UNIVERSIDAD GABRIELA MISTRAL
FACULTAD DE INGENIERIA

“RADIOGRAFÍA A LOS PROYECTOS DE SOFTWARE DESARROLLADOS EN INSTITUCIONES PÚBLICAS DE SALUD DE ATENCIÓN CERRADA PERTENECIENTES A LA REGIÓN METROPOLITANA DURANTE EL PERIODO COMPRENDIDO ENTRE LOS AÑOS 2016 Y EL PRIMER TRIMESTRE DEL 2017”.

Memoria para optar al título de Ingeniero de Ejecución en Informática

Autor : Daniel Peñailillo Ceballos.
Profesor Guía : Franco González Lecaros.
Profesor Integrante : Roberto Carú Cisternas.

Santiago – Chile
Julio, 2017

INDICE

AGRADECIMIENTOS

Si nos ponemos a recordar los momentos que vivimos en la universidad, esos raros pensamientos de no saber lo que nos esperaba más adelante, los grandes trasnoches de estudio, el cambio de sensaciones donde cada uno poco a poco se va dando cuenta que va cambiando y madurando un poco a la vez, así como también las grandes amistades, en este momento es cuando uno siente y está dispuesto a vivir todo esto por segunda vez.

Sin duda existen muchas personas a las que se debe agradecer, tal vez algunas sin relevancia alguna con nuestra vida, pero aun así cada una cumplió su labor esencial en nuestra formación. Sin embargo, no olvidemos los más grandes apoyos, quienes fueron el pedestal perfecto para mantenerme firme en todo momento.

Mi padre Daniel estricto, orgulloso, inteligente, generoso y con una forma distinta de entregar su conocimiento a los demás, inculcó en mí valores que muy difícilmente serán olvidados y seguirá aportándolos por que como dice él: “yo tengo la experiencia”.

Mi madre Nadia, cuyas cualidades de bondad, generosidad, amor, esfuerzo, compasión y un sinfín más que supo traspasar a cada uno de sus hijos, han dejado huella en mí y cuyo trabajo incesante en pos de nosotros demuestra cada día que con esfuerzo todo es posible. Ambos son la base de lo que soy y seré siempre.

Y, por supuesto, mis hermanos Dianna, Luis, Sebastián, Leonardo y Alfonso, quienes son los pilares fundamentales de mí estructura. Son parte importante en mi vida, una mezcla de alegría, orden, generosidad, juventud, entrega, compromiso, inteligencia. Son los más grandes ejemplos de personas que quiero llegar a ser y yo a su vez tratar de ser un ejemplo para Uds.

No puedo dejar de mencionar a mis sobrinos, Leonor y Mateo, quienes me dan la energía necesaria para esforzarme y superarme día a día. Gracias por su amor, por entender mi ausencia y por creer en mí. Sin su cariño nada de esto sería posible.

Por último, no puedo de dejar de mencionar a mis amigos, parte importante de mi vida, dispuestos a estar contigo en la buenas y en las malas, como hermanos, sin pedir nada a cambio, y de los cuales estoy seguro seguirán siendo amigos, aun así, pasen los años.

A todos aquellos que creyeron en mí y lo seguirán haciendo.

GRACIAS.

CAPITULO I: INFORMACIÓN GENERAL.

INTRODUCCIÓN.

La necesidad de buscar mejoras, es una acción inherente a la Gestión en Calidad, inicialmente esta solo era practicada en la industria, y posteriormente se extendió a empresas comerciales y de Salud. El objeto de esta práctica, era obtener una mayor producción y mejores productos aumentando así sus ingresos, reconocimiento y prestigio. La metodología consiste en identificar situaciones de riesgo para prevenir o mitigar eventos no deseados y así estos no interfieran en el curso normal de procesos. Con el tiempo, esta práctica se ha facilitado gracias a la utilización de softwares, los cuales inicialmente comenzaron siendo proyectos con fines científicos o gubernamentales, pero que gradualmente comenzaron a ser parte del entorno empresarial utilizados principalmente en administración de finanzas, recurso humano y producción. Dentro de los productos ofrecidos por el mercado se encuentran los softwares comerciales y los desarrollados a medida, ambos cuentan con ventajas y desventajas, y hasta la fecha, tanto su posibilidad de éxito como de fracaso es incierto ya que esta condición puede presentarse en cualquier etapa de la línea de tiempo de su desarrollo, obligando muchas veces a la reingeniería de procesos y sus correspondientes herramientas. Se han realizado numerosos esfuerzos por conocer la real causa de esto, pero sin resultados concretos hasta el momento. Esto queda demostrado con el hecho de que aún el número de proyectos fallidos y con evolución incierta han superado a la cantidad de proyectos exitosos según lo expuesto por Standish Group en su CHAOS Report 2015.

En las Instituciones de Salud, si bien, la informática forma parte de sus procesos, no se ha desarrollado en su máxima capacidad, y prueba de esto, es el hecho de que aún prevalece el papel por sobre el almacenamiento digital. Referencias aportadas por expertos en el área, señalan que de cada 10 proyectos informáticos solicitados a las Unidades de Tecnología e Informática (TI), solo 1 es documentado y las restantes

solicitudes quedan a la espera de mejores condiciones que permitan su desarrollo. Aun cuando no se cuenta con un respaldo escrito de tal referencia, se especula que las dificultades experimentadas se relacionan con el usuario, experto y entorno, lo cual coincide con lo expuesto por Standish Group quienes han investigado este fenómeno en la Industria y Empresas de gran embergadura.

Resulta interesante el desafío de poder documentar la experiencia de los expertos TI que se desempeñan en Instituciones Públicas de Salud, para así identificar situaciones que nos permitan idear estrategias de mejora, es por este motivo, que esta tesis busca realizar un diagnóstico y conocer así la experiencia de los profesionales informáticos durante el desarrollo de proyectos de software en Instituciones de Salud de Atención Cerrada en la Región Metropolitana, esto nos permitirá contar un mayor número de antecedentes que a futuro alimentarán nuevas estrategias de trabajo que faciliten y asegure la interacción Experto TI, Usuario y Ambiente Hospitalario.

Inicialmente se elaboraró y validaró un cuestionario el cual permitió el levantamiento de la experiencia en Unidades TI. La población objeto, corresponde a los Profesionales Informáticos pertenecientes a las Instituciones de Salud de Atención Cerrada de la Región Metropolitana a quienes se les solicito su participación en forma voluntaria. La información obtenida fue orientada en la búsqueda de posibles factores que pudiesen influir en el éxito y/o fracaso de proyectos de software los cual se espera contribuya a la generación de estrategias de mejora reflejadas en acciones preventivas o mitigadoras.

PLANTEAMIENTO DEL PROBLEMA.

En las Instituciones de Salud, se estima que un porcentaje importante de proyectos informáticos solicitados a las Unidades de TI experimentan dificultades durante su ciclo de vida. Si bien los procesos y necesidades hospitalarias son transversales para todas las instituciones, se requiere de un diagnóstico que refleje la realidad local y global y así nos permita desarrollar metodologías de trabajo adecuadas que contribuyan al éxito del proyecto a desarrollar desde el punto de vista de su gestión, del producto a desarrollar y de su permanencia y adaptación al cambio en el tiempo.

OBJETIVO GENERAL.

Conocer la experiencia de los expertos en Tecnologías en Información y Comunicaciones durante el desarrollo de proyecto de software en sus Instituciones Públicas de Salud de Atención Cerrada en la Región Metropolitana durante el periodo comprendido entre el año 2016 y el primer trimestre del 2017.

OBJETIVOS ESPECÍFICOS.

- Elaborar un cuestionario para realizar el levantamiento de experiencias de expertos TI durante el desarrollo de proyectos de software en Instituciones de Salud de Atención Cerrada en la Región Metropolitana.
- Validar cuestionario mediante análisis de expertos considerando parámetros de Utilidad y comprensión del mismo.
- Levantar información en Profesionales Informáticos que se desempeñan en Instituciones de Salud mediante aplicación de Cuestionario previamente elaborado y validado por expertos.

- Describir el comportamiento de los proyectos de software realizados en Instituciones Públicas de Salud.
- Evaluar la tendencia de los principales atributos de un proyecto de software durante los periodos 2016 y 2017.
- Identificar los principales factores que puedan influir en el éxito o fracaso de proyectos de software desarrollados en instituciones de Salud de Atención Cerrada.
- Realizar un Diagnóstico en base a los proyectos de software desarrollados en Instituciones Públicas de Salud de Atención Cerrada.

JUSTIFICACIÓN.

Los Proyectos Informáticos son fundamentales para que las organizaciones tengan éxito en los negocios y productividad. Actualmente, dentro del ámbito de la Salud, las instituciones privadas lideran esta evolución desarrollando proyectos innovadores y de crecimiento, mientras las instituciones Públicas desde un punto de vista más conservador invierten su mayor esfuerzo en Proyectos Vitales, los cuales no representan un riesgo para el archivo en papel, debido a que buscan dar respuesta inmediata a requerimientos emanados de instituciones encargadas de normar y fiscalizar sus procesos, tales como el Ministerio y Superintendencia de Salud respectivamente.

Claramente, las nuevas legislaciones, los avances tecnológicos y los requerimientos, cada vez más complejos y variados de sus usuarios, obligarán al sector público a la automatización de procesos e informatización, para así mantener los estándares mínimos de calidad en su atención y así ser competitivos con el sector privado. Cuando llegue este momento, los expertos en Tecnología de la Información y Comunicación(TIC) deben contar con las herramientas necesarias que le permitan detectar y desarrollar exitosamente estas oportunidades de negocio.

En un primer acercamiento, esta tesis realiza un diagnóstico sobre los proyectos de software desarrollados en Instituciones Públicas de Salud de Atención Cerrada identificando, desde el punto de vista de su gestión, los principales atributos que determinan su éxito o fracaso posterior a su implementación. Esta información, permitirá a futuro, elaborar estrategias que potencien las metodologías empleadas mejorando los resultados obtenidos y optimizando tres recursos fundamentales en las instituciones de Salud como lo son el tiempo, hora hombre y recurso monetario.

Por otra parte, las instituciones en Salud tienen una estructura transversal lo cual permitiría que un mismo proyecto de software exitoso pueda ser implementado en cualquier institución de la misma naturaleza generando mayor demanda y ampliando el nicho de nuestro negocio. Los beneficios generados contribuirán tanto a la institución requirente del proyecto informático como a los equipos TIC participantes.

CAPITULO II: MARCO TEÓRICO.

ROL DE DEPARTAMENTOS DE GESTIÓN DE TECNOLOGÍAS DE LA INFORMACIÓN EN INSTITUCIONES DE SALUD.

Desde una perspectiva organizacional, el Departamento de Gestión en Tecnologías de Información (TI) depende jerárquicamente de la Subdirección Administrativa y de Finanzas, junto a los Departamentos de Gestión Financiera y Contable, Recursos Físicos, Abastecimiento, Recursos Humanos, Oficina de partes, Convenios y Estadísticas.

En una mirada funcional, este Departamento apoya a toda la organización en lo referente a Tecnologías de Información y Comunicaciones aplicadas en Salud, para la implementación y explotación de iniciativas estratégicas.

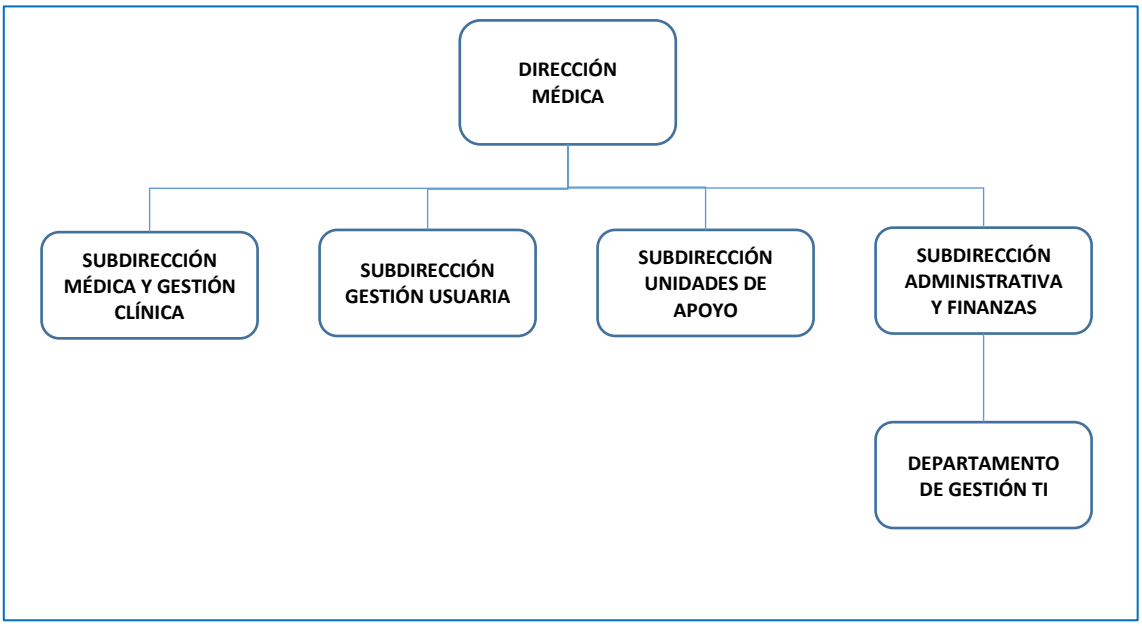


Fig. 1. Organigrama Abreviado en Instituciones de Salud.

Técnicamente están capacitados para desarrollar variadas actividades con el objeto de generar y promover iniciativas de índole tecnológica que permitan optimizar recursos y contribuyan a mejorar la atención del paciente. Dentro de estas destacan:

El estudiar, formular y proponer alternativas de solución para la informatización de procesos hospitalarios manuales o semi-automatizados.
Generar aplicaciones informáticas que den respuesta a los requerimientos generados dentro de la Institución Hospitalaria.
Mantener actualizadas las aplicaciones existentes, adecuándolas a su vez a los nuevos cambios y niveles de gestión que se necesiten.
Ejecutar el proceso de implementación de sistemas de información, asumiendo con esto la capacitación usuaria.
Asesorar y colaborar con la Subdirección Administrativa del Complejo en la gestión de la información, proponiendo modelos de desarrollo del área informática.
Dado este perfil, es difícil comprender por qué aún reina el papel por sobre el disco duro dentro de instituciones de salud.

EXPERIENCIAS EN PROYECTOS DE SOFTWARE DESARROLLADOS POR DEPARTAMENTOS DE GESTIÓN DE TECNOLOGÍAS DE LA INFORMACIÓN.

El conocimiento sobre el “Como se está realizando” se encuentra en la experiencia de los actores, para poder llegar al “como debería Realizarse” es necesario conocer y analizar la experiencia existente para luego con la ayuda de la literatura generar las propuestas que nos permitan enunciar una nueva propuesta metodológica.
Existen variadas herramientas para realizar levantamiento experiencias, y para poder seleccionar la más idónea es necesario realizar un breve análisis de los actores, y su entorno.

Los Actores: son todos los profesionales que han participado de proyectos de software desarrollados en instituciones de salud. En este caso ese grupo de persona está constituida por los Ingenieros que se desempeñan en Unidades de Gestión de Tecnologías de la Información y los referentes Clínicos dueños del requerimiento que

forman parte del recurso humano de cada Institución Hospitalaria Públicas. Cabe señalar que los profesionales pertenecientes a consultorías externas, fueron excluidos de este diagnóstico debido a que pueden otorgar sus servicios a instituciones públicas y privadas ampliando y sesgando la experiencia estudiada. En Chile aún existe diferencia en la gestión realizada entre Instituciones Públicas y Privadas. Claramente existe una notoria brecha tecnológica entre ellas, mientras un resistente sistema público aún lucha por encontrar la Calidad con la ayuda de herramientas análogas, un innovador y creciente sistema privado trabaja duro para lograr y mantener la Excelencia utilizando herramientas digitales.

El Entorno: Las instituciones en Salud pertenecen al sistema Público de Atención y se encuentran distribuidas en distintos puntos geográficos en la Región Metropolitana. Su objetivo principal, es dar atención oportuna, accesible y continua a sus usuarios, lamentablemente, la demanda de atención supera a la oferta y el mayor tiempo en su gestión está destinado a paliar necesidades dirigidas a la atención clínica, es por este motivo, que toda iniciativa no relacionada con los problemas reales de la institución pierde relevancia y es postergado en el tiempo, tal es el caso de las soluciones informáticas pequeñas.

ANTECEDENTES HISTÓRICOS VINCULADOS AL ÉXITO O FRACASO DE PROYECTOS DE SOFTWARE.

Antes de situarnos en lo que respecta al contexto histórico propiamente tal, es necesario definir éxito, el cual inicialmente fue considerado como el cumplimiento de las 3 constantes del clásico triángulo de gestión formado por Tiempo, Presupuesto y Alcance (Project Management Institute (PMI)). Cabe destacar que aun cumpliendo con dichas características los proyectos no lograban los beneficios esperados por los usuarios y la organización, es por este motivo, que se recomiendo complementar la búsqueda del Alcance con el logro de Resultados Satisfactorios debido a que las expectativas varían durante el desarrollo del proyecto y merecen ser gestionadas.

Dentro de estudios realizados con miras a mostrar el comportamiento de los proyectos de software destaca el Informe de CHAOS, desarrollado por Standish Group en grandes, medianas y pequeñas empresas ubicadas en los principales segmentos de la industria como la banca, ventas al por menor y mayor, fábricas, seguros, entre otras. La herramienta para recolección de datos fue una encuesta aplicada a ejecutivos responsables de TI y los resultados fueron agrupados en tres tipos de resolución: exitosos, fracasados e inciertos.

El Informe es publicado anualmente, y en el 2015 se señala que el 19% de los proyectos evaluados se consideraron un fracaso, frente a un 22% obtenido en el levantamiento realizado en el año 2011, a pesar de los grandes y acelerados avances tecnológicos existentes solo existió un 3% de mejora con respecto al escenario anterior. A su vez, el 29% de los proyectos de 2011 fueron considerados un éxito obteniendo el mismo porcentaje en el año 2015 dejando ver una fase de meseta que se mantiene constante en el tiempo. El porcentaje restante estaría centrado en proyectos considerados como problemáticos o inciertos que a diferencia de las dos situaciones mencionadas anteriormente estaría en aumento (Standish Group, 1995).

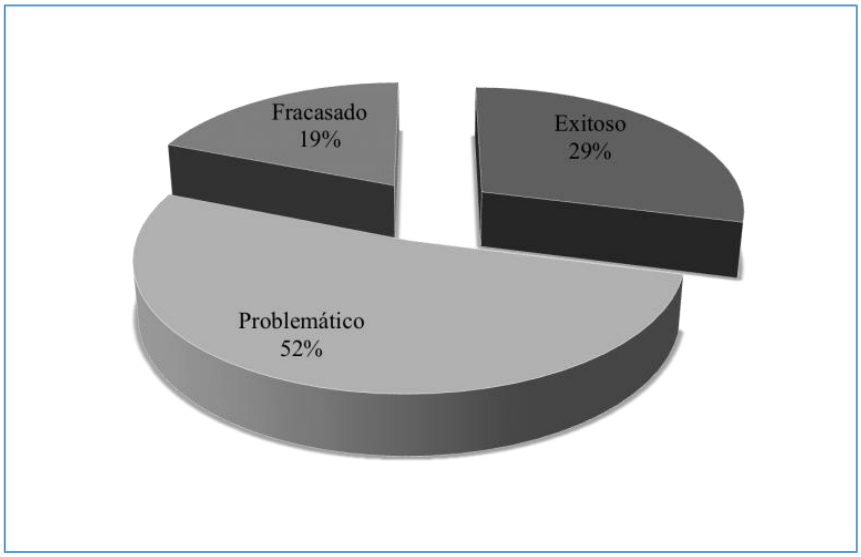
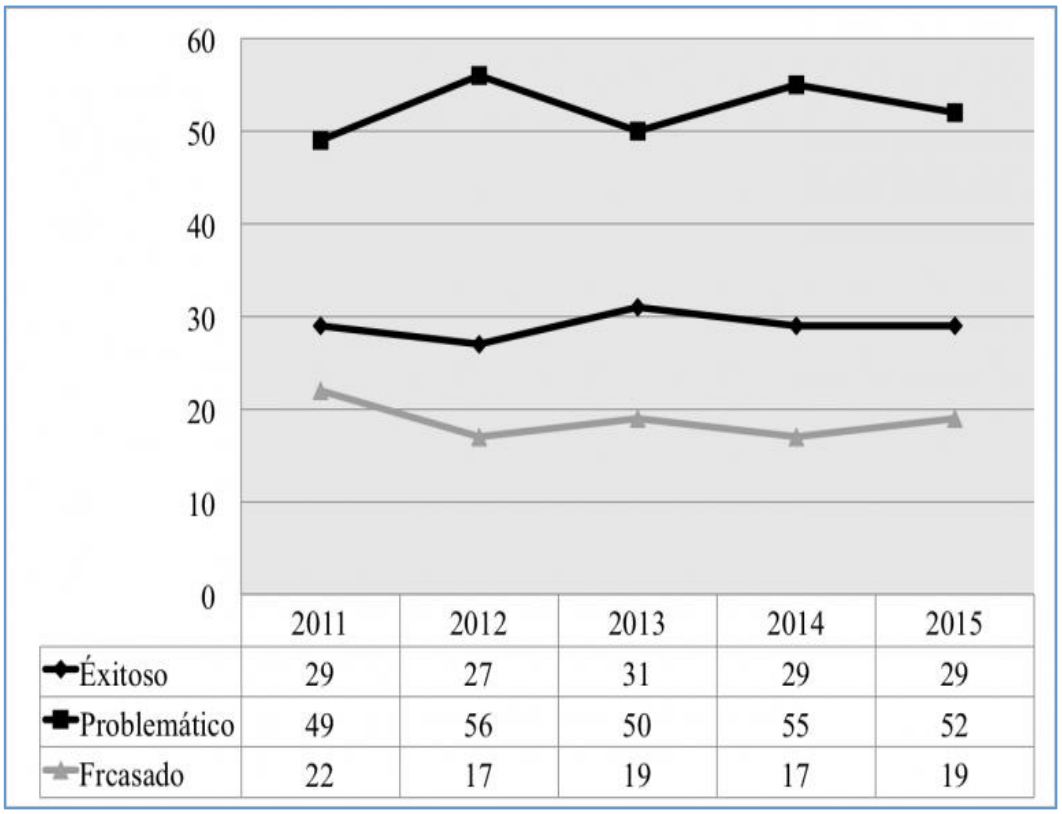


Fig. 2.- **Resultados CHAOS Report 2015.**

Claramente, estos resultados demandan acciones inmediatas destinadas a mejorar esta realidad. Dentro de las conclusiones planteadas el informe señala que la probabilidad de éxito de un proyecto parece inversamente relacionada al tamaño del mismo, cuanto más grande es el proyecto, más probable es que no sea un éxito. Estos resultados no difieren mucho de los pesquisados en años anteriores (Fig. 3), pero aun cuando numéricamente los porcentajes se mantienen existe una percepción por parte de la gerencia de que existe más fracaso ahora que hace 5 y 10 años atrás.



	2011	2012	2013	2014	2015
Éxitoso	29	27	31	29	29
Problemático	49	56	50	55	52
Frcasado	22	17	19	17	19

Fig. 3.- Evolución de los Resultados según CHAOS Report.

Dentro de las principales causales de fracaso o incertidumbre se señalan el reinicio de estos, el sobrecosto, Incumplimiento de los requisitos de especificación iniciales y retraso en el tiempo estimado de término. Cabe destacar que estas fallas se presentaron en forma aislada y combinadas.

En cuanto a las causas de éxito detectadas los factores más relevantes tienen relación con la participación del Usuario y el apoyo de la gerencia, pero también es importante contar con una clara declaración de los requerimientos y necesidades, una adecuada planificación, expectativas realistas, hitos de menor tamaño, personal competente, propiedad, personal enfocado en el trabajo duro, visión y objetivos claros.

Por consiguiente, los factores enumerados anteriormente generaron un listado con los elementos a evitar durante el desarrollo de un proyecto de software como lo son: falta de participación usuaria, requisitos o especificaciones incompletos, cambios en las necesidades y especificaciones, falta de apoyo de la gerencia, incompetencia tecnológica, falta de recursos, expectativas poco realistas, objetivos poco claros, marcos de tiempo poco realistas, y nueva tecnología.

Si bien, muchos de estos factores parecieran evidentes y mejorables, existió un factor no mencionado por Standish Group que tiene relación con la capacidad de comunicación y convencimiento entre los actores participantes, en donde muchas veces, prevalecía una orden jerárquica por sobre la opinión del ejecutivo en TI o una expertiz inflexible de esta misma persona ante un requerimiento viable por parte del usuario. No debemos dejar de mencionar el entorno o procesos ignorados por el usuario con el objeto de generar requerimientos personales que no lograrán una satisfacción colectiva.

Joseph Gulla(2012), en IBMSystems magazine, señala que existen 7 razones principales para el fracaso del proyecto las cuales son Pobre Planificación y dirección de proyectos, comunicación insuficiente, gestión ineficaz, no alineación entre los constituyentes del proyecto y las partes interesadas, participación ineficaz de la gestión ejecutiva, falta de habilidades blandas o capacidad de adaptarse, y pobre o nulo uso de Metodologías y Herramientas (Gulla, 2012). Coincide con lo expuesto por Standish Group, con respecto a la necesidad de una Planificación precisa, objetivos claramente definidos, gerentes proactivos, pero a la vez suma la necesidad de asignaciones claras y una comunicación

eficaz que le permitan al equipo de trabajo identificar y enfrentar los riesgos potenciales para así evitar retrasos y demoras costosas en el futuro. Sutilmente, atribuye la responsabilidad del éxito del proyecto a la administración, señalando que su inadecuada gestión produce el 54% de los fracasos, y que los métodos y problemas técnicos solo representan un 8% y 3% respectivamente.

