Alumno/Tesista: Ing. Silvana A. Santos

Director: Dra. Gabriela Robiolo

Co-Director: Lic. Alejandro Oliveros

Estimación de proyectos de software pequeños basada en el juicio de expertos

Tesis presentada para obtener el grado de

Magister en Ingeniería de Software

Facultad de Informática - Universidad Nacional de La Plata

October, 2014

A mis padres

y a mi compañero de vida que significa tanto para mí, Christian

Agradecimientos

Principalmente, a mi Directora de Tesis, Dra Gabriela Robiolo por su guía, e invaluable contribución. También por la paciencia, la motivación y el generoso apoyo que me brindó durante el proceso de elaboración de la presente Tesis; sin ella hubiera sido imposible la realización de la misma.

A mi Co-Director de Tesis, Licenciado Alejandro Oliveros por el gran valor de su contribución y aportes de conocimiento a la presente Tesis.

A mis padres, por el apoyo y los valores que me transmitieron desde pequeña.

A Christian por su apoyo, paciencia y amor

Contenido

1. Introducción 8

1.1. Motivación 8

1.2. Objetivo de la Tesis 9

1.3. Publicaciones 10

1.4. Organización del trabajo 10

2. Estado del arte 11

3. Marco teórico de los métodos de estimación más utilizados en la industria. Combinación de Juicio de Expertos 14

3.1. Juicio de Expertos 14

3.1.1. Evidencias de aplicación en empresas 14

3.1.2. Descripción del método 15

3.1.3. Juicio de Expertos vs Modelos formales de estimación 21

3.2. Delphi 22

3.2.1. Evidencias de aplicación en empresas 22

3.2.2. Descripción del método 23

3.3. Wideband Delphi 24

3.3.1. Evidencias de aplicación en empresas 24

3.3.2. Descripción del método 27

3.4. Analogías 30

3.4.1. Evidencias de aplicación en empresas 30

3.4.2. Descripción del método 30

3.5. Planning Poker 31

3.5.1. Evidencias de aplicación en empresas 31

3.5.2. Descripción del método 32

3.6. Comparación de métodos de estimación basados en Juicio de Expertos 34

4. Descripción del problema 38
4.1. Contexto 38
4.2. Pequeños Proyectos de Software 38
4.3. Problema 39
5. Descripción de la propuesta: Método de Estimación Basado en la Especificación de Requerimientos (MEBER) 39
5.1. Características del MEBER 39
5.2. MEBER. Diagrama de estados 42
5.3. ERS (Especificación de Requerimientos de Software) 44
6. Caso de estudio 45
6.1. Características de los proyectos 45
6.2. Aplicación del método MEBER 49
6.3. Análisis de los resultados 51
7. Discusión 52
8. Conclusión 53
9. Trabajos de investigación futuros 54
Bibliografía básica relacionada 55
Glosario de siglas y términos 60
ANEXOS 62
ANEXO I - ¿Es posible superar la precisión basada en el juicio de expertos de la estimación de esfuerzo de productos de software? 63
1 Introducción 64
2 El juicio de expertos 65
3 El juicio de expertos vs los métodos formales 66
4 Caso de Estudio 67
4.1 Estimación de CU de una aplicación 69
4.2 Estimación sucesivas de versiones de una aplicación 71
4.3 Amenazas de validez 72

4.4 Conclusiones del caso de estudio 72
5 Conclusión final 73
ANEXO II - Expert Estimation and Historical Data: An Empirical Study 75
I. INTRODUCTION 76
II. ESTIMATION METHODS 77
A. Expert Estimation Method (ExE) 77
B. Analogy-Based Method (AbM) 78
C. Historical Productivity 78
III. DESCRIPTION OF OUR EMPIRICAL STUDY 78
A. Definition 78
B. Planning 79
C. Execution 81
D. Threats to validity 82
IV. DATA ANALYSIS AND INTERPRETATION 82
A. Result analysis 83
1) Undergraduate 83
2) Practitioners 83
3) The statistical significance of the results 84
4) Discussion 84
V. RELATED WORK 85
VI. CONCLUSION AND FUTURE WORK 86
AKNOWLEDGMENTS 86
REFERENCES 86
ANEXO III - Método de Estimación Basado en la Disponibilidad de Horas persona (MEBDHP) 88
1. Descripción de método 89
1.1 Estimación Inicial 89
1.2 Estimaciones detalladas 91

1.3 Comparación de las estimaciones detalladas y los valores de la Tabla de Horas Disponibles 93

1.4 MEBDHP. Diagrama de Estados 93

2. Debilidades del MEBDHP 96

ANEXO IV - Software Requirements Specification 97

Introduction 98

Purpose 98

Document Conventions 98

Intended Audience and Reading Suggestions 98

Scope 98

References 98

Overall Description 98

Product Perspective 98

User Characteristics 99

Operating Environment 99

Design and Implementation constraints 99

User Documentation 99

Assumptions and Dependencies 99

System Features 99

System feature 1 99

Descriptions and priorities 99

Functional Requirements 100

System feature 2 100

Descriptions and priorities 100

Functional Requirements 100

External Interface Requirements 100

User Interface 100

Software Interfaces 100

Other Nonfunctional Requirements 100
Performance Requirements 100
Safety Requirements 101
Security Requirements 101
Software Quality 101
Other Requirements 101
Appendix – Glossary 101

