Variables de crecimiento y decrecimiento de la productividad, “Estamos descubriendo mejores formas de desarrollar software haciéndolo y ayudando a que otros lo hagan. A través de este trabajo hemos llegado a valorar:

Individuos e interacciones sobre procesos y herramientas; el software funcionando sobre una amplia documentación; la colaboración con el cliente sobre la negociación de contratos y responder ante el cambio sobre el seguimiento de un plan. Es decir, mientras que hay valor en los elementos de la derecha, se valoran más los elementos de la izquierda” (Beck et al., 2001)

“Desarrollo ágil de software es un término genérico para un conjunto de métodos y prácticas basadas en los valores y principios expresados en el manifiesto ágil” (Beck et al., 2001).

El significado de “Estamos descubriendo…” puede entenderse como que aún no termina el descubrimiento dado que está escrito en presente continuo, es decir, que la firma del manifiesto ágil comprende los valores y principios como mejor solución a los problemas de proyectos tradicionales encontrados hasta el momento.

El primer valor “Individuos e interacciones sobre procesos y herramientas” explica que en la disciplina del desarrollo de software son más importes las personas y sus relaciones, los equipos de trabajo, la colaboración y la excelencia, que tener un proceso predefinido y pedirle a las personas que lo sigan, o tener un conjunto de herramientas y esperar que por sí solas generen valor para clientes y/o usuarios.

El segundo valor “Software funcionando sobre una amplia documentación” explica que la principal medida de progreso de los proyectos de desarrollo de software son el producto en sí, incrementos que de forma incremental satisfagan las necesidades de clientes y/o usuarios, esto es más importante que tener una herramienta de gestión de proyectos que nos da la ilusión de control sobre el proyecto, entregando información del avance del proyecto, que podría no ser real, pues el trabajo en desarrollo de software como se explicó en la introducción, depende de múltiples factores emocionales que no permiten estandarizar de forma precisa las estimaciones del trabajo.

El tercer valor “Colaboración con el cliente sobre negociación de contratos” explica que los cambios son bienvenidos en todo momento, incluso en etapas avanzadas del proyecto, e

incluso son deseables para generar valor a clientes permitiendo una real adaptación a las condiciones cambiantes del entorno. Es por esto, que se prefiere una continua colaboración con el cliente, aceptando sus cambios y recomendaciones a medida que se avanza en el proyecto y se ve producto funcionando. Esto es más importante que un contrato definido y cerrado generalmente desde el principio del proyecto, esperando una visión completa del producto, sin lugar a incertidumbre y con condiciones claramente definidas, incluso, en algunos casos con cláusulas de penalidad, las cuales hacen que lo importante no sea la generación de valor, sino el no incurrir en penalidades.

El cuarto valor “Responder ante el cambio sobre seguir un plan” explica quizá el verdadero significado de la palabra agilidad, permitiendo reaccionar de forma temprana a cambios no estimados, por ejemplo, los regulatorios. Cuando sometemos los proyectos al cumplimiento de un plan, toma más relevancia el cumplir con el plan que generar valor para los clientes y/o usuarios, cerrándonos ante cambios inesperados que podrían encontrarse en el transcurso del proyecto y que podrían generar mucho más valor.

1.2.1 Principios

El manifiesto ágil además de los cuatro valores ya descritos, está acompañado por doce principios (Beck et al., 2001). Estos principios hacen una gran diferencia en una organización, pues representan el cambio de hábitos y de cultura para que lograr los beneficios de usar métodos ágiles.

1. “Nuestra Mayor prioridad es satisfacer al cliente mediante la entrega temprana y continua de software con valor” (Beck et al., 2001): Para lograr satisfacer a clientes y por ende mejorar la productividad, es necesario realizar pequeños incrementos de producto que por sí solos tengan valor y sentido y aporten a la necesidad general. Esto, debido a que los métodos ágiles no son hacer mucho código en poco tiempo, en cambio, es hacer el incremento de producto que mayor valor genera al cliente y así, satisfacer la necesidad de forma incremental.
2. “Aceptamos que los requisitos cambien, incluso en etapas tardías del desarrollo. Los procesos ágiles aprovechan el cambio para proporcionar ventaja competitiva al cliente” (Beck et al., 2001): Este valor guarda estrecha relación con el tercero y cuarto valor.

3. “Entregamos software funcional frecuentemente, entre dos semanas y dos meses, con preferencia al periodo de tiempo más corto posible” (Beck et al., 2001): La entrega del software debe generar algún beneficio al cliente pues podría entregarse segmentos de software que no agregarían valor, por ejemplo, un bloque de backend, o frontend, sin embargo, por sí solos no proporcionan una solución. Para esto existen técnicas que permiten a los involucrados de los proyectos dividir sus incrementos de producto como el user story.
4. “Los responsables del negocio y los desarrolladores trabajamos juntos de forma cotidiana durante todo el proyecto” (Beck et al., 2001): Este principio evidencia la colaboración entre clientes y/o usuarios y los equipos de desarrollo. Esto mejora la comunicación, el entendimiento y la validación.
5. “Los proyectos se desarrollan en torno a individuos motivados. Hay que darles el entorno y el apoyo que necesitan y confiarles la ejecución del trabajo” (Beck et al., 2001): La motivación es otro factor que se identifica y que se incluye en el modelo de productividad. En torno a la motivación se encuentran el propósito, la seguridad psicológica, la autonomía y la aceptación.
6. “El método más eficiente y efectivo de comunicar información al equipo de desarrollo y entre sus miembros es la comunicación cara a cara” (Beck et al., 2001): Cualquiera sea el medio de comunicación, el propósito es comunicarse bien, sin embargo, este principio resta relevancia a métodos como los documentos de requisitos, ampliamente usado en la ingeniería de software en métodos tradicionales como cascada. El problema no es el documento en sí, sino que no existe una explicación cara a cara de este a los involucrados en la solución.
7. “El software funcionando es la medida principal de progreso” (Beck et al., 2001): Este principio está directamente ligado al segundo valor y su explicación está allí dada.
8. “Los procesos ágiles promueven el desarrollo sostenible. Los promotores, desarrolladores y usuarios debemos ser capaces de mantener un ritmo constante de forma indefinida” (Beck et al., 2001): El ritmo sostenible es una condición fundamental para la productividad de los equipos de desarrollo, pues suele ser una constante ver equipo en continuo sobre-esfuerzo dada la premura de los proyectos, sin embargo, esta condición puede aumentar la productividad sólo de forma temporal y no sostenible y a largo plazo, decrecer la productividad a niveles no adecuados para las organizaciones. El equipo debe poder determinar cuánto

trabajo cubrir en cada iteración o periodo de tiempo, pues será el equipo quien lo ejecutará.

9. “La atención continua a la excelencia técnica y al buen diseño mejora la agilidad” (Beck et al., 2001): La excelencia técnica es una condición que los miembros del equipo ya sean de negocio, desarrolladores o testers, deben obtener. Técnica en cuanto a su área de desempeño y de esta forma desenlazar todo el potencial en las soluciones que el equipo genera. En esta práctica se encuentran: El refactoring y la integración continua (Beck, 1999; Dyba & Dingsoyr, 2009; Shull et al., 2010) y demás prácticas que el equipo debe adquirir para mejorar sus desarrollos.
10. “La simplicidad o el arte de maximizar la cantidad de trabajo no realizado, es esencial” (Beck et al., 2001): Este principio hace referencia a que no somos más productivos cuando hacemos mucho código o muchas funcionalidades, lo realmente importante es hacer el software requerido y que genere valor para nuestros clientes y/o usuarios. Es así, que debemos siempre preguntarnos si cada funcionalidad o deseo de cliente es realmente su necesidad y podríamos dejar de hacerlo para enfocar al equipo en lo realmente indispensable.
11. “Las mejores arquitecturas, requisitos y diseños emergen de equipos auto-organizados” (Beck et al., 2001): La auto-organización es un factor que afecta la productividad en el sentido de mejorar la distribución del trabajo, la motivación y la autonomía de los miembros del equipo para proponer las soluciones. Un equipo auto-organizado es un equipo motivado, y a su vez, un equipo que se identifica con el producto que se está construyendo. Esta condición genera un mejor trabajo.
12. “A intervalos regulares el equipo reflexiona sobre cómo ser más efectivos para a continuación ajustar y perfeccionar su comportamiento en consecuencia” (Beck et al., 2001): Este principio está ligado a la productividad a través del mejoramiento continuo. Scrum soporta este principio a través de retrospectivas en cada Sprint.

Se evidencia que el manifiesto no se trata sólo de la implantación de las prácticas ágiles sino de la verdadera adopción de esta (Eloranta, Koskimies, & Mikkonen, 2016). Por el contrario, se plantea desde el aprendizaje organizacional como un cambio cultural profundo organizacional, acompañado de cambios a nivel de estructura, cambios en instalaciones físicas, cambios en el método de desarrollo y en la arquitectura de los sistemas son necesarios para generar diferencia respecto a la productividad en equipos

de desarrollo ágil de software. Estos cambios empiezan a ilustrar como el Pensamiento Sistémico –PS- es una herramienta que permite introducir los conceptos de productividad en la industria del software.

2. Marco de trabajo para el desarrollo ágil de software: SCRUM

Ser ágil es muy diferente de hacer agilidad, el primero requiere cambiar tus hábitos y tu cultura, el segundo una lectura de pocos minutos

En este capítulo se describe que son los marcos de trabajo ágil y además el más usado en el mundo, Scrum. Este marco soporta de manera coherente la cultura de agilidad, proponiendo una serie de artefactos y ceremonias para materializarla. Scrum ha sido ampliamente estudiado, sin embargo, es necesaria su descripción ya que es parte de la base de análisis de este trabajo. En el capítulo 3 se identifican algunos de los factores más relevantes que afectan la productividad de los equipos de desarrollo derivados de este marco de trabajo.

2.1 ¿Qué son los marcos de trabajo ágil?

Los marcos de trabajo para el desarrollo ágil de software son variados. Estos constituyen una serie de prácticas que sus autores han consolidado en una propuesta lista para usarse en la industria, sin embargo, estos mismos marcos son de fácil interpretación pero de una extremada dificultad para aplicarlos de forma adecuada. Esto se debe, a que la aplicación de un marco de trabajo puede hacerse sin llegar a generar ningún valor, siempre que este no se acompañe de un cambio profundo respecto a los hábitos y la cultura organizacional.

Clasificados como marcos de trabajo para el desarrollo ágil de software encontramos dos ampliamente usados, Scrum y XP. El primero propone un conjunto de valores, ceremonias, roles y artefactos que promueven la cultura de agilidad. El segundo propone una serie de principios, prácticas o disciplinas para mejorar el trabajo realizado.

2.2 Exploración del marco de trabajo ágil más usado:

2.2.1 Scrum

El término Scrum fue adoptado por sus creadores por la analogía de (Takeuchi, Hirotaka, 1986) de comparar equipos de alto desempeño con la posición que adoptan los equipos de rugby en melé (scrum en inglés), dado que todos los integrantes del equipo lucharán por alcanzar el mismo objetivo.

Scrum es un marco de trabajo para la gestión de proyectos creado y presentado por Jeff Sutherland y Ken Schwaber en 1995 (Dingsøyr et al., 2012; Schwaber & Sutherland, 2016) cuyo propósito es abordar los proyectos cuya incertidumbre es alta o se mueven en el dominio de lo complejo (Alaimo, 2013; Schwaber & Sutherland, 2016; Snowden & Boone, 2007), sin embargo a menudo se usa para todos los tipos de proyecto. La alta incertidumbre generalmente se presenta cuando los requisitos de la necesidad son desconocidos y emergen o se descubren a medida que se avanza en los desarrollos, lo que hace que los resultados sean cada vez más impredecibles y/o porque la tecnología con la cual se desarrollará también puede ser desconocida.

Scrum se conoce como un marco de trabajo, más que como una metodología, ya que este no es prescriptivo, en su lugar, Scrum presenta una serie de reglas con las cuales los equipos de desarrollo se orientarán para gestionar el proyecto. Esto significa que a pesar de que varios equipos estén cumpliendo con las reglas de Scrum, pueden diferir en la estrategia (Kniberg, Jeff, & Cohn, 2007).

Scrum propone un método de desarrollo de software iterativo e incremental, basado en pilares como la transparencia, la inspección y la adaptación del producto a medida que este avanza en iteraciones (sprints) no mayores a un mes de trabajo (Schwaber & Sutherland, 2016)

La transparencia hace referencia a que cada aspecto del proceso debe ser conocido y estar disponible para todos, tanto integrantes del equipo como interesados del producto, clientes y usuarios. Además se debe usar un lenguaje común y las mismas reglas de juego para facilitar la comunicación y la interacción de todas las personas.

La inspección está dada en varias dimensiones. La primera es la inspección del producto, pues todo el equipo responsable del resultado está continuamente realizando validaciones en periodos cortos de tiempo. La segunda dimensión está dada a nivel de inspección del proceso que se sigue para desarrollar el producto, pues este debe ser mejorado continuamente y para ello se realiza introspección. Se realiza inspección de las personas, del proceso, del equipo, del método, de la técnica, entre otros.

La adaptación es en gran medida el corazón de la agilidad, pues se realiza constantemente mejora de los productos y del proceso, adecuándolos a las necesidades y al entorno del momento. Para ello se realiza experimentos, se permiten y se flexibilizan los cambios repentinos, de los requisitos iniciales, contrario a lo propuesto por métodos en cascada.

Scrum, también provee las herramientas para facilitar la inspección y la adaptación continua a través de reuniones con propósitos bien definidos.

A estos tres pilares se le pueden sumar cualquier número de valores, y a continuación describiremos cuatro de ellos.

- Compromiso: El equipo adquiere el compromiso de búsqueda constante de la excelencia técnica. En cada iteración también se compromete con una cantidad de trabajo constante y que puede entregar en un incremento de producto. Compromiso entre los integrantes del equipo a aportar todo lo posible para conseguir las metas.
- Coraje: Valentía para adquirir compromiso y para no adquirirlo.
- Respeto: Para trabajar en equipo.
- Foco: Acotar el trabajo en pequeñas porciones, sin interrupciones para lograr la terminación de las tareas.