La revista IEEE Spectrum, en el estudio publicado por Robert Charette (2005) hacía alusión a los mismos elementos refiriendo que los proyectos fallaban por metas poco realistas o mal articuladas, estimaciones imprecisas de los recursos, requisitos del sistema mal definidos, deficiente información sobre el estado del proyecto, riesgos no gestionados, falta de comunicación entre los clientes, desarrolladores y usuarios; uso de tecnología inmadura, incapacidad para manejar la complejidad del proyecto, prácticas descuidadas de desarrollo, mala gestión del proyecto, políticas de las partes interesadas, y presiones comerciales (Charette, 2005). Si la comunidad Informática ya había sido advertida de los posibles factores de fracaso es desalentador ver que a la fecha aún se sigan discutiendo y cuantificando de la misma forma, dejando en evidencia que las mejoras realizadas no han generado un impacto significativo.

El 2011 Gartner entrega otra visión a esta línea investigativa proporcionando recomendaciones sobre cómo aumentar el éxito de proyectos TI, para esto analizó 845 proyectos con la colaboración de un proveedor de servicios externos, la encuesta fue aplicada el 2010 en los estados Unidos, Reino Unido, Francia y Alemania. Dentro de sus resultados destaca un 42,5% del total de proyectos que no generan los beneficios esperados (fig 4), 44% sobrepasaron el presupuesto definido inicialmente (fig 5) y el 42% no fueron entregados a tiempo (fig 6), estos factores se presentaron en forma aislada y asociada en la muestra analizada.

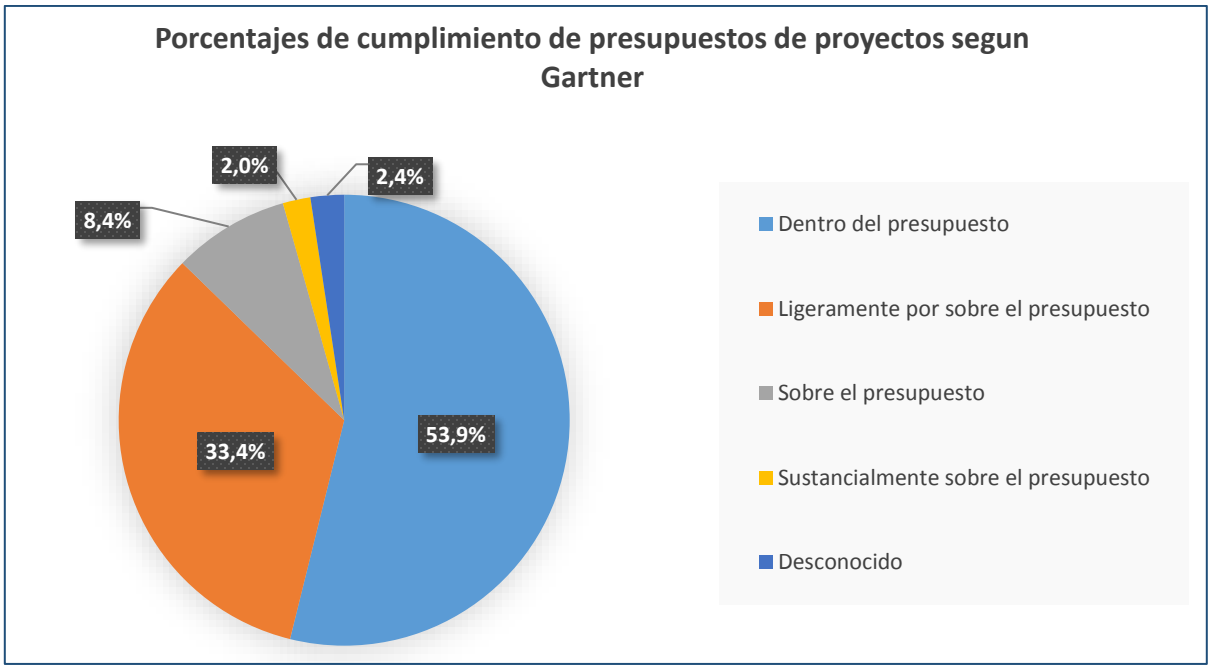


Fig. 4 Presentación de porcentajes de cumplimiento de Presupuesto de proyectos evaluados por Gartner en el año 2011.

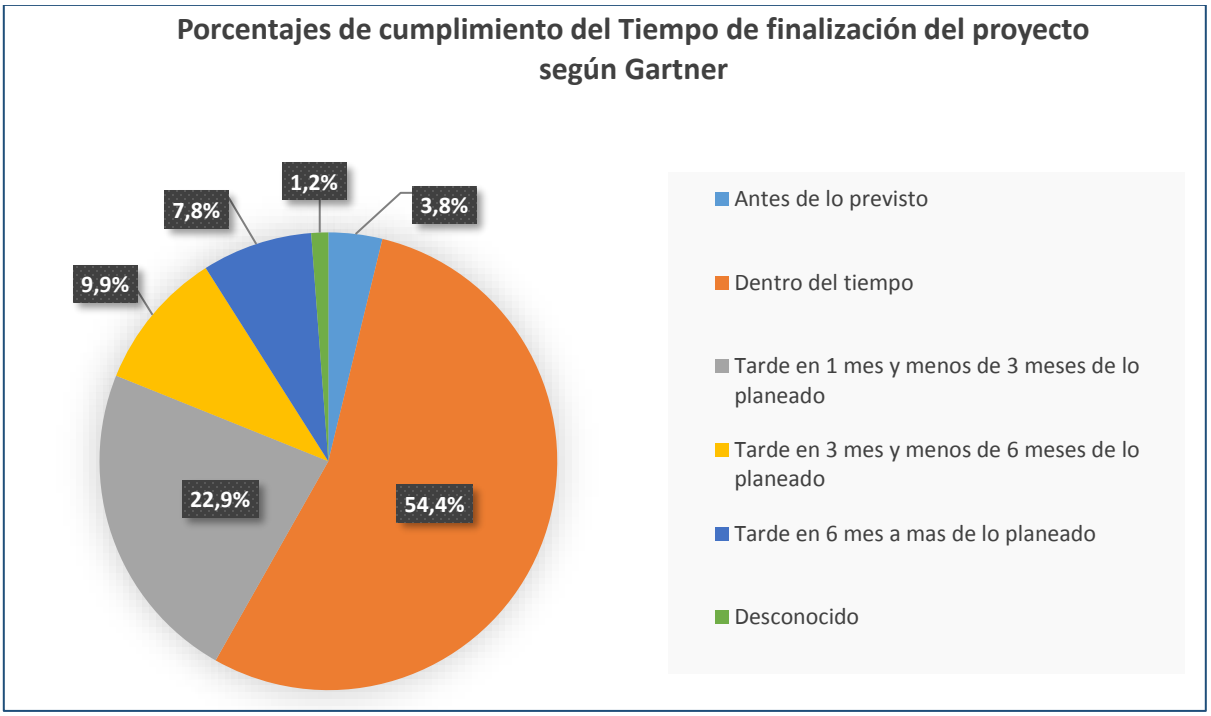


Fig. 5 Presentación de porcentajes de cumplimiento de Tiempo de finalización de proyectos evaluados por Gartner en el año 2011.

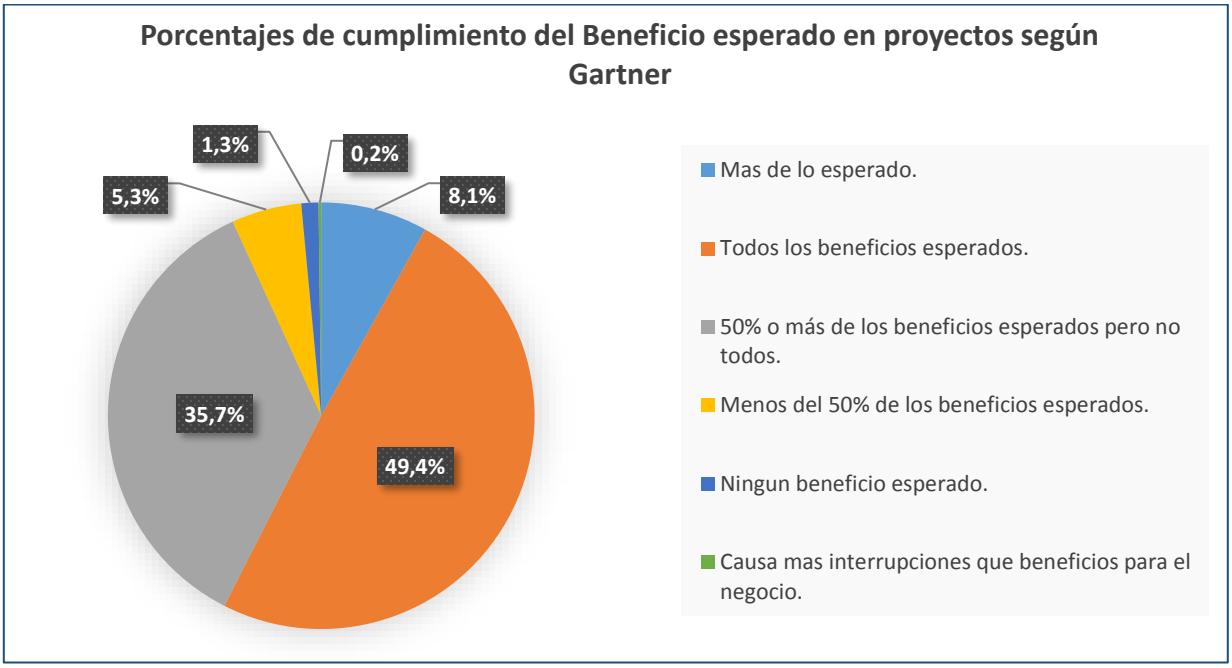


Fig. 6 Presentación de porcentajes de cumplimiento de Beneficios esperados en proyectos evaluados por Gartner en el año 2011.

Gartner sugiere que, para obtener el éxito de un proyecto, estos no solo deben centrarse en cumplir con estos tres factores mencionados, sino además con un conjunto de atributos que van a influir en su resultado final. Es así como diseñan el denominado “Five Natural Clusters” constituido por planificación, gestión de proyectos, Habilidades y experiencia técnica, gestión de usuarios, y habilidades cualitativas. Aun cuando existen elementos que contribuyen más que otros al proyecto, es necesario considerarlos todos ya que la ausencia o deficiente gestión de cualquiera de ellos reducen la probabilidad de éxito de manera significativa.

Otro aspecto decisivo en el futuro de un proyecto informático, tiene relación con el uso de Metodologías Agiles, las cuales han demostrado un aumento en la probabilidad de éxito independientemente del tamaño del proyecto, esto se asocia a los resultados

pesquisados en el mismo estudio que alimenta al Informe Chaos Report, en donde se observó un incremento del 11% al 39% de proyectos exitosos y a la vez una baja del 29% al 9% de fracasos durante el periodo 2011 y 2015. A primera vista, pareciera indicar que existen factores de éxito relacionados con la metodología empleada por los actores.

INFLUENCIA DE LA METODOLOGÍA EMPLEADA EN EL ÉXITO DE UN PROYECTO DE SOFTWARE.

Otro aspecto decisivo en el futuro de un proyecto informático, tiene relación con la elección de la metodología a utilizar, ya que el éxito de los proyectos de software depende en gran medida de que tenga un buen inicio. El requisito principal de la metodología a elegir se relaciona con la capacidad de esta para adecuarse a las características y circunstancias del proyecto, lo cual permitirá realizar un buen uso del tiempo y mejorar la calidad de los sistemas a producir.

Cuando el Ingeniero comienza con la comparación y clasificación de las metodologías existentes le resulta una actividad compleja debido a la gran diversidad de propuestas y diferencias en el nivel de detalle, información disponible y alcance de cada una de ellas.

Si se tiene en cuenta su filosofía de desarrollo, se pueden distinguir dos fuertes corrientes metodológicas: las ágiles, que dan prioridad a la interacción entre los individuos y a la comunicación con el cliente; y las tradicionales, también llamados pesados que son los que promueven la disciplina por medio de la planificación y la comunicación escrita.

La elección correcta de una metodología no asegura el éxito de un proyecto, lo que realmente marca la diferencia, es el nivel de conocimiento por parte del equipo de proyecto, para así, asegurar una correcta selección y aplicación de esta.

DESCRIPCIÓN DE METODOLOGIAS TRADICIONALES.

Las metodologías tradicionales llevan asociado un marcado énfasis en el control de procesos mediante una rigurosa definición de roles, actividades y artefactos, incluyendo modelado y documentación detallada (Canós, Letelier, & Penadés, Metodologías Ágiles en el desarrollo de software, 2003).

Estas metodologías imponen una disciplina de trabajo sobre el proceso de desarrollo del software, con el fin de conseguir un producto más eficiente. Para ello se hace énfasis en la planificación total de todo el trabajo a realizar y una vez se encuentra todo detallado comienza el ciclo de desarrollo.

Este esquema ha demostrado ser efectivo y necesario en proyectos de gran tamaño, respecto a tiempo y recursos, donde por lo general se exige un alto grado de ceremonia en el proceso. Sin embargo, este enfoque no resulta ser el más adecuado para muchos de los proyectos actuales donde el entorno del sistema es muy cambiante, y en donde se exige reducir drásticamente los tiempos de desarrollo, pero manteniendo una alta calidad.

Dentro de las metodologías tradicionales existentes destacan: Modelo en cascada, Modelo evolutivo, Modelo de componentes Reutilizables, Modelos Híbridos.

Modelo En cascada.

El modelo en cascada es uno de los primeros modelos utilizados en el desarrollo de software, el cual fue popularizado por Winston Royce en el año 1970 y aún sigue vigente. Se caracteriza por seguir una secuencia de fases ordenadas, en donde cada etapa tiene una entrada y una salida, relacionándose linealmente (Royce, 1970). La estrategia principal de este modelo es definir y seguir el progreso del desarrollo hacia puntos de revisión bien definidos, es decir es un proceso continuo de codificación y reparación.
Las fases que contempla este modelo son:

Ingeniería y análisis del sistema, debido a el software es siempre parte de un sistema mayor, el trabajo comienza estableciendo los requisitos de todos los elementos del sistema y luego asignando algún subconjunto de estos al software.

análisis y especificaciones de requerimiento, el proceso de recopilación de los requisitos se centra e intensifica especialmente en el software. Se trabaja con los clientes y usuarios finales del sistema para determinar el dominio de la aplicación y los servicios que debe proporcionar el sistema, así como sus restricciones. Con esta información se elabora el documento de Especificación de Requerimientos.

Diseño del sistema y del software, en esta fase se identifican los requerimientos que son del software y los del hardware, posterior a esto establece estructura de los datos, arquitectura completa del sistema, detalle procedimental y caracterización de la Interfaz. Con estas acciones correctamente ejecutadas se logra identificar los subsistemas existentes y se describe cómo funciona cada uno y sus relaciones.

Codificación, consiste en codificar y probar los diferentes subsistemas por separado. La prueba de unidades implica verificar que cada uno cumpla su especificación.

Integración y validación del sistema, una vez probadas cada una de las unidades, estas son integradas para formar el sistema completo, el cual debe cumplir con todos los requisitos del software. Si las pruebas efectuadas al sistema completo son exitosas, el software es entregado al cliente.

Liberación y mantenimiento, el software sufrirá cambios después de que se entrega al cliente. Esto se debe a que en esta fase se corrigen errores detectados, se realizaran cambios para una mejor adaptación al entorno cambiante, o bien, el cliente requiera ampliaciones funcionales y/o del rendimiento.

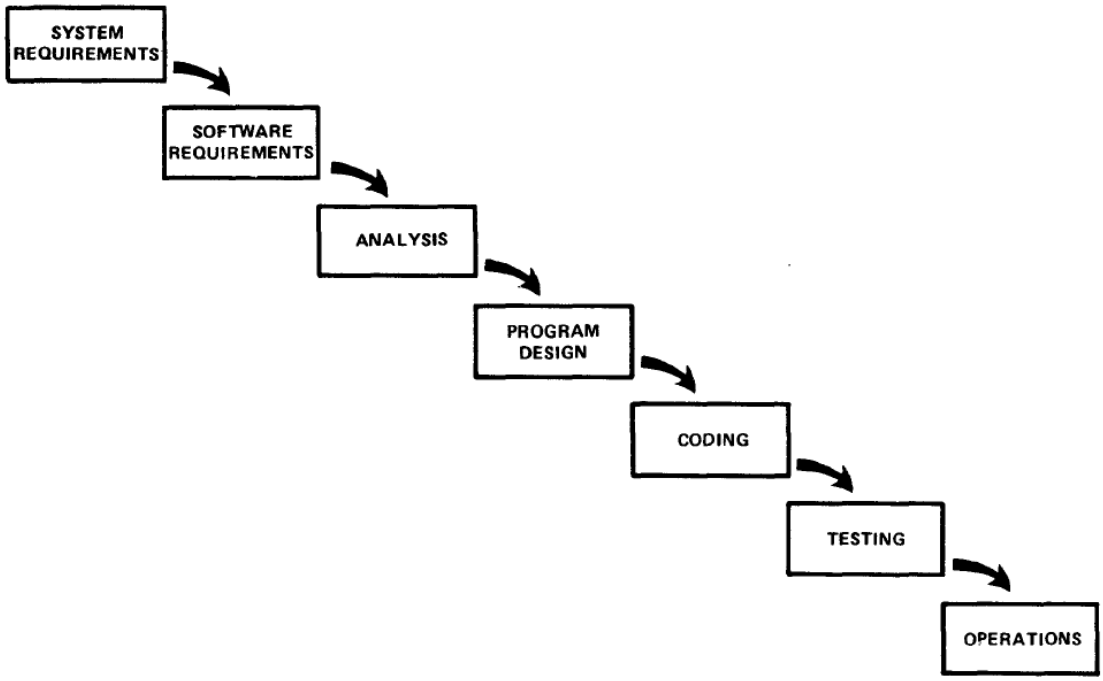


Fig. 7 Modelo en Cascada propuesto por Winston Walker Royce en 1970.

En principio, el resultado de cada fase es uno o más documentos aprobados(“firmados”). La siguiente fase no debe empezar hasta que la fase previa haya finalizado. En la práctica, estas etapas se superponen y proporcionan información a las otras. Durante el diseño se identifican los problemas con los requerimientos; durante el diseño del código se encuentran problemas, y así sucesivamente. El proceso del software no es un modelo lineal simple, sino que implica una serie de iteraciones de las actividades del desarrollo (Sommerville, 2005).

A este modelo pertenecen el modelo en cascada puro, cascada con fases solapadas, cascadas con subproyectos y cascada con reducción de riesgo. Todos se caracterizan por generar documentación completa del sistema y seguir una secuencia serial de actividades ya descritas anteriormente. El modelo en cascada con fases solapadas permite realizar actividades de las siguientes fases en paralelo a las últimas actividades de la fase anterior sin romper la secuencia. El modelo de cascada con subproyectos,

conserva el carácter secuencia de las actividades aun cuando lo divide en proyectos más pequeños los cuales se desarrollan en paralelo integrándolos todos al final. Por último, en el modelo de cascada con reducción de riesgo este es controlado en la fase de requerimientos con una espiral que los identifica, mitiga y prevé la posibilidad de retroceder en la secuencia de actividades manteniéndolas en el mismo orden que una cascada pura.

Si la gran ventaja de los modelos en cascada es que la documentación se produce en cada fase y que éste cuadra con otros modelos del proceso de ingeniería. Su principal problema es su inflexibilidad al dividir el proyecto en distintas etapas. Al hacer compromisos en las etapas iniciales, se hace difícil responder a los cambios en los requerimientos del cliente.

Por lo tanto, el modelo en cascada sólo se debe utilizar cuando los requerimientos se comprendan bien y sea improbable que cambien radicalmente durante el desarrollo del sistema. Sin embargo, el modelo refleja el tipo de modelo de proceso usado en otros proyectos de la ingeniería. Los procesos del software que se basan en este enfoque se siguen utilizando para el desarrollo del software, particularmente cuando este es parte de proyectos grandes de ingeniería de sistemas.

Modelo Evolutivo.

Estos modelos se caracterizan por ser iterativos, lo cual permite a los ingenieros de software desarrollar versiones cada vez más eficientes, efectivas y eficaces del sistema. La iteración está centrada sobre las actividades de especificación, desarrollo y validación.

Inicialmente el desarrollo se realiza a partir de los requerimientos prioritarios del sistema, los cuales deben ser aquellos que estén mejor definidos, una vez obtenida, esta primera versión es refinada en una nueva iteración con las peticiones del cliente para generar un sistema que logre satisfacer sus necesidades.

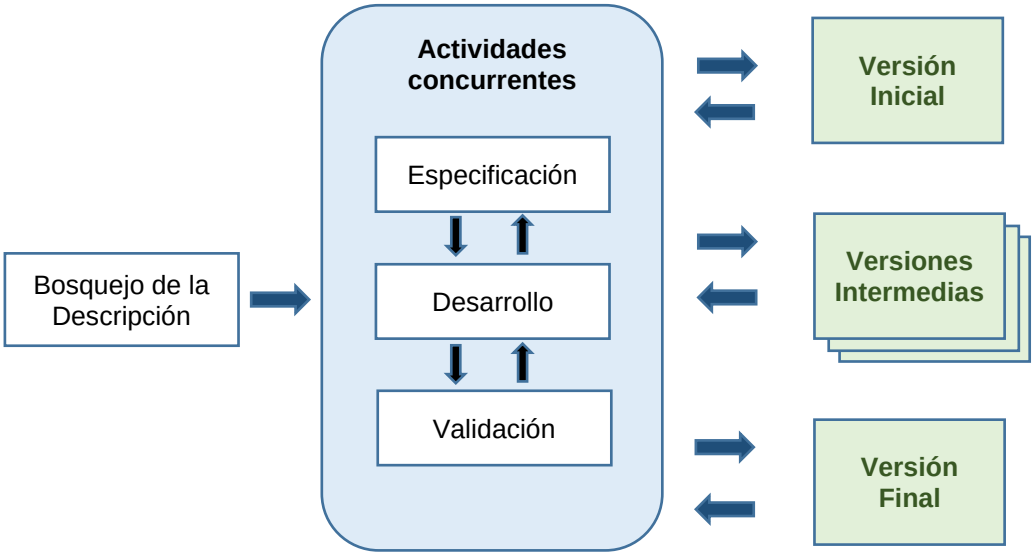


Fig. 8 Modelo Evolutivo.

Existen dos tipos de desarrollo evolutivo:

Desarrollo exploratorio, en donde el objetivo del proceso es trabajar con el cliente para explorar sus requerimientos y refinar el sistema hasta que se logra un producto adecuado. El desarrollo comienza con los elementos que se comprenden mejor y el sistema evoluciona agregando nuevos atributos propuestos por el o los clientes.

Prototipos desechables, ya sea para descubrir o terminar de comprender los requerimientos del cliente se construye un prototipo con funcionalidad simulada y, si este no es lo que el cliente espera, se construye otro prototipo el cual inclusive puede ser desde cero con una definición mejorada de los requerimientos para el sistema. En el modelo evolutivo de tipo prototipo desechable, se comienza definiendo los requisitos que no están claros para el usuario y se utiliza un prototipo para experimentar con ellos. El diseño va evolucionando según se vayan entendiendo los requerimientos, aunque la

funcionalidad siga siendo simulada. Cuando se aclaran los requerimientos se completa la funcionalidad según el último prototipo.

Sommerville señala que, en la producción del sistema, un enfoque evolutivo para el desarrollo del software suele ser más efectivo que el enfoque en cascada descrito recientemente, ya que satisface las necesidades inmediatas de los clientes. La ventaja de un proceso de software que se basa en un enfoque evolutivo es que la especificación se puede desarrollar de forma creciente. Tan pronto como los usuarios desarrollen un mejor entendimiento de su problema, éste se puede reflejar en el sistema de software. Sin embargo, desde una perspectiva de ingeniería y de gestión, el enfoque evolutivo tiene dos problemas:
El proceso no es visible, esto se debe a que los administradores tienen que hacer entregas regulares para medir el progreso. Si los sistemas se desarrollan rápidamente, no es rentable producir documentos que reflejen cada versión del sistema.

A menudo los sistemas tienen una estructura deficiente debido a que los cambios continuos tienden a corromper la estructura del software y el incorporar cambios en él se convierte cada vez más en una tarea difícil y costosa.

Modelo de Componentes Reutilizables.

“Un componente es una unidad de composición de aplicaciones de software, que posee un conjunto de interfaces y un conjunto de requisitos, y que ha de poder ser desarrollado, adquirido, incorporado al sistema y compuesto con otros componentes de forma independiente, en tiempo y espacio” (Szyperski, 1998).

El modelo de componentes reutilizables se basa en la existencia de un número significativo de componentes que pueden ser utilizados nuevamente. El reuso tiene como finalidad utilizar nuevamente ideas, arquitecturas, diseños o códigos de una aplicación

para elaborar otras. El proceso de desarrollo del sistema se enfoca en integrar estos componentes en el sistema, y así, no comenzar a desarrollarlos desde cero.
En muchos proyectos de software existen elementos que pueden ser reutilizados, pero el disponer de componentes no es suficiente a menos que seamos capaces de reutilizarlos. El reutilizarlo no significa tan solo que será utilizado más de una vez, sino además implica que su capacidad permitirá utilizarlo en contextos distintos a aquellos para los que fue diseñado. Comúnmente, esto sucede cuando los ingenieros que trabajan en el proyecto conocen diseños o códigos similares al requerido, es en ese momento que lo buscan, lo modifican y luego es incorporado en el nuevo sistema. Esta reutilización informal es independiente del proceso de desarrollo que se utilice, sin embargo, en estos últimos años ha surgido un enfoque de desarrollo de software denominado ingeniería de software basada en componentes(CBSE) que se basa en la reutilización, el cual se está utilizando ampliamente.

Las etapas de especificación de requerimientos y de validación son comparables con los otros procesos, las etapas intermedias en el proceso orientado a la reutilización son diferentes. Estas etapas son:

Análisis de componentes: consiste en encontrar componentes que sirvan para desarrollar la especificación de Requerimientos. En general, los componentes que se utilizan sólo proporcionan parte de la funcionalidad requerida por lo que se necesita modificarlo.

Modificación de Requerimientos: Con la información que se obtiene de los componentes identificados, se analizan los requerimientos para ver si concuerdan con los componentes existentes y disponibles. Si las modificaciones no pueden ejecutarse, entonces se lleva a cabo nuevamente el análisis de componentes para buscar soluciones alternativas.