1. Introducción

1.1. Motivación

En el contexto de la ingeniería de software persiste la preocupación de la causa de los fracasos de proyectos de software. Charette [11] encuentra las siguientes causas: objetivos de proyectos no realistas o inarticulados; estimaciones de los recursos necesarios inexactas, requerimientos mal definidos; estado del proyecto mal reportado; riesgos no administrados; comunicación ineficiente entre clientes, desarrolladores y usuarios; inmadurez en las tecnologías utilizadas; inhabilidad para controlar la complejidad de los proyectos; descuido en las prácticas de desarrollo de software; administración de proyectos insuficiente y la deficiente armonización de las políticas de los actores involucrados y las presiones comerciales con los objetivos del proyecto.

En las últimas décadas el aumento del tamaño y la complejidad de los proyectos de software han aumentado el valor económico de los fracasos de proyectos. Además, la presencia de realidades informatizadas en todos los ámbitos de la vida social lleva a impactar en pequeña y gran escala a numerosas personas e instituciones.

Dentro de las causas de fracasos de proyectos de software enunciadas anteriormente, que se entienden muy cercanas a la realidad, en este trabajo interesa poner foco en las estimaciones de recursos, focalizando el estudio en el costo de desarrollo medido principalmente en horas persona.

Ha sido muy valiosa la recopilación de información realizada por Jørgensen y Shepperd [39], quienes realizan una revisión sistemática de estudios ya hechos en este tema y proveen recomendaciones para investigaciones futuras. Su estudio abarca 304 artículos de 76 Journals y revistas de investigación de Ingeniería de Software publicados hasta Abril del 2004. Identifican y seleccionan con muy buen criterio los artículos que describen investigaciones en estimaciones de costos en el desarrollo de software y/o aquellos artículos donde no siendo éste el tema central de la investigación, igualmente realizan un aporte en esta área de conocimiento.

Jørgensen y Shepperd afirman que existen pocos investigadores en esta área y que además es necesario definir un marco adecuado para el desarrollo de trabajos de calidad. Los autores proponen realizar las siguientes mejoras en este tema de investigación:

- *Profundizar temas básicos de las estimaciones de software*. Boehm et al. [8] y Foss et al. [15] resaltan la importancia de la comparación y evaluación de los métodos de estimación. Se destacan: la evaluación de la precisión de un método de estimación y la selección apropiada de un método de estimación.
- *Ampliar las investigaciones sobre métodos de estimación más comúnmente usados actualmente en la industria del software.* El método de estimación dominante es el basado en el juicio experto (bajo la forma de analogías, experiencias, intuición). Jørgensen [25] indica que en su revisión sistemática de estimación experta no se han logrado encontrar

estudios que reportasen que la mayoría de las estimaciones fueron basadas en métodos de estimación formales. Hay evidencia disponible [25] que sugiere que la precisión en la estimación no mejora con el uso formal de modelos de estimación. A pesar de esto, las investigaciones sobre estimación experta es escaza.

- *Estudios que den soporte al método de estimación basadas en el juicio de expertos en lugar de reemplazarlos por otros métodos de estimación.* Siendo el juicio de expertos el método más usado en la industria del software, sería conveniente potenciar este juicio usando los métodos formales de estimación.
- *Aplicación de métodos de estimaciones de costos aplicados en situaciones reales.* Existen pocos estudios donde los métodos de estimación son evaluados en situaciones reales, puesto que la mayoría se han aplicado en contextos no reales de laboratorios.

Otro tema importante que queremos destacar en el presente trabajo de tesis es el cambio de enfoque que Jørgensen et.al. [35]. plantean para evitar desviaciones en las estimaciones. Este enfoque indica que es necesario usar el formato tradicional de pregunta a la hora de requerir estimaciones: "¿Cuántas horas se necesitan para completar la tarea X?" versus el enfoque alternativo que se suele usar: "¿Cuántas tareas se pueden realizar en tal período de tiempo?" El formato alternativo genera lo que Jørgensen llama efectos de anclaje. Independientemente de los mecanismos responsables de tales efectos de anclaje, los resultados obtenidos en el estudio realizado por Jørgensen et.al. [35] sugieren que el enfoque alternativo debe evitarse, especialmente en proyectos de corta duración y en situaciones donde se ve una tendencia hacia las estimaciones de esfuerzo muy optimistas.

Las consideraciones anteriores nos han llevado a interesarnos por el método de estimación de esfuerzo de proyectos de software basada en el juicio de expertos, analizando un caso de estudio en el área de desarrollo de software de una empresa multinacional, utilizado para dar soporte a sus actividades comerciales.