En Scrum, todo el trabajo que se debe realizar en el proyecto y que hace parte del o de los productos hacen parte de la lista de producto*,* en forma de ítems de la lista de producto, a menudo escritos en forma de historias de usuario, estos son priorizados por el *dueño de producto* con el fin de mantener enfocado al *equipo de desarrollo*. En la primera reunión de cada *iteración*, llamada *reunión de planeación*, se realiza una explicación de cada ítem de *la lista de producto* que de acuerdo a la velocidad del *equipo de desarrollo* podrán ser implementados. A continuación, el *equipo de desarrollo* se compromete con los ítems de

la lista de producto que desarrollarán en el *sprint* para entregar un *incremento de producto*. El trabajo entre los integrantes del equipo se sincroniza en la reunión diaria. Posteriormente se realiza una reunión de revisión del producto, con el fin de adaptarse a los cambios que se puedan presentar y de ajustar dicho producto. Por último, se realiza la reunión de retrospectiva, cuyo propósito es que el equipo se revise a si mismo, adaptar el proceso, identificar opciones de mejora en el trabajo realizado por el equipo de desarrollo, identificar acciones acertadas que se deben continuar realizando y definir el plan de acción para la implementación de las posibles mejoras (Schwaber & Sutherland, 2016)

Scrum propone tres roles principales: El dueño de producto (product owner), el Scrum Master y el equipo de desarrollo (development team) (Schwaber & Sutherland, 2016)

Dueño de producto: Es el canal de gestión de la demanda entre el o los clientes y el equipo. Es una persona con conocimiento del negocio, pero también conoce con suficiente nivel de detalle de la tecnología en la cual se implementará la solución. En general se encarga de cuatro cosas:

a. Asegurar que exista entendimiento suficiente por parte del equipo para resolver el problema de los clientes y facilitar la transparencia en cuanto a la visibilidad de la lista de producto.
b. Asegurar que exista validación del producto constantemente con el o los clientes. Esto responde la inspección y la adaptación del producto.
c. Asegurar que exista prioridad en todo el trabajo que se debe realizar. Esto con el fin de que el equipo tenga foco en las características del producto más importantes para los clientes. Generalmente son las que maximizan la rentabilidad de la inversión del cliente las primeras cosas que el equipo desarrollará.
d. Asegurar que exista información al exterior del equipo y para todos los interesados, clientes y sponsors del proyecto. Esto apoya el valor de la transparencia.

Scrum master: Es un líder facilitador o líder servil, quien ayuda al equipo a obtener su mejor rendimiento de largo plazo. El scrum master es quien constantemente cuestiona al equipo en cómo pueden ser mejores, y evalúa el MC en torno al desarrollo personal, desarrollo del equipo, mejoramiento del proceso de desarrollo, adopción de la cultura, conocimiento en agilidad. También es quien asegura que el método scrum se siga como es debido.

Equipos de desarrollo: Es un equipo de profesionales, que auto-gestionan su trabajo y contiene todas las personas con el conocimiento suficiente para desempeñar las tareas y desarrollar el producto, a esta última característica se le conoce como multi-funcionalidad. El tamaño del equipo generalmente está entre 5 y 9 personas, donde la comunicación funciona bien entre sus miembros y tienen el número de personas suficientes para desarrollar una buena cantidad de tareas. (Kniberg et al., 2007; Kniberg & Skarin, 2010; Schwaber & Sutherland, 2016)

Se plantean cuatro ceremonias o reuniones: La reunión de planeación (Sprint planning meeting), la reunión diaria (daily meeting), la reunión de revisión (Spint review meeting) y la reunión de retrospectiva (Sprint retrospective) (Schwaber & Sutherland, 2016)

La reunión de planeación: Esta reunión tiene lugar una vez al principio de cada iteración, su duración es de 2 horas por cada semana que dura la iteración, su objetivo es identificar cuáles ítems de la lista de producto podrán ser desarrollados en la iteración y qué tareas deben hacerse para poder entregar un incremento de producto completamente terminado. Esta reunión tiene dos momentos: en el primero, el dueño de producto explica a todo el equipo cada ítem de la pila (la cual previamente pudo haber sido estimada) durante este proceso, los miembros del equipo realizan todas las preguntas necesarias para obtener la vista única y completa de las necesidades de los clientes. Si no existen dudas y el ítem explicado está completamente claro y se tiene todo lo necesario para convertirlo en parte del producto, se dice que está “lista” para desarrollarse. En caso contrario, se podrá negociar con el dueño del producto la aclaración del ítem o su re-priorización. En el segundo momento de la reunión, el equipo de desarrollo discute qué tareas deben realizar para poder terminar el incremento dar por “Terminado” cada ítem de la lista de producto. Por último el equipo adquiere un compromiso con todo el equipo Scrum (interesados, PO, SM, Equipo de desarrollo) en cuáles ítems podrán terminar en el sprint. El resultado de las tareas que se deben ejecutar y los ítems de la lista de producto comprometidos se le llama Lista de pendientes del sprint.

La reunión diaria: Es una reunión de sincronización entre los miembros del equipo, tiene una duración de 15 minutos. Con el fin de simplificar su duración se hace de pie y se responden tres preguntas: ¿Qué hice ayer? ¿Qué haré hoy? Y ¿Qué impedimentos tengo para hacer mis tareas? Esta reunión cumple con los principios de inspección y adaptación.

La reunión de revisión: Dado que los métodos ágiles son adaptativos y exploratorios en respuesta a la incertidumbre que tienen los productos, se debe realizar de forma continua la validación del producto con los interesados del mismo. Esta reunión tiene una duración de una hora por cada semana de duración de la iteración y su objetivo principal es obtener realimentación de los interesados respecto a que el software entregado está o no resolviendo el problema.

La reunión de retrospectiva: Esta reunión tiene una duración de una hora por cada semana de trabajo de duración del sprint. Su objetivo principal es realizar una introspección acerca de qué se está haciendo bien y se debe seguir haciendo o qué estamos haciendo mal y que no nos puede volver a pasar, además de definir las acciones para resolver y mantener las prácticas malas y buenas respectivamente. En esta reunión se verifica también el proceso que se está siguiendo, las competencias de las personas con el ánimo de mejorar continuamente. (Kniberg et al., 2007; Kniberg & Skarin, 2010; Schwaber & Sutherland, 2016).

Por último, se plantean tres artefactos: Lista de producto (product baklog), Lista de pendientes del sprint (sprint backlog), y el incremento de producto (potentially shippable product increment) (Schwaber & Sutherland, 2016).

Lista de producto: Lista conformada por ítems priorizados por valor de negocio, donde un ítem es una necesidad del cliente. Se mantiene activa durante todo el proyecto, con el fin de permitir a dueño de producto introducir nuevos ítems, modificación o eliminación a los existentes (Permite cambios). Esto significa que Scrum permite realizar cambios constantemente. Esta lista es mantenida por el dueño de producto y se alimenta con las necesidades de los interesados. Generalmente los ítems son escritos en forma de historia de usuario, cuya sintaxis es: Como “rol” – Necesito “funcionalidad” – Para “Valor”, esta estructura permite al equipo entender desde la forma de lenguaje natural del cliente, entender sus necesidades. También puede ser una lista de ítems, donde estos pueden ser entre otros: Requisitos o Casos de uso (Schwaber & Sutherland, 2016).

Lista de pendientes del sprint: Lista de las tareas necesarias para satisfacer o cumplir con los ítems de la lista de producto comprometidos en la iteración. Esta lista se reinicia cada

iteración. El equipo de desarrollo es quién define sus tareas, dado que este es auto-organizado (Schwaber & Sutherland, 2016).

Incremento de producto: Este es un conjunto de funcionalidades desarrolladas y probadas, que son potencialmente entregables o puestas en producción, si así lo decide el dueño de producto. El incremento de producto es la suma de todos los ítems de la lista de producto comprometidos para la iteración (Kniberg et al., 2007; Kniberg & Skarin, 2010; Schwaber & Sutherland, 2016). Este no constituye todas las funcionalidades deseadas del producto, sin embargo, los primeros incrementos serán los más relevantes para satisfacer la necesidad del cliente mediante entregas frecuentes y continuas que generan valor

Son varios los factores que afectan la productividad que pueden identificarse con la correcta aplicación de este marco ágil de trabajo, entre los más importantes está la continua realimentación del producto a través de revisiones en periodos cortos, también la entrega continua, la sincronización constante y la priorización por valor. Estos factores serán estudiados con más detalle en los siguientes capítulos.

3. Factores de productividad en equipos de desarrollo ágil de software

En este capítulo se listan, clasifican y describen los factores de productividad encontrados en la revisión de la literatura que están relacionados a la cultura de agilidad o al marco de trabajo ágil Scrum. Respecto a la clasificación, se detalla en la literatura varias formas usadas para agrupar estos factores, sin embargo, la propuesta realizada de agrupación en este capítulo se realizó respecto a los cuatro pilares descritos como el corazón de la agilidad (Cockburn, 2016): Entrega frecuente de producto con valor, Colaboración, Adaptación y Mejoramiento continuo. Respecto a los factores y su descripción, se encuentran múltiples fuentes que los analizan, en algunos artículos revisados, los factores son identificados por los mismos equipos como relevantes en su productividad (Melo et al., 2011).

3.1 Propuesta de clasificación:

El corazón de la agilidad propuesto por (Cockburn, 2016) identifica cuatro cimientos en los cuales se enmarca la agilidad de las organizaciones. A partir de este concepto, este trabajo realiza una propuesta de análisis de la productividad de los equipos de desarrollo ágil de software. El concepto del corazón de la agilidad servirá para determinar los pilares en los cuales se clasifican los factores encontrados en la revisión de la literatura.

Los cuatro pilares propuestos en el corazón de la agilidad y en los cuales se clasifican los factores hallados son:

1. Entrega frecuente de producto con valor
2. Adaptación
3. Colaboración
4. Mejoramiento continuo (MC).

Para cada factor mapeado en su correspondiente pilar se describirán algunas características como el nombre del factor, su descripción, si el factor fue identificado de un marco de trabajo o de la cultura de agilidad y por qué se considera que afecta la productividad.

3.1.1 Pilar de Entrega frecuente de producto con valor:

En este pilar se clasifican cuatro factores, siendo el pilar además en sí mismo un factor de productividad

La entrega frecuente de producto con valor consiste en la entrega de pequeños incrementos del producto en periodos cortos de tiempo. Los primeros incrementos son los que mayor valor agregan para el cliente (Beck, 1999; Larusdottir, Gulliksen, & Cajander, 2016). Este factor se identifica de los principios del manifiesto ágil y se determina su afectación a la productividad pues al entregar pequeños incrementos de producto se tienen mejoras en la posibilidad de adaptación temprana, en ganancias en oportunidad en el mercado, en reducción de la cantidad de errores, en reducción del dolor y la incertidumbre del éxito de la puesta en producción y por último, de la capacidad de ciclar rápidamente el foco del equipo en otro proyecto o funcionalidad. Todas estas mejoras entregan ganancias en tiempo, generación de valor o reducción de costos. Esto significa que se aumenta el producido o se disminuye los recursos para generarlo como se describe en el capítulo 1.

El foco consiste en que el equipo de desarrollo se preocupe por un pequeño grupo de funcionalidades trabajadas de principio a fin en cada iteración (Dyba & Dingsoyr, 2009; Kniberg & Skarin, 2010), de esta forma, la entrega de valor se favorece, ya que no se trata de avances en documentación o definiciones, sino de entregar funcionalidades listas para usarse. Este factor se identifica con Scrum a través de su planeación del Sprint, y está ligada a la ejecución de un desarrollo iterativo e incremental. La incidencia del foco en la

productividad básicamente responde a que mediante se entregue valor al cliente y/o usuarios continuamente aumentando las utilidades o la reducción de costos de forma temprana.

La priorización por valor, la cual determina que lo más importante para el usuario será lo primero que se desarrollará (Alaimo, 2013; Deemer, n.d.; Kniberg et al., 2007), de esta forma se maximiza al valor entregado a clientes y/o usuarios en etapas tempranas. También facilita el re-enfoque del equipo, pues al terminar las funcionalidades de mayor valor del proyecto 1, podrían enfocarse en las funcionalidades de mayor valor del proyecto 2, siendo estas nuevas funcionalidades, más valiosas que continuar con las restantes del proyecto 1. De esta forma la priorización por valor aumentará la productividad, generando mayores ingresos para la organización. Este factor se identifica también de Scrum, donde se propone una priorización de la Lista de producto por valor y posteriormente desarrollarlo de forma iterativa e incremental.

El timeboxing es una práctica que permite a los equipos dilatar o contraer el trabajo a realizar sólo hasta el máximo permitido por el timeboxig. Esta práctica se identifica de Scrum al determinar los tiempos máximos de las ceremonias a realizar y afecta la productividad al presionar la finalización de las tareas en el tiempo permitido, por ejemplo, en el caso de las reuniones, e incluso, en el tiempo del Sprint para realizar un incremento de producto. Los equipos se adecuan a dichos tiempos y pueden con facilidad determinar cuánto trabajo pueden realizar.

3.1.2 Pilar de Adaptación:

La adaptación como factor de productividad promueve una continua inspección y adaptación del producto (Dyba & Dingsoyr, 2009; Vlietland & van Vliet, 2015). Este factor es lo que realmente se denomina agilidad como la capacidad de adaptarse a entornos cambiantes y con alta incertidumbre. La adaptación afecta directamente la productividad al permitir a los equipos dirigir su esfuerzo a lo que realmente se ha identificado como generador de valor, incluso, en etapas tardías de los proyectos. Se identifica que la cultura de agilidad promueve esta práctica, al igual que el marco de trabajo Scrum, con ceremonias como la Revisión del producto y la Retrospectiva e incluso, en la Reunión diaria, la Planeación del Sprint y el Refinamiento del backlog. Se puede afirmar que todo

el marco de trabajo Scrum está diseñado para promover una continua inspección y adaptación del producto, del proceso, de las personas, de la tecnología entre otros.

El desarrollo iterativo e incremental es una forma de materializar y facilitar la inspección y adaptación (Schwaber & Sutherland, 2016). Se trata de generar incrementos del producto de software que generan valor a los clientes y/o usuarios en periodos acotados de tiempo. Scrum llama a estos periodos Sprints y los establece de una a cuatro semanas. Estos periodos acotados tienen completa relación con el Timeboxing.

3.1.3 Pilar de Colaboración:

La colaboración, un factor de productividad en sí mismo, cuya aplicación se presenta entre clientes y equipo de desarrollo trabajando juntos para lograr el mejor resultado (Heikkilä et al., 2015), entre los integrantes del equipo y entre los diferentes equipos. La colaboración es continuamente mencionada en la cultura de agilidad, específicamente el manifiesto ágil. Una muy buena colaboración implica tener una muy buena comunicación entre los interesados e involucrados en el producto, es por esto que se obtienen diferentes mejoras que afectan de forma positiva la productividad de los equipos, entre las cuales se encuentran: el mejoramiento del entendimiento del problema, además de su validación. En estos dos casos lo que se identifica importante para la cultura de agilidad y los marcos de trabajo ágil es hacer inspección y adaptación de forma continua, sin embargo, más adelante se abordarán estos aspectos. También se identifican mejoras en la priorización, la sincronización, la transferencia de conocimiento y la auto-organización. Todas estas prácticas afectan de forma directa la productividad al verse impactada la generación de valor, la cantidad de software construido, o en otros casos, la reducción de costos u otro recurso para mejorar la producción.