Diseño del sistema con reutilización: Se diseña o reutiliza un marco de trabajo para el nuevo sistema teniendo en cuenta los componentes que serán completamente nuevos.

Desarrollo e integración: El software que no se tiene disponible y que no se puede adquirir externamente se desarrolla integrando los componentes reutilizables disponibles. En este modelo, la integración de los sistemas es parte del desarrollo más que una actividad separada.

La ingeniería de software basada en componentes, bien ejecutada, tiene la ventaja de reducir la cantidad de componentes a desarrollar, disminuir los costos, minimizar riesgos, reducir el tiempo de entrega. Si las nuevas versiones de los componentes reutilizables no están bajo control de la organización que los utiliza, se pierde parte del control sobre la evolución del sistema lo que puede ocasionar el no cumplimiento de las necesidades globales del usuario, lo cual es considerado uno de los grandes factores de fracaso en los proyectos de software.

Modelos Híbridos.

Los modelos híbridos se caracterizan por reunir elementos de todos los modelos de procesos genéricos (Sommerville, 2005), dentro de estos citamos Rational Unified Process(RUP), propuesto por IBM Rational en donde proponen el desarrollo de software basado en las mejores prácticas recopiladas de un conjunto de proyectos exitosos. El proceso Unificado, a diferencia de los modelos de procesos genéricos, se describe desde tres perspectivas:

Perspectiva Dinámica: Muestra las etapas del modelo sobre el tiempo. El RUP cuenta con cuatro fases:

Inicio: El objetivo es establecer un caso de negocio para el sistema. Se deben identificar todas las entidades externas que interaccionarán con el sistema,

definiendo dichas interacciones. Esta información levantada es empleada para evaluar el aporte que el sistema hace al negocio. Si este no es importante se puede cancelar el proyecto después de esta etapa.

Elaboración: Los objetivos de esta etapa son lograr una comprensión del dominio del problema, desarrollar el plan de proyecto e identificar los riesgos claves de este. Al finalizar esta etapa, los productos esperados son: un modelo de requerimientos del sistema, una descripción arquitectónica y un plan de desarrollo de software.

Construcción: esta etapa comprende el diseño, programación y pruebas del sistema; se realiza mediante una serie de iteraciones en donde se extraen algunos casos de uso, se refinan e implementan para nuevas pruebas. El producto esperado es un sistema de software operativo, y la documentación correspondiente para ser entregada al o los usuarios.

Transición: el sistema es probado en un entorno real, previamente este debe ser trasladado desde la comunidad de desarrollo hacia la comunidad del usuario. Esta actividad se considera de alto costo. El objetivo final de esta etapa es un sistema documentado que funciona correctamente en su entorno operativo.

Perspectiva Estática: Muestra las actividades que tienen lugar durante el proceso de desarrollo. Éstas se denominan flujos de trabajo en la descripción del RUP. Existen seis principales flujos de trabajo del proceso identificados en el proceso y tres principales flujos de trabajo de soporte. EL RUP se ha diseñado conjuntamente con el UML, por lo que la descripción del flujo de trabajo se orienta alrededor de los modelos UML asociados. Sommerville señala que los flujos de trabajo empleados son:

Modelado del Negocio: Los procesos del negocio se modelan utilizando casos de uso de negocio.

Requerimientos: Se definen los actores que interactúan con el sistema y se desarrollan casos de uso para modelar los requerimientos del sistema.
Análisis y diseño: Se crea y documenta un modelo del diseño utilizando modelos arquitectónicos, modelos de componentes, modelos objetivos y modelos de secuencias.

Implementación: Se implementan y estructuran en subsistemas los componentes del sistema. La generación automática de código de los modelos de diseño ayuda a acelerar este proceso.

Pruebas: Las pruebas son un proceso iterativo que se llevan a cabo conjuntamente con la implementación.

Despliegue: Se crea una reléase del producto, se distribuye a los usuarios y se instala en su lugar de trabajo.

Configuración y cambios de gestión: Este flujo de trabajo de soporte gestiona los cambios del sistema.

Gestión del proyecto: Este flujo de trabajo de soporte gestiona el desarrollo del sistema.

Entorno: Este flujo de trabajo se refiere a hacer herramientas de software apropiadas disponibles para los equipos de desarrollo de software.

Perspectiva Práctica: Sugiere 6 buenas prácticas a utilizar durante el proceso, estas son:

Gestión de Requisitos: RUP brinda una guía para encontrar, organizar, documentar y seguir los cambios de los requisitos funcionales y restricciones. Emplea una notación de casos de uso y escenarios para representar los requisitos.
Desarrollo del Software Iterativo: Durante el desarrollo se realizan iteraciones con hitos bien definidos, en donde las actividades son repetidas, pero con distinto énfasis según la fase del proyecto en la que se encuentre.

Desarrollo basado en componentes: La creación de sistemas intensivos requiere de la división del sistema en componentes con interfaces bien definidas, que luego serán ensambladas para generar el sistema. Esta característica permite que el sistema se vaya creando a medida que se obtienen sus componentes.

Modelado visual: Las herramientas de modelado visual facilitan la gestión de dichos modelos, permitiendo ocultar o exponer detalles cuando sea necesario. El modelado visual también mantiene la consistencia. En resumen, mejora la capacidad del equipo para gestionar la complejidad del software.

Verificación continua de la Calidad: Es muy importante que durante el proceso de desarrollo se evalúe la calidad en varios puntos, especialmente al momento de finalizar cada iteración. En esta verificación las pruebas juegan un rol fundamental y es por este motivo que deben ser integrada a lo largo de todo el proceso.

Gestión de los Cambios: En todo proyecto de software el cambio es un factor de riesgo crítico debido a que este no solo se presenta durante las acciones de mantenimiento posteriores a la entrega del producto, sino que también durante el proceso de desarrollo. Dentro de los cambios experimentados, los más importantes, debido a su impacto, son los que se producen en los requisitos. Por otra parte, el hecho de que exista la participación simultanea de múltiples desarrolladores implica disciplina ya que la ausencia de esta puede conducir al caos del proyecto.

DESCRIPCIÓN DE METODOLOGIAS ÁGILES.

Por más que en los últimos años, han evolucionado diversas metodologías para asegurar un mejor control del proceso, los clientes quedan frecuentemente insatisfechos con el resultado (Aguilar, 2002). Si bien, la mala administración ha sido la principal causa de fracaso en los proyectos de software, también se atribuye a la metodología empleada. Las metodologías Ágiles surgen como otra alternativa de desarrollo para contrarrestar estos fracasos. En la década del 90, se crea The Ágil Alliance, con el objeto de ayudar a la profesión a pensar en desarrollo de software, metodologías y organizaciones, de manera más Ágil. La primera contribución de dicha alianza fue elaborar el Manifiesto para el desarrollo de software Ágil, el cual consta de 12 principios comunes a las metodologías Ágiles de desarrollo (Beck, y otros, 2001), estos eran:

- Nuestra mayor prioridad es satisfacer al cliente mediante la entrega temprana y continua de software con valor.
- Son bienvenidos los requisitos cambiantes, incluso si llegan tarde al desarrollo. Los procesos ágiles se doblegan al cambio como ventaja competitiva para el cliente.
- Entregar con frecuencia software que funcione, en periodos de un par de semanas hasta un par de meses, con preferencia en los periodos breves.
- Las personas del negocio y los desarrolladores deben trabajar juntos de forma cotidiana a través del proyecto.
- Construcción de proyectos en torno a individuos motivados, dándoles la oportunidad y el respaldo que necesitan y procurándoles confianza para que realicen la tarea.
- La forma más eficiente y efectiva de comunicar información de ida y vuelta dentro de un equipo de desarrollo es mediante la conversación cara a cara.
- El software que funciona es la principal medida del progreso.
- Los procesos ágiles promueven el desarrollo sostenido. Los patrocinadores, desarrolladores y usuarios deben mantener un ritmo constante de forma indefinida.

- La atención continua a la excelencia técnica enaltece la agilidad.
- La simplicidad como arte de maximizar la cantidad de trabajo que no se hace, es esencial.
- Las mejores arquitecturas, requisitos y diseños emergen de equipos que se auto organizan.
- En intervalos regulares, el equipo reflexiona sobre la forma de ser más efectivo y ajusta su conducta en consecuencia.

La utilización de todas las buenas prácticas enumeradas en el manifiesto Ágil no implica ser Ágil, sin embargo, el hecho de incumplir una de ellas te transforma en no Ágil.

Dentro de las metodología Ágiles existentes destacan:

Programación Extrema (Extreme Programming, XP)

La metodología denominada XP, es la primera metodología ágil y la que le dio conciencia al movimiento actual de metodologías ágiles, se centra en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y permitiendo un buen clima laboral. La programación Extrema se basa en la realimentación continua entre cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y valentía para enfrentar los cambios. Es especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico.

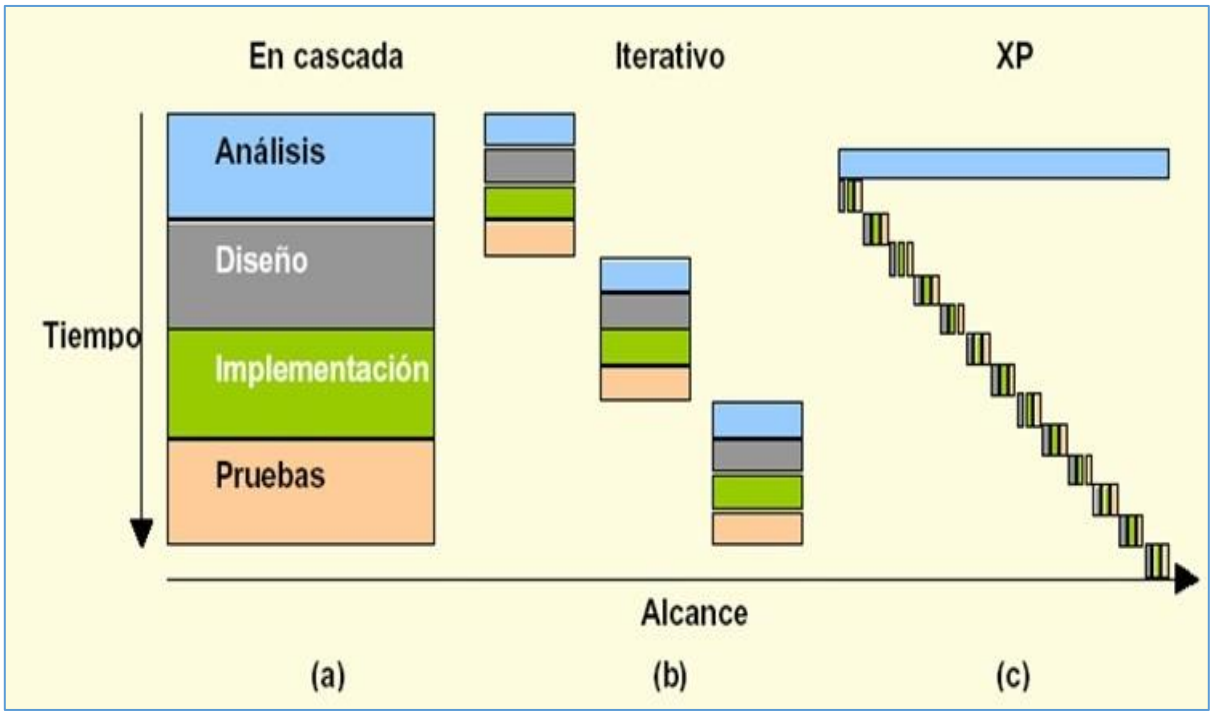


Fig. 9 Evolución de los largos ciclos de desarrollo en cascada (a) a ciclos iterativos más cortos (b) y a la mezcla que hace XP (c).

Los principios y prácticas son de sentido común pero llevadas al extremo, de ahí proviene su nombre. Kent Beck, el padre de XP, describe la filosofía de XP en “Extreme Programming Explained. Embrace Change” sin cubrir los detalles técnicos y de implantación de las prácticas. Posteriormente, otras publicaciones de experiencias se han encargado de dicha tarea (Canós, Letelier, & Penadés, Metodologías Ágiles en el desarrollo de software, 2003).

Los roles propuestos por Beck son:

Programador: El programador escribe las pruebas unitarias y produce el código del sistema. Debe existir una comunicación y coordinación adecuada entre los programadores y otros miembros del equipo (Letelier & Penadés, 2006).

Cliente: El cliente escribe las historias de usuario y las pruebas funcionales para validar su implementación. Además, asigna la prioridad a las historias de usuario y decide cuáles se implementan en cada iteración centrándose en aportar mayor valor al negocio. El cliente es sólo uno dentro del proyecto pero puede corresponder a un interlocutor que está representando a varias personas que se verán afectadas por el sistema.

Encargado de pruebas (Tester): El encargado de pruebas ayuda al cliente a escribir las pruebas funcionales. Ejecuta las pruebas regularmente, difunde los resultados en el equipo y es responsable de las herramientas de soporte para pruebas.

Encargado de seguimiento (Tracker): El encargado de seguimiento proporciona realimentación al equipo en el proceso XP. Su responsabilidad es verificar el grado de acierto entre las estimaciones realizadas y el tiempo real dedicado, comunicando los resultados para mejorar futuras estimaciones. También realiza el seguimiento del progreso de cada iteración y evalúa si los objetivos son alcanzables con las restricciones de tiempo y recursos presentes. Determina cuándo es necesario realizar algún cambio para lograr los objetivos de cada iteración.

Entrenador (Coach): Es responsable del proceso global. Es necesario que conozca a fondo el proceso XP para proveer guías a los miembros del equipo de forma que se apliquen las prácticas XP y se siga el proceso correctamente.

Consultor: Es un miembro externo del equipo con un conocimiento específico en algún tema necesario para el proyecto. Guía al equipo para resolver un problema específico.

Gestor (Big boss): Es el vínculo entre clientes y programadores, ayuda a que el equipo trabaje efectivamente creando las condiciones adecuadas. Su labor esencial es de coordinación.

El ciclo de desarrollo consiste en los siguientes pasos:

1. El cliente define el valor de negocio a implementar.
2. El programador estima el esfuerzo necesario para su implementación.
3. El cliente selecciona qué construir, de acuerdo con sus prioridades y las restricciones de tiempo.
4. El programador construye ese valor de negocio.
5. Vuelve al paso 1.

En todas las iteraciones de este ciclo tanto el cliente como el programador aprende. No se debe presionar al programador a realizar más trabajo que el estimado, ya que se perderá calidad en el software o no se cumplirán los plazos. De la misma forma el cliente tiene la obligación de manejar el ámbito de entrega del producto, para asegurar que el sistema tenga el mayor valor de negocio posible con cada iteración (Amaro Calderón & Valverde Rebaza, 2007).

El ciclo de vida ideal de XP consiste en seis fases: Exploración, Planificación de la entrega, Iteraciones, Producción, Mantenimiento y Muerte del proyecto (Canós, Letelier, & Penadés, Metodologías Ágiles en el desarrollo de software, 2003).

La principal suposición que se realiza en XP es la posibilidad de disminuir la mítica curva exponencial del costo del cambio a lo largo del proyecto, lo suficiente para que el diseño evolutivo funcione. XP apuesta por un crecimiento lento del costo del cambio y con un comportamiento asintótico. Esto se consigue gracias a las tecnologías disponibles para ayudar en el desarrollo de software y a la aplicación disciplinada de las siguientes prácticas:

El juego de la planificación: Es un espacio frecuente de comunicación entre el cliente y los programadores. El equipo técnico realiza una estimación del esfuerzo requerido para la implementación de las historias de usuario y los clientes deciden sobre el ámbito y tiempo de las entregas y de cada iteración. Esta práctica se puede ilustrar como un juego, donde existen dos tipos de jugadores: Cliente y Programador. El cliente establece la prioridad de cada historia de usuario, de acuerdo con el valor que aporta para el negocio. Los programadores estiman el esfuerzo asociado a cada historia de usuario. Se ordenan las historias de usuario según prioridad y esfuerzo, y se define el contenido de la entrega y/o iteración, apostando por enfrentar lo de más valor y riesgo cuanto antes. Este juego se realiza durante la planificación de la entrega, en la planificación de cada iteración y cuando sea necesario reconducir el proyecto.

Entregas pequeñas: La idea es producir rápidamente versiones del sistema que sean operativas, aunque obviamente no cuenten con toda la funcionalidad pretendida para el sistema, pero sí que constituyan un resultado de valor para el negocio. Una entrega no debería tardar más 3 meses.

Metáfora: En XP no se enfatiza la definición temprana de una arquitectura estable para el sistema. Dicha arquitectura se asume evolutiva y los posibles inconvenientes que se generarían por no contar con ella explícitamente en el comienzo del proyecto se solventan con la existencia de una metáfora. El sistema es definido mediante una metáfora o un conjunto de metáforas compartidas por el cliente y el equipo de desarrollo. Una metáfora es una historia compartida que describe cómo debería funcionar el sistema. Martin Fowler explica que la práctica de la metáfora consiste en formar un conjunto de nombres que actúen como vocabulario para hablar sobre el dominio del problema. Este conjunto de nombres ayuda a la nomenclatura de clases y métodos del sistema (Fowler, The New Methodology, 2001).

Diseño simple: Se debe diseñar la solución más simple que pueda funcionar y ser implementada en un momento determinado del proyecto. La complejidad innecesaria y

el código extra debe ser removido inmediatamente. Kent Beck dice que en cualquier momento el diseño adecuado para el software es aquel que: supera con éxito todas las pruebas, no tiene lógica duplicada, refleja claramente la intención de implementación de los programadores y tiene el menor número posible de clases y métodos.

Pruebas: La producción de código está dirigida por las pruebas unitarias. Las pruebas unitarias son establecidas antes de escribir el código y son ejecutadas constantemente ante cada modificación del sistema. Los clientes escriben las pruebas funcionales para cada historia de usuario que deba validarse. En este contexto de desarrollo evolutivo y de énfasis en pruebas constantes, la automatización para apoyar esta actividad es crucial.

Refactorización (Refactoring): La refactorización es una actividad constante de reestructuración del código con el objetivo de remover duplicación de código, mejorar su legibilidad, simplificarlo y hacerlo más flexible para facilitar los posteriores cambios. La refactorización mejora la estructura interna del código sin alterar su comportamiento externo (Poppendieck & Poppendieck, 2003). Robert Martin señala que el diseño del sistema de software es una cosa viviente. No se puede imponer todo en un inicio, pero en el transcurso del tiempo este diseño evoluciona conforme cambia la funcionalidad del sistema (Martin, 2002). Para mantener un diseño apropiado, es necesario realizar actividades de cuidado continuo durante el ciclo de vida del proyecto. De hecho, este cuidado continuo sobre el diseño es incluso más importante que el diseño inicial. Un concepto pobre al inicio puede ser corregido con esta actividad continua, pero sin ella, un buen diseño inicial se degradará.

Programación en parejas: Toda la producción de código debe realizarse con trabajo en parejas de programadores. Según Cockburn y Williams en un estudio realizado para identificar los costos y beneficios de la programación en parejas (Cockburn & Williams, 2000), las principales ventajas de introducir este estilo de programación son: muchos errores son detectados conforme son introducidos en el código (inspecciones de código

continuas), por consiguiente la tasa de errores del producto final es más baja, los diseños son mejores y el tamaño del código menor (continua discusión de ideas de los programadores), los problemas de programación se resuelven más rápido, se posibilita la transferencia de conocimientos de programación entre los miembros del equipo, varias personas entienden las diferentes partes sistema, los programadores conversan mejorando así el flujo de información y la dinámica del equipo, y finalmente, los programadores disfrutan más su trabajo. Dichos beneficios se consiguen después de varios meses de practicar la programación en parejas. En los estudios realizados por Cockburn y Williams este lapso de tiempo varía de 3 a 4 meses.

Propiedad colectiva del código: Cualquier programador puede cambiar cualquier parte del código en cualquier momento. Esta práctica motiva a todos a contribuir con nuevas ideas en todos los segmentos del sistema, evitando a la vez que algún programador sea imprescindible para realizar cambios en alguna porción de código.

Integración continua: Cada pieza de código es integrada en el sistema una vez que esté lista. Así, el sistema puede llegar a ser integrado y construido varias veces en un mismo día. Todas las pruebas son ejecutadas y tienen que ser aprobadas para que el nuevo código sea incorporado definitivamente. La integración continua a menudo reduce la fragmentación de los esfuerzos de los desarrolladores por falta de comunicación sobre lo que puede ser reutilizado o compartido. Martin Fowler afirma que el desarrollo de un proceso disciplinado y automatizado es esencial para un proyecto controlado, el equipo de desarrollo está más preparado para modificar el código cuando sea necesario, debido a la confianza en la identificación y corrección de los errores de integración (Fowler & Foemmel, Continuous Integration, 2001).

40 horas por semana: Se debe trabajar un máximo de 40 horas por semana. No se trabajan horas extras en dos semanas seguidas. Si esto ocurre, probablemente está ocurriendo un problema que debe corregirse. El trabajo extra desmotiva al equipo. Los proyectos que requieren trabajo extra para intentar cumplir con los plazos suelen al final

ser entregados con retraso. En lugar de esto se puede realizar el juego de la planificación para cambiar el ámbito del proyecto o la fecha de entrega.

Cliente in-situ: El cliente tiene que estar presente y disponible todo el tiempo para el equipo. Gran parte del éxito del proyecto XP se debe a que es el cliente quien conduce constantemente el trabajo hacia lo que aportará mayor valor de negocio y los programadores pueden resolver de manera inmediata cualquier duda asociada. La comunicación oral es más efectiva que la escrita, ya que esta última toma mucho tiempo en generarse y puede tener más riesgo de ser mal interpretada. Jeffries indica que se debe pagar un precio por perder la oportunidad de un cliente con alta disponibilidad. Algunas recomendaciones propuestas para dicha situación son las siguientes: intentar conseguir un representante que pueda estar siempre disponible y que actúe como interlocutor del cliente, contar con el cliente al menos en las reuniones de planificación, establecer visitas frecuentes de los programadores al cliente para validar el sistema, anticiparse a los problemas asociados estableciendo llamadas telefónicas frecuentes y conferencias, reforzando el compromiso de trabajo en equipo (Jeffries, Anderson, & Hendrickson, 2001).

Estándares de programación: XP enfatiza la comunicación de los programadores a través del código, con lo cual es indispensable que se sigan ciertos estándares de programación (del equipo, de la organización u otros estándares reconocidos para los lenguajes de programación utilizados). Los estándares de programación mantienen el código legible para los miembros del equipo, facilitando los cambios.

Comentarios respecto de las prácticas: El mayor beneficio de las prácticas se consigue con su aplicación conjunta y equilibrada puesto que se apoyan unas en otras. Esto se ilustra en la Figura 10, donde una línea entre dos prácticas significa que las dos prácticas se refuerzan entre sí.

La mayoría de las prácticas propuestas por XP no son novedosas, sino que en alguna forma ya habían sido propuestas en ingeniería del software e incluso demostrado su valor en la práctica. El mérito de XP es integrarlas de una forma efectiva y complementarlas con otras ideas desde la perspectiva del negocio, los valores humanos y el trabajo en equipo.

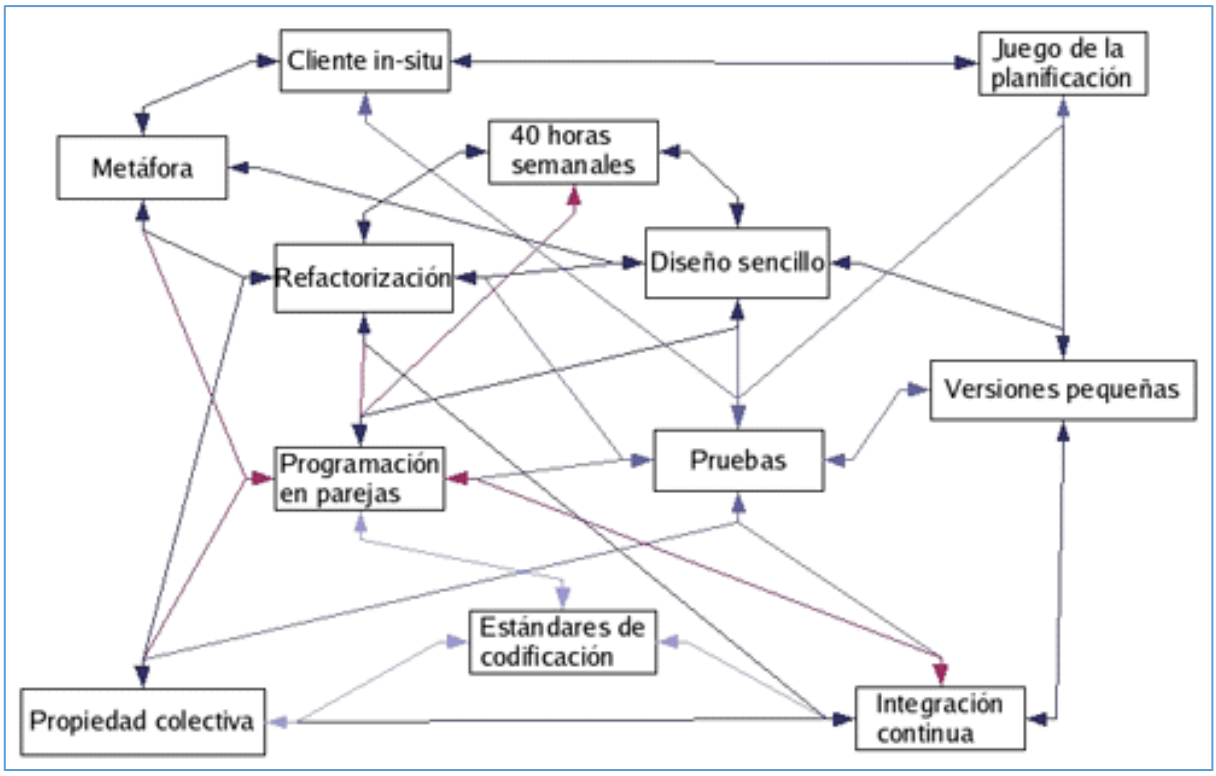


Fig. 10 Las Practicas se refuerzan entre sí.