La composición del equipo, es un factor clave en la productividad, ya que determina quiénes serán requeridos para desarrollar el proyecto (Melo et al., 2011), sus capacidades técnicas y de relacionamiento, no sólo de los integrantes que desarrollarán el producto, sino también de los usuarios de negocio que expondrán la situación problemática y validarán el producto. Una mala composición del equipo afectará de forma directa la productividad del equipo, por ejemplo, al no poseer las habilidades de desarrollo requeridas para el proyecto, o no interactuar de forma adecuada con los demás integrantes del equipo

impedirá la colaboración. El no tener la suficiente diversidad en el equipo también evitará que este tenga diferentes perspectivas para resolver los problemas y sólo se enfoquen en las habilidades limitadas que el equipo posee.

La auto organización, como la capacidad de los integrantes del equipo para gestionar su trabajo y la autonomía suficiente para decidir sobre cómo llevar a cabo el desarrollo del proyecto (Bass, 2016; Fontana, Fontana, da Rosa Garbuio, Reinehr, & Malucelli, 2014; Olszewska (née Pląska), Heidenberg, Weijola, Mikkonen, & Porres, 2016; Pikkarainen et al., 2008; von Wangenheim, Savi, & Borgatto, 2013). La auto-organización es una característica que los equipos adquieren con el tiempo y que requiere ser entrenada y mejorada. Esta afecta la productividad porque al mejorarla el equipo puede resolver de forma autónoma sus problemas, lo que genera una solución a la correcta distribución del trabajo entre los integrantes del equipo y mejoras en las decisiones de cómo resolver los proyectos. Esta característica se evidencia en marcos de trabajo como Scrum cuando se hace referencia al Equipo Scrum.

El Tamaño del equipo hace referencia a la cantidad de personas ideal para generar una buena porción de trabajo y no agregar complejidad a la comunicación (Melo et al., 2011; Schwaber & Sutherland, 2016). El tamaño ideal para componer el equipo está entre 5 y 9 personas según Scrum, no significa que equipos más pequeños o más grandes no funcionen bien, sin embargo, puede afectarse la productividad en términos de riesgo y por ende una posibilidad de aumento de costos en equipos más pequeños por los riesgos de enfrentarse a incapacidades médicas de los integrantes u otras ausencias justificadas y los equipos grandes tienen grandes dificultades en la comunicación ya que al aumentar el número de personas aumentará también los tiempos del daily y otras ceremonias Scrum, además de la complejidad en toma de decisiones y lograr acuerdos.

La Cross-funcionalidad del equipo es un factor que afecta la productividad de los equipos de desarrollo ágil de software ya que debe asegurar que entre sus integrantes, se encuentra el conocimiento suficiente y necesario para desarrollar el producto (Schwaber & Sutherland, 2016) y las capacidades individuales se complementan en el equipo para lograr la satisfacción de un objetivo común. De no promoverse la cross-funcionalidad en un equipo, se corre el riesgo de generar dependencias de expertos, o de no facilitar la colaboración en algunas tareas ya que cada miembro se dedica a realizar lo correspondiente a su especialidad y esto generaría un problema de productividad al no

generar valor para usuarios o aumentar costos por la generación de tiempos ociosos. Esta característica se evidencia en el marco de trabajo Scrum al referirse al Equipo Scrum.

La comunicación es una característica fundamental en el proceso de desarrollo y la cultura de agilidad la propone preferiblemente cara a cara, evitando extensos documentos para transferir conocimiento (Beck et al., 2001; Pikkarainen et al., 2008). La comunicación efectiva afecta la productividad al mejorar el entendimiento del problema, la sincronización a todos los niveles y la validación.

La rotación de personal podría generar equipos inmaduros y sin experiencias comunes (Melo et al., 2011). Esto impide que los integrantes del equipo logren acuerdos de trabajo y generalmente inicien en un nivel de formación según (Tuckman, 1965). Esto indudablemente afectará la productividad al no permitir a los equipos maximizar su rendimiento por falta de acuerdos.

Las dependencias externas es un importante factor, ya que equipos que dependen de tareas de otras personas pueden ver afectada su productividad por efectos de la mala sincronización o no tener los insumos para terminar el incremento de producto (Melo et al., 2011).

Por último se encuentra un factor que también se presenta como un valor Scrum y es la transparencia, la cual es fundamental que se presente en los equipos ágiles y esta se manifiesta en la productividad cuando la información está disponible para todos en todo momento acelerando la toma de decisiones (Petersen, 2011; Purna Sudhakar et al., 2011; Schwaber & Sutherland, 2016).

3.1.4 Pilar de Mejoramiento continuo

El mejoramiento continuo MC se describe como la capacidad de los equipos de desarrollo para identificar, en ciclos de realimentación cortos, lo que deben mejorar (Beck, 1999; Fontana, Meyer, Reinehr, & Malucelli, 2015; Pino, Pedreira, García, Luaces, & Piattini, 2010). Este factor se identifica regularmente en la cultura de agilidad, cuando se hace referencia a que en intervalos regulares el equipo se reúne a identificar mejoras en su proceso, su producto, el equipo entre otros. También se identifica que está soportado por el marco de Scrum, al proponerse ceremonias como Retrospectiva y Revisión del producto.

Ambas son puntos de inspección y adaptación, la primera más de la mejora continua de las condiciones del proceso y lo necesario para ser un mejor equipo de desarrollo, la segunda más enfocada al producto que se desarrolla. Se considera que el MC influye de forma positiva en la productividad de los equipos de desarrollo a largo plazo, al permitir identificar posibles mejoras y factores de éxito que el equipo esté realizando bien para continuar con la práctica.

La motivación del equipo (de O. Melo et al., 2013; Melo et al., 2011; Pino et al., 2010), es un factor de productividad que se describe como la felicidad al ejecutar las tareas de desarrollo, la identidad con el equipo y con el producto a desarrollar, la autonomía y la maestría que cada integrante siente que obtiene al trabajar en el producto. Este factor se identifica en los principios de la cultura de agilidad y afecta la productividad de forma directa.

La excelencia técnica es un factor de productividad que se obtiene en equipos maduros a través de la mejora continua, busca siempre lograr la excelencia técnica, la disminución de los errores y la velocidad de entregas (Beck et al., 2001; Campanelli & Parreiras, 2015). Este factor se encuentra en los principios de la cultura de agilidad y se promueve en Scrum en espacios de MC. También es visto como mejora de la productividad a largo plazo.

La introspección es un factor de productividad identificado que se define como la capacidad que el equipo Scrum desarrolla para analizar sus aciertos y sus fallas (Dyba & Dingsoyr, 2009; Holmström, Fitzgerald, Ågerfalk, & Conchúir, 2006). Este concepto mezclado con la premisa de hacerlo en periodos cortos es altamente efectivo para lograr que el equipo se enfoque en lo que debe cambiar para mejorar. Scrum hace referencia a este factor en la ceremonia de Retrospectiva y los principios de la cultura de agilidad también los mencionan.

La experiencia grupal, la cual se fortalece entre los equipos que permanecen a través de los proyectos y se logra experiencia y conexión entre sus integrantes (Dyba & Dingsoyr, 2009). Se complementa con la experiencia individual, la cual se describe como el aprendizaje de cada individuo en la labor desarrollada (Rasch & Tosi, 1992).

Otros factores se identifican como influyentes en la productividad de los equipos, estos factores son externos a los equipos, y describen las condiciones de entorno, ambiente de

trabajo y cultura. Entre estos se identifican: La cultura organizacional, la cual implica una fuerte cultura de colaboración entre los integrantes del equipo y del negocio con las áreas de TI, cultura de MC, cultura de entrega frecuente de software con valor, cultura de constante inspección y adaptación.

Las herramientas, ya que prácticas como la integración continua del software, la liberación continua en ambientes productivos, el control de versiones adecuado y prácticas de desarrollo como TDD no realizadas o realizadas de forma incorrecta podrían no obtener la mejor productividad (Shull et al., 2010)

Se evidencian varios estudios de factores influyentes en la productividad (Purna Sudhakar et al., 2011; Rasch & Tosi, 1992; Shull et al., 2010). También dos estudios anteriores que analizan algunos factores que afectan la productividad en equipos ágiles (de O. Melo et al., 2013; Melo et al., 2011). Sin embargo, no se evidencia que estos artículos aborden todos los posibles factores que pueden afectar la productividad, además de un análisis sistémico, es por esto, que este trabajo puede verse como un complemento usando PS para identificar ciclos de realimentación y balance y poder así proponer prácticas para el mejoramiento de la productividad en equipos de desarrollo ágil de software.

Como conclusión de este capítulo, se evidencia una fuerte adherencia de los factores identificados en su correspondiente clasificación, sin embargo, por lo menos el pilar de MC es un pilar al que casi cualquier factor podría atribuírsele, pues el mejoramiento continuo, aplica para todo lo que hace en la disciplina de ingeniería de software, todo es susceptible de mejora.

4. Propuesta de explicación de la productividad basada en los cuatro pilares de la agilidad

En este capítulo se describe detalladamente el análisis realizado para explicar la afectación a la productividad de los equipos cuando actúan bajo las cuatro dimensiones descritas en el anterior capítulo. Este análisis se realiza usando la herramienta de PS, la cual ha sido ampliamente usada para el análisis de sistemas y problemas en los cuales se presentan diferentes causas que pueden explicar diferente efectos.

Se explica cada pilar de la agilidad con sus respectivos factores y ciclos de realimentación más relevantes- Estos son presentados como prácticas, y se analiza cuál es su afectación sobre la variable de la productividad. Finalizado cada modelo se presentan las principales recomendaciones resultantes de la validación realizada con los “Agile coaches” de los equipos de desarrollo ágil de software

4.1 Enfoque sistémico como herramienta para el análisis de la productividad

Esta tesis usa un enfoque sistémico para el análisis de la productividad de los equipos de desarrollo de software. Describe el comportamiento de los factores identificados en la cultura de agilidad y en el marco de trabajo Scrum y sus relaciones, centrándose en ciclos de realimentación. Estos modelos son representados con Diagramas causales (DC), los cuales nos permiten entender de una forma rápida la hipótesis planteada en cada modelo. Estos diagramas además, son bastante útiles para facilitar la comunicación, lo que permitirá ampliar el entendimiento de un problema.

Acerca de la notación de los DC, estos consisten de variables con nombres claros, conectadas por flechas dirigidas, las cuales denotan la influencia causal. Su propósito

principal es capturar los ciclos de realimentación del problema modelado. Cada flecha tiene un símbolo de incremento o disminución, el cual determina la relación creciente o decreciente entre las variables de causa y efecto. Un vínculo positivo significa que si la causa se incrementa, el efecto aumenta también por encima de lo que podría haber ocurrido en un caso diferente. Un vínculo negativo significa que si la causa se incrementa, el efecto decrece por debajo de lo que podría ocurrir en otro caso.

Los ciclos en los cuales la multiplicación de los símbolos de sus relaciones, positiva o negativa, resulten positivos, se denominan ciclos de *Refuerzo*, y los ciclos que resulten negativos, se denominan ciclos de *Balance.* Un ciclo de refuerzo implica un crecimiento en el comportamiento del sistema. Los ciclos deben tener un nombre que los caracterice, lo que facilitará aún más la comunicación entre los interesados.

Los retardos, cuya notación es un par de líneas perpendiculares a la flecha de relación, explican que ante un incremento en la causa, existe una afectación en otra variable, sin embargo, este efecto no se verá inmediatamente reflejado en el sistema.

Los diagramas causales pueden expresar diferentes comportamientos conocidos como *Arquetipos* (Senge, 1990), estos sin embargo, no se introducen en esta tesis.

4.2 Modelamiento de la productividad

4.2.1 Propósito del modelo

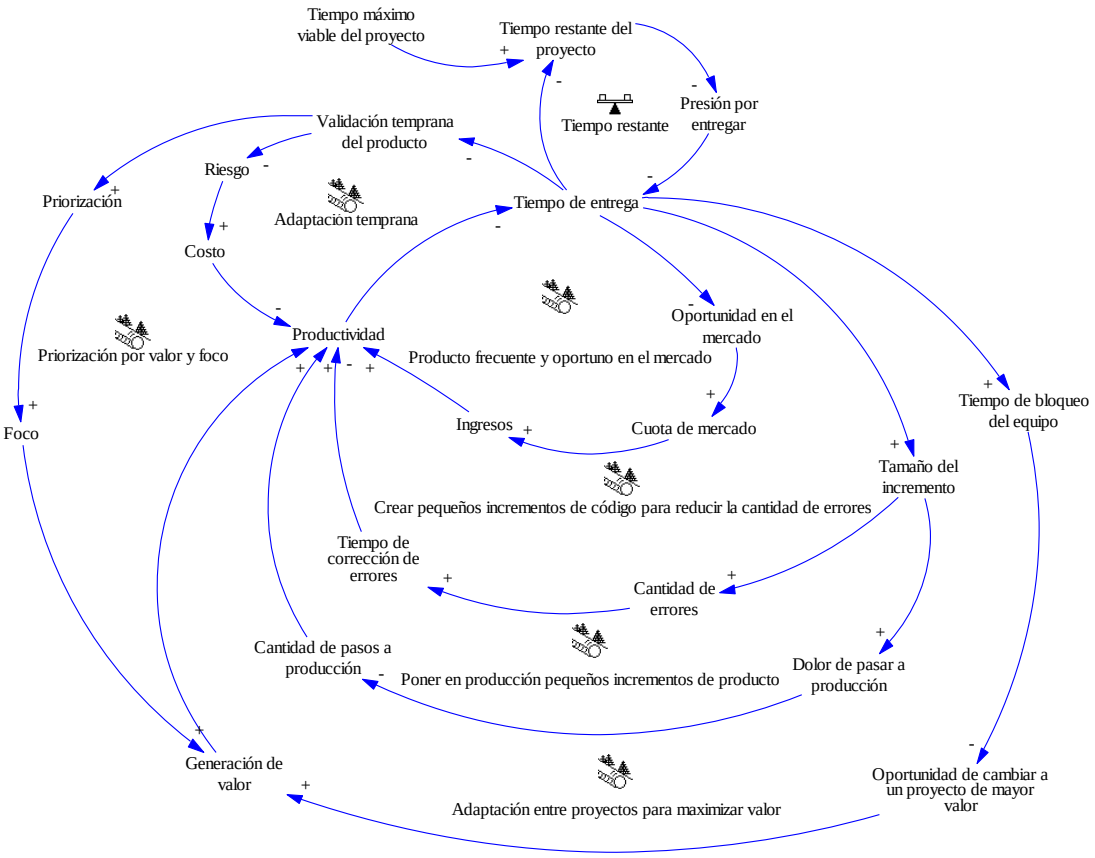
La variable clave a explicar es la *Productividad* como consecuencia de las acciones que se toman al interior de los equipos de desarrollo de software que utilizan las prácticas ágiles. El propósito de estos modelos conceptuales, mapeados con diagramas causales, es explicar cómo los pilares de *Adaptación*, *Mejoramiento continuo* (MC), *Entrega frecuente de producto con valor* y *Colaboración*, generan un conjunto de condiciones que complementadas con políticas y prácticas adecuadas, pueden llevar a un incremento en la productividad de los equipos de desarrollo de software.