Dentro de las otras metodologías Ágiles disponibles se encuentran:

SCRUM: Desarrollada por Ken Schwaber, Jeff Sutherland y Mike Beedle. Define un marco para la gestión de proyectos, que se ha utilizado con éxito durante los últimos 10 años. Está especialmente indicada para proyectos con un rápido cambio de requisitos. Sus principales características se pueden resumir en dos. El desarrollo de software se realiza mediante iteraciones, denominadas sprints, con una duración de 30 días. El resultado de cada sprint es un incremento ejecutable que se muestra al cliente. La

segunda característica importante son las reuniones a lo largo proyecto. Éstas son las verdaderas protagonistas, especialmente la reunión diaria de 15 minutos del equipo de desarrollo para coordinación e integración.

Crystal Methodologies: Se trata de un conjunto de metodologías para el desarrollo de software caracterizadas por estar centradas en las personas que componen el equipo (de ellas depende el éxito del proyecto) y la reducción al máximo del número de artefactos producidos. Han sido desarrolladas por Alistair Cockburn. El desarrollo de software se considera un juego cooperativo de invención y comunicación, limitado por los recursos a utilizar. El equipo de desarrollo es un factor clave, por lo que se deben invertir esfuerzos en mejorar sus habilidades y destrezas, así como tener políticas de trabajo en equipo definidas. Estas políticas dependerán del tamaño del equipo, estableciéndose una clasificación por colores, por ejemplo Crystal Clear (3 a 8 miembros) y Crystal Orange (25 a 50 miembros).

Dynamic Systems Development Method (DSDM): Define el marco para desarrollar un proceso de producción de software. Nace en 1994 con el objetivo el objetivo de crear una metodología RAD unificada. Sus principales características son: es un proceso iterativo e incremental y el equipo de desarrollo y el usuario trabajan juntos. Propone cinco fases: estudio viabilidad, estudio del negocio, modelado funcional, diseño y construcción, y finalmente implementación. Las tres últimas son iterativas, además de existir realimentación a todas las fases.

Adaptive Software Development (ASD): Su impulsor es Jim Highsmith. Sus principales características son: iterativo, orientado a los componentes software más que a las tareas y tolerante a los cambios. El ciclo de vida que propone tiene tres fases esenciales: especulación, colaboración y aprendizaje. En la primera de ellas se inicia el proyecto y se planifican las características del software; en la segunda desarrollan las características y finalmente en la tercera se revisa su calidad, y se entrega al cliente. La revisión de los componentes sirve para aprender de los errores y volver a iniciar el ciclo de desarrollo.

Feature-Driven Development (FDD): Define un proceso iterativo que consta de 5 pasos. Las iteraciones son cortas (hasta 2 semanas). Se centra en las fases de diseño e implementación del sistema partiendo de una lista de características que debe reunir el software. Sus impulsores son Jeff De Luca y Peter Coad.

Lean Development (LD): Definida por Bob Charette's a partir de su experiencia en proyectos con la industria japonesa del automóvil en los años 80 y utilizada en numerosos proyectos de telecomunicaciones en Europa. En LD, los cambios se consideran riesgos, pero si se manejan adecuadamente se pueden convertir en oportunidades que mejoren la productividad del cliente. Su principal característica es introducir un mecanismo para implementar dichos cambios.

DIFERENCIAS ENTRE METODOLOGÍAS ÁGILES Y TRADICIONALES.

Las principales diferencias entre metodología Ágil y Tradicional se resumen en la siguiente tabla.

Tabla 1.- Comparación entre metodologías Ágiles y Tradicionales.

Metodologías Ágiles	**Metodologías Tradicionales**
Se encuentran basadas en heurística provenientes de prácticas de producción de código.	Basadas en normas provenientes de estándares seguidos por el entorno de desarrollo.
Están especialmente preparadas para afrontar cambios durante el proyecto.	Presentan cierta resistencia al cambio.
Son impuestas internamente, por el Equipo.	Son impuestas externamente.
Cuenta con procesos menos controlados.	Cuentan con procesos mucho más controlados, con numerosas políticas y normas.
No existe contrato tradicional o al menos es bastante flexible.	Existe un contrato prefijado.
El cliente es parte del equipo de desarrollo.	El cliente interactúa con el equipo de desarrollo mediante reuniones.
Se trabaja en grupos pequeños (<10 integrantes) y trabajando en el mismo sitio.	Se trabaja en grupos grandes y posiblemente distribuidos.
Cuenta con pocos artefactos.	Cuenta con mayor cantidad de artefactos.
Existen pocos roles.	Existen muchos roles.
Se pone menos énfasis en la arquitectura del software.	La arquitectura del software es esencial y se expresa mediante modelos.

El uso de Metodologías Agiles, han demostrado un aumento en la probabilidad de éxito independientemente del tamaño del proyecto. Dentro de los resultados que alimentan al Informe CHAOS 2015, se observó un incremento del 11% al 39% de proyectos exitosos y a la vez una baja del 29% al 9% de fracasos durante el periodo 2011 y 2015. A primera vista, pareciera indicar que existen factores de éxito relacionados con la metodología empleada por los actores.

CAPITULO III: PRESENTACIÓN DE MÉTODOS.

LEVANTAMIENTO DE EXPERIENCIAS EN DEPARTAMENTOS DE GESTIÓN TI FRENTE A SOLICITUDES DE DESARROLLO DE SOLUCIONES INFORMÁTICAS.

Para realizar el levantamiento de información se elaboró un cuestionario (Anexo 1) dirigido a Ingenieros que se desempeñan en departamentos de Gestión de Tecnologías de la Información con el objeto de conocer la experiencia de los profesionales informáticos durante el desarrollo de proyectos de software en instituciones de salud.

CONSTRUCCIÓN DEL CUESTIONARIO.

El cuestionario fue elaborado pensando en dar respuesta a la pregunta de investigación, para esto se utilizaron los antecedentes existentes en la literatura en torno a la confección de proyectos de software.

Para darle lógica al levantamiento se formularon inicialmente preguntas orientadas al origen, objetivo y actores participantes de la solicitud y levantamiento del requerimiento informático. Dentro de las posibles áreas hospitalarias que requieren de una herramienta se encuentran:

Gerencia o Dirección Hospitalaria, es la autoridad máxima dentro de una Institución de Salud, responsable del funcionamiento global desde el punto de vista administrativo, clínico y financiero. Posee el conocimiento general de todos los procesos existentes y sus interacciones. Es quien autoriza la planeación y ejecución de un proyecto de Software el cual es presentado previamente por la Unidad de Informática Local. Rinde cuentas a instituciones que se encuentran por sobre su línea de mando como lo son: Servicio de Salud y Ministerio de Salud.

Usuarios, en este caso son los propios pacientes quienes manifiestan indirectamente la necesidad de agilizar las actividades realizadas y mejorar las

herramientas empleadas en los variados procesos existentes para así optimizar su atención. Estos requerimientos los centraliza a través de la Oficina de Informaciones, Reclamos y Sugerencias(OIRS), quienes presentan la información a los Sub departamentos involucrados para definir una estrategia de mejora.

Clientes Internos, en este grupo se encuentran los servicios administrativos, clínicos y de apoyo quienes manifiestan la necesidad de elaborar herramientas que permitan optimizar el producto generado ya sea a través de la reducción de hora hombre, o bien, asegurando y disponibilizando la información recabada en cada proceso desarrollado.

Unidad de Informática, su rol es fundamental debido a que su conocimiento en el área permite proponer estrategias eficientes, efectivas y eficaces, que contribuyan a mejorar los resultados obtenidos mediante el levantamiento de procesos hospitalarios los cuales se caracterizan por ser dinámicos y de gran magnitud.

Instituciones Externas, si bien los hospitales son empresas autónomas que generan productos y servicios, muchas veces deben rendir cuenta a Instituciones externas que se encuentran por sobre su nivel jerárquico en nuestro sistema de Salud Nacional. Por ejemplo, las instituciones públicas pertenecen a un Servicio de Salud y a su vez estos pertenecen al Ministerio de Salud. En otras situaciones particulares toda institución pública debe rendir cuenta a organismos fiscalizadores como lo son la Superintendencia de Salud y SEREMI.

En cuanto a los objetivos de los distintos proyectos de software desarrollados en salud, esto varían dependiendo del giro de negocio y evolución de la institución, dentro de estos destacan:

i. Crear reservorios de información.

ii. Centralizar y organizar la información.
iii. Asegurar la información.
iv. Facilitar el acceso a la información.
v. Mejorar la supervisión y control de diversos procesos tales como: procesos clínicos, procesos de apoyo, procesos administrativos y/o financieros.
vi. Permitir difusión orientada a usuarios, clientes internos o externos.

Cada una de las opciones mencionadas anteriormente deben desarrollarse pensando en satisfacer a la Institución Hospitalaria como un todo, la cual está conformada por Gerencia o Dirección, Clientes Internos, y a sus Usuarios quienes se benefician de la atención o servicio brindado. En casos excepcionales se considera satisfacer a Instituciones externas. Aunque la literatura existente no lo avale, la evidencia demuestra que cada vez que se desarrolla un proyecto de software la balanza tiende a favorecer a un área hospitalaria inespecífica generando a largo plazo inestabilidad e insatisfacción.

Cuando se conforma el equipo de trabajo, lo óptimo es que éste sea idóneo para así lograr el cumplimiento del objetivo del proyecto, dentro de los actores posibles se encuentran:

Representantes de la Unidad de Informática: Ingenieros en Informática, y Técnicos en informática.

Representantes del Proceso a Optimizar: Jefes, Supervisores o Coordinadores de Servicios Clínicos o de Apoyo.
Representantes de la Gerencia o Alta dirección: Director o Gerente, quienes aprueban el requerimiento.

Representantes de los responsables de ejecutar las tareas y acciones contenidas en cada proceso: Profesionales de la Salud, Personal Clínico, y/o Administrativo.

El diseño participativo en los procesos de desarrollo de software es un método que permite utilizar los conocimientos de los usuarios (Estayno & Panizzi, 2009). Como bien señala este autor, la participación de los usuarios o de quienes realizan el requerimiento, en el caso de los proyectos de salud, contribuye a la satisfacción de ellos mismos por considerarse constructores del producto que utilizarán para satisfacer sus necesidades. Este por sí solo no asegura el éxito del proyecto, ya que la participación directa de la dirección o la gerencia es fundamental para el logro de los objetivos. La gran incógnita actualmente es en qué etapa del proyecto deben estar presente para ser un real aporte.

Otro elemento considerado en la herramienta de levantamiento de experiencia es la importancia de la comunicación entre los distintos actores participantes. Entender los requerimientos de un problema es una de las tareas más difíciles que enfrenta el ingeniero de software. Cuando se piensa por primera vez, no parece tan difícil desarrollar un entendimiento claro de los requerimientos. Después de todo ¿acaso no sabe el cliente lo que se necesita?, ¿No deberían tener los usuarios finales una buena comprensión de las características y funciones que le darán un beneficio? Sorprendentemente, en muchas instancias la respuesta a estas preguntas es “no”. E incluso, si los clientes y los usuarios finales explican sus necesidades, éstas cambiarán mientras se desarrolla el proyecto (Pressman & Troya, 2010).

Esta brecha de comunicación es consecuencia de las diferencias en las especialidades del usuario y el analista. Así, los usuarios son expertos en el dominio del problema, pero, por lo general conocen poco de las diferentes técnicas de modelamiento; y los analistas, por su parte, son expertos en el lenguaje de modelamiento, pero en términos generales poco conocen del problema (Jaramillo & Isaza, 2012). Si bien, la literatura no señala en qué etapa del desarrollo de software el requerimiento es comprendido completamente resulta complejo para el experto TI el determinarlo ya que esto está influenciado por las personas, procesos y tecnologías cambiantes, negocio complejo difícil de definir o acotar, y la propia organización.

Para extraer los requerimientos se utilizan variadas técnicas. Una técnica, es una serie de pasos documentados que van de la mano con unas reglas para su uso y criterios para verificar su corrección (Aristizábal & Torres, 2009). Dentro de estas las más comúnmente utilizadas son:

Entrevista: consiste en realizar preguntas a los stakeholders sobre el sistema que utilizan y el sistema a desarrollar. Las entrevistas pueden ser cerradas o abiertas, dependiendo de si tienen que responder a un conjunto de preguntas predefinidas o no.

Modelo y Notación de Procesos de Negocio(BPMN): se define como un conjunto de actividades conectadas, las cuales colectivamente representan un objetivo de negocio o meta, normalmente dentro del contexto de una estructura organizacional definida por responsabilidades funcionales y relaciones (Quelopana, Vega, Gallardo, & Meneses, 2009). Esta notación estándar facilita el entendimiento de las colaboraciones y transacciones de negocio entre grupos de trabajo, departamentos, áreas y organizaciones. No se requiere que los participantes cuenten con un perfil técnico en sistemas, por esta razón BPMN constituye la solución a los problemas comunes de comunicación entre ingenieros de software y clientes, pues al posibilitar que todos hablen el mismo idioma, el éxito del proceso está garantizado (Maure & Castillo, 2011).

Cuestionario: Los cuestionarios son esencialmente una entrevista escrita en papel u otro medio que la persona debe responder (Pytel, y otros, 2011). Esta herramienta tiene la ventaja de ser distribuida a un gran número de personas que pueden estar ubicadas en lugares remotos.

Casos de Uso: son una descripción gráfica y/o escrita de un conjunto de posibles esenarios de acciones y eventos que describen una parte del comportamiento del sistema. Los diagramas obtenidos durante el levantamiento de requerimientos

representan la comunicación y el comportamiento de un sistema que interactúa con los usuarios u otros sistemas bajo diversas condiciones. Para que esta herramienta pueda ser empleada en su máxima expresión se sugiere contar previamente con el modelado del negocio, ya sea con BPMN u otra.

Tormenta de ideas (Brainstorming): Es una técnica de reuniones en grupo, cuyo objetivo es la generación de ideas en un ambiente libre de críticas o juicios (Zapata & Carmona, 2010). Ayuda a generar una gran variedad de vistas del problema y a formularlo de diferentes formas.

Prototipos: Son simulaciones del posible producto, que luego son utilizados por el usuario final, permitiéndonos conseguir una importante retroalimentación en cuanto a si el sistema diseñado con base a los requerimientos recolectados le permite al usuario realizar su trabajo de manera eficiente y efectiva (Arias, 2011).

Tal como señala Arias en su artículo, a través de los años, se ha podido constatar que los requerimientos o requisitos son la pieza fundamental en un proyecto de desarrollo de software, ya que marcan el punto de partida para actividades como la planeación, básicamente en lo que refiere a las estimaciones de tiempos y costos, así como la definición de recursos necesarios y la elaboración de cronogramas que será uno de los principales mecanismos de control con los que se contará durante la etapa de desarrollo (Arias, 2011).

El otro gran desafío del cuestionario es lograr identificar el comportamiento de características comunes de proyectos de software en instituciones de salud, tales como:

Metodología o Modelo empleado: considerando que las instituciones hospitalarias no cuentan con todos sus procesos levantados, es importante identificar que metodología de trabajo se utiliza, y cuál de estas, se adapta de mejor forma a la naturaleza del proyecto y al ambiento hospitalario. Dentro de las

alternativas que se les presentará a los expertos TI se encuentran: Modelo Ágil, Modelo Tradicional, y Modelo Extreme.

Cumplimiento del Tiempo: es interesante conocer si se logra cumplir con los plazos establecidos a cabalidad, si existieron retrasos, o incluso, si el proyecto logró anticiparse a los tiempos proyectados.

Presupuesto proyectado: igualmente importante es el logro del cumplimiento del presupuesto definido inicialmente, o si existió la necesidad de realizar un aumento de este durante el transcurso del proyecto. Debido a la existencia de proyectos pequeños dentro del ámbito hospitalario existe la posibilidad que el presupuesto no sea un ítem relevante para el experto TI a cargo del proyecto, no cuantificándolo.

Comportamiento del Alcance: Todo proyecto posee un alcance, si consideramos el hecho, de que muchas veces los requerimientos no son comprendidos o identificados completamente en su etapa inicial, es esperable que ocurran modificaciones en el alcance lo cual no asegura su cumplimiento a cabalidad, debido a esto, debido a esto es importante conocer si los resultados fueron satisfactorios para conocer si existe una relación directa entre estos dos elementos.

Nivel de Satisfacción obtenido: Luego de la puesta en marcha del software desarrollado es necesario conocer la percepción del requirente y los operadores de la nueva herramienta quienes manifestarán su nivel de satisfacción. Esta nueva métrica servirá de parámetro para considerar el nivel de éxito del proyecto desde una perspectiva distinta a la propuesta por Standish Group.

Clasificación del proyecto desde el punto de vista del Requirente: Esta clasificación permite conocer el nivel de importancia que el requirente le da al

proyecto informático desde el punto de vista de su utilidad y contribución al sistema considerándolos de la siguiente forma:

- Vital: Proyecto que debe ser realizado sin importar del costo o tiempo invertido ya que su utilidad es requerida con urgencia.
- Crecimiento: Proyecto destinado a contribuir al crecimiento de la Institución de Salud, y por ende, a la mejor dirección y control de sus procesos siendo altamente competitiva.
- Innovación: Proyecto destinado a potenciar nuevos nichos nunca antes desarrollados por Instituciones de Salud Públicas.
- Desempeño: Proyecto que busca mejorar administración y rendimiento de la Institución de Salud manteniéndose dentro del estándar nacional.

Por último, es necesario identificar las principales dificultades experimentadas por los expertos TI durante el desarrollo de proyectos de software en instituciones de salud. Para esto, se construyó una pregunta cerrada con 13 dificultades descritas en la literatura en donde los participantes podrán seleccionar más de una de ellas. Dentro de las dificultades más comunes se encuentran:

i. Requerimiento incompletos e insuficientes.
ii. Falta de involucramiento del Usuario.
iii. Falta de involucramiento de la gerencia o Dirección.
iv. Expectativas no realistas.
v. Requerimientos cambiantes.
vi. Plazos no realistas.
vii. Usuarios resistentes.
viii. Cambios en la tecnología elegida.
ix. Falta de recursos.
x. Riesgos no identificados.
xi. Pérdida de información.
xii. Omisión de información.
xiii. Dificultad para recepcionar y comprender lo solicitado.

METODOLOGÍA PARA VALIDACIÓN DE CUESTIONARIO.

En la validación del cuestionario participaron 12 expertos con el siguiente perfil:

- Profesión: Ingeniero.
- Antecedente de participación en Proyectos de desarrollo de software.

La participación fue voluntaria y su experiencia fue fundamental para realizar la Validación de Contenido y Comprensión.

Validación de Contenido.

La validez de contenido descansa generalmente en el juicio de expertos. Se define como el grado en que los ítems que componen el test representan el contenido que el test trata de evaluar. Por lo tanto, la validez de contenido se basa en la definición precisa del dominio y en el juicio sobre el grado de suficiencia con que ese dominio se evalúa.
Para realizar la validación de contenido se empleó el modelo de Lawshe (1975), para esto, se conformó un panel de expertos compuesto por profesionales vinculados a proyectos de software desarrollados en instituciones de salud. Cada participante evaluó todos los ítems del cuestionario emitiendo su opinión en tres categorías:

- **Esencial:** se considerará esencial para el cuestionario cuando la pregunta ayuda a dar cumplimiento al objetivo de la herramienta y es necesario que forme parte de esta.
- **Útil pero no esencial**: en este caso, la pregunta contribuye a complementar los resultados obtenido e indirectamente permite dar respuesta a la pregunta de investigación.
- **No necesario o innecesaria:** solo se categorizará el ítem como innecesario cuando su presencia o ausencia no favorece ni perjudica el objetivo planteado.

Una vez registradas las opiniones respecto a cada pregunta evaluada se determinó el número de coincidencia para cada una de ellas esperando alcanzar más del 50% de

acuerdos para la categoría esencial para considerar que el ítem evaluado tiene cierta validez de contenido (Lawshe, 1975). Para definir el consenso de los panelistas, se empleó la Razón de Validez de contenido (Content Validity Ratio, CVR) propuesta por Lawshe:

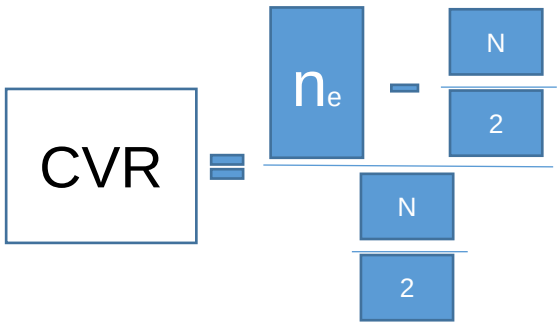


Fig. 11 Expresión de Razón de Validez de Contenido **(Lawshe, 1975)**.

Donde, **ne**: es el número de panelistas que tienen acuerdo en la categoría “Esencial”;y N : representa al número total de panelistas participantes.

Esta expresión es interpretada como una correlación, por tomar valores entre -1 y +1; de esta manera, si el CVR es negativo quiere decir que el acuerdo ocurrió en menos de la mitad de los jueces; si es igual a 0, CVR es nula por tener exactamente la mitad de acuerdos entre los panelistas; y por último, CVR es positiva cuando existe más de la mitad de acuerdos entre los expertos.

Una vez que cada ítem evaluado cuenta con su resultado de CVR y que a su vez fue aceptado por presentar un valor que nos indica que tiene cierta validez de contenido, se procede a calcular la media de todos ellos obteniéndose el Indice de Validez de Contenido (Content Validity Index, CVI), el cual se interpreta como la concordancia entre la habilidad, competencia y conocimiento solicitadas en un dominio específico y el desempeño solicitado en la prueba que trata de medir dicho dominio.

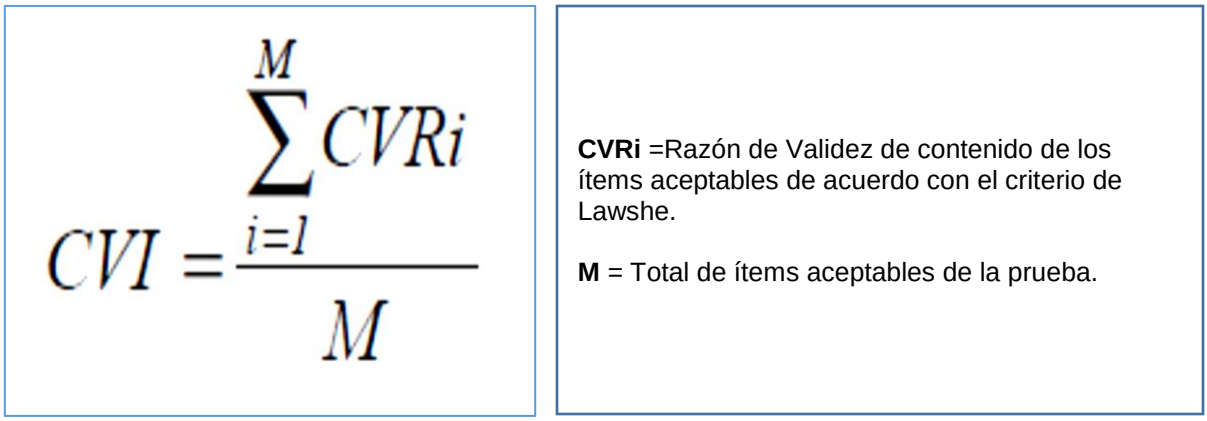


Fig. 12 Fórmula de Índice de Validez de Contenido(CVI) propuesta por Lawshe (1975).

Para definir el número mínimo de participantes se consideró la formula presentada por Tristán López (2008), debido a que el modelo normalizado expuesto en su artículo admite un desacuerdo en el caso de tres y cuatro panelistas y de dos desacuerdos cuando se tiene de cinco a siete jueces, lo cual es una condición bastante más razonable para emplear en una situación práctica.

Una vez obtenidos los valores para CVR', se puede determinar el Índice de Validez de Contenido (CVI) propuesto por Lawshe como promedio simple de los ítems aceptables (Tristán-López, 2008). En resumen, para que un instrumento o un banco de ítems pueda considerarse como aceptable, requiere contar por lo menos con un 58% de los ítems en condición satisfactoria del CRV'.

Validez de comprensión

Otro elemento a evaluar en conjunto con la validez de contenido, fue la validez de comprensión para evitar que una formulación errónea o confusa influyera en la respuesta aportada por el panel de expertos. Para esto se les solicitó que indicarán dudas o consultas complementarias en la casilla observaciones indicando además si la redacción les pareció clara, confusa o incomprensible.

Evaluación del tiempo de ejecución

En cada evaluación del cuestionario por parte de los expertos se controló el tiempo de ejecución con el objeto de estimar el tiempo promedio de desarrollo de las preguntas. Para esto se controló el tiempo de inicio y término del test.

Dado que se deseaba estimar un tiempo real de ejecución se solicitó que se contestaran las preguntas y luego de registrado el tiempo de término se realizara la evaluación de contenido y comprensión.

El tiempo promedio de ejecución de la versión final del cuestionario fue de 34 minutos.

EVALUACIÓN DEL CUESTIONARIO.

Se aplicó el cuestionario a los expertos y se realizó la correspondiente validación de contenido y de comprensión obteniéndose los siguientes resultados.

Tabla 2. Resultados de Validación de Contenido.

Pregunta evaluada	Porcentaje de cumplimiento categoría "Esencial"	Porcentaje de cumplimiento categoría "Útil pero no Esencial"	Porcentaje de cumplimiento categoría "Innecesaria"	Razón de Validez de Contenido (CVR)
1	83%	17%	0%	0,67
2	100%	0%	0%	1,00
3	83%	17%	0%	0,67
4	83%	17%	0%	0,67
5	100%	0%	0%	1,00
6	83%	17%	0%	0,67

7	100%	0%	0%	1,00
8	83%	17%	0%	0,67
9	100%	0%	0%	1,00
10	83%	17%	0%	0,67
11	100%	0%	0%	1,00
12	92%	8%	0%	0,83
13	100%	0%	0%	1,00
14	100%	0%	0%	1,00
15	100%	0%	0%	1,00
16	92%	8%	0%	0,83
17	92%	8%	0%	0,83

Luego de la evaluación, se aplica la fórmula de Lawshe para obtener el Índice de Validez de Contenido(CVI) que resultó ser 0,85, lo cual considera al instrumento como aceptable.

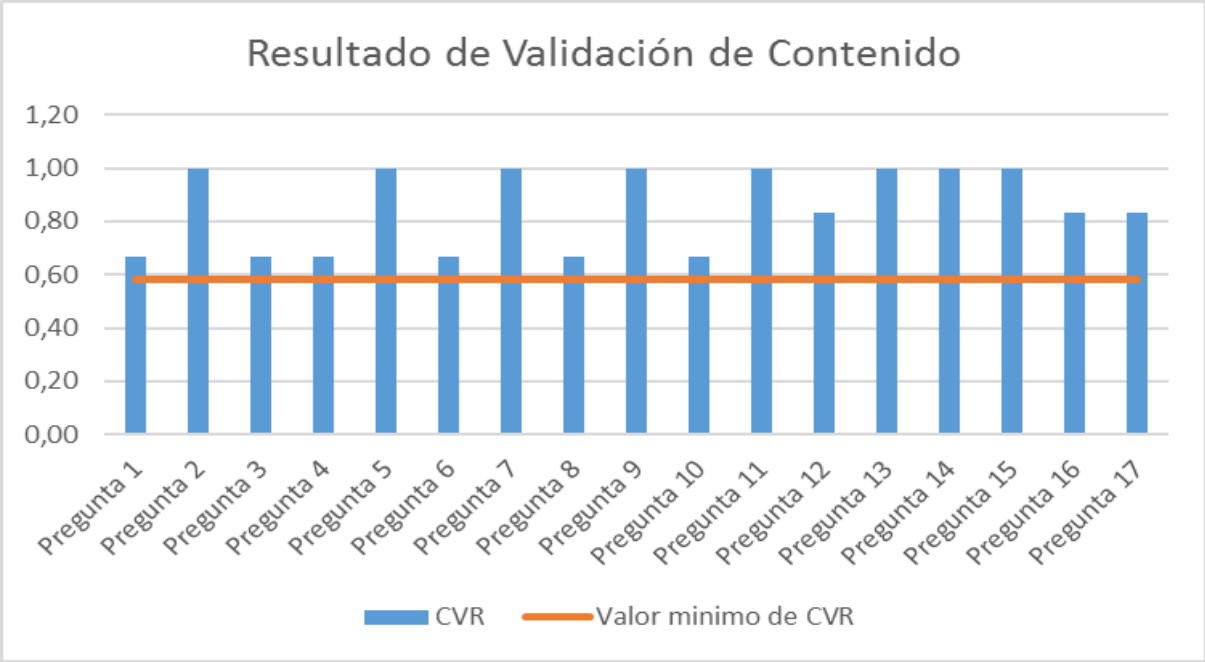


Fig. 13 Resultado de Validación de Contenido.

Resultados de Validación de Comprensión

La comprensión de las preguntas debe realizarse lo más claro posible debido a que podrían ser causal de una interpretación errónea de lo que se desea evaluar, en este caso, dado que las preguntas presentaban una redacción breve y precisa se obtuvieron los siguientes resultados.

Tabla 3. Resultados de Validación de Comprensión del Cuestionario.

Pregunta evaluada	Pregunta formulada de manera Clara	Pregunta formulada de manera Confusa	Pregunta formulada de manera Incomprensible
1	92%	8%	0%
2	100%	0%	0%
3	100%	0%	0%
4	100%	0%	0%
5	100%	0%	0%
6	92%	8%	0%
7	100%	0%	0%
8	100%	0%	0%
9	83%	17%	0%
10	100%	0%	0%
11	100%	0%	0%
12	100%	0%	0%
13	100%	0%	0%
14	100%	0%	0%
15	100%	0%	0%
16	92%	8%	0%
17	100%	0%	0%

DESCRIPCIÓN DE CUESTIONARIO POSTVALIDACIÓN.

Luego de realizar las modificaciones sugeridas durante la Validación el cuestionario quedó conformado de la siguiente manera:

Pregunta 1	**Alternativas propuestas**
Dentro de una Institución de Salud, ¿Desde dónde nace la necesidad de un software?	a) Unidad de Informática. b) Gerencia o Dirección. c) Usuarios. d) Clientes Internos. e) Institución Externa. f) Otro (señale desde donde).
Objetivo: Identificar quien es el dueño de la iniciativa de implementar una herramienta informática en proyectos de software desarrollados en instituciones de salud. **Característica:** El participante puede considerar más de una alternativa.	

Pregunta 2	**Alternativas propuestas**
¿Cuál o cuáles de los siguientes objetivos específicos se encontraban dentro de la solicitud del requirente en su proyecto de software realizado? (Puede seleccionar más de una opción).	a) Crear un reservorio de Información. b) Disminuir Carga Laboral. c) Centralizar y Ordenar la Información. d) Asegurar la Información. e) Facilitar el acceso a Información. f) Difusión de Buenas Prácticas. g) Mejorar Supervisión y Control. h) Otro (señale cual).

Objetivo:

Identificar objetivos específicos comunes dentro de los proyectos de software evaluados.

Característica:

El participante puede considerar más de una alternativa.

Pregunta 3	**Alternativas propuestas**
Indique con una “X”, ¿Cuál o cuáles fueron los actores participantes en sus proyectos de software realizado en su Institución de Salud?	a) Ingeniero Civil o Ejecución Informática. b) Técnico en Informática. c) Jefe, Supervisor, o Coordinador de la Unidad o Servicio Clínico. d) Director Médico, Gerencia o Subdirección de la Institución. e) Personal Clínico (Médicos, Odontólogos, Enfermeras, Matronas, Técnicos, Otros)

Objetivo:

Identificar los principales actores participantes en el proyecto de desarrollo de software en Instituciones de Salud.

Característica:

El participante puede considerar más de una alternativa.

Pregunta 4	**Alternativas propuestas**
El producto generado en el Proyectos de Software en el cual participó, era inicialmente para:	a) Satisfacer requerimientos de los Usuarios (Pacientes). b) Satisfacer requerimientos de Clientes Internos (Funcionarios). c) Satisfacer requerimientos de la Gerencia o Dirección. d) Satisfacer requerimientos de Organizaciones Externas. e) Otro (señale cual).
Objetivo: Identificar que actores o sistemas se verán beneficiados con la Herramienta informática a desarrollar. **Característica:** El participante puede considerar más de una alternativa.	

Pregunta 5	**Alternativas propuestas**
¿En que etapa del proyecto de desarrollo de software se centró la participación de Clientes Internos y/o Usuarios?.	a) Solicitud de una solución Informática. b) Análisis de Requerimientos. c) Diseño. d) Desarrollo. e) Pruebas. f) Implementación.

	g) Ninguna de las Anteriores
Objetivo: Identificar en qué etapa del desarrollo del proyecto de software en Instituciones de Salud se centró la participación del cliente interno y/o Usuarios. **Característica:** El participante puede considerar más de una alternativa.	

Pregunta 6	**Alternativas propuestas**
¿En que etapa del proyecto de desarrollo de software se centró la participación de la Gerencia o Dirección?	a) Solicitud de una solución Informática. b) Análisis de Requerimientos. c) Diseño. d) Desarrollo. e) Pruebas. f) Implementación. g) Ninguna de las Anteriores
Objetivo: Identificar en qué etapa del desarrollo del proyecto de software en Instituciones de Salud se centró la participación del Director o Gerente Hospitalario. **Característica:** El participante puede considerar más de una alternativa.	

Pregunta 7	**Alternativas propuestas**
En cuanto al levantamiento de requerimientos y a la comunicación con los actores participantes. Señale la	a) El Requerimiento fue claro y de fácil comprensión. b) El Requerimiento fue comprendido parcialmente y requirió más de una

experiencia vivida en el proyecto de software realizado.	intervención por parte del solicitante. c) El Requerimiento no fue comprendido y requirió de la interpretación de un tercero (cliente interno, usuario, solicitante). d) El Requerimiento no fue comprendido por el experto en TIC y no se contó con la interpretación de terceros
Objetivo: Identificar la dinámica de la comunicación entre el experto TI y el usuario o cliente durante el levantamiento de requerimientos del proyecto de software realizado. **Característica:** El participante puede considerar solo una alternativa.	

Pregunta 8	**Alternativas propuestas**
En cuanto al levantamiento de requerimientos y a la comunicación con los actores participantes. ¿En que etapa del proyecto el requerimiento fue comprendido completamente?	a) Solicitud de una solución Informática. b) Análisis de Requerimientos. c) Diseño. d) Desarrollo. e) Pruebas. f) Implementación. g) Ninguna de las Anteriores.

Objetivo: Identificar en qué etapa del desarrollo del proyecto de software desarrollado en Instituciones de Salud se logra comprender completamente el requerimiento. **Característica:** El participante puede considerar solo una alternativa.

Pregunta 9	**Alternativas propuestas**
¿Qué Técnica Emplea Su Unidad Para Realizar El Levantamiento De Requerimientos?	a) Entrevista. b) BPMN. c) Cuestionario. d) Casos de Uso. e) Tormenta de Ideas. f) Puntos de vista. g) Otra.
Objetivo: Identificar que técnicas son las más usadas para realizar el levantamiento de requerimientos en el desarrollo de proyectos de software en instituciones de Salud. **Característica:** El participante puede considerar más de una alternativa.	

Pregunta 10	**Alternativas propuestas**
Según la perspectiva del requirente, ¿Cómo clasifica él este proyecto?	a) Vital. b) Crecimiento. c) Innovador. d) Desempeño.
Objetivo: Conocer la visión del Requirente con respecto al Proyecto de Software solicitado por él. **Característica:** El participante puede considerar solo una alternativa.	

Pregunta 11	**Alternativas propuestas**
¿Cuál fue la metodología de desarrollo empleada en su proyecto?	a) Ágil b) Tradicional. c) Otra.
Objetivo: Identificar que metodología es la más utilizada por los expertos TI durante un desarrollo de software en Instituciones de Salud de atención cerrada. **Característica:** El participante puede considerar solo una alternativa.	

Pregunta 12	**Alternativas propuestas**
¿Cuál es el estado del proyecto de software evaluado a la fecha?	a) Finalizado. b) En curso. c) Inconcluso.
Objetivo: Identificar el estado del proyecto de software evaluado. **Característica:** El participante puede considerar solo una alternativa.	

Pregunta 13	**Alternativas propuestas**
En relación al tiempo definido en la planificación del proyecto de software evaluado, ¿Cómo es su cumplimiento a la fecha?	a) Cumple. b) Sufre retraso. c) Se anticipa.
Objetivo: Identificar que proporción de los proyectos de software evaluados presenta cumplimiento o incumplimiento del Tiempo definido en carta gantt. **Característica:** El participante puede considerar solo una alternativa.	

Pregunta 14	**Alternativas propuestas**
En relación al presupuesto definido en la planificación del proyecto de software evaluado, ¿Cómo es su cumplimiento a la fecha?	a) Cumple. b) Se excede. c) No se cuantifica.
Objetivo: Identificar que proporción de los proyectos de software evaluados presenta cumplimiento o incumplimiento del Presupuesto definido durante la Planificación. **Característica:** El participante puede considerar solo una alternativa.	

Pregunta 15	**Alternativas propuestas**
En relación al alcance definido en la planificación del proyecto de software evaluado, ¿Cómo es su cumplimiento a la fecha?	a) Cumple con el alcance definido. b) Sufre modificaciones y de esta forma es cumplido a cabalidad. c) Con o sin modificaciones y no es cumplido.
Objetivo: Identificar que proporción de los proyectos de software evaluados presenta cumplimiento o incumplimiento del Alcance definido durante la Planificación. **Característica:** El participante puede considerar solo una alternativa.	

Pregunta 16	**Alternativas propuestas**
En Relación a los Resultados Obtenidos, ¿Cómo se observa la satisfacción del requirente a la fecha?	a) El producto entregado logra satisfacer completamente al requirente. b) El producto entregado logra satisfacer parcialmente al requirente. c) El producto entregado no logra satisfacer al requirente.
Objetivo: Identificar que proporción de los proyectos de software evaluados presenta cumplimiento o incumplimiento del Alcance definido durante la Planificación. **Característica:** El participante puede considerar solo una alternativa.	

Pregunta 17	**Alternativas propuestas**
¿Qué dificultades ha experimentado al participar en el proyecto de software evaluado?	a) Requerimientos incompletos e Insuficientes. b) Falta de Involucramiento del Usuario. c) Falta de Involucramiento de la Gerencia o Dirección. d) Expectativas no Realistas. Requerimientos Cambiantes. e) Plazos no realistas.

	f) Usuarios resistentes. g) Cambios en la Tecnología elegida. h) Falta de Recursos. Riesgos no identificados. i) Pérdida de Información. j) Omisión de Información. k) Dificultad para recepcionar y comprender lo solicitado. l) Ninguna dificultad. m) Otras(Señale cual o cuales).
Objetivo: Identificar las principales dificultades experimentadas por los expertos TI durante un proyecto de desarrollo de software en instituciones de salud. **Característica:** El evaluador puede considerar más de una alternativa.	

APLICACIÓN DEL CUESTIONARIO.

El Instrumento Validado y ajustado queda compuesto finalmente por 17 preguntas cerradas. Cada participante podrá seleccionar más de una alternativa si lo consideran necesario en las preguntas que lo indiquen.

El número de cuestionarios aplicados refleja la participación en proyectos de desarrollo de Software para Instituciones de Salud realizados en el periodo comprendido entre el segundo trimestre del año 2015 y el primer trimestre del 2017.

La muestra quedó constituida por 99 cuestionarios respondidos por 33 expertos TI que se desempeñan en Unidades de Informática pertenecientes a Instituciones de Salud de

Atención Cerrada de la Región Metropolitana, la participación fue voluntaria y se realizó en forma presencial.

Previo al desarrollo del cuestionario los participantes leen y firman el compromiso de confidencialidad, con el cual se declara que no existe conflicto de interés, que la información obtenida es de carácter confidencial, y sus resultados solo serán utilizados con fines académicos y docentes sin hacer referencia directa a Ingenieros participantes o instituciones evaluadas.

El tiempo de ejecución de la Actividad fue fijado en 45 minutos. Este valor fue determinado de la suma del tiempo promedio de ejecución, obtenido durante la validación de la herramienta, más la holgura, definida como el tiempo probable de interrupción el cual resultó en promedio 11 minutos.

Esta Herramienta permitió elaborar un perfil que denota la tendencia general de los proyectos de software que se desarrollan en Salud, el cual describe:

i. Cuáles son las principales necesidades de nuestros usuarios para requerir de una herramienta informática.
ii. Quienes son los dueños de la iniciativa de elaborar un proyecto de software.
iii. Cómo los Interesados catalogan los proyectos de software en Salud desde el punto de vista Cualitativo.
iv. A qué área pertenecen los principales actores participantes en proyectos de software desarrollados en instituciones de Salud.
v. En qué etapas se centra la participación del Usuario o Cliente dentro del proyecto.
vi. En qué etapa y de qué manera es visualizada la participación de la Gerencia o la Dirección en el proyecto.
vii. Quienes son los realmente satisfechos con el proyecto desarrollado.

viii. Cuál es la realidad del experto TI al momento de realizar el levantamiento de requerimientos, en cuanto a la comunicación generada con los requirentes.
ix. En qué momento, comúnmente, se logra la completa comprensión de los requerimientos levantados al inicio del proyecto.
x. Cuáles son las técnicas más empleadas para realizar el levantamiento de requerimientos que alimentan a los proyectos desarrollados.

Con la información aportada por la misma herramienta se construyó un perfil que expone la tendencia individual de los proyectos de software que se desarrollan en Salud, el cual describe:

i. Cuál es el modelo de desarrollo de software más utilizado por los expertos TI.
ii. Qué proporción de los proyectos descritos se encuentran en estado: en curso, finalizado o cancelado.
iii. Antecedentes con respecto al tiempo definido.
iv. Antecedentes con respecto al presupuesto definido.
v. Antecedentes con respecto al alcance definido.
vi. Antecedentes con respecto al grado de satisfacción de los resultados obtenidos.
vii. Antecedentes con respecto a las dificultades que han experimentado los expertos TI durante el desarrollo de proyectos de software en instituciones de Salud.

Por último, se trazó una línea de tiempo comparativa, pre y post Diciembre del 2016, para identificar la tendencia de los principales atributos de los proyectos de software desarrollado en Instituciones Públicas de atención cerrada.

CAPITULO IV: PRESENTACIÓN Y ANÁLISIS DE RESULTADOS.

La presentación y análisis de resultados se estructuró en dos perspectivas: por preguntas en forma individual, y asociadas.

PRESENTACIÓN Y ANÁLISIS DE RESULTADOS POR PREGUNTA:

1.1. Pregunta 1: DENTRO DE UNA INSTITUCIÓN DE SALUD, ¿DESDE DONDE NACE LA NECESIDAD DE DESARROLLAR UN SOFTWARE?

Los actores o sistemas que expresaron la necesidad de contar con una herramienta informática como parte de un plan de mejora se presentaron en las siguientes proporciones:

1.1.1. En un 67% del total de proyectos evaluados, la Gerencia o Dirección fue parte de la iniciativa de este.

1.1.2. En un 67% del total de proyectos evaluados, Usuarios, Clientes o Pacientes formaron parte de la iniciativa de este.

1.1.3. En un 48% los Clientes Internos, formaron parte de la iniciativa de desarrollar los proyectos evaluados.

1.1.4. En un 45% de los proyectos evaluados la Unidad de informática estuvo vinculada a la iniciativa de su desarrollo.

1.1.5. En un 18% de los casos fueron Instituciones Externas las que iniciaron la solicitud de desarrollo de un proyecto informático.

Estos resultados demuestran que en la actualidad la necesidad que impulsa el desarrollo de un software proviene de mayoritariamente de la Gerencia o Dirección y los Pacientes o Usuarios. Estos últimos manifiestan sus inquietudes a través de la Oficina de Orientación, Información, Sugerencias y Reclamos(OIRS)

señalando alguna debilidad, que posteriormente, gracias al análisis realizado por la Gerencia se traduce en un nuevo requerimiento informático.

Esta pregunta permitió, además, identificar las proporciones que representan necesidades locales y colectivas. Dentro del total de proyectos evaluados se observa que un 24% de ellos nacieron desde un área específica, mientras en el 76% restante, el requerimiento proviene simultáneamente desde un conjunto de áreas que requieren de la misma solución informática. Cabe destacar, que no siempre las necesidades locales implican un menor costo y tiempo que las necesidades colectivas.

Los Clientes Internos son quienes se desempeñan en la línea de mando bajo la Gerencia. Sus tareas están orientadas a actividades clínicas y administrativas, y frecuentemente, detectan vulnerabilidades y amenazas que permiten identificar factores de riesgos que pueden resolverse mediante la generación de un requerimiento informático. Cabe destacar que poseen conocimiento cabal de todos los procesos que se desarrollan dentro de la Institución de Salud. En el tercer lugar, encontramos a las Unidades de Informática, las cuales están constituidas por los expertos TI quienes poseen las competencias necesarias para materializar los requerimientos provenientes del área clínica.

Por último, las instituciones Externas, tales como Ministerio de Salud, Superintendencia de Salud, Secretaría Regional Ministerial, y Entidades Acreditadoras, generan requerimientos transversales a todas las instituciones de Salud existentes, pero con una frecuencia menor.

1.2. Pregunta 2: ¿CUÁL O CUÁLES DE LOS SIGUIENTES OBJETIVOS ESPECÍFICOS SE ENCONTRABAN DENTRO DE LA SOLICITUD DEL REQUIRENTE EN SU PROYECTO DE SOFTWARE REALIZADO?

Dentro de los objetivos específicos de los proyectos de Software evaluados se manifestaron en las siguientes proporciones:

1.2.1. Crear un Reservorio de Información, se manifestó en un 24% de los proyectos evaluados.
1.2.2. Disminuir Carga Laboral, se encontraba en un 27% de los proyectos evaluados.
1.2.3. Centralizar y ordenar la Información, se observó en un 67% del total de proyectos de software evaluados.
1.2.4. Asegurar la Información, se encontró en un 52% de los proyectos evaluados.
1.2.5. Facilitar el Acceso a la Información, se manifestó en un 64% de los proyectos de software evaluados.
1.2.6. Difundir buenas prácticas, se propuso en un 21% de los proyectos evaluados.
1.2.7. Mejorar Supervisión y control, se encontraba en un 55% de los proyectos evaluados.

Al realizar un Diagrama de Pareto, se evidencia que el 80% de las necesidades presentadas buscan garantizar: Centralización, Acceso, Aseguramiento, Supervisión y Control (Fig. 12).

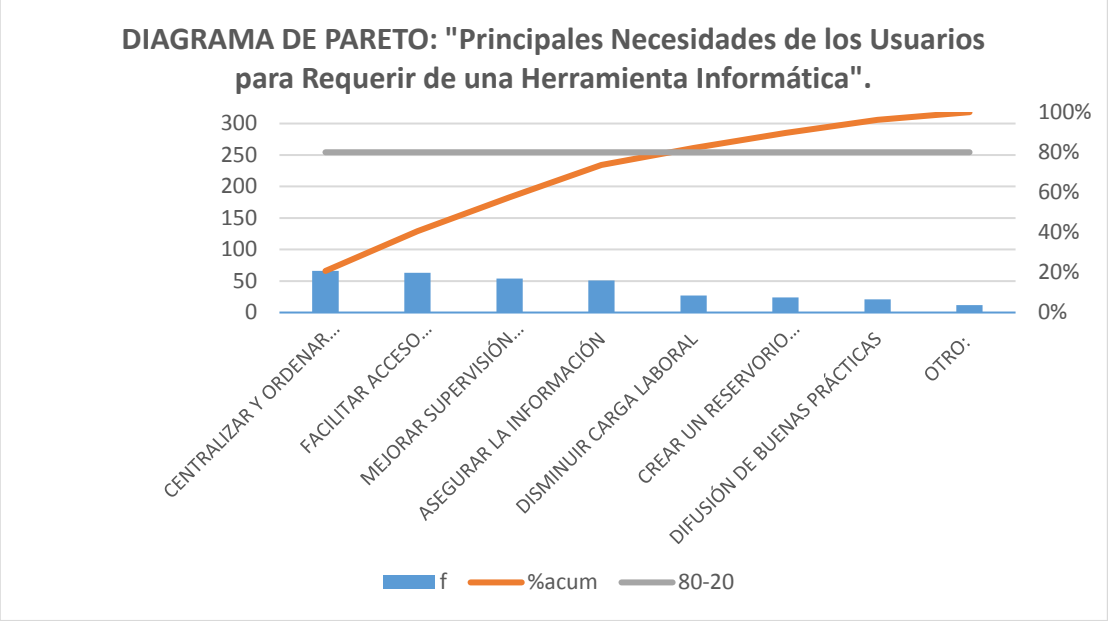


Fig. 14 Diagrama de Pareto que presenta la distribución 80 - 20 de los objetivos específicos de los proyectos evaluados.

1.3. Pregunta 3: ¿CUÁL O CUALES FUERON LOS ACTORES PARTICIPANTES EN EL PROYECTO DE SOFTWARE REALIZADO EN SU INSTITUCIÓN DE SALUD?

La participación de los actores en el total de proyectos de software evaluados está distribuida de la siguiente manera:

1.3.1. Ingeniero Civil o Ejecución Informática: 88%.
1.3.2. Técnico en Informática: 33%.
1.3.3. Jefe, Supervisor o Coordinador de Servicio Clínico: 79%.
1.3.4. Director Médico, Gerencia o Subdirección de la Institución: 36%.
1.3.5. Personal Clínico (Médicos, Odontólogos, Enfermeras, Matronas, Técnicos paramédicos, Otros): 76%.

Mayoritariamente los proyectos de software cuentan con la participación de Ingenieros, Jefes de Servicio y Personal Clínico. Esto permite mayor claridad en los requerimientos planteados debido a que se interrelacionan distintas visiones de los procesos existentes en una institución de Salud. Cabe destacar que la participación de Técnicos en Informática es menor debido a que sus tareas dentro de una Unidad de TI están avocada principalmente a mantención y soporte. En el caso de la Gerencia o Dirección, su rol se relaciona con la aprobación del requerimiento, el proyecto y su producto final.

Solo en un 9% de los proyectos analizados existieron equipos conformados únicamente por Ingenieros Civil o Ejecución en Informática. El 81% restante de proyectos conto con equipos multidisciplinarios.

1.4. Pregunta 4: EL PRODUCTO GENERADO EN EL PROYECTO DE SOFTWARE EN EL CUAL PARTICIPÓ, ERA INICIALMENTE PARA:

Los proyectos de Software no pertenecen a quienes los crean sino a quienes los requieren. Los siguientes resultados señalan a quienes buscaba satisfacer exclusivamente el producto final en los proyectos analizados:

1.4.1. Usuarios o Pacientes: 15%.
1.4.2. Clientes Internos o Funcionarios: 33%.
1.4.3. Gerencia o Dirección: 15%.
1.4.4. Organizaciones Externas: 0%
1.4.5. Otros: 3%.

El 33% faltante, se centra en la búsqueda de Satisfacción Institucional, lo que se traduce en el beneficio directo o indirecto de Pacientes, Funcionarios y gerencia en conjunto.

1.5. Preguntas 5 y 6: ¿EN QUÉ ETAPA DEL PROYECTO DE DESARROLLO DE SOFTWARE SE CENTRÓ LA PARTICIPACIÓN DE CLIENTES INTERNOS Y LA GERENCIA?

Estos dos actores están relacionados directamente con los procesos hospitalarios y con estas preguntas se busca conocer en que etapas se centró su participación tal como se expone en la siguiente tabla:

Tabla 4. Porcentaje de Proyectos que presentaron participación de Clientes Internos y Dirección en las distintas etapas del Desarrollo de Software en Instituciones de Salud.

ETAPA	Cliente Interno	Gerencia
Solicitud de una solución informática.	**64%**	**55%**
Análisis de Requerimientos.	**70%**	**33%**
Diseño.	**15%**	**12%**
Desarrollo.	**9%**	**3%**
Pruebas.	**42%**	**12%**
Implementación.	**42%**	**27%**
Ninguna de las Anteriores.	**3%**	**15%**

Lo expuesto en la Tabla 1, demuestra que las participaciones de los actores vinculados a los procesos hospitalarios participan frecuentemente de la Solicitud y Análisis de los requerimientos en una primera instancia, y que posteriormente, en menor proporción existe un porcentaje de proyectos que consideró su participación en las etapas de Pruebas e Implementación, dejando el Diseño y Desarrollo exclusivamente a los expertos TI. El Cliente interno está presente en mayor cantidad de proyectos, pero esto se debe a que la Gerencia o Dirección, al

ser la Autoridad máxima de la institución, tiene asignada labores de gestión que le impiden una mayor participación.