4.2.2 Modelo de productividad a partir del pilar de Entrega frecuente de producto con valor:

En este modelo se presentan cinco ciclos determinantes para el análisis de la productividad al momento de desarrollar software. Estos ciclos explican cómo las prácticas ágiles apuntan a la satisfacción del cliente mediante la entrega continua de incrementos de producto que le generen valor.

En la cultura y los marcos de trabajo ágil se identifica que es una constante el desarrollo iterativo e incremental, lo que apalanca la entrega frecuente de producto con valor. Scrum genera periodos de tiempo llamados Sprints (Schwaber & Sutherland, 2016), XP posee una práctica llamada Pequeñas entregas (Beck, 1999) y Kanban es un flujo continuo de entregas (Kniberg & Skarin, 2010; Lei et al., 2015; Sugimori et al., 1977). Estas prácticas nos muestran que este es un pilar fundamental de los métodos ágiles. A continuación se explica por qué el pilar de *Entrega frecuente de producto con valor* es un catalizador de la productividad en los equipos de desarrollo de software.

Figura 4-1: Modelo de productividad a partir del pilar de Entrega frecuente de producto con valor



Fuente: Elaboración propia

El diagrama causal presenta cómo la productividad es afectada en cada una de las seis prácticas:

- La adaptación temprana
- Producto frecuente y oportuno en el mercado
- Crear pequeños incrementos de código para reducir la cantidad de errores
- Poner en producción pequeños incrementos del producto
- Adaptación entre los proyectos para maximizar el valor
- Priorización por valor y foco

Ciclos de refuerzo:

Adaptación temprana: Esta práctica, la cual hace parte tanto de la dimensión de Adaptación como de Entrega frecuente de producto con valor, se materializa cuando los equipos entregan pequeños incrementos de producto dado que la cantidad de código fuente que genera impacto en el cliente es controlable. De esta manera se identifican fácilmente los errores ocasionados al someterlo a pruebas y en su posterior puesta en producción. También hace posible el fallo y el aprendizaje rápido, además de la constante validación del incremento de producto entregado con los clientes y usuarios para adaptarlo en caso de ser necesario mediante una priorización adecuada, tal y como está descrito en el factor de productividad en el capítulo 3. Esta priorización también determina un foco constante para el equipo, lo que mejora la entrega y por ende la productividad. Por otro lado se reduce el riesgo de mercado, al obtener realimentación del mismo, se identifica de manera temprana si el producto será adquirido, si satisface las necesidades de los clientes, si es viable, usable y si les genera valor y tomar un nuevo curso de ser necesario. También se puede probar la hipótesis del caso de negocio, al verse materializado con el uso del incremento del producto por parte de los clientes.

Cuando el *Tiempo de entrega* se incrementa, se pierde la posibilidad de realizar *Validación temprana del producto*, lo que aumenta el *Riesgo*, dado que se podría construir un producto que no satisface la necesidad o no se tienen las recomendaciones de forma oportuna. El aumento del *Riesgo* incrementa el *Costo* del producto, dada la posibilidad de re-procesos innecesarios o cambios en etapas tardías, el incremento en el *Costo* disminuye la *Productividad* y si esta última aumenta, el equipo de desarrollo podrá disminuir los *Tiempos de entrega*.

Producto frecuente y oportuno en el mercado: Esta práctica genera mejoras en la productividad cuando se logra liberar producto rápidamente en el mercado. Esto es posible si el producto entregado cuenta solamente con el principal conjunto de funcionalidades, esto permite al consumidor obtener el producto de forma oportuna y al productor obtener información del mercado como la aceptación del producto, el valor generado al cliente, la promesa de uso, entre otros. Esta dinámica posibilita la generación rápida de ingresos vía el aumento en la cuota de mercado por ser pioneros.

Cuando el *Tiempo de entrega* se incrementa, se pierde la *Oportunidad del producto en el mercado* y posiblemente, la competencia gane la mayor cuota de mercado por ser pioneros. Si la *Oportunidad en el mercado* se aumenta, aumentará la *Cuota de mercado*,

derrotando en primera instancia a los competidores, generando *Ingresos* tempranos. Si los *Ingresos* aumentan, aumentará efectividad del equipo y por ende su *Productividad*. Por último, si la *Productividad* aumenta, los equipos de desarrollo disminuirán sus *Tiempos de entrega*.

Crear pequeños incrementos de código para reducir la cantidad de errores: Esta práctica invita a la generación de pequeños incrementos de producto y tomar realimentación lo más pronto posible de la validez y este. Para ello, el equipo de desarrollo puede apoyarse en herramientas y flujos automáticos como integración continua. Las ventajas ofrecidas por esta práctica son muy valiosas para aportar a la productividad, dado que a menor cantidad de código, menor es la cantidad de errores posiblemente inyectados y por ende toma menos tiempo la identificación de los mismos.

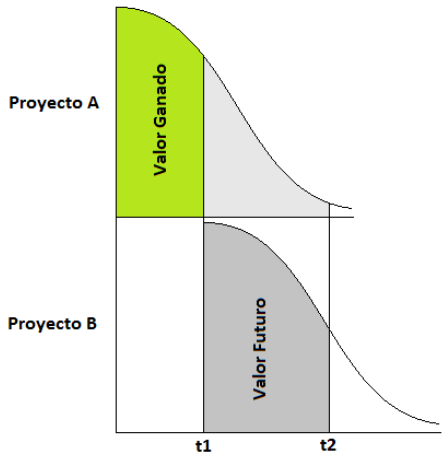
A medida que el *Tiempo de entrega aumenta*, el *Tamaño del incremento* de producto también incrementará, pues el equipo de desarrollo tendrá más tiempo para crear más producto, esto hará que la cantidad de código construido, es decir el *Tamaño del incremento* de producto, sea también muy grande y dada la poca validación temprana que se ha tenido en ese tiempo, la *Cantidad de errores* que se han inyectado será mayor. Este aumento de errores provocará un incremento en el *Tiempo de hallazgo y corrección* de los mismos y todo incremento en el tiempo de desarrollo disminuirá la *Productividad*. Por último, si la *Productividad* aumenta, el *Tiempo de entrega* del producto tenderá a reducirse.

Poner en producción pequeños incrementos de producto: Esta práctica se presenta ante la situación que a menudo viven los equipos de desarrollo al experimentar miedo en el momento de poner el incremento de producto en el ambiente productivo. Este miedo es inducido por la falta de seguridad de los desarrolladores y testers de la cohesión que el incremento de producto tendrá en el paso del ambiente de calidad al ambiente de producción. Un incremento de producto muy grande ocasionará gran dolor al equipo, en términos de dificultad y complejidad, pues tendrán incidentes resultantes del paso, difícilmente detectables y una mayor presión para resolverlos, ya que estarían afectando el servicio al cliente. Este dolor motiva al equipo de desarrollo a evitar los pasos, por ende se presentan pasos cada vez más grandes por la acumulación del código no desplegado que a su vez generarán más dolor. Este es el círculo vicioso que trata de romper la práctica descrita.

Cuando el *Tiempo de entrega* aumenta, el equipo tendrá más tiempo para aumentar el *Tamaño del incremento.* Este aumento hará que la acumulación de código no desplegado sea mayor y por ende se presente mayor *Dolor de pasar a producción* dada la cantidad de errores y parametrizaciones, la presión y la dificultad de encontrar un error en grandes cantidades de código. Este aumento de dolor, ocasiona que el equipo tienda a evitar los pasos a producción y por lo tanto la *Cantidad de pasos a producción* disminuyan. Esta disminución afectará la *Productividad*, pues no se tendrá producto generando valor para los clientes y/o usuarios. El ciclo es cerrado dado que un aumento en la *Productividad*, disminuirá el *Tiempo de entrega* del incremento de producto.

Adaptación entre proyectos para maximizar el valor: Esta práctica genera mucho valor para las organizaciones dada la capacidad de adaptación que estas adquieren, habilitando los fondos para los proyectos que mayor valor les generen. Esta práctica, representada en el ciclo de realimentación, es sinónimo de agilidad, dando la posibilidad a directivos de tener la opción de decidir continuar invirtiendo en un proyecto que ya ha percibido el mayor valor en las entregas realizadas, o poder dirigir los esfuerzos de personas y recursos para embarcarse en una nueva idea de negocio, que en un horizonte de tiempo resulta ser más beneficiosa para la organización que haber decidido terminar todo el conjunto deseado de funcionalidades del proyecto anterior. La **Figura 4-2**: Adaptación entre proyectos para maximizar el valor, ejemplifica una captación de mayor valor al cambiar de foco del proyecto A al proyecto B, entre los periodos de tiempo t1 y t2.

Figura 4-2: Adaptación entre proyectos para maximizar el valor



Fuente: Elaboración propia.

Al aumentar el *Tiempo de entrega* del incremento de producto, el equipo de desarrollo permanecerá todo ese tiempo bloqueado, por lo tanto el *Tiempo de bloqueo del equipo* aumentará. Este incremento del tiempo de bloqueo del equipo disminuirá la *Oportunidad de cambiar a un proyecto de mayor valor*, pues no será posible el cambio de foco y dirigir sus personas y recursos a un nuevo proyecto de mayor valor dado que no se ha terminado un incremento usable del producto, esto retribuye en una menor *Generación de valor* para organización y por ende a una baja en la *Productividad*. Cerrando el ciclo, un aumento en la *Productividad*, disminuirá el *Tiempo de entrega* del incremento de producto.

Priorización por valor y foco: Esta práctica afectará fuertemente a la productividad de los equipos dado que estos se enfocarán en los ítems que mayor beneficio traigan para los clientes y/o usuarios. Dar la posibilidad al negocio que determine la prioridad de las funcionalidades que el equipo trabajará según su prioridad aportará al valor que al final se entregue. Algunos métodos o marcos ágiles de trabajo proponen realizar cambios a las tareas de los equipos sólo en periodos de tiempo adecuados para favorecer el foco de estos en un conjunto acotado de tareas, sin embargo, otros marcos de trabajo más continuo como Kanban, favorecen estos cambios si su prioridad es definitivamente mayor a la que poseen las tareas que actualmente ejecute el equipo.

Cuando el *Tiempo de entrega* aumenta, disminuye la oportunidad de hacer *Validación temprana del producto,* y un aumento en esta, mejora la *Priorización*, ya que se realizaría con mayor frecuencia en la duración completa del proyecto. Al entregar un incremento de producto, el cliente podrá verlo en funcionamiento y hacer retroalimentación al equipo, además de actualizar la Lista de producto, incluso ingresando nuevos ítems, o retirando existentes. Esto permite aceptar los cambios como el Manifiesto ágil lo propone, además de ofrecer ventaja competitiva al cliente por la posibilidad de realizar cambios y adecuar sus prioridades, lo que aumentaría el valor. Al aumentar la *Priorización* también aumentará el *Foco*, pues el equipo conoce que funcionalidades o la parte del proceso más importante para el negocio y así enfocarse en el desarrollo de esta. El aumento del *Foco* aumentará la *Generación de valor* para los clientes, al recibir el conjunto de funcionalidades más importantes del proyecto. Un aumento en la *Generación de valor* aumentará la *Productividad* y aun aumento de esta disminuirá el *Tiempo de entrega*.

Ciclos de balance:

Tiempo restante: Este ciclo en conjunto con cada ciclo de refuerzo del modelo, representa una adaptación del arquetipo de *Límites al crecimiento*, el cual explica que una entrega podría ser tan grande sólo hasta donde las restricciones del proyecto lo permitan, es decir, el tiempo de entrega podrá irse incrementando, sólo hasta que el dinero o lo que es lo mismo, el tiempo máximo del proyecto, lo permita o pasará a ser un proyecto inviable. Esto son los factores que limitan el crecimiento en las entregas.

Este arquetipo de límites al crecimiento también nos muestra que a medida que la fecha máxima permitida, antes de pasar a ser un proyecto inviable se acerca, aumentará la presión por tener producto que supla la necesidad y esto hará que las entregas se aceleren.

La variable limitante del sistema es el *Tiempo máximo viable del proyecto*, cuando esta aumenta también aumentará el *Tiempo restante del proyecto*, el cual es también el tiempo disponible para entregar un producto que satisfaga la necesidad. Cuando el *Tiempo restante del proyecto* aumenta, disminuirá la *Presión por entregar el producto*, pues estamos confiados en que el tiempo restante es suficiente para resolver la situación problemática. Cuando la *Presión por entregar el producto* aumenta, disminuirá el *Tiempo de entrega*, posiblemente a un ritmo no sostenible del cual se hablará más adelante. Por último, cuando el *Tiempo de entrega aumenta*, el *Tiempo restante del proyecto disminuirá*.

Validación del modelo:

Según el análisis realizado por cada “Agile coach”, a quienes se les presentó el modelo, demuestra que el modelo captura los principales componentes que afectan la productividad, siendo un modelo adecuado respecto a los ciclos, variables y relaciones. También arrojó un grupo de sugerencia presentadas por completo en el Anexo: **Encuesta 5-1**: Encuesta de validación de los modelos de productividad. Estas sugerencias fueron incorporadas en la versión final aquí presentada y se resumen como más relevantes las siguientes:

- Se sugiere hacer explícita la variable *Calidad* como una causante adicional de la afectación a la *productividad*, sin embargo, este modelo generaliza que la entrega es completa e integral. La variable *Calidad* se menciona a lo largo de este trabajo, teniendo en cuenta su relevancia y se incluye de forma explícita en el modelo de

mejoramiento continuo como un efecto de la causante *Ritmo de trabajo* acelerado o presionado.