Junto con esto se determinó el porcentaje promedio de participación del Cliente Interno y la Dirección en cada Proyecto evaluado. (Tabla 3)

Tabla 5. Rango Mínimo, Rango Máximo, Promedio y Moda de los Porcentajes de Participación de Clientes Internos y Dirección por cada proyecto.

ETAPA	Cliente Interno	Gerencia
Rango Mínimo de Participación en el Proyecto.	**0%**	**0%**
Rango Máximo de Participación en el Proyecto.	**83%**	**67%**
% Promedio de Participación en el Proyecto.	**40%**	**24%**
% de Participación más frecuente en el Proyecto.	**50%**	**17%**

1.6. Pregunta 7: EN CUANTO AL LEVANTAMIENTO DE REQUERIMIENTOS Y A LA COMUNICACIÓN CON LOS ACTORES PARTICIPANTES. SEÑALE LA EXPERIENCIA VIVIDA EN EL PROYECTO DE SOFTWARE REALIZADO.

Debido a que el levantamiento de requerimientos es una etapa clave dentro del desarrollo de un proyecto de software, se consultó a los expertos cual ha sido su experiencia al respecto obteniendo los siguientes resultados:

1.6.1. El Requerimiento fue claro y de fácil comprensión: 24%.

1.6.2. El Requerimiento fue comprendido parcialmente y requirió más de una intervención por parte del solicitante: 58%.

1.6.3. El Requerimiento no fue comprendido y requirió de la interpretación de un tercero (cliente interno, usuario, solicitante): 12%.

1.6.4. El Requerimiento no fue comprendido por el experto en TIC y no se contó con la interpretación de terceros:6%.

Estos resultados demuestran el nivel de complejidad que reviste esta actividad, y especialmente en el caso de los requerimientos presentados por instituciones de salud su dificultad radica en la variabilidad de los procesos existentes, muchos de estos desconocidos o de difícil comprensión para el Equipo experto de la Unidad de Informática. Esto causa iteraciones inclusive desde etapas avanzadas del proyecto generando retraso y modificación de Alcance y Presupuesto.

1.7. Pregunta 8: EN CUANTO AL REQUERIMIENTO Y A LA COMUNICACIÓN CON LOS ACTORES PARTICIPANTES. ¿EN QUE ETAPA DEL PROYECTO EL REQUERIMIENTO FUE COMPRENDIDO COMPLETAMENTE?

Si bien es cierto, el escenario ideal para el Levantamiento de Requerimientos es comprenderlos en su totalidad durante el Análisis de estos aún existen proyectos de software realizados en Salud que no logran sortear sus dificultades tal como se presenta en el siguiente gráfico en donde existe un 46% de Proyectos Evaluados que obtienen la información requerida durante las etapas de Diseño, Desarrollo, Pruebas, e Implementación.

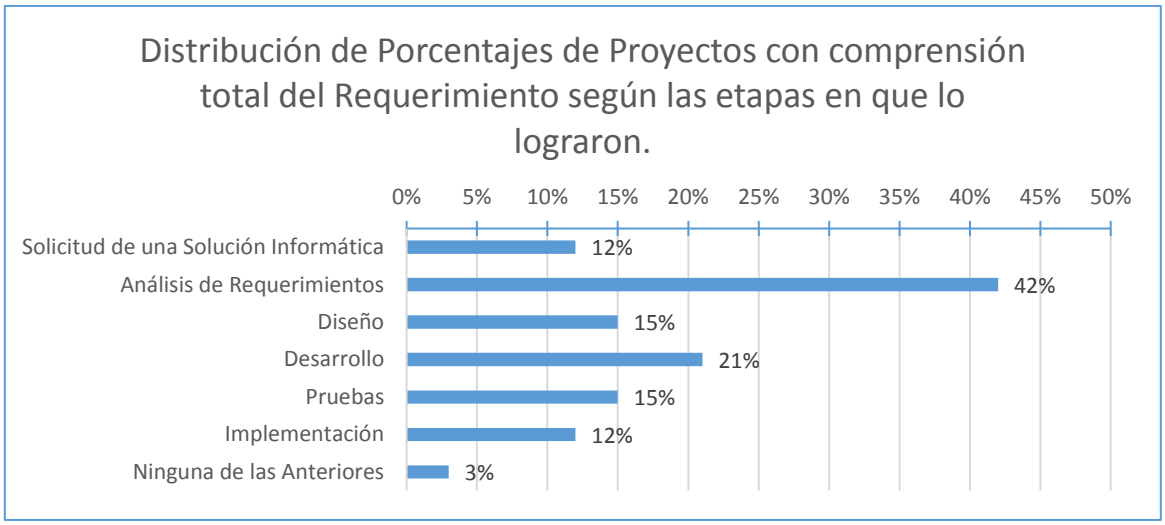


Fig. 15 Distribución de porcentajes de Proyectos con comprensión total del requerimiento según su etapa.

1.8. Pregunta 9: ¿QUÉ TÉCNICA EMPLEA SU UNIDAD PARA REALIZAR EL LEVANTAMIENTO DE REQUERIMIENTOS?

Otros elementos a considerar en este diagnóstico, son las Técnicas empleadas para el Levantamiento de Requerimientos por los distintos equipos de Expertos durante el desarrollo de un Proyecto de Software en Instituciones de Salud. Muchos de los proyectos evaluados utilizaron combinaciones de Técnicas las cuales se agruparon en el siguiente diagrama de Pareto:

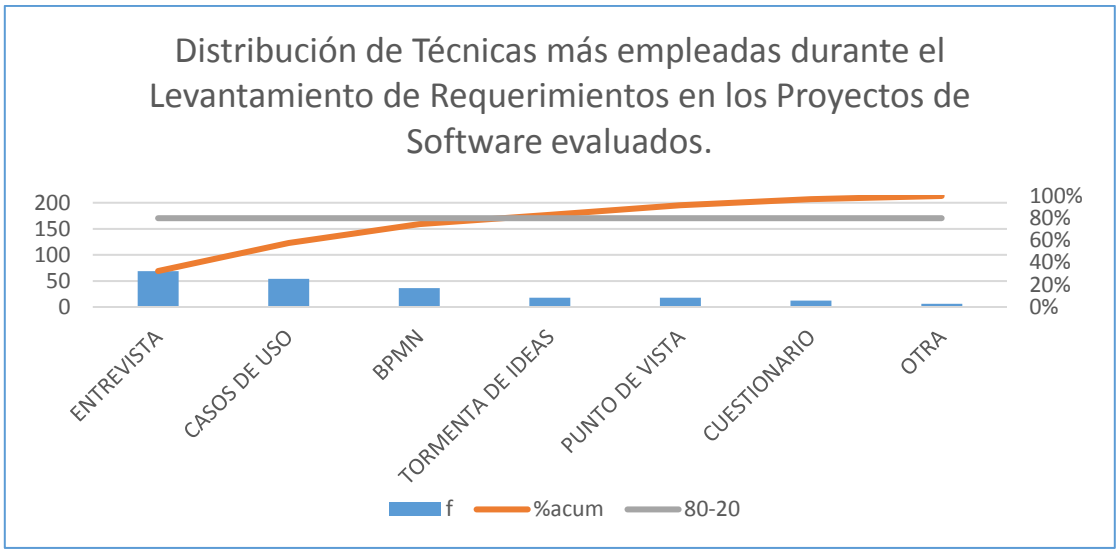


Fig. 16 Distribución de Técnicas más empleadas durante el levantamiento de requerimientos en los proyectos de software evaluados.

Dentro de las Técnicas más empleadas se encuentra la entrevista, Casos de Uso y BPMN (Business Process Model and Notation) las cuales representan el 75% de las Técnicas empleadas en los distintos proyectos evaluados.

1.9. Pregunta 10: SEGÚN LA PERSPECTIVA DEL REQUIRENTE, ¿CÓMO CLASIFICA ÉL ESTE PROYECTO?

Al evaluar la perspectiva del Requirente se observa que el nivel de importancia que reviste el proyecto para la Institución de Salud está en estricta relación con sus atributos, los cuales los hacen imprescindibles o postergables. De los 99 proyectos evaluados, el 58% de estos fueron considerados vitales, requiriendo el producto final a cualquier costo y al menor tiempo posible; mientras, por otra parte, ninguno contaba con características de un proyecto de desempeño, evidenciando que por el momento son los menos atractivos.

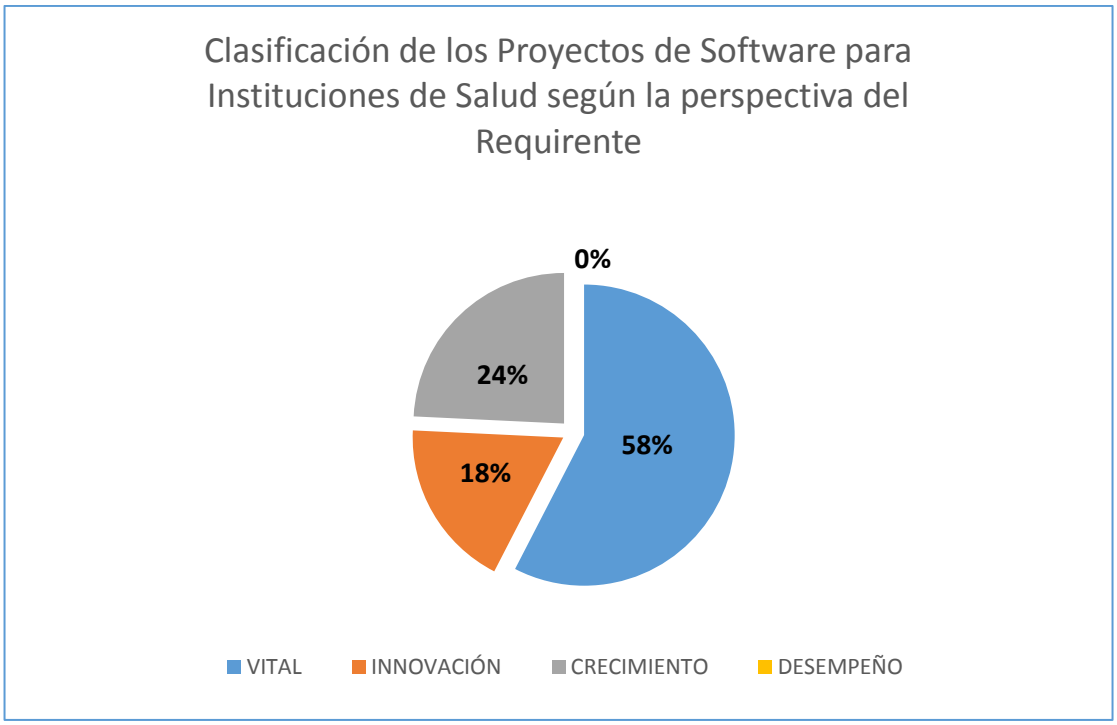


Fig. 17 Clasificación de Proyectos de software para Instituciones de Salud según la perspectiva del Requirente.

1.10. Pregunta 11: ¿CUÁL FUE LA METODOLOGÍA DE DESARROLLO EMPLEADA EN SU PROYECTO DE SOFTWARE?

Dentro de las Metodologías empleadas se observa un equilibrio con una leve ventaja que favorece a las metodologías Ágiles por sobre las Tradicionales. Dentro del 11% que representa a las Otras metodologías empleadas destaca el uso de metodologías propias o una combinación de las señaladas anteriormente.

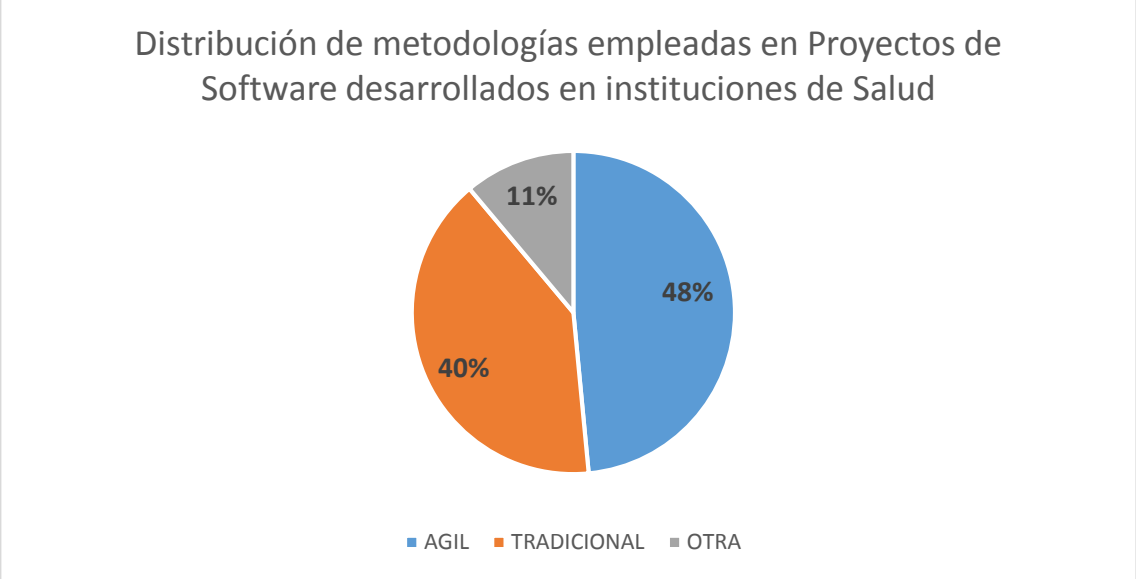


Fig. 18 Distribución de metodologías empleadas en Proyectos de Software desarrollados en Instituciones de Salud.

1.11. Pregunta 12: ¿CUÁL ES EL ESTADO DEL PROYECTO DE SOFTWARE EVALUADO A LA FECHA?

En cuanto al estado del proyecto evaluado a la fecha de la aplicación del cuestionario nos encontramos con los siguientes resultados:

a) Proyectos Finalizados: 59%.
b) Proyectos en curso: 39%.
c) Proyectos cancelados:2%.

Este antecedente nos permite agrupar los proyectos evaluados en 2 grupos en donde los proyectos finalizados representan los proyectos realizados hasta diciembre del año 2016 y los proyectos en curso representan a los proyectos que serán finalizados durante el año 2017.

Por otra parte, es importante destacar que la cantidad de proyectos cancelados es despreciable en comparación a los finalizados y en curso demostrando un bajo

porcentaje de fracaso desde el punto de vista de su gestión, desarrollo, ejecución e implementación.

1.12. Pregunta 13: EN RELACIÓN AL TIEMPO DEFINIDO EN LA PLANIFICACIÓN DEL PROYECTO DE SOFTWARE EVALUADO, ¿CÓMO ES SU CUMPLIMIENTO A LA FECHA?

Es interesante observar que al cierre de la aplicación del cuestionario los proyectos se comportaron de la siguiente forma en relación al cumplimiento del tiempo planificado:

a) Un 29% de los proyectos cumplieron con el tiempo definido en la planificación.
b) Un 68% de los proyectos evaluados presenta retraso en comparación el tiempo definido en su planificación.
c) Y solo un 3% de los proyectos evaluados se anticipó al tiempo presupuestado en su planificación.

Aquellos proyectos que se encontraban en curso, consideraron el cumplimiento de su planificación a la fecha de aplicación del cuestionario.

1.13. Pregunta 14: EN RELACIÓN AL PRESUPUESTO DEFINIDO EN LA PLANIFICACIÓN DEL PROYECTO DE SOFTWARE EVALUADO, ¿CÓMO ES SU CUMPLIMIENTO A LA FECHA?

Con respecto al comportamiento del presupuesto en los proyectos de software evaluados los expertos TI refieren lo siguiente:

a) Un 73% del total de proyectos evaluados cumple con el presupuesto definido en la planificación del proyecto.
b) Un 15% de los proyectos excede el presupuesto definido.

c) Y un 12% de los proyectos no cuantifica el presupuesto debido a que es realizado con recursos locales disponibles.

El hecho que una Unidad no considere la cuantificación de los recursos empleados en un proyecto impide que este sea evaluado empleando la triada de gestión del proyecto la cual considera tiempo, alcance y presupuesto; de esta manera, los esfuerzos se centran en conseguir un producto necesario e indispensable para la Institución de Salud.

1.14. Pregunta 15: EN RELACIÓN AL ALCANCE DEFINIDO EN LA PLANIFICACIÓN DEL PROYECTO DE SOFTWARE EVALUADO, ¿CÓMO ES SU CUMPLIMIENTO A LA FECHA?

El alcance del proyecto se encuentra presente en todos los proyectos de desarrollo de software en instituciones de Salud, su comportameinto es variable dependiendo de la naturaleza del proyecto. A nivel general se obtuvieron los siguientes resultados:

a) Un 34% de los proyectos evaluados cumple con el alcance definido en el proyecto.
b) Un 52% de los proyectos evaluados cuentan con un alcance que sufre modificaciones y de esta forma es cumplido a cabalidad.
c) Un 14% de los proyectos presentaron alcances que aun cuando fueron modificados no fueron cumplidos completamente.

De esta pregunta se puede rescatar que el cumplimiento del alcance es una preocupación constante en los expertos debido a que cuando visualizaban su no cumplimiento evaluaron la necesidad de realizar cambios, independiente de su resultado final. Este fenómeno se pesquisó en 65 de los 99 proyectos evaluados.

1.15. Pregunta 16: EN RELACIÓN A LOS RESULTADOS OBTENIDOS, ¿CÓMO SE OBSERVA LA SATISFACCIÓN DEL REQUIRENTE A LA FECHA?

Los resultados obtenidos sumados a la satisfacción del Requirente demuestran el éxito del proyecto desde un punto de vista funcional, aun cuando durante la gestión de este, no se cumpla a cabalidad con alcance, tiempo o presupuesto.
Del total de proyectos evaluados solo un 3% no logra satisfacer al requirente versus un 51% de satisfacción completa y un 46% con satisfacción parcial del producto entregado.

1.16. Pregunta 17: ¿QUÉ DIFICULTADES HA EXPERIMENTADO AL PARTICIPAR EN EL PROYECTO DE SOFTWARE EVALUADO?

Así como cada proyecto posee su propia identidad, también tienen sus propias dificultades. Muchas de estas, son las mismas para más de un grupo de expertos TI, pero lo que realmente varía, es la combinación de estas dificultades, su nivel de visibilidad, el momento en el cual se presentan, su magnitud e impacto.

El siguiente diagrama de Pareto representa las dificultades más frecuentes dentro de los proyectos de software evaluados.

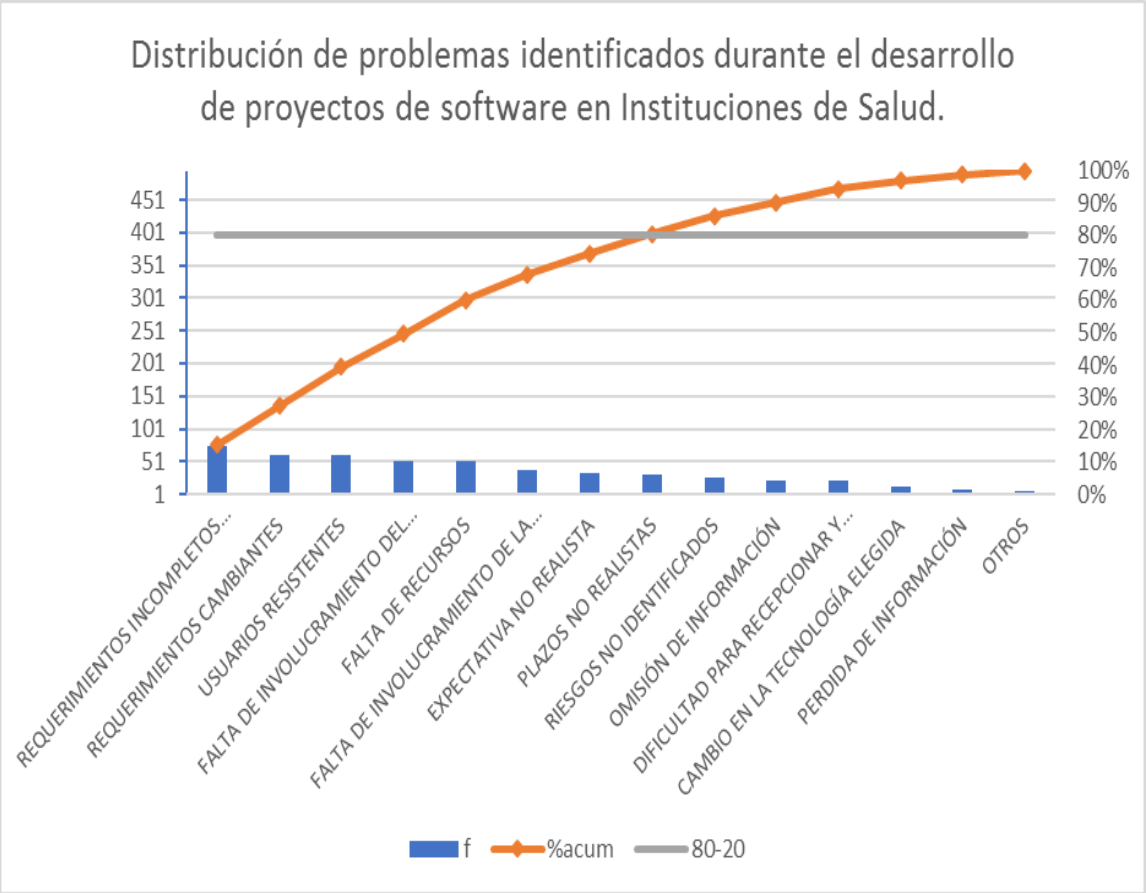


Fig. 19 Distribución de problemas identificados durante el desarrollo de proyectos de software en instituciones de Salud.

Dentro del 80% de los problemas contabilizados se encuentran los siguientes:

a) Requerimientos incompletos e insuficientes.
b) Requerimientos cambiantes.
c) Usuarios resistentes.
d) Falta de involucramiento del Usuario.
e) Falta de recursos.
f) Falta de involucramiento de la Gerencia o Dirección.
g) Expectativas no realistas.

Estos representan a los problemas que se presentan con mayor frecuencia en los proyectos evaluados. Dentro del 20% de problemas restantes se encuentran:

a) Plazos no realistas.
b) Riesgos no identificados.
c) Omisión de información.
d) Dificultad para recepcionar y comprender lo solicitado.
e) Cambios en la tecnología elegida.
f) Perdida de información.
g) Otros.

PRESENTACIÓN Y ANÁLISIS DE RESULTADOS POR ASOCIACIÓN DE PREGUNTAS:

1.1. Evolución del desarrollo de proyectos de software en instituciones de salud en los periodos 2016 y 2017.

Para obtener estas conclusiones se estructuraron 2 grupos: Proyectos finalizados al 30 de diciembre del 2016 versus proyectos finalizados o en curso durante el año 2017.

En cuanto a conocer desde donde nace la necesidad de realizar un proyecto de software en Instituciones de Salud, se observa que esta se mantiene liderada por la Gerencia y los Usuarios, pero se observa que durante el 2017 los Clientes Externos toman un rol protagónico convirtiéndose en Requirentes y desplazando levemente a las Unidades de informática tal como se expone en los siguientes gráficos comparativos.

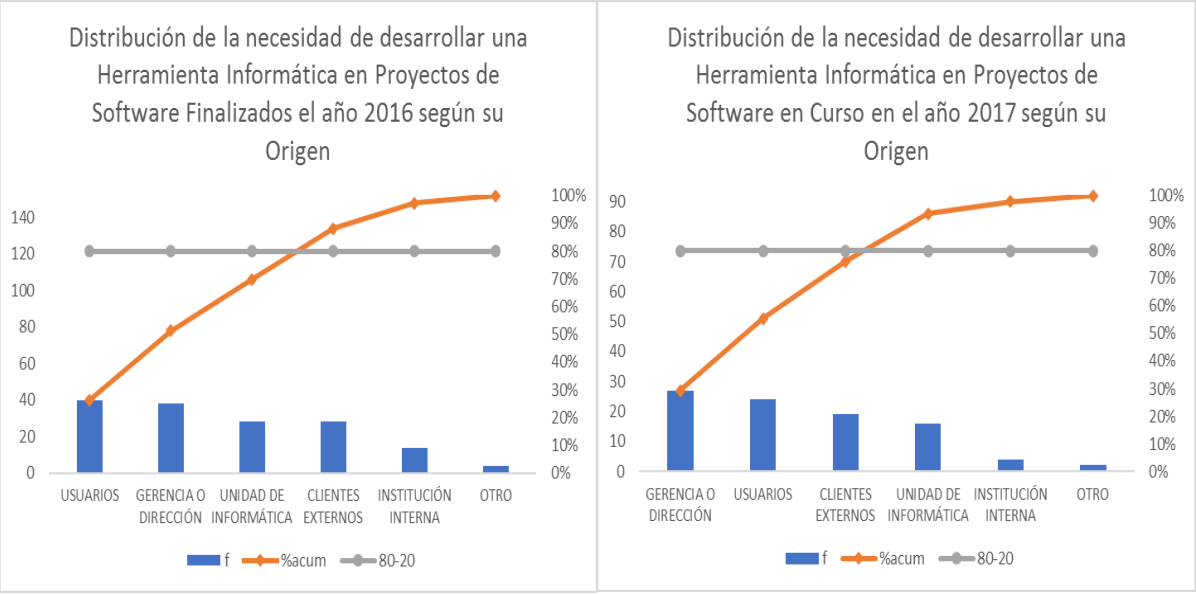


Fig. 20 Comparación gráfica de las distribuciones de necesidades de desarrollar una herramienta informática en proyectos de software finalizados en los años 2016 y 2017.

Por otra parte, sus objetivos mantienen su tendencia sin demostrar cambios hasta el momento en donde principalmente los proyectos desarrollados, buscan a nivel de Instituciones de Salud, centralizar y ordenar, facilitar el acceso y asegurar la información, y mejorar la supervisión.

Los actores participantes siguen siendo los mismos, esto se debe a que en el último periodo no se han realizado cambios organizacionales manteniendo el rol del técnico informático vinculado al área de soporte y el del director o gerente a la gestión global hospitalaria.

Al comparar el porcentaje de participación del Cliente Interno y Dirección, esta se mantiene en los periodos 2016 y 2017.

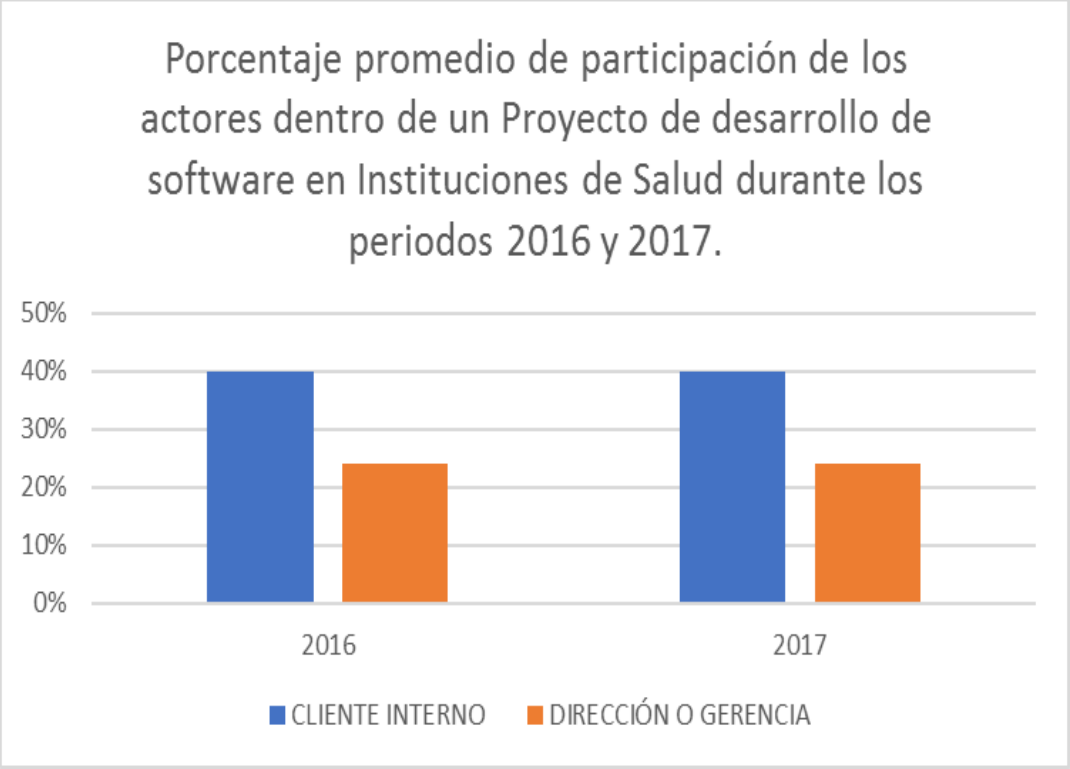


Fig. 21 Porcentaje promedio de participación de los actores dentro de un proyecto de desarrollo de software en instituciones de salud durante los periodos 2016 y 2017.

Y dentro de las etapas del desarrollo de software se observa que la participación de clientes internos ya no se centra solo en las etapas iniciales, sino que se están haciendo presentes en la implementación y pruebas. En el caso de las etapas de diseño y desarrollo se observa que estas aun pertenecen mayoritariamente a los expertos TI, con escasa participación de algún representante del sector salud.

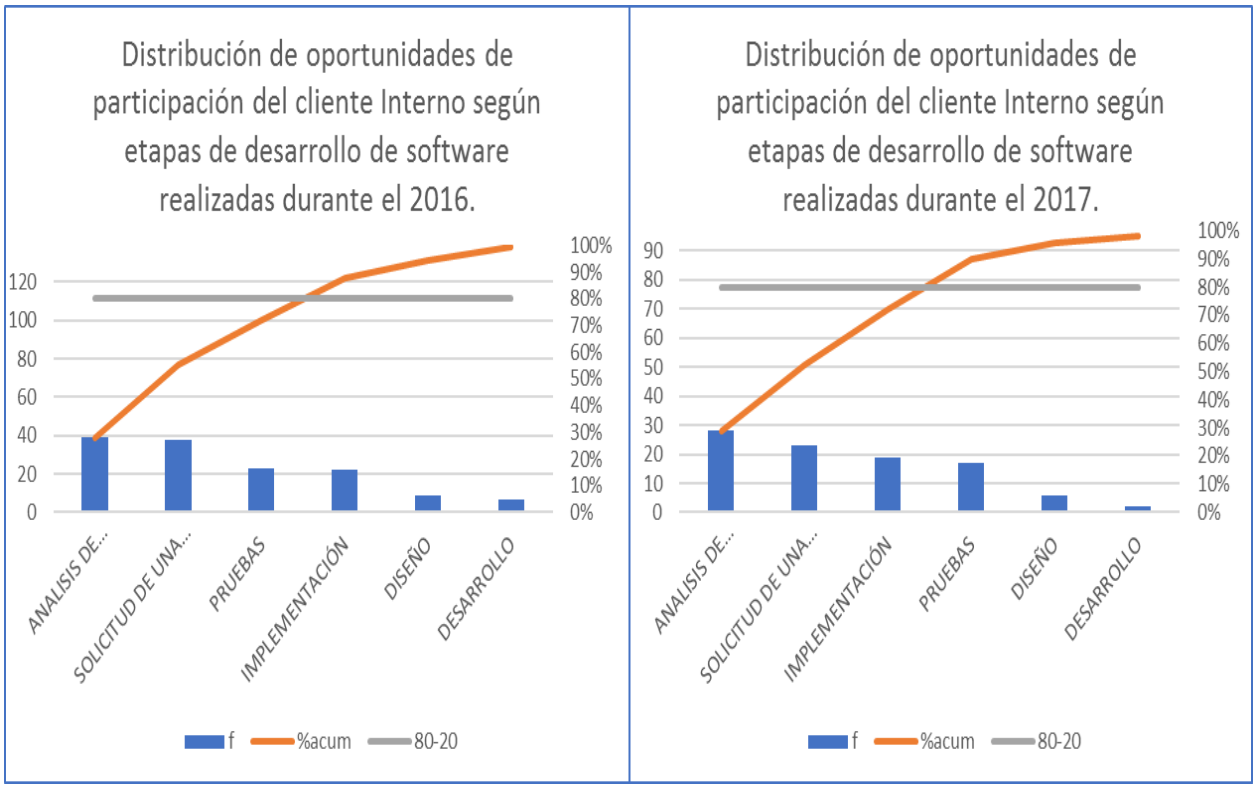


Fig. 22 Comparación gráfica de la distribución de oportunidades de participación del Cliente Interno según las etapas de desarrollo de software realizadas durante el periodo 2016 y 2017.

En cuanto a la participación de la gerencia o la dirección en los proyectos de software se mantiene la tendencia de participación entre los años 2016 y 2017, en donde su rol es más bien fiscalizador, consistente principalmente en la aprobación del proyecto, control de estados de avance y recepción una vez finalizado.

EL levantamiento de Requerimientos se considera como una de las etapas vitales para desarrollar un proyecto que represente lo más fielmente posible lo solicitado por el requirente. Por el momento no se observan cambios entre los proyectos realizados *ex ante* y *ex post* Diciembre del 2016.

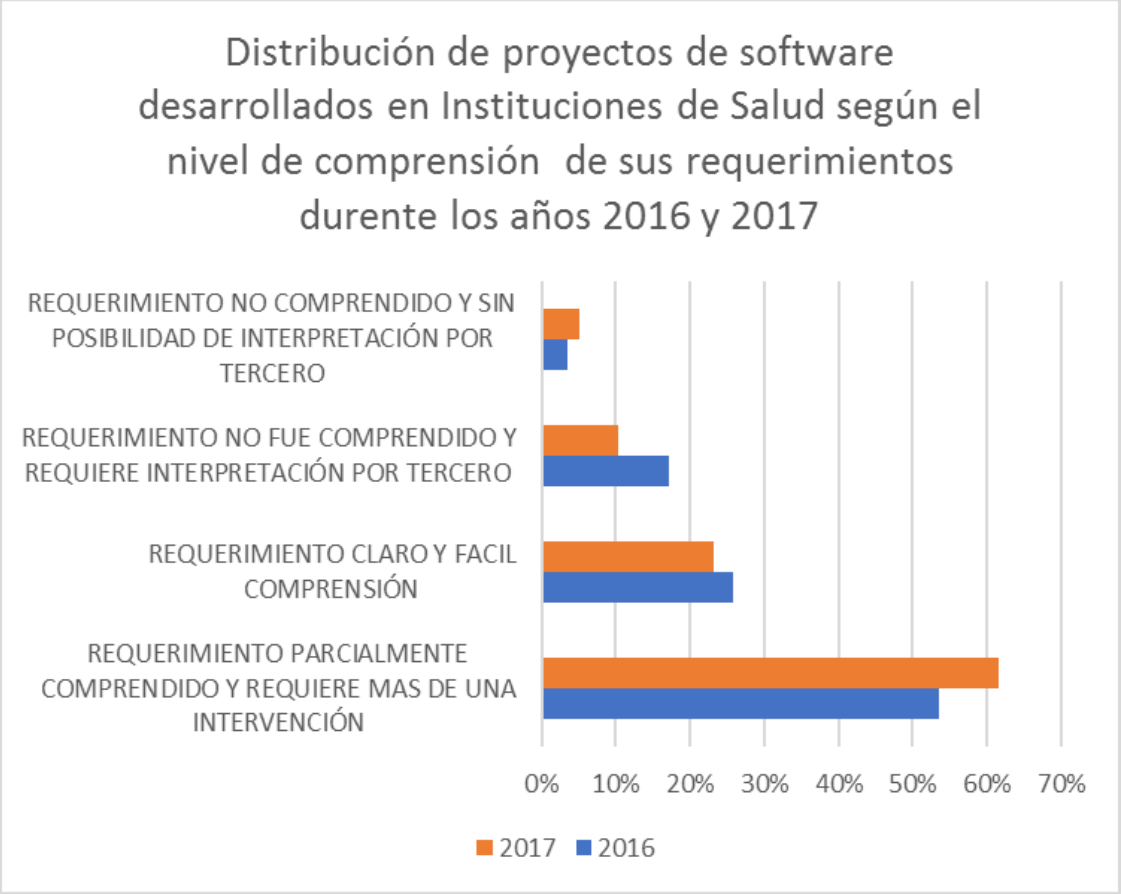


Fig. 23 Distribución de proyectos de software desarrollados en instituciones de salud según el nivel de comprensión de sus requerimientos durante los años 2016 y 2017.

A la fecha, los expertos TI aceptan que existe cierto grado de dificultad en la comprensión de los requerimientos, lo cual implica, que la información levantada sea parcialmente comprendida y requiera más de un levantamiento. Lo esperado, es que, en la medida en que se realicen un mayor número de proyectos de software en Instituciones de Salud, esta brecha comunicacional y, por ende, de comprensión se

acorte. Pero el factor que impide esto, tiene relación con la gran variedad y magnitud de los procesos hospitalarios, muchos de estos no documentados y cambiantes a mediano corto plazo debido a los constantes avances tecnológicos y científicos existentes, obligando al experto TI a levantar el proceso, comprenderlo y posterior a esto interpretar el requerimiento. Actualmente los Equipos TI consideran importante la existencia de un actor que facilite esta comprensión, pero por el momento esto no se ha formalizado.

Al comparar los proyectos 2016 y 2017, observamos que en ambos se continua comprendiendo totalmente el requerimiento durante el análisis de este, y lo destacable es, que se observa una inflexión que retrasa su comprensión siendo completada en etapas posteriores como se muestra en la siguiente gráfica.

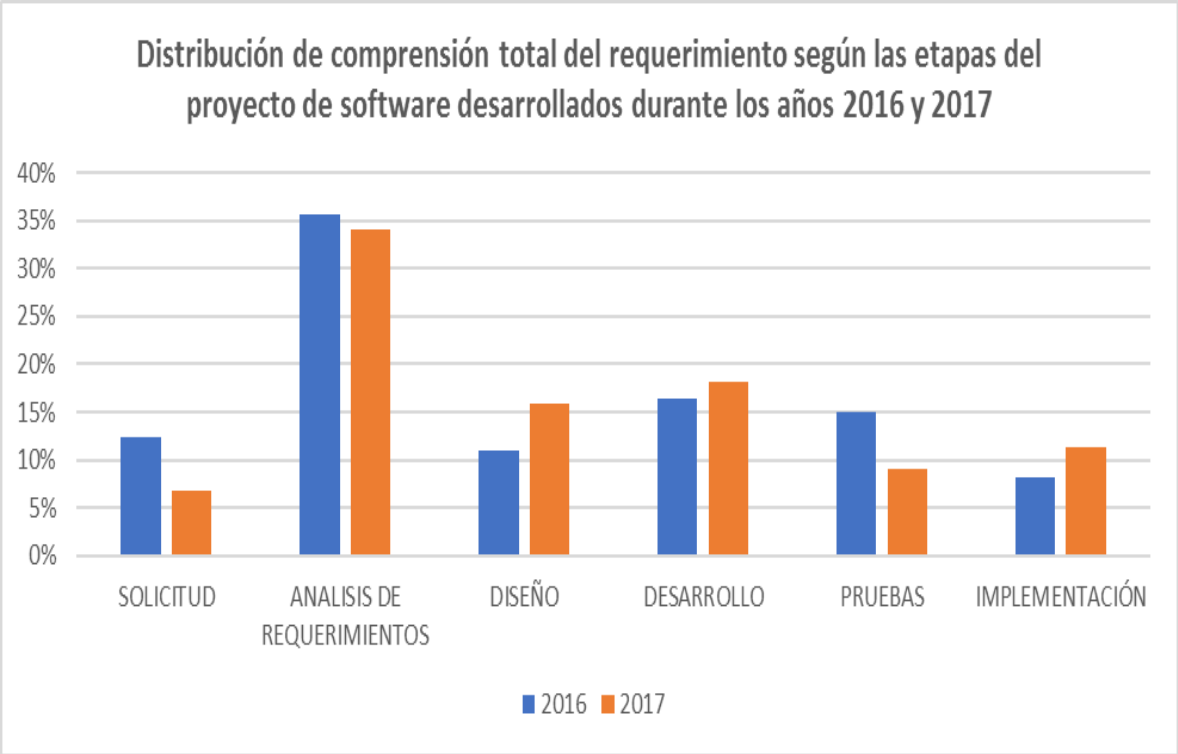


Fig. 24 Distribución de obtención de comprensión total del requerimiento según etapas del proyecto de software desarrollado durante los periodos 2016 y 2017.

Al comparar las técnicas empleadas para realizar el levantamiento de requerimientos, se observa una tendencia conservadora en donde las más empleadas son: Entrevistas, Casos de Uso y BPMN. El hecho, de que se continúe empleando esta última, se asocia a la necesidad de levantar procesos previos al análisis y validación del Requerimiento (Tabla 6).

Tabla 6. Técnicas empleadas en el levantamiento de Requerimientos en proyectos realizados en los periodos 2016 a primer trimestre 2017.

TECNICA EMPLEADA	*PROYECTOS 2016*	*PROYECTOS 2017*
ENTREVISTA	34%	31%
CASOS DE USO	24%	27%
BPMN	16%	18%
CUESTIONARIO	8%	2%
TORMENTA DE IDEAS	8%	9%
PUNTO DE VISTA	8%	8%
OTRA	2%	5%

En cuanto a la metodología empleada para el desarrollo del proyecto de software existe variación entre los años 2016 y 2017 la cual evidencia una preferencia por las metodologías Ágiles por sobre las Tradicionales tal como se muestra en el siguiente gráfico:

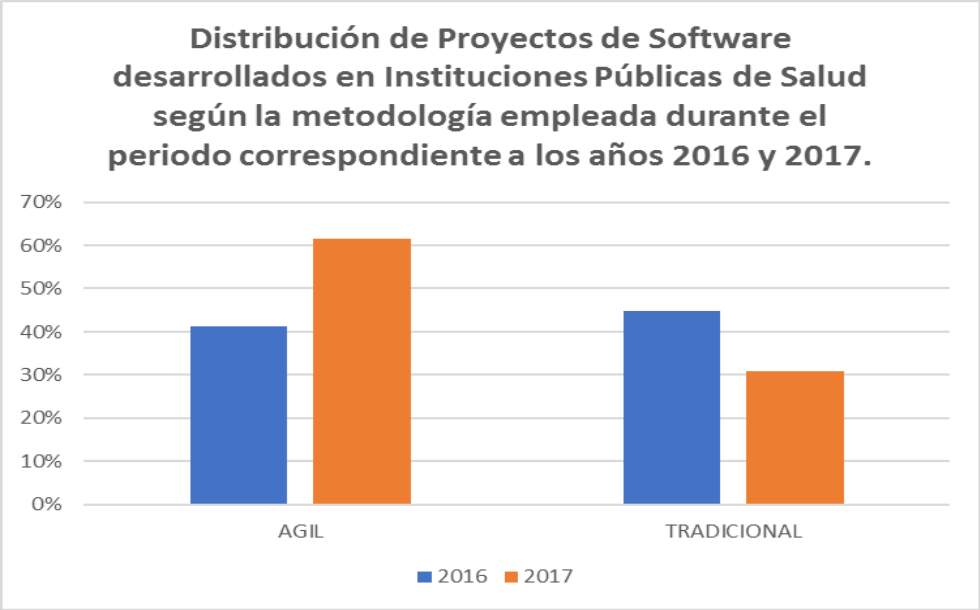


Fig. 25 Distribución de Proyectos de Software desarrollados en Instituciones Públicas de Salud según la metodología empleada durante el periodo correspondiente a los años 2016 y 2017.

También se observó variaciones en cuanto a la gestión del proyecto, en donde, solo un 20% del total de proyectos evaluados cumple con la triada: tiempo, presupuesto y alcance. De estos, al situarlos en nuestra línea de tiempo podemos visualizar que el número de proyectos que cumplen con la triada es cada vez menor bajando de un 28% obtenido en el año 2016 a un 10% de cumplimiento durante el año 2017.

Al analizar cada componente de la triada y su comportamiento en el periodo 2016 a 2017 se observa el siguiente fenómeno.

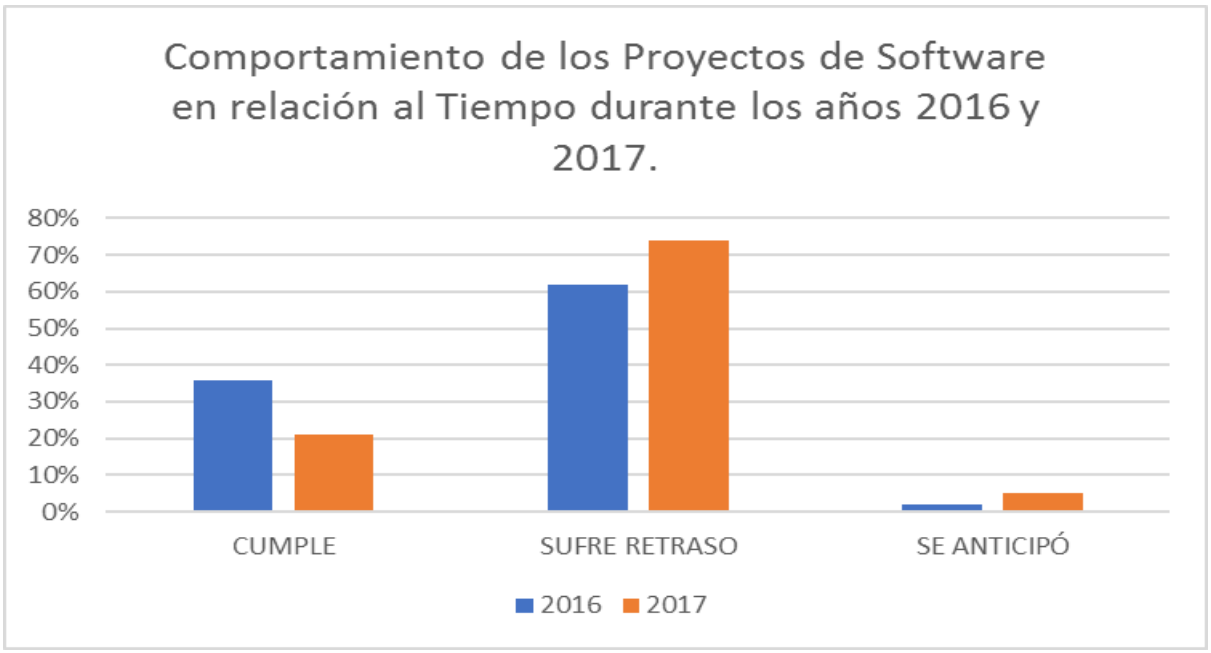


Fig. 26 Comportamiento de los Proyectos de Software en relación al tiempo durante los años 2016 y 2017.

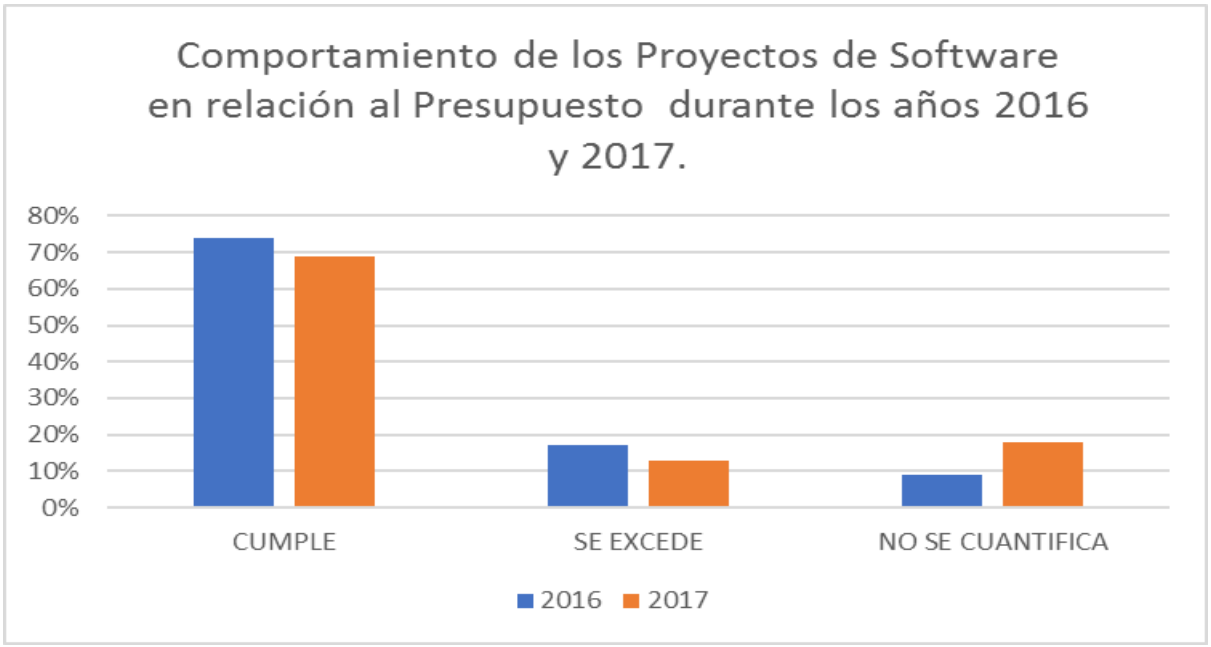


Fig. 27 Comportamiento de los Proyectos de Software en relación al Presupuesto durante los años 2016 y 2017.

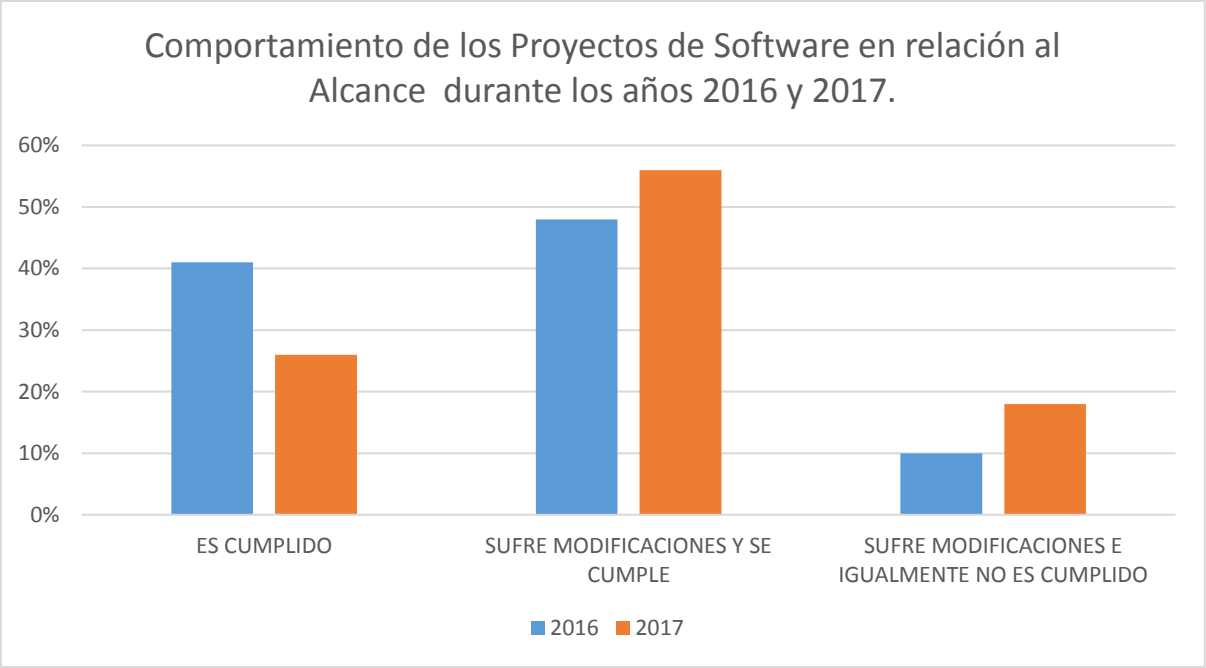


Fig. 28 Comportamiento de los Proyectos de Software en relación al Alcance durante los años 2016 y 2017.