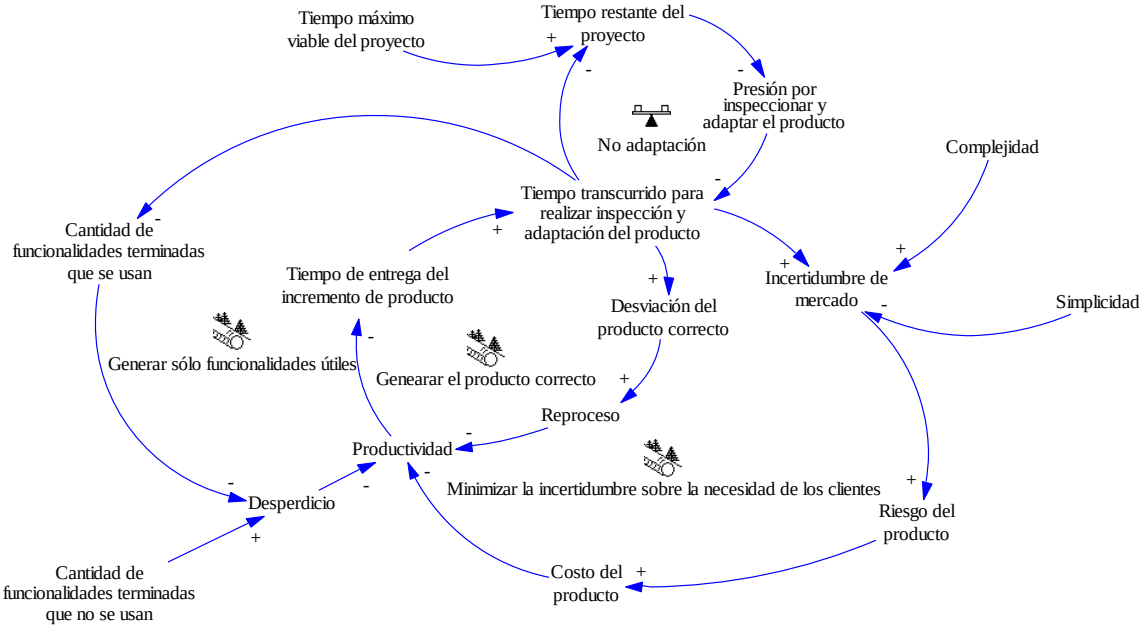
- Se sugiere un conjunto de relaciones como efecto ante la variable *Presión por entregar*, ya que esta presión inyecta una cantidad de errores mayor a la tasa normal del equipo. Este comportamiento se expresa, tal y como se sugiere, en el modelo de mejoramiento continuo.

4.2.3 Modelo de productividad a partir del pilar de Adaptación:

En este modelo se presentan tres ciclos de realimentación que muestran la afectación a la productividad desde la perspectiva de adaptarnos a los cambios de entorno, de regulación, de estrategia, entre otros. Las organizaciones que aprenden a adaptarse rápidamente tienen a ser más flexibles y evitan el riesgo mucho más rápido. La cultura de agilidad y los marcos de trabajo ágil promueven una constante adaptación. En Scrum se identifican las reuniones de Planeación, Refinamiento, Revisión y Retrospectiva (Bash, 2015; Moe, Dingsøyr, & Dybå, 2010; Schwaber & Sutherland, 2016) y XP promueve una práctica llamada On-site Customer (Beck, 1999), que promueve que el cliente se siente muy cerca del equipo de desarrollo para facilitar la comunicación, colaboración y adaptación continua. La adaptación se da en diferentes niveles, tales como la adaptación del producto, del proceso, de la cultura, de las personas, de los equipos, entre otros. Este modelo describe la adaptación en términos del producto o incremento de producto que se está construyendo para satisfacer las necesidades de los clientes y/o usuarios. Los demás niveles de adaptación se abordarán en el modelo de productividad a partir del pilar de *Mejora continua*.

Adaptarse es en sí el significado de la agilidad, pues es la mejor manera de abordar la incertidumbre y la complejidad consecuente a los proyectos de software evitando desbordar en proyectos más costosos de lo esperado o con una duración que no se adapta a las necesidades del mercado y productos o funcionalidades que no se usan (Alaimo, 2013).

Figura 4-3: Modelo de productividad a partir del pilar de Adaptación



Fuente: Elaboración propia

Este modelo nos presenta tres ciclos principales sobre la productividad de los equipos de desarrollo de software promovidos por la dimensión de adaptación:

- Generar sólo funcionalidades útiles
- Generar el producto correcto
- Minimizar la incertidumbre sobre la necesidad de los clientes

Ciclos de refuerzo:

Generar sólo funcionalidades útiles: Esta práctica se logra a través de una inspección y adaptación continua del producto que se está generando, combinado con el anterior pilar de entrega frecuente de producto con valor. De hecho, una gran cantidad de producto de software que se genera en el mundo no es usado, debido a una baja capacidad de adaptación de las organizaciones para hacer sólo funcionalidades que generan valor a los clientes y usuarios. Existe la tendencia de generar en cada proyecto un conjunto de funcionalidades que desde la perspectiva del usuario serían muy buenas, ya sea porque a él le parece que agregan valor sin hacer validaciones con los clientes, o porque

aprovechando que el proyecto fue aprobado por la organización es la oportunidad de agregarle todo lo que sea posible, desembocando sin duda en un mal aprovechamiento de los recursos y de las personas.

A medida que el *Tiempo transcurrido para realizar inspección y adaptación* aumenta, la *Cantidad de funcionalidades terminadas que se usan* disminuirá, debido a la baja adaptación a las necesidades del cliente en el transcurso de tiempo. Muchas de estas funcionalidades aportaban valor al principio del periodo de desarrollo; pero estas pueden cambiar debido a variaciones en el entorno y dejar de generar valor para el final del periodo. Por este motivo resulta conveniente realizar inspección y adaptación temprana del producto e identificar con la oportunidad suficiente estas desviaciones y poder re-priorizar la lista de producto para enfocar el equipo siempre en la generación de valor de nuevas funcionalidades. Un aumento en la *Cantidad de funcionalidades que se usan* generará una disminución en el *Desperdicio* ya se evitan los re-procesos y los sobre costos, y un aumento en el *Desperdicio*, causado por un aumento en la *Cantidad de funcionalidades terminadas que no se usan*, disminuirá la Productividad. Cuando la *Productividad* aumenta se retribuye en una evidente disminución del *Tiempo de entrega del incremento del producto* y finalmente el ciclo es cerrado cuando un aumento en el *Tiempo de entrega del incremento de producto* tiene como efecto el aumento en el Tiempo transcurrido para realizar inspección y adaptación.

Generar el producto correcto: Esta práctica también se centra en la continua inspección y adaptación del producto y en la generación de pequeños incrementos del mismo. Esta práctica está relacionada con la adaptación del producto de acuerdo al cumplimiento de su propósito respecto a la satisfacción de las necesidades de los clientes y/o usuarios. No es la generación de funcionalidades que se usen, como en la anterior práctica, es que las que se usan sean las más adecuadas para los clientes y/o usuarios.

Cuando el *Tiempo transcurrido para realizar inspección y adaptación* aumenta, es más probable que el equipo de desarrollo esté teniendo *Desviaciones del producto correcto* respecto a la necesidad del cliente al transcurrir determinado tiempo. En otras palabras, una necesidad que nació en un periodo de tiempo, no necesariamente es la misma al final de dicho periodo, debido a cambios en el entorno, en el caso de negocio, en los clientes entre otros. El equipo de desarrollo deberá realizar validación frecuentemente para saber

si es requerido un cambio en las prioridades que se están trabajando o si se requiere modificar alguna definición, diseño o desarrollo. El no realizar con la debida oportunidad esta validación podrá resultar en un *Reproceso* para adaptar el producto en etapas tardías y este *Reproceso* se traduce en un costo adicional de tiempo y dinero, lo que disminuirá la *Productividad*. Un aumento en la *Productividad* del equipo de desarrollo disminuirá el *Tiempo de entrega del incremento de producto*.

Minimizar la incertidumbre sobre la necesidad de los clientes: Esta práctica introduce elementos de incertidumbre de los productos, que puede ocasionar grandes riesgos y posteriormente pérdidas. Los métodos y marcos ágiles de trabajo proponen cubrirla mediante una inspección y adaptación continua. Si no existiera incertidumbre, el equipo de desarrollo podría realizar grandes etapas de desarrollo y pruebas y liberar el producto mucho tiempo después, sin embargo, en un entorno cambiante y con alta complejidad, la incertidumbre crecerá. Algunos escenarios encontrados se refieren a que los usuarios ya podrían no necesitar lo que se está construyendo, el nacimiento de una nueva tecnología descontinua nuestro producto en construcción sin haberse liberado en el mercado, la regulación cambia, las empresas encentran nuevos negocios, etc. Esta incertidumbre, de no ser controlada, se traduce en riesgos y altos costos en caso de que alguno de los anteriores escenarios se cumpliera.

Cuando el *Tiempo de entrega del incremento de producto aumenta*, aumenta también el *Tiempo transcurrido para realizar inspección y adaptación*, lo que hace que no se esté validando constantemente y por ende que esté aumentando la *Incertidumbre en el mercado* y alguno de los escenarios descritos podría estarse materializando. Este aumento de *Incertidumbre del mercado* hace que los *Riesgos del producto* aumenten y por ende un incremento los *Costos del producto* que se traduce finalmente en una disminución de la *Productividad*. El aumento en la *Productividad* del equipo de desarrollo hace que baje el *Tiempo de entrega del incremento de producto*.

Ciclo de balance:

Al igual que en el modelo de productividad a partir del pilar de entrega continua de producto con valor, el modelo de adaptación también tiene la condición de cumplir con el arquetipo de límites al crecimiento que se conforma entre cada ciclo de refuerzo y el ciclo de balance: *No adaptación*, sumado a la condición limitante. El tiempo transcurrido para realizar

inspección y adaptación del producto puede ser tan grande sólo hasta que el tiempo máximo viable del proyecto lo permita, en cuyo caso sería similar a realizar un proyecto usando un método en cascada, pues sólo tendríamos una única inspección del producto sin lugar a adaptación y este sería al final proyecto.

Validación del modelo:

La validación general del modelo demuestra que las prácticas presentadas son adecuadas y afectan de forma positiva la productividad de los equipos de desarrollo, las variables del modelo también son las adecuadas, igual que sus relaciones. La recomendación principal realizada por los Agile coach de los equipos con los cuales se realizó la validación de este modelo, es que se sugiere la inclusión de la variable *Actualización del plan de entregas* para mejorar la adaptación. Esta variable no se considera en este modelo pues la actualización de este artefacto es un medio para conseguir adaptación no una práctica obtenida como resultado de realizar adaptación continua.

4.2.4 Modelo de productividad a partir del pilar de Colaboración:

En este modelo se presentan cinco ciclos explicados desde la perspectiva del pilar de Colaboración y que afectan la productividad de forma directa. La colaboración aquí explicada se da desde diferentes niveles:

La colaboración entre el equipo de desarrollo y los clientes y/o usuarios: En este nivel se requiere una constante comunicación y colaboración para desarrollar un producto ajustado a las necesidades de los clientes y además que sea desarrollado de forma eficiente. El equipo de desarrollo en constante comunicación con el cliente indaga y valida continuamente el desarrollo para tener éxito, mientas los clientes y usuarios comprenden la complejidad del sistema a desarrollar y ayuda al equipo manteniendo claras las prioridades, el entendimiento de la situación problemática y a realizar constante inspección y adaptación de la solución.

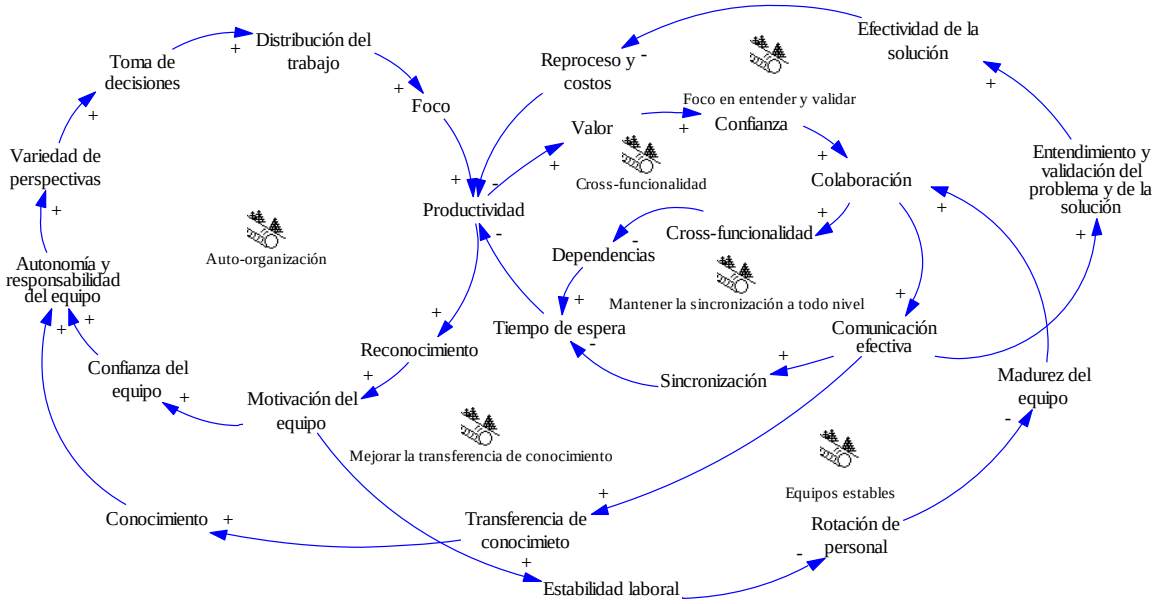
La colaboración entre los integrantes del equipo: En este nivel los integrantes del equipo deben interactuar y aportar desde sus habilidades para entregar las mejores soluciones a clientes y usuarios. Esta característica es llamada *Cross-funcionalidad* (Eloranta et al., 2016; Heikkilä et al., 2015; Larusdottir et al., 2016; Schwaber & Sutherland, 2016).

Desarrolladores, testers, diseñadores y demás actores involucrados en el desarrollo del sistema requieren complementar sus habilidades y experiencias para ponerlas a disposición de la solución y agregar valor a los interesados. Esto hace importante que la composición del equipo de desarrollo sea efectiva (de O. Melo et al., 2013). También es necesario que el equipo de desarrollo comprenda sus funciones y sus objetivos y en pro de conseguir dichos objetivos en un periodo de tiempo acotado (timeboxing), el equipo solucionador decida la mejor forma de ejecutar su trabajo. A esta característica se le llama Auto-organización (Bass, 2016; Fontana et al., 2014; Olszewska (née Pląska) et al., 2016; Pikkarainen et al., 2008; von Wangenheim et al., 2013).

Se identifica también un nivel de colaboración entre diferentes equipos, donde diferentes equipos de desarrollo interactúan para resolver una única situación problemática o satisfacer una necesidad. En este nivel se encuentran diferentes situaciones que agregan complejidad al modelo de trabajo, es por esto que las dependencias son uno de los factores que afectan radicalmente la productividad en el desarrollo de software (de O. Melo et al., 2013). Algunos marcos de escalamiento de desarrollo ágil como SAFE (“Shared Services – Scaled Agile Framework,” 2016), y NEXUS (Escalable, 2015; “Scrum.org,” 2016), tratan de abordar esta complejidad donde un único producto es desarrollado por múltiples equipos ya que se requieren diferentes niveles de sincronización.

Una característica identificada como factor de productividad es la rotación de personal (de O. Melo et al., 2013) la cual afectará todo el ecosistema de colaboración, sea cliente y/o usuario o miembro del equipo de desarrollo. Una alta rotación impedirá que los equipos logren un adecuado nivel de madurez, logren acuerdos de trabajo y lleguen a convertirse en un equipo de alto desempeño. Según (Tuckman, 1965) es necesario un equipo de trabajo pasará por diferentes etapas de madurez antes de convertirse en un equipo de alto desempeño y claramente la rotación de personal sería un impedimento para lograrlo.

Figura 4-4: Modelo de productividad a partir del pilar de Colaboración



Fuente: Elaboración propia

Los seis ciclos presentados desde la dimensión de Colaboración tienen una gran influencia en la productividad de los equipos de desarrollo de software.

- Auto-organización
- Mejorar la transferencia de conocimiento
- Mantener la sincronización a todo nivel
- Foco en entender y validar
- Equipos estables
- Cross-funcionalidad

Ciclos de refuerzo:

Auto-organización: Esta práctica o característica de los equipos de trabajo promueve que sea el equipo quien toma las decisiones respecto a cómo ejecutar el trabajo o lograr el objetivo propuesto. Esto desincentiva la figura de experto o el tipo de liderazgo en el que el líder le dice a cada persona qué realizar y cómo. El equipo es en sí mismo un sistema, tiene tanta complejidad como integrantes e interdependencias entre ellos se tengan. Para controlar un equipo con dicha complejidad se requiere un sistema de control igual de

complejo (Ashby, 1956). Es por esto, que una única persona, generalmente un jefe, no absorbe la complejidad del sistema y por ende, no podría controlarlo. La mejor solución a este problema, es considerar que el equipo mismo sea quien se controle. A partir de esta idea surge la *Auto-organización*, como la medida adecuada para optimizar la distribución del trabajo, la asignación de las tareas, mejorar la toma de decisiones, entre otras ventajas.

La auto-organización es la base para mejorar la toma de decisiones al permitir analizar el problema con una amplia variedad de perspectivas y lograr acuerdos que beneficien más el producto a desarrollar y a las personas que lo desarrollan. También permite mejorar la distribución del trabajo porque cada integrante del equipo, en pleno conocimiento de sus facultades y limitaciones, podrá proponer qué hacer y con quién hacerlo, de esta forma, todos logran un mejor acuerdo.

Cuando la variable *Autonomía y responsabilidad del equipo* aumenta, ya sea porque se crea el ambiente para que esto suceda y se da al equipo mayor confianza para la toma de decisiones o porque el equipo se empodera de la responsabilidad ante la consecución del objetivo común, la *Variedad de perspectivas* también aumenta, al no ser una persona, generalmente un líder, sino que son todos los integrantes del equipo aportando a la solución. Un incremento en la *Variedad de perspectivas* mejora la *Toma de decisiones*, pues se evitan sesgos o experiencias de una única fuente y por la colaboración de todos los integrantes se llegará a una mejor decisión. Cuando la *Toma de decisiones aumenta*, también lo hará la *Distribución del trabajo*, pues el equipo que participa en las decisiones tomadas, es quien determine quién realiza cada tarea para lograr el objetivo. Al mejorar la *Distribución del trabajo*, también mejora el *Foco*, pues el equipo tiene un objetivo que alcanzar y se centra en las decisiones más importantes. Un aumento en el *Foco* de los equipos aumenta la *Productividad*, pues un conjunto acotado de trabajo da como resultado más trabajo terminado y no solamente mucho trabajo empezado. Este trabajo terminado aumenta el valor para los clientes. Cuando la *Productividad* aumenta, se incrementa el *Reconocimiento* para el equipo y un aumento en el *Reconocimiento*, genera incremento en la *Motivación* de todos sus integrantes, ya que el reconocimiento es una condición que los integrantes de un equipo valoran. Un aumento en la *Motivación* del equipo genera un aumento en la *Confianza* del equipo para tomar decisiones, distribuir su trabajo y enfocarse y esto además, mejora la *Autonomía y responsabilidad del equipo* pues se forman lazos en un equipo con confianza y mejores relaciones que aumenta la seguridad para opinar y aportar al equipo.

Mejorar la transferencia de conocimiento: Esta práctica es fundamental para aumentar la productividad de los equipos. Generalmente los equipos inician en un estado de poca madurez y muchas personas aún no se adaptan a trabajar con otros para alcanzar un objetivo común. El conocimiento puede estar centralizado en pocas personas del equipo y frecuentemente resaltan expertos. Esta práctica busca crear un equipo en el que a través de la colaboración se generen los espacios de confianza, empoderamiento y difusión del conocimiento, en los cuales los integrantes del equipo no sean expertos sólo de su campo de acción, sino que comprenda con un nivel de detalle apropiado las funciones y especialidad de sus demás compañeros. Esto genera independencia de los expertos y del conocimiento centralizado y mitiga los riesgos por ausencia de los que saben, también evita el desplazamiento de los integrantes de un equipo a otro porque se requiere apoyo pues sólo esa persona es quien conoce. Adicional, se resalta especialmente una práctica de XP llamada Desarrollo en pares, la cual incentiva a la transferencia de conocimiento entre integrantes del equipo.

Cuando aumenta la *Transferencia de conocimiento* entre los integrantes del equipo, aumenta también la base de *Conocimiento* con lo cual el equipo podría afrontar una necesidad, tal y como lo es la sintaxis de los lenguajes de programación, los métodos de desarrollo, patrones, etc. El crecimiento del *Conocimiento* del equipo incrementa la *Autonomía y responsabilidad del equipo*, ya que este tiene mejores bases para tener independencia de expertos y líderes. Cuando la *Autonomía y responsabilidad del equipo* aumenta, lo hará también la *Variedad de perspectivas* y esto conlleva a un aumento en la *Toma de decisiones* que a su vez mejorará la *Distribución del trabajo* y el *Foco* del equipo. En este nivel se mejora la *Productividad*, lo que generará un mayor *Valor* para los usuarios y/o clientes. Cuando el *Valor* aumenta, los clientes y/o usuarios aumentan su *Confianza* hacia el equipo y así estarán dispuestos a aumentar su *Colaboración* con el mismo. Un aumento en la *Colaboración* se traduce en un aumento de la *Comunicación efectiva* lo que desemboca en un aumento de *Transferencia de conocimiento*.

Mantener la sincronización a todo nivel: Esta práctica explica por qué cuando clientes y/o usuarios trabajan junto a uno o varios equipos de desarrollo, debe existir sincronización entre todas las partes, en sí, comportarse con un único gran equipo. Sin embargo, las dependencias siguen siendo una de las causas más frecuentes que afectan de forma negativa la productividad de los equipos de desarrollo (de O. Melo et al., 2013).

Cuando existen dependencias entre actores es más probable que se incrementen los tiempos de espera. Sin embargo, el problema no es que existan dependencias, sino que estas no funcionen adecuadamente. Si un equipo depende de un insumo de otro, el problema se presenta cuando no está listo dicho insumo y el equipo con la dependencia debe esperarlo, pues esto aumentará los tiempos de entrega y por ende una reducción a la productividad.

Otra dependencia común es que el o los equipos dependan de validaciones de usuarios y/o clientes, o de definiciones funcionales o reglas de negocio que el equipo puede no dominar en ese momento y depende de un actor externo. También se presentan dependencias a un nivel mucho más interno del equipo y es la que posiblemente se presente entre desarrolladores y testers respecto a un artefacto que se requiera para que uno de los dos continúe su trabajo. Todas estas dependencias requieren ser gestionadas y optimizadas de tal forma que la productividad no se vea afectada por tiempos de espera, retrasos o fallas en calidad del producto por asumir definiciones no validadas.

Cuando la *Sincronización a todo nivel* aumenta, el *Tiempo de espera* disminuye pues la información fluye a través de todos los involucrados e interesados facilitando la toma de decisiones. Cuando el *Tiempo de espera* aumenta por falta de sincronización entre los actores del sistema, existe una afectación a la *Productividad*. Si la *Productividad* aumenta, también lo hará el *Valor* para nuestros clientes, al recibir productos que les generan impacto positivo en lo producido o disminución en los recursos para producir más frecuentemente. Si el *Valor* aumenta, mejora la *Confianza* de clientes y/o usuarios, lo que desemboca en una mejor *Colaboración* con el equipo de trabajo. Este aumento en la *Colaboración* incrementa la *Comunicación efectiva* y facilita el incremento de la *Sincronización*.

Foco en entender y validar: Esta práctica consiste en incrementar la colaboración entre los actores del equipo, tanto clientes y/o usuarios como desarrolladores, testers, entre otros. Esto facilita la comunicación y por ende, el buen entendimiento de la situación problemática a resolver, que acompañado de una continua validación de la solución, contribuyen de manera inminente a un producto mucho más efectivo, a la medida, que realmente satisfaga las necesidades de los interesados. El foco en entender el problema y una continua validación del producto, evitará en gran medida re-procesos por la reconstrucción de productos no validados a tiempo o por haber resulto muy bien el problema incorrecto,

identificando un mal entendimiento de la necesidad. La productividad de un equipo que entrega un producto, cuyo valor es entendido y validado de manera temprana aumentará en cuestión de evitar los re-procesos.

Cuando el *Entendimiento y validación del problema y de la solución* aumentan, la *Efectividad de las soluciones* también lo hace, debido a la disminución de desperdicios, re-procesos, enfocarse en un problema que no es, o en una forma de resolverlo que no es la adecuada. El aumento en la *Efectividad de la solución* disminuirá los *Re-procesos y costos* y esto a su vez reduce la *Productividad*. Un aumento en la *Productividad*, incrementa el *Valor* para clientes y usuarios, su *Confianza* y su *Colaboración* con el equipo, lo que traerá mejoras en la *Comunicación* y por ende mejora el *Entendimiento y validación del problema y su solución*.

Equipos estables: Esta práctica asegura una colaboración constante en los equipos. El factor clave es la madurez de los mismos, cuyo incremento mejora los acuerdos entre sus integrantes, encontrando mejores formas de trabajar y de colaborar. La rotación de las personas de los equipos son uno de los factores de productividad identificados por (Melo et al., 2011) que afectan la productividad. Adicional, (Tuckman, 1965) propone cuatro fases o estados por los cuales pasan los integrantes de un equipo antes de llegar al estado de alto desempeño. Una continua rotación del equipo no permitirá que este estado se desarrolle.

Cuando la *Rotación de personal* aumenta, la *Madurez del equipo* disminuirá, dado que el reemplazo de un integrante del equipo hará que los acuerdos logrados hasta ese momento deban ser puestos nuevamente en discusión y práctica con el nuevo integrante. Según Tuckman, los equipos entran nuevamente en la primera etapa de su modelo llamada forming. Un equipo que logra un buen grado de madurez tendrá mejores acuerdos de colaboración, por lo tanto, un amento en la *Madurez del equipo* aumentará también la *Colaboración* entre sus miembros. Esto aumenta la *Confianza* y a su vez la *Comunicación efectiva*. A mayor *Comunicación* entre sus integrantes aumenta la *Sincronización* en la ejecución de las tareas y un aumento en la *Sincronización* disminuye el *Tiempo de espera* para realizar la entrega del incremento de producto. Un aumento en el *Tiempo de espera* disminuirá también la *Productividad*, ya que puede existir falta de oportunidad, pérdida de mercado, entre otros. El aumento en la *Productividad* acrecienta el *Reconocimiento* para

el equipo y esta a su vez, incrementa su *Motivación* afectando de forma directa la *Estabilidad laboral* del empleado al sentir que su trabajo le hace feliz. El aumento en la *Estabilidad laboral* disminuye la *Rotación de personal*.

Cross-funcionalidad: Esta práctica puede también expresarse como equipos multidisciplinarios y promueve que las capacidades técnicas y de relacionamiento de sus integrantes, sean las suficientes y necesarias para resolver la situación problemática de principio a fin, evitando al máximo las dependencias de personas externas al equipo. Es así, como desarrolladores, testers, usuarios de negocio, diseñadores entre otros, ponen en común sus habilidades y el equipo complementará dichas habilidades en pro de la consecución de un objetivo común.

Cuando la *Cross-funcionalidad* aumenta en el equipo, generalmente a través del entrenamiento externo, mejorando así sus capacidades, se reducen las *Dependencias* que se tienen con esos externos. En el caso en que las *Dependencias* aumenten, lo que significaría incapacidad del equipo para resolver problemas de forma autónoma, el *Tiempo de espera* también aumenta, producto de que los actores externos no entreguen los insumos de forma oportuna. Un incremento en el *Tiempo de espera*, se refleja en un decremento en la *Productividad*. El aumento de la *Productividad* causa un aumento en el *Valor* para los clientes y esto a su vez una mejora en la *Confianza* del cliente en el equipo. El mejorar la *Confianza* causa un aumento en la *Colaboración* y esto provoca, un aumento en la *Cross-funcionalidad*, ya que las personas están más dispuestas a realizar transferencia de conocimiento.

Validación del modelo:

La validación general muestra una aceptación de los ciclos presentados, las variables y sus relaciones, además de las prácticas y su relación con la productividad de los equipos. La principal sugerencia realizada por los “Agile Coaches” es limitar el crecimiento del modelo basado en el cumplimiento del propósito del equipo. Esta sugerencia se validó y en conclusión, la productividad definitivamente aumentará con las prácticas presentadas, sin embargo, el crecimiento de esta productividad se dará con un límite superior que consiste en el cumplimiento de las metas del equipo, allí la productividad se hace más lenta por la sensación tranquilidad de terminar el trabajo.