Los tres gráficos expuestos demuestran que existe una tendencia en el tiempo la cual indica que los proyectos están sufriendo cada vez más retrasos, que existen proyectos que no cuantifican sus presupuestos y que se ven forzados a modificar el alcance poder cumplirlo.

En cuanto al grado de satisfacción obtenida, se observa que durante el 2016 existe un mayor nivel de satisfacción completa del requirente(56%) que durante el 2017(44%). Esto se debe a que durante el último año existen más requirentes con satisfacción parcial del producto entregado. Esto no demuestra una baja en la calidad del trabajo realizado, sino, un aumento de la exigencia del solicitante ya que en el 2017 no se pesquisan proyectos que generen insatisfacción a diferencia del año 2016.

Tabla 7. Nivel de Satisfacción del Requirente al momento de la entrega del Proyecto de Software durante los periodos 2016 y 2017.

	Periodo Evaluado	
	2016	**2017**
Satisfacción completa	56%	44%
Satisfacción Parcial	40%	56%
Insatisfecho	3%	0%

Dentro de los problemas identificados durante el desarrollo de un proyecto de software en instituciones públicas de salud se observa una tendencia mantenida en donde continuamos evidenciando requerimientos incompletos, cambiantes, y usuarios resistentes lo cual implica más de una intervención para lograr un levantamiento que disminuya las iteraciones durante el proyecto.

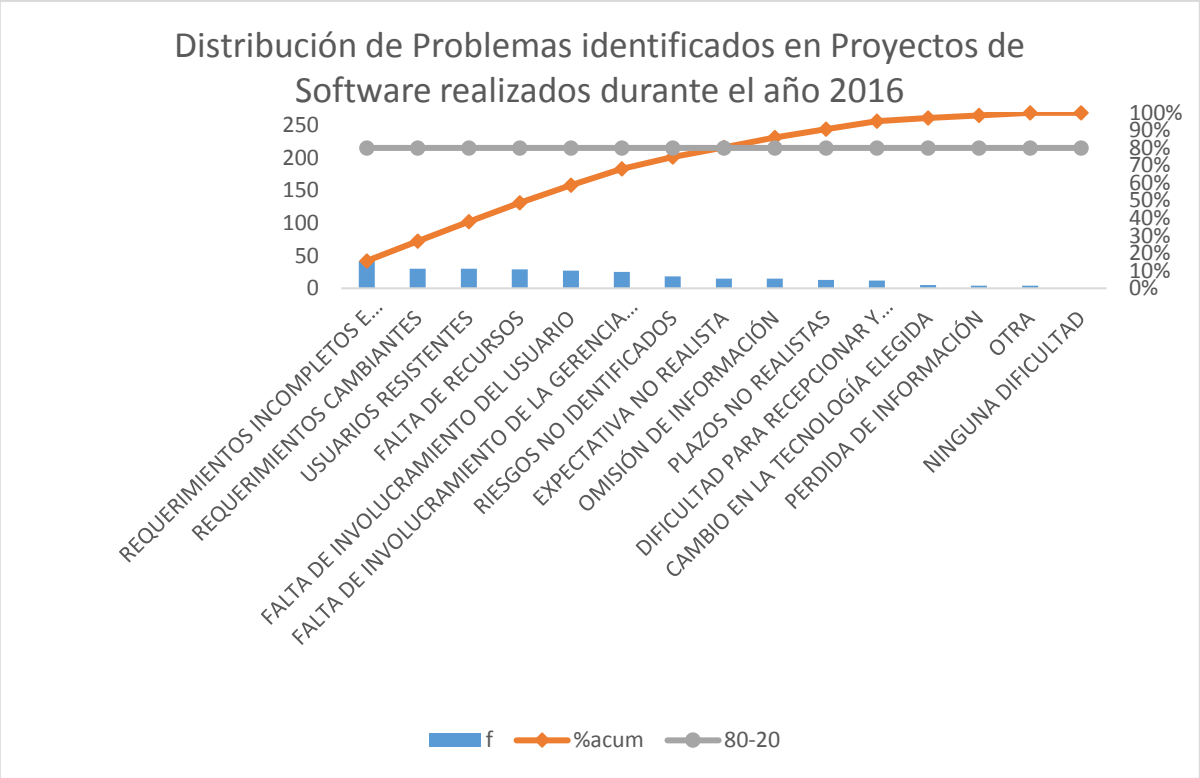


Fig. 29 Distribución de Problemas Identificados en Proyectos de Software realizados durante el 2016.

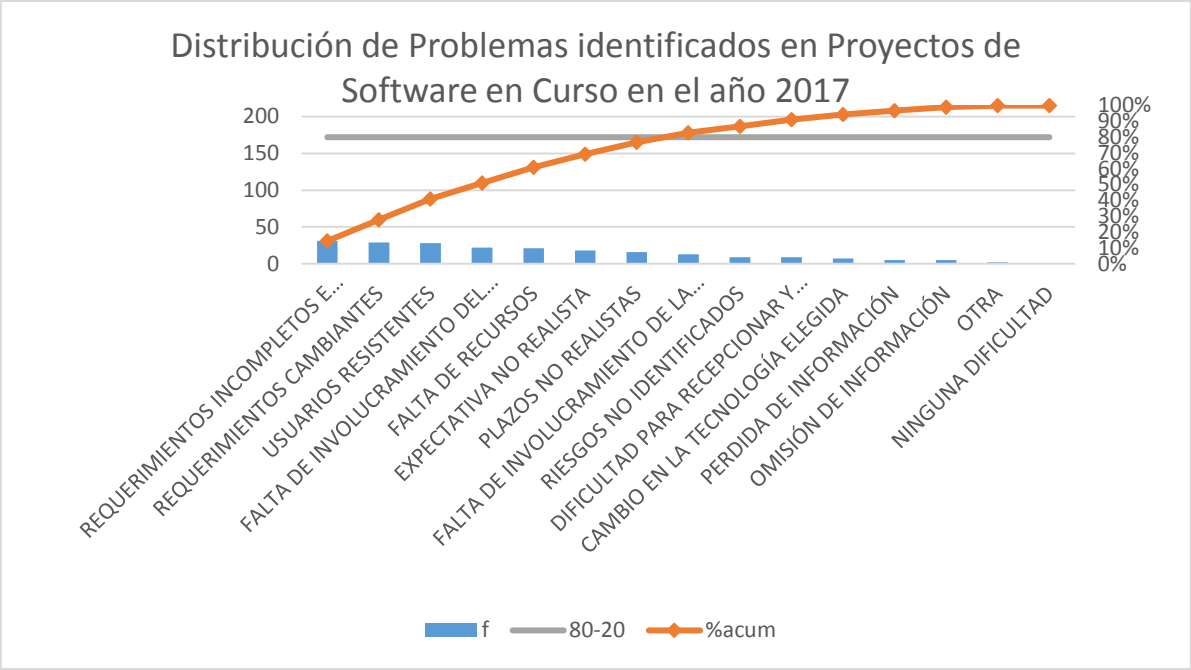


Fig. 30 Distribución de Problemas Identificados en Proyectos de Software realizados durante el 2017.

CAPITULO V: PRESENTACIÓN DE CONCLUSIONES.

Esta tesis nos ha permitido levantar, analizar, concentrar la experiencia de los expertos en Tecnologías en Información y Comunicaciones durante el desarrollo de Proyectos de Software en sus Instituciones Públicas de Salud de Atención Cerrada en la Región Metropolitana durante el periodo comprendido entre el año 2016 y el primer trimestre del 2017.

Las conclusiones obtenidas fueron agrupadas en lo que se denominaron 3 momentos del proyecto de desarrollo informático en instituciones de salud:

Momento 1: Interacción Requirente-Experto-Entorno.

- El sector Salud se ha convertido en una empresa de consumo en donde se observó que nuestros principales clientes son los propios Pacientes, quienes nos comunican día a día sus necesidades forzándonos a mejorar continuamente. Dichos clientes generan requerimientos análogos, que luego de ser identificados y analizados por la Gerencia o Dirección se decide formar un equipo de trabajo en donde en conjunto con los clínicos, se realiza una solicitud a las Unidades TI para que evalúen la situación y generen un producto digital que nos permita evitar futuras complicaciones.

- La Gerencia o Dirección, es la dueña de la iniciativa al momento de solicitar un proyecto de software. Esto se debe a que es la Autoridad máxima a nivel organizacional, tiene acceso a información privilegiada y genera un puente e interlocución entre lo que desean los pacientes y quienes tienen la capacidad de ayudarlos (Clientes Internos y Unidades de Informática). La tendencia observada indica que es el paciente quien presenta una disconformidad, la gerencia evalúa y delega, las Unidades de Informa.

- Claramente, la primera interacción entre un Clínico y un experto TI es compleja, ambos son profesionales, que pertenecen a distintas áreas de desempeño, cada uno

posee su entorno propio y cuenta con su propio idioma, lo cual dificulta la comunicación efectiva bidireccionalmente. No siempre el interlocutor logra transmitir fielmente el mensaje que contiene el requerimiento, y a esto, hay que adicionarle que el experto TI no siempre logra recepcionar fielmente lo que el requirente desea.

- Al observar los requerimientos presentados a las Unidades TI, queda en evidencia una necesidad por centralizar, ordenar, facilitar acceso y asegurar la información, en conjunto con mejorar la supervisión y control de los procesos existentes. Actualmente, son la Unidad de Archivo y las Fichas Clínicas Análogas quienes dan conformidad a todos estos atributos solicitados, pero el aumento de la demanda asistencial, los avances tecnológicos en medicina, y la evolución continua de sus procesos, nos impulsará, a mediano-largo plazo, a la informatización y automatización de estos.

- Los actores más frecuentes de encontrar dentro de un proyecto de software desarrollado en una institución de salud son: Ingeniero Informático, jefe o Supervisor de Servicio Clínico, y Personal Clínico. Aun cuando, la tendencia es formar equipos multiestamentales, de igual forma, queda expuesta brecha comunicacional existente, la cual, nos impide optimizar tiempo y recursos invertidos en el levantamiento de requerimientos obligándonos a realizar más de una iteración, muchas veces inoportuna, debido a dificultad de agendar reuniones de trabajo con el personal clínico el cual esta demandada continuamente por sus pacientes.

- Dentro de las dificultades más frecuentes se encuentran, Requerimientos Incompletos, los cuales son una consecuencia de comunicación disfuncional, discontinua y muchas veces postergada, debido a que las instituciones de salud de atención cerrada no cuentan con un experto TI que cuente con las competencias necesarias que le permitan salir a terreno y levantar requerimientos desde los procesos clínicos mediante la observación directa.

- Toda actualización científico técnica afecta a los procesos clínicos, modificándolos o reformulándolos. Esto afecta directamente a los proyectos de software en desarrollo ocasionando una reevaluación de los requerimientos iniciales modificando el alcance, generando retraso, pero manteniendo el presupuesto inicial.

- Indirectamente se observó la existencia de Usuarios o Clínicos Resistentes a la Automatización, por miedo a modificar los procesos existentes. Por otra parte, cuando no hay resistencia existe ausencia, manifestándose en el poco involucramiento de profesionales clínicos y la dirección en los proyectos evaluados dejándolos en manos de los Ingenieros Informáticos existentes en la Institución.

- Mientras no exista un idioma común entre experto TI y Profesional Clínico, siempre estarán presentes los problemas relacionados al levantamiento de requerimientos lo cual genera retraso del proyecto y confusión entre los actores participantes. El hecho de que los procesos clínicos no estén debidamente documentados e interrelacionados, impide que el experto TI pueda visualizar un correcto mapa de procesos intrahospitalario y sus correspondientes trastiendas.

Momento 2: Planeación, Estructuración Y Desarrollo.

- La planeación es una etapa informal previa a la documentación de la planificación, es considerada a nivel local como indispensable para una adecuada ejecución de las acciones definidas ya que se realiza una simulación verbal y no escrita del proyecto a realizar entre el experto TI y el solicitante. Este ejercicio permite contar con una idea general del requerimiento, previo a su levantamiento y análisis, y su efecto contribuye a no desviarnos hacia otras situaciones a abordar al interactuar con el o los requirentes.

- Los equipos multiestamentales permiten mantener una comunicación efectiva, directa y continua con los procesos clínicos. Esto se potenciaría al contar con la asesoría de un Profesional de la Salud con experiencia clínica y formación en Ingeniería Informática.

- La participación del cliente interno actualmente se restringirse a participar en las acciones de solicitud y análisis del requerimiento, aun cuando existe una leve tendencia que nos indica que han comenzado a integrarse a las etapas de diseño, desarrollo e implementación aumentando así su porcentaje de participación en el proyecto.

- Por otra parte, la Dirección o Gerencia, demostró ser un personaje secundario que debido a sus funciones en la institución solo se limita a aprobar la solicitud, aun cuando debiera de participar de etapas claves del desarrollo, evaluando continuamente si existe necesidad de apoyar con recursos financieros, infraestructura, tecnología y personal competente en pos del éxito del proyecto.

- En cuanto a la metodología de desarrollo más adecuada para asegurar el éxito del proyecto, los resultados expuestos dejan en evidencia que la tendencia está apuntando a desarrollar proyectos de menor tamaño mediante una metodología ágil, la cual es más compatible con el sector salud ya que no requiere de la existencia de procesos documentados a diferencia de una metodología tradicional, considerada más rígida y estructurada, y que solo debiera reservarse para proyectos de gran envergadura y con procesos documentados.

- El hecho de que el levantamiento de procesos se realice en cada proyecto por el experto TI empleando BPMN, no asegura el éxito de estos, debido a que existe riesgo de generar una visión e interpretación sesgada del funcionamiento global de la Institución de Salud dado su gran variedad y magnitud en sus procesos.

- Se pudo observar, que la mayor parte de los expertos TI emplean las mismas herramientas para realizar el levantamiento de requerimientos, sin realizar innovaciones o mejoras que aseguren el éxito y eficiencia. Si consideramos que el entorno hospitalario varía según el servicio o producto generado, veremos que difícilmente se podrán representar fielmente empleando una misma herramienta. Existe levantamiento mediante entrevistas, revisión documental, pero en ningún caso señalaron herramientas que se emplean en el levantamiento de acciones mediante la observación directa.

Momento 3: Evaluación De La Gestión Del Proyecto Realizado.

- Se concluye que a pesar de que las instituciones son una empresa de servicio y consumo, la percepción del proyecto por parte del profesional de salud es un tanto inmadura comparada con la percepción del ingeniero.

- El profesional de la Salud cataloga el éxito de su proyecto basándose exclusivamente en la satisfacción del producto final y la motivación de su premura es alimentada de la necesidad imperiosa de la institución. Es así como un gran porcentaje de proyectos son considerados vitales, de crecimiento y por último de innovación.

- La evaluación de la gestión del proyecto por parte del ingeniero continúa centrándose en Tiempo, Alcance y Presupuesto, en donde la tendencia observada es cumplir con prioritariamente con el Presupuesto, flexibilizando el cumplimiento del Tiempo y el Alcance. Esto radica en que, dado que el Alcance es necesario para no desviarnos de la línea de trabajo, se invierten esfuerzos conjuntos para hacer las modificaciones necesarias que permitan su cumplimiento manteniendo el mismo Presupuesto. Esto inevitablemente genera retraso en el producto final, pero, permite satisfacer parcial o completamente al requirente al momento de la entrega.

ANEXOS.

ANEXO 1: CUESTIONARIO: "RADIOGRAFÍA A LOS PROYECTOS DE SOFTWARE DESARROLLADOS EN INSTITUCIONES PÚBLICAS DE SALUD DE ATENCIÓN CERRADA DURANTE EL PERIODO COMPRENDIDO ENTRE LOS AÑOS 2016 Y EL PRIMER TRIMESTRE DEL 2017".

El siguiente cuestionario cuenta con 17 preguntas cerradas y pretende cumplir con el siguiente objetivo:

- Conocer la experiencia de los expertos en Tecnologías en Información y Comunicaciones durante el desarrollo de proyecto de software en sus Instituciones Públicas de Salud de Atención Cerrada en la Región Metropolitana durante el periodo comprendido entre el año 2016 y el primer trimestre del 2017.

  .

Cabe recordar que su participación es voluntaria y anónima. La información recabada será utilizada para fines docentes y académicos.

Cuentan con 40 minutos para el desarrollo. Agradezco su participación y colaboración en este estudio.

1. Dentro de una Institución de Salud, ¿Desde dónde nace la necesidad de un software?

a) Unidad de Informática.
b) Gerencia o Dirección.
c) Usuarios.
d) Clientes Internos.
e) Institución Externa.
f) Otro (señale desde donde).

2. ¿Cuál o cuáles de los siguientes objetivos específicos se encontraban dentro de la solicitud del requirente en su proyecto de software realizado? (Puede seleccionar más de una opción).

a) Crear un reservorio de Información.
b) Disminuir Carga Laboral.
c) Centralizar y Ordenar la Información.
d) Asegurar la Información.
e) Facilitar el acceso a Información.
f) Difusión de Buenas Prácticas.
g) Mejorar Supervisión y Control.
h) Otro (señale cual).

3. Indique con una “X”, ¿Cuál o cuáles fueron los actores participantes en sus proyectos de software realizado en su Institución de Salud?

a) Ingeniero Civil o Ejecución Informática.
b) Técnico en Informática.
c) Jefe, Supervisor, o Coordinador de la Unidad o Servicio Clínico.
d) Director Médico, Gerencia o Subdirección de la Institución.

e) Personal Clínico (Médicos, Odontólogos, Enfermeras, Matronas, Técnicos, Otros).

4. El producto generado en el Proyectos de Software en el cual participó, era inicialmente para:

a) Satisfacer requerimientos de los Usuarios (Pacientes).
b) Satisfacer requerimientos de Clientes Internos (Funcionarios).
c) Satisfacer requerimientos de la Gerencia o Dirección.
d) Satisfacer requerimientos de Organizaciones Externas.
e) Otro (señale cual).

5. ¿En que etapa del proyecto de desarrollo de software se centró la participación de Clientes Internos y/o Usuarios?

a) Solicitud de una solución Informática.
b) Análisis de Requerimientos.
c) Diseño.
d) Desarrollo.
e) Pruebas.
f) Implementación.
g) Ninguna de las Anteriores

6. ¿En que etapa del proyecto de desarrollo de software se centró la participación de la Gerencia o Dirección?

a) Solicitud de una solución Informática.
b) Análisis de Requerimientos.
c) Diseño.
d) Desarrollo.

e) Pruebas.
f) Implementación.
g) Ninguna de las Anteriores.

7. En cuanto al levantamiento de requerimientos y a la comunicación con los actores participantes. Señale la experiencia vivida en el proyecto de software realizado.

a) El Requerimiento fue claro y de fácil comprensión.
b) El Requerimiento fue comprendido parcialmente y requirió más de una intervención por parte del solicitante.
c) El Requerimiento no fue comprendido y requirió de la interpretación de un tercero (cliente interno, usuario, solicitante).
d) El Requerimiento no fue comprendido por el experto en TIC y no se contó con la interpretación de terceros

8. En cuanto al levantamiento de requerimientos y a la comunicación con los actores participantes. ¿En que etapa del proyecto el requerimiento fue comprendido completamente?

a) Solicitud de una solución Informática.
b) Análisis de Requerimientos.
c) Diseño.
d) Desarrollo.
e) Pruebas.
f) Implementación.
g) Ninguna de las Anteriores.

9. ¿Qué Técnica Emplea Su Unidad Para Realizar El Levantamiento De Requerimientos?

a) Entrevista.
b) BPMN.
c) Cuestionario.
d) Casos de Uso.
e) Tormenta de Ideas.
f) Puntos de vista.
g) Otra.

10. Según la perspectiva del requirente, ¿Cómo clasifica él este proyecto?

a) Vital.
b) Crecimiento.
c) Innovador.
d) Desempeño.

11. ¿Cuál fue la metodología de desarrollo empleada en su proyecto?

a) Ágil
b) Tradicional.
c) Otra.

12. ¿Cuál es el estado del proyecto de software evaluado a la fecha?

a) Finalizado.
b) En curso.
c) Inconcluso.

13. En relación al tiempo definido en la planificación del proyecto de software evaluado, ¿Cómo es su cumplimiento a la fecha?

a) Cumple.
b) Sufre retraso.
c) Se anticipa.

14. En relación al presupuesto definido en la planificación del proyecto de software evaluado, ¿Cómo es su cumplimiento a la fecha?

a) Cumple.
b) Se excede.
c) No se cuantifica.

15. En relación al alcance definido en la planificación del proyecto de software evaluado, ¿Cómo es su cumplimiento a la fecha?

a) Cumple con el alcance definido.
b) Sufre modificaciones y de esta forma es cumplido a cabalidad.
c) Con o sin modificaciones, el alcance definido no es cumplido.

16. En Relación a los Resultados Obtenidos, ¿Cómo se observa la satisfacción del requirente a la fecha?

a) El producto entregado logra satisfacer completamente al requirente.
b) El producto entregado logra satisfacer parcialmente al requirente.
c) El producto entregado no logra satisfacer al requirente.

17. ¿Qué dificultades ha experimentado al participar en el proyecto de software evaluado?

a) Requerimientos incompletos e Insuficientes.
b) Falta de Involucramiento del Usuario.
c) Falta de Involucramiento de la Gerencia o Dirección.
d) Expectativas no Realistas. Requerimientos Cambiantes.
e) Plazos no realistas.
f) Usuarios resistentes.
g) Cambios en la Tecnología elegida.
h) Falta de Recursos. Riesgos no identificados.
i) Pérdida de Información.
j) Omisión de Información.
k) Dificultad para recepcionar y comprender lo solicitado.
l) Ninguna dificultad.
m) Otras(Señale cual o cuales).

BIBLIOGRAFIA.

Aguilar, A. S. (2002). Introducción a la programación extrema. *3*.

Amaro Calderón, S. D., & Valverde Rebaza, J. C. (2007). *Metodologías Ágiles.* Universidad Nacional de Trujillo. Perú: Escuela de Informática.

Arias, M. C. (2011). La ingeniería de requerimientos y su importancia en el desarrollo de proyectos de software. *InterSedes, 6*(10).

Aristizábal, N. M., & Torres, M. M. (2009). técnicas de Levantamiento de Requerimientos con Innovación. *En Cuarto Congreso Colombiano de Computación 4CCC*.

Beck, K., Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., . . . Thomas, D. (2001). *http://agilemanifesto.org/*.

Canós, J. H., Letelier, P., & Penadés, M. C. (2003). Metodologías Ágiles en el desarrollo de software. *Metodologías Ágiles en el desarrollo de software*, 1.

Canós, J. H., Letelier, P., & Penadés, M. C. (2003). Metodologías Ágiles en el desarrollo de software. En P. Letelier Torres, & E. A. Sanchez Lopez (Ed.), *Metodologías Ágiles en el desarrollo de software*, (págs. 1-8). Alicante.

Charette, R. N. (2005). Why software fails. *IEEE Spectrum, 42*(9), 36.

Cockburn, A., & Williams, L. (2000). The Costs and Benefits of Pair Programming. *Humans and Technology Technical Report*, 1-11.

Estayno, M. G., & Panizzi, M. D. (2009). Participación de los usuarios en el proceso de desarrollo de software. *En XI Workshop de Investigadores en Ciencias de la Computación*.

Fowler, M. (2001). The New Methodology. *Wuhan University Journal of Natural Sciences, 6*(1-2), 12-24.

Fowler, M., & Foemmel, M. (2001). *Continuous Integration*. Obtenido de www.martinfowler.com: www.martinfowler.com/articles/designDead.html

Gulla, J. (2012). Seven reasons IT proyectFail. *IBM Systems Magazine*.

Jaramillo, C. Z., & Isaza, F. A. (2012). los modelos verbales en el lenguaje natural y su utilización en la elaboración de esquemas conceptuales para el desarrollo de software: una revisión crítica. *Revista Universidad EAFIT, 41*(137), 77-95.

Jeffries, R., Anderson, A., & Hendrickson, C. (2001). *Extreme Programming Installed.* Addison-Wesley.

Lawshe, C. H. (1975). A quantitative approach to content validity 1. *Personnel Psychology, 28*(4), 563-575.

Letelier, P., & Penadés, M. (abril/junio de 2006). Metodologías ágiles para el desarrollo de software: eXtreme Programming (XP). *Tecnica Administrativa, 5*(26), ISSN 1666-1680. Obtenido de http://www.cyta.com.ar.

Martin, R. C. (2002). Continuous Care vs. Initial Design.

Maure, Y. D., & Castillo, M. S. (2011). Estrategia de validación de calidad de un producto software.Modelado del proceso según las pautas establecidas por BPMN. *Iberoamerican Journal of Proyect Management, 2*(1).

Poppendieck, M., & Poppendieck, T. (2003). Lean Software Development:An Agile Toolkit for Software Development Managers. *Addison-Wesley Longman Publishing Co*, 240.

Pressman, R. S., & Troya, J. (2010). *Ingeniería del Software* (Séptima ed.). McGraw Hill.

Pytel, P., Uhalde, C., Ramón, H., Castello, H., Tomasello, M., Pollo, M. C., . . . García, R. M. (2011). Ingeniería de requisitos basada en técnicas de ingeniería del conocimeinto. *En workshop de Investigadores en Ciencias de la Computación*.

Quelopana, A. R., Vega, V. Z., Gallardo, J. A., & Meneses, C. V. (2009). Una propuesta metodológica para modelar procesos de negocio de decision como técnica de elicitación de requisitos para sistemas Business Intelligence. *En WER*.

Royce, W. W. (1970). Managing the development of large software systems. *IEEE WESCON*, 1-9.

Sommerville, I. (2005). *Ingeniería del Software* (7 ed.). Madrid: Pearson Addison Wesley.

Standish Group. (1995). *The CHAOS report*. Obtenido de www.standishgroup.com.

Tristán-López, A. (2008). Modificación al modelo de Lawshe para dictamen cuantitativo de la validez de contenido de un instrumento objetivo. *Avances en Medición, 6*, 37-48.

Zapata, C., & Carmona, N. (2010). Un modelo de diálogo para la educción de requisitos de software. *Dyna, 77*(164), 209-219.