4.2.5 Modelo de productividad a partir el pilar de Mejoramiento continuo–MC:

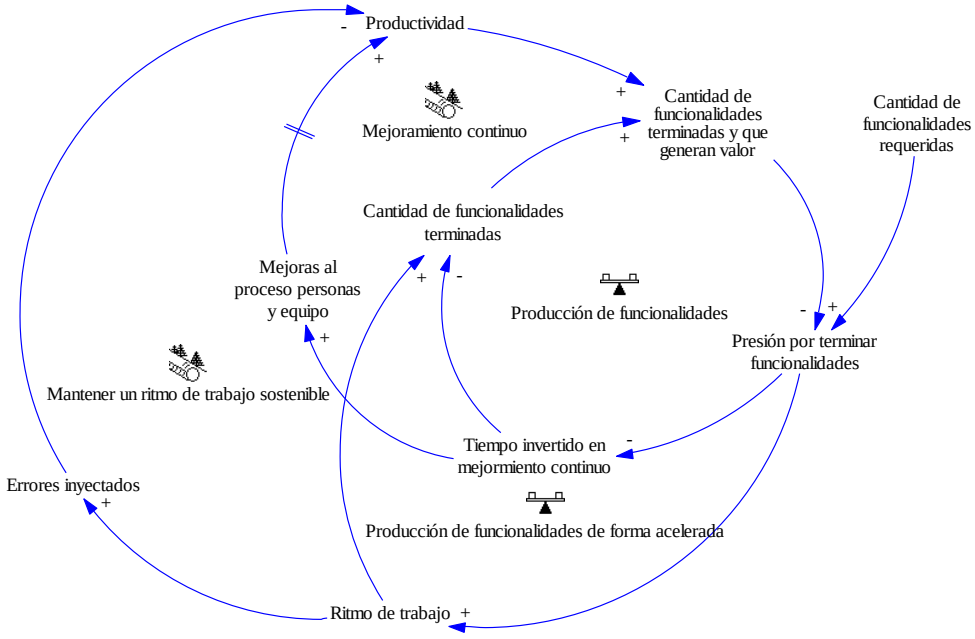
En este modelo se presentan cuatro ciclos significativos como factores que afectan la productividad de los equipos de desarrollo. El MC tiene múltiples perspectivas, el modelo es simplificado a un ámbito general y se explicará con mayor detalle cada una de las prácticas identificadas en los múltiples factores de productividad encontrados y probados en este pilar.

El MC es una práctica en sí, tomando las anteriores prácticas y llevándolas cada vez a un mejor desempeño, identificando un grupo acotado de focos para aplicar la mejora y desarrollándolos a medida que los ciclos de desarrollo avanzan. Son susceptibles de mejorar las personas, los procesos y el comportamiento de los equipos y estas mejoras deben aplicarse de manera frecuente en cada Sprint, guardando un balance entre el tiempo requerido para desarrollar el incremento de producto y el tiempo requerido para realizar mejoramiento continuo

La productividad de los equipos de desarrollo depende de las personas que lo conformen y de factores como su experiencia personal y en equipos de desarrollo de software y en el manejo de sus relaciones (Alaimo, 2014). También de sus capacidad técnicas y del desarrollo de las mismas como lenguajes de programación, patrones de diseño, técnicas modernas de desarrollo como TDD, BDD, integración continua, automatización de pruebas, despliegue continuo, desarrollo de micro-servicios, APIs, entre otros. Por último, pero no menos importantes están los factores motivacionales (de O. Melo et al., 2013; Melo et al., 2011; Pino et al., 2010), que se promueven a través de la generación de maestría, como la capacidad de las personas de aprender un arte o una función hasta el nivel de poder enseñarla. La generación de conocimiento en las personas produce motivación por desarrollar sus actividades. Otros factores atados a la motivación son el propósito común, el cual podría describirse como sentir que el trabajo del individuo genera valor para alguien, ya sea una organización o clientes y/o usuarios, saber por qué se está haciendo lo que se hace y quien se beneficia de los resultados, además de sentirse identificado con el producto creado. La experiencia que el individuo ha adquirido a través de la puesta en práctica de los conocimientos de su actividad (Dyba & Dingsoyr, 2009; Rasch & Tosi, 1992). La Seguridad psicológica de los individuos del equipo para expresar sus emociones, alertar de problemas e innovar con el ánimo de aprender y sin temor al fracaso. La

autonomía para desempeñar su trabajo, como la posibilidad del individuo de proponer y resolver los problemas de ingeniería que se presentan en su proceso y de tomar decisiones con independencia de líderes o burocracia. La aceptación en un equipo de trabajo y el reconocimiento también aumentan la motivación de los individuos. La colaboración y la comunicación efectiva son factores susceptibles de mejora continua. El ritmo sostenible de trabajo es un factor motivacional al que siempre se le debe apuntar, la mejora en las prácticas de desarrollo y a la optimización del ciclo de vida de desarrollo son fundamentales para lograr reducciones en tiempos de entrega y calidad.

Figura 4-5: Modelo de productividad a partir del pilar de Mejora continua



Fuente: Adaptación (Repenning & Sterman, 2002)

De los cuatro ciclos presentados desde el pilar de MC, dos influyen directamente en la variable productividad. Estas serán las prácticas presentadas. Los dos ciclos adicionales se describirán como parte indispensable del balance del modelo.

- Mantener un ritmo de trabajo sostenible
- Mejoramiento continuo (MC)
- Producción de funcionalidades (Balance)
- Producción de funcionalidades de forma acelerada (Balance)

Ciclos de refuerzo:

Mantener un ritmo de trabajo sostenible: Esta práctica describe por qué los equipos de trabajo deben mantener un ritmo adecuado en su trabajo. El equilibrio entre el tiempo invertido a lo personal y lo laboral debe existir para que el desempeño de sus integrantes sea el adecuado y no se afecte el factor motivacional.

Típicamente los directores de proyectos se enfrascan en conservar los tres vértices del llamado triángulo de hierro, el cual está constituido por el alcance, el tiempo y el presupuesto. Este triángulo es una práctica que obliga a los equipos a que en un tiempo estimado se desarrolle todo el alcance deseado y sin sobrepasar los costos presupuestados. A pesar de esto, la realidad de los proyectos es que estos se mueven en un entorno de alta incertidumbre, por lo que generalmente se obtiene una mala estimación del tiempo y del presupuesto suficiente para desarrollar todo el alcance, es allí donde se obtienen desfases en tiempo y/o dinero o se reduce el alcance para conservar el equilibrio.

Los métodos ágiles proponen que uno de los tres vértices debe ser variable, para así permitir al equipo afrontar la incertidumbre y desempeñarse sin romper en otras variables no consideradas en el triángulo como la calidad. Esto significa que si el alcance, el tiempo y presupuesto están cerrados, el trabajo desbordará en sobre-esfuerzo de los integrantes de los equipos, lo que se traduce en ritmo no sostenible, o en problemas de calidad del código. El ritmo no sostenible y los problemas de calidad se pone en evidencia cuando la exigencia a los integrantes del equipo se centra en tener las funcionalidades esperadas por los clientes y/o usuarios para cumplir el alcance y en el tiempo y costo deseado, no en la calidad con la que estas se construyen. Generalmente el vértice que la cultura de agilidad propone sea variable es el alcance, pues el tiempo y el costo son limitados para las organizaciones. Escenarios como lanzar la campaña antes de la competencia, obliga a las empresas a tener listo el producto en un tiempo finito, además el costo no deberá sobrepasar lo esperado pues tendría afectaciones al plan de negocio, lo que haría el proyecto posiblemente inviable. Con el alcance variable, lo que la cultura de agilidad propone, es generar el mejor producto posible que resuelva en gran medida la situación problemática, en el tiempo deseado y con el costo deseado, sin embargo, este producto puedo no contener todas las características o funcionalidades deseadas, pero si las más

importantes para su éxito. Esta filosofía es la misma que determina el pilar de entrega frecuente de producto con valor para clientes y/o usuarios.

Cuando la *Cantidad de funcionalidades terminadas* y que generan valor aumenta, la Presión por terminar funcionalidades de directores o gerentes de proyecto baja, pues existe la sensación de tranquilidad de que el equipo está avanzando al ritmo deseado. Entre mayor sea el desfase entre la *Cantidad de funcionalidades requeridas* y la *Cantidad de funcionalidad terminadas* y que generan valor, mayor será la Presión por terminar funcionalidades ya que los directores y gerentes están tratando de cumplir con el alcance completo del proyecto. Un aumento en la *Presión por terminar funcionalidades*, incrementa el Ritmo de trabajo, generalmente representado en sobre-esfuerzos del equipo. El aumento en el Ritmo de trabajo y el sobre-esfuerzo hará los Errores inyectados aumente, pues los integrantes del equipo no estarán en óptimas condiciones para mantener la calidad del código, evitando cubrirlo por pruebas unitarias ya que estas requieren un esfuerzo adicional. Los *Errores inyectados* representan un deterioro a la calidad del incremento de producto y por ende la Productividad se verá disminuida por todos los re-procesos para corregirlo. Un aumento de la Productividad, aumentará la Cantidad de funcionalidades terminadas y que generan valor para clientes y/o usuarios.

Mejoramiento continuo (MC): Este ciclo formado por cinco variables se convierte en una práctica para aumentar la productividad de los equipos de desarrollo ágil de software. Esta explica como la inversión de tiempo en MC influye positivamente en la productividad del equipo dado las mejoras que se realizan a los comportamientos personales, a los procesos y a la interacción como equipo.

Cuando la *Cantidad de funcionalidades terminadas* y que generan valor aumenta, disminuye la *Presión por terminar funcionalidades*. El aumento en la *Presión por terminar funcionalidades*, disminuye el *Tiempo invertido en mejoramiento continuo*, ya que el equipo entra en un grado de presión superior y el tiempo es invertido completamente en producir producto final para clientes y/o usuarios. Un aumento en el *Tiempo invertido en mejoramiento continuo*, incrementa las *Mejoras al proceso, personas y equipo*, en todos los posibles factores identificados. Este incremento mejora la *Productividad* de los equipos de desarrollo de software, sin embargo, dichas mejoras generalmente no se evidencian en periodos cortos como un sprint, sino que se hacen evidentes incluso para proyectos posteriores. El equipo deberá trabajar en su MC durante el sprint si desea aumentar su

nivel para el futuro. Un incremento en la *Productividad* se refleja en un incremento en la *Cantidad de funcionalidades terminadas y que generan valor*.

Ciclos de balance:

Producción de funcionalidades: Este ciclo describe el foco que generalmente se tiene por la producción normal de funcionalidades. En este modelo se evidencia que si se dedica tiempo para el MC, se tendrán un menor número de funcionalidades terminadas a corto plazo, sin embargo, como lo vimos en el ciclo de refuerzo de MC, esto puede traer un incremento en la productividad a largo plazo. Además, de las funcionalidades terminadas, se debe considerar que no todas agregan valor o el mayor valor para los usuarios, esto significa que no necesariamente producir muchas funcionalidades generará valor para los usuarios finales por lo tanto no tiene efecto en ser más productivos.

Cuando la *Cantidad de funcionalidades terminadas* y que generan valor aumenta, disminuye la Presión por terminar funcionalidades y si esta última aumenta, se reduce el Tiempo invertido en mejoramiento continuo. Si esta variable aumenta, se producen menos funcionalidades, pues los equipos tienen que distribuir sus tiempo entre generar funcionalidades y mejorar, por lo tanto se presenta una disminución de la Cantidad de funcionalidades terminadas. Si la producción de funcionalidades aumenta, también los hará la Cantidad de funcionalidades terminadas y que generan valor.

Producción de funcionalidades de forma acelerada: Este ciclo tiene un efecto similar a la Producción de funcionalidades. Este describe cómo un ritmo acelerado de trabajo también permite aumentar el grupo de funcionalidades terminadas y de igual forma, se puede producir valor por este medio. Sin embargo, tal ritmo no representa un aumento a la productividad, pues producir mucho no significa entregar el mayor valor. Por otra parte, está el agotamiento del equipo por no mantener un rito de trabajo sostenible, lo que a largo plazo, podría representar una baja en la productividad.

Validación del modelo:

Para el modelo de MC se encuentra también adecuado respecto a la coherencia, prácticas, variables y relaciones, además de representar de forma adecuada la productividad de los equipos. Las sugerencias más relevantes son las siguientes:

- Se sugiere incluir la variable motivación. Esta no está incluida explícitamente, sin embargo, el modelo considera una generalidad del MC y se explica que las mejoras obtenidas al invertir tiempo en MC están relacionadas con personas, procesos y equipo de trabajo. En la dimensión de las personas se encuentra la motivación como factor fundamental analizado en este trabajo.
- Se sugiere incluir la experimentación y validación de hipótesis, sin embargo, esta se evidencia más en la dimensión de adaptación.

Como conclusión a este capítulo, cabe destacar las 17 prácticas encontradas y representadas en los cuatro modelos de productividad. Estas prácticas representan el incremento productivo que los equipos obtendrán al adoptarla y cómo estas la afectan, ya sea por impacto a una variable de incremento o una de decremento en la función de productividad según la **Figura 1-1**: Variables de crecimiento y decrecimiento de la productividad

5. Conclusiones y recomendaciones

5.1 Conclusiones

En esta tesis se realiza un análisis de cómo la cultura de agilidad y el marco de trabajo Scrum, promueven a través de 25 factores, la productividad de los equipos que desarrollan software. Este análisis se realizó usando la herramienta de pensamiento sistémico, demostrando ser adecuada para analizar la relación entre los factores identificados en la literatura.

Se realizó la descripción de la cultura de agilidad basada en el manifiesto ágil, además del marco de trabajo Scrum para el desarrollo de software, identificando 25 factores que podrían afectar la productividad de los equipos de forma directa. Con los factores identificados se realizó un mapeo a cuatro pilares que Cockburn (2016) define como el corazón de la agilidad, pues es un contenedor de lo que significa ser ágil. A partir de allí, se tomaron los cuatro pilares y se realizó un mapeo de los 25 factores, identificando que es posible realizar un mapeo directo, con lo cual el corazón de la agilidad es buen contenedor de la definición de agilidad.

A partir del análisis realizado con PS se concluye, que tanto la cultura de agilidad como el marco de trabajo Scrum, promueven de forma eficaz la productividad en los equipos de desarrollo ágil de software. Esto a partir de enseñanzas obtenidas de los errores cometidos con métodos tradicionales basados en cascada.

Se realizó un análisis con PS de los cuatro pilares respecto a la variable Productividad. Este análisis mostró los principales ciclos de realimentación que a su vez se convierten en las prácticas sugeridas para maximizar la productividad de los equipos de desarrollo. Esto se mostró en diagramas causales que fueron validados con cinco “Agile Coaches”, quienes realizaron el ejercicio con sus respectivos equipos de desarrollo ágil de software del ámbito

financiero colombiano. A estos se les aplicó una encuesta, encontrando que los equipos encuentran coherente el modelo, sus variables y sus relaciones, además de las prácticas (ciclos de realimentación) del sistema mismo. Esto permitió comprobar que cada práctica presentada afecta de forma positiva la productividad de los equipos de desarrollo.
Las prácticas definidas en los modelos son: Adaptación temprana, Producto frecuente y oportuno en el mercado, La adaptación temprana, Crear pequeños incrementos de código para reducir la cantidad de errores, Poner en producción pequeños incrementos del producto, Adaptación entre los proyectos para maximizar el valor, Priorización por valor y foco, Generar sólo funcionalidades útiles, Generar el producto correcto, Minimizar la incertidumbre sobre la necesidad de los clientes, Auto-organización, Mejorar la transferencia de conocimiento, Mantener la sincronización a todo nivel, Foco en entender y validar, Equipos estables, Cross-funcionalidad, Mantener un ritmo de trabajo sostenible, Mejoramiento continuo (MC).

El uso de PS como herramienta de modelado y análisis de la productividad se encuentra altamente efectiva para comprender la dinámica que por sí solos no reflejan los factores identificados en la literatura. Esta dinámica resulta indispensable para comprender las prácticas propuestas en esta tesis y aplicarlas de manera adecuada en los equipos de desarrollo de software.

5.2 Recomendaciones

Como trabajos futuros y como parte del análisis de productividad se desprenden los siguientes posibles trabajos futuros:

- Análisis y priorización de la demanda de la organización: Una de las prácticas encontradas en este trabajo es “priorizar lo importante”. Esto debe realizarse a nivel equipo, sin embargo, para la generación de valor a nivel organización, este ejercicio debe realizarse en niveles superiores. Priorizar los proyectos de la compañía es una práctica relevante. Esto podría incluirse en los factores influyentes en análisis posteriores.
- Análisis de la capacidad general de empresas con múltiples equipos de desarrollo: La capacidad en términos de la oferta con relación a la demanda de proyectos de la organización y su buena administración influirán en la productividad de la

empresa, más allá de la productividad de cada equipo, ya que esta, estará en función de lo producido y los recursos necesarios para producir como se explica en la **Figura 1-1**: Variables de crecimiento y decrecimiento de la productividad, es decir, que de no administrarse bien la capacidad, la organización podría estar enfocada en hacer lo que se puede hacer y no lo que se debe hacer.

- En términos de colaboración se identifican limitantes legales en la regulación Colombiana cuando en los equipos están trabajando personas de diferentes empresas, práctica que es común en organizaciones donde su CORE no es propiamente hacer software. En estos casos, la empresa contratante tiene algunas limitantes por temas de subordinación y esto no facilita la colaboración, entrenamiento, compartir prácticas y demás al interior de los equipos.
- Extender los modelos y hacer el paso siguiente hacia la construcción de modelos formales que permitan hacer simulación. Usando diagramas de flujos y niveles, se daría lugar un análisis cuantitativo de la productividad.

A. Anexo: Encuesta de validación de los modelos de productividad

Se anexa la encuesta usada para la validación de cada modelo. Esta encuesta está compuesta por el modelo de cada pilar y cinco preguntas realizadas a cinco diferentes equipos a través de su agile coach asignado. Las respuestas se conservan de acuerdo a como el Agile coach las entrega, posteriormente se realizó validación con cada uno de ellos. Los hallazgos encontrados se presentan en el capítulo 4 en la validación de cada modelo.

Encuesta 5-1: Encuesta de validación de los modelos de productividad

Bibliografía

Alaimo, D. M. (2013). *Proyectos ágiles con #Scrum : flexibilidad, aprendizaje, innovación y colaboración en contextos complejos*.

Alaimo, D. M. (2014). *Equipos más productivos*.

Ashby, W. R. (1956). AN INTRODUCTION TO CYBERNETICS. *Director*, *80*(4), 295. http://doi.org/10.2307/3006723

Balijepally, V., Mahapatra, R., Nerur, S., & Price, K. H. (2009). Are two heads better than one for software development? The productivity paradox of pair programming. *MIS Quarterly: Management Information Systems*, *33*(1), 91–118. Retrieved from http://www.scopus.com/inward/record.url?eid=2-s2.0-60649093502&partnerID=tZOtx3y1

Bash, E. (2015). SCRUM AND XP FROM THE TRENCHES. *PhD Proposal*, *1*. http://doi.org/10.1017/CBO9781107415324.004

Bass, J. M. (2016). Artefacts and Agile Method Tailoring in Large-Scale Offshore Software Development Programmes. *Information and Software Technology*, *75*, 1–16. http://doi.org/10.1016/j.infsof.2016.03.001

Beck, K. (1999). *Extreme Programming Explained: Embrace Change*. *XP Series*. http://doi.org/10.1136/adc.2005.076794

Beck, K., Beedle, M., Bennekum Van, A., Cockburn, A., Cunningham, W., Fowler, M., … Thomas, D. (2001). Manifesto for Agile Software Development. Retrieved from http://agilemanifesto.org/

Campanelli, A. S., & Parreiras, F. S. (2015). Agile methods tailoring – A systematic literature review. *Journal of Systems and Software*, *110*, 85–100. http://doi.org/10.1016/j.jss.2015.08.035

Chulani, S., Boehm, B. W., & Steece, B. (1999). Bayesian Analysis of Empirical Software Engineering Cost Models. *IEEE Transactions on Software Engineering*, *25*(4), 573–583. http://doi.org/10.1109/32.799958

Cockburn, A. (2016). The heart of agile. *CrossTalk*, *29*(6).

de O. Melo, C., S. Cruzes, D., Kon, F., & Conradi, R. (2013). Interpretative case studies on agile team productivity and management. *Information and Software Technology*, *55*(2), 412–427. http://doi.org/10.1016/j.infsof.2012.09.004

Deemer, P. (n.d.). Una introducción básica a la teoría y práctica de Scrum.

Dingsøyr, T., Nerur, S., Balijepally, V., & Moe, N. B. (2012). A decade of agile methodologies: Towards explaining agile software development. *Journal of Systems and Software*, *85*(6), 1213–1221. http://doi.org/10.1016/j.jss.2012.02.033

Dyba, T., & Dingsoyr, T. (2009). What Do We Know about Agile Software Development? *IEEE Software*, *26*(5), 0–3. http://doi.org/10.1109/MS.2009.145

Eloranta, V.-P., Koskimies, K., & Mikkonen, T. (2016). Exploring ScrumBut—An empirical study of Scrum anti-patterns. *Information and Software Technology*, *74*, 194–203. http://doi.org/10.1016/j.infsof.2015.12.003

Escalable, D. S. (2015). Guia Nexus ™.

Fontana, R. M., Fontana, I. M., da Rosa Garbuio, P. A., Reinehr, S., & Malucelli, A. (2014). Processes versus people: How should agile software development maturity be defined? *Journal of Systems and Software*, *97*, 140–155. http://doi.org/10.1016/j.jss.2014.07.030

Fontana, R. M., Meyer, V., Reinehr, S., & Malucelli, A. (2015). Progressive Outcomes: A framework for maturing in agile software development. *Journal of Systems and Software*, *102*, 88–108. http://doi.org/10.1016/j.jss.2014.12.032

Heikkilä, V. T., Paasivaara, M., Rautiainen, K., Lassenius, C., Toivola, T., & Järvinen, J. (2015). Operational release planning in large-scale Scrum with multiple stakeholders – A longitudinal case study at F-Secure Corporation. *Information and Software Technology*, *57*, 116–140. http://doi.org/10.1016/j.infsof.2014.09.005

Holmström, H., Fitzgerald, B., Ågerfalk, P. J., & Conchúir, E. Ó. (2006). Agile Practices Reduce Distance inGlobal Software Development. *Information Systems Management*, *530*(May 2012), 37–41. http://doi.org/10.1201/1078.10580530/46108.23.3.20060601/93703.2

IFPUG. (2000). International Function Point User Group. Retrieved January 1, 2016, from www.ifpug.com

Kniberg, H., Jeff, P. De, & Cohn, M. (2007). *Scrum Y Xp Desde Las*. *Online*. Retrieved from http://www.proyectalis.com/wp-content/uploads/2008/02/scrum-y-xp-desde-las-

trincheras.pdf

Kniberg, H., & Skarin, M. (2010). *Kanban and Scrum-making the most of both*. *Work* (Vol. 1). Retrieved from http://www.infoq.com/minibooks/kanban-scrum-minibook

Larusdottir, M., Gulliksen, J., & Cajander, Å. (2016). A License To Kill - Improving UCSD In Agile Development. *Journal of Systems and Software*. http://doi.org/10.1016/j.jss.2016.01.024

Lei, H., Ganjeizadeh, F., Jayachandran, P. K., & Ozcan, P. (2015). A statistical analysis of the effects of Scrum and Kanban on software development projects. *Robotics and Computer-Integrated Manufacturing*. http://doi.org/10.1016/j.rcim.2015.12.001

Lo Giudice, D., & Angel, N. (2013). *Q3 2013 Global Agile Software Application Development Online Survey*. Retrieved from https://www.forrester.com/

Maurer, F., & Martel, S. (2002). Extreme programming: Rapid development for web-based applications. *IEEE Internet Computing*, *6*(1), 86–90. http://doi.org/10.1109/4236.989006

Melo, C., Cruzes, D. S., Kon, F., & Conradi, R. (2011). Agile team perceptions of productivity factors. *Proceedings - 2011 Agile Conference, Agile 2011*, 57–66. http://doi.org/10.1109/AGILE.2011.35

Moe, N. B., Dingsøyr, T., & Dybå, T. (2010). A teamwork model for understanding an agile team: A case study of a Scrum project. *Information and Software Technology*, *52*(5), 480–491. http://doi.org/10.1016/j.infsof.2009.11.004

Nguyen, V., Deeds-Rubin, S., Tan, T., & Boehm, B. (2007). A SLOC Counting Standard. *COCOMO II Forum.*, *2007*, 1–15.

Olszewska (née Pląska), M., Heidenberg, J., Weijola, M., Mikkonen, K., & Porres, I. (2016). Quantitatively Measuring A Large-Scale Agile Transformation. *Journal of Systems and Software*. http://doi.org/10.1016/j.jss.2016.03.029

Petersen, K. (2011). Measuring and predicting software productivity: A systematic map and review. *Information and Software Technology*, *53*(4), 317–343. http://doi.org/10.1016/j.infsof.2010.12.001

Pikkarainen, M., Haikara, J., Salo, O., Abrahamsson, P., & Still, J. (2008). The impact of agile practices on communication in software development. *Empirical Software Engineering*, *13*(3), 303–337. http://doi.org/10.1007/s10664-008-9065-9

Pino, F. J., Pedreira, O., García, F., Luaces, M. R., & Piattini, M. (2010). Using Scrum to guide the execution of software process improvement in small organizations. *Journal*

of Systems and Software, *83*(10), 1662–1677. http://doi.org/10.1016/j.jss.2010.03.077

Poppendieck, M., & Llc, P. (1991). Lean Software Development : A Tutorial, 26–32.

Purna Sudhakar, G., Farooq, A., & Patnaik, S. (2011). Soft factors affecting the performance of software development teams. *Team Performance Management: An International Journal*, *17*(3/4), 187–205. http://doi.org/10.1108/13527591111143718

RAE. (2014). Real Academia Española. Retrieved from http://www.rae.es/

Rasch, R. H., & Tosi, H. L. (1992). Factors Affecting Software Developers' Performance: An Integrated Approach. *MIS Quarterly*, *16*(3), 395. http://doi.org/10.2307/249535

Repenning, N. P., & Sterman, J. D. (2002). Nobody ever gets credit for fixing problems that never happened: Creating and sustaining process improvement. *IEEE Engineering Management Review*, *30*(4), 64–78. http://doi.org/10.1109/EMR.2002.1167285

Royce, W. W. (1970). Managing the development of large software systems. *Electronics*, *26*(August), 1–9. http://doi.org/10.1016/0378-4754(91)90107-E

Schwaber, K., & Sutherland, J. (2016). The Scrum Guide. *Scrum.Org and ScrumInc*, (July), 17. http://doi.org/10.1053/j.jrn.2009.08.012

Scrum.org. (2016). Retrieved from https://www.scrum.org/

Senge, P. M. (1990). The fifth discipline. *Measuring Business Excellence*, *1*(3). http://doi.org/10.1108/eb025496

Shared Services – Scaled Agile Framework. (2016). Retrieved from http://www.scaledagileframework.com/shared-services/

Shull, F., Melnik, G., Turhan, B., Layman, L., Diep, M., & Erdogmus, H. (2010). What do we know about test-driven development? *IEEE Software*, *27*(6), 16–19. http://doi.org/10.1109/MS.2010.152

Snowden, D. J., & Boone, M. E. (2007). A leader's framework for decision making. *Harvard Business Review*, *85*(11). Retrieved from http://www.scopus.com/inward/record.url?eid=2-s2.0-36348971683&partnerID=tZOtx3y1

Sugimori, Y., Kusunoki, K., Cho, F., & Uchikawa, S. (1977). Toyota production system and Kanban system Materialization of just-in-time and respect-for-human system. *International Journal of Production Research*, *15*(6), 553–564.

http://doi.org/10.1080/00207547708943149

Sverrisdottir, H. S., Ingason, H. T., & Jonasson, H. I. (2014). The Role of the Product Owner in Scrum-comparison between Theory and Practices. *Procedia - Social and Behavioral Sciences*, *119*, 257–267. http://doi.org/10.1016/j.sbspro.2014.03.030

Takeuchi, Hirotaka, and I. N. (1986). The New New Product Development Game. *Harvard Business Review 64, No. 1*, *1*.

Tuckman, B. W. (1965). Developmental sequence in small groups. *Psychological Bulletin*, *63*(6). http://doi.org/10.1037/h0022100

Vlietland, J., & van Vliet, H. (2015). Towards a governance framework for chains of Scrum teams. *Information and Software Technology*, *57*, 52–65. http://doi.org/10.1016/j.infsof.2014.08.008

von Wangenheim, C. G., Savi, R., & Borgatto, A. F. (2013). SCRUMIA—An educational game for teaching SCRUM in computing courses. *Journal of Systems and Software*, *86*(10), 2675–2687. http://doi.org/10.1016/j.jss.2013.05.030