Nosotros pudimos ver la diferencia que genera la utilización de ambos enfoques planteados por Jørgensen et.al. [35]. en el caso de estudio planteado en este trabajo ya que comparamos dos métodos de estimación. El método aplicado en la empresa anteriormente estaba guiado por un enfoque donde se evaluaba qué cantidad de requerimientos se podían cubrir en una cierta cantidad de tiempo o bajo un cierto monto de dinero disponible por año. La directriz de nuestra propuesta, es en cambio, trabajar poniendo foco en cada requerimiento y estimarlos independientemente de la cantidad de dinero o tiempo que se tuviera disponible.

1.2. Objetivo de la Tesis

El presente trabajo busca mejorar el método de estimación de esfuerzo de pequeños proyectos de software basado en el juicio de expertos, aplicado en una empresa con la finalidad de reducir el error en las estimaciones de esfuerzo medido en horas persona y aumentar la satisfacción del cliente.

Concretamente, la gerencia de dicha empresa solicitó al área de desarrollo trabajar con una mayor precisión en las estimaciones para hacer un uso más eficiente de los recursos.

Por lo tanto se plantea la siguiente pregunta de investigación:

¿Es posible mejorar la estimación de esfuerzo medido en horas persona de los pequeños proyectos de Software?

1.3. Publicaciones

Fue posible publicar dos artículos con referato en congresos afín a la Ingeniería de Software. Si bien el tema central de estos artículos no es el centro del presente trabajo de tesis, la tesista realizó su aporte al desarrollo teórico de los mismos.

En el Anexo I se detalla la publicación: "Es posible superar la precisión del juicio de expertos de la estimación de esfuerzo de proyectos de software?" por Robiolo, G.,Castillo, O.,Rossi, B. y Santos, S., Experimental Software Engineering Latin American Workshop (ESELAW 2013), Uruguay, 2013.

En el Anexo II se detalla la publicación: "Expert Estimation and Historical Data: an Empirical Study, in proceedings", The Eighth International Conference on Software Engineering Advances, ICSEA 2013, October 27 - November 1, 2013 - Venice, Italy.

1.4. Organización del trabajo

A continuación se describen los contenidos de los próximos capítulos.

El Capítulo 2 presenta el estado del arte del método de estimación basado en el juicio de expertos destacando principalmente el trabajo de Jørgensen, quien ha realizado un importante aporte en los últimos años en esta área.

El Capítulo 3 describe los métodos de estimación más utilizados en la industria del software, su historia, evidencia de aplicación en empresas y autores destacados.

El Capítulo 4 describe el contexto en el que se desarrolló el caso de estudio sobre el que se basa la presente tesis, la categorización de *Pequeño Proyectos de Software* en el marco de la empresa y el problema que se presentó en la misma.

El Capítulo 5 describe el nuevo método de estimación propuesto llamado Método de Estimación Basado en la Especificación de Requerimientos (MEBER).

El Capítulo 6 presenta un caso de estudio basado en datos recolectados de proyectos representativos de la realidad que se ejecutaron bajo los lineamientos del método propuesto MEBER.

El Capítulo 7 describe la discusión sobre los resultados obtenidos.

El Capítulo 8 presenta las conclusiones y el Capítulo 9 destaca las contribuciones y recomendaciones para la puesta en marcha de trabajos futuros.

2. Estado del arte

En términos de estimaciones de proyectos de software, Magne Jørgensen ha realizado grandes aportes en esta área con sus innumerables publicaciones. Para el desarrollo de la presente tesis, se han consultado varios artículos de este autor.

Dicho autor se ha interesado en la estrategia de estimación basada en el juicio de expertos, realizando una revisión de estudios detallada en su artículo "A review of studies on expert estimation of software development effort" [25]. Los resultados arrojados por dicho proceso de revisión sugieren que las estimaciones basadas en juicio de expertos es la modalidad más utilizada para proyectos de software. También afirma que no existe evidencia sustancial en favor del uso de modelos de estimación y que hay situaciones donde se puede esperar que las estimaciones expertas sean mucho más precisas que los métodos formales de estimación. Según el caso de estudio presentado en Robiolo et.al. [53], no se pudo superar la precisión de juicio de expertos al compararlo con otros métodos formales tales como Regresión Lineal y Analogías. Además [25] propone una lista de 12 "best practices" o principios de estimación experta validados empíricamente y provee sugerencias sobre cómo implementar estas guías en las organizaciones.

Jørgensen [25] identifica los siguientes puntos como ventajas del juicio de expertos sobre el uso de modelos:
(a) Los expertos en el dominio del desarrollo de software típicamente tienen más información específica del contexto que los modelos.
(b) Las predicciones realizadas por expertos en situaciones inestables (por ejemplo: dada en la relación esfuerzo-tamaño) resultan ser mejores mientras que los modelos funcionan mejor en situaciones estables, conocidas.

También se reconoce que el juicio de expertos tiene sus aspectos negativos, como por ejemplo: el grado de inconsistencia y la ponderación de variables. Si estos efectos negativos se pudieran reducir, la mejora en la precisión de las estimaciones sería mucho mayor.

Jørgensen y Boehm [29], en el artículo Estimación de esfuerzo de Desarrollo de Software: ¿modelos formales o juicio de expertos?, toman dos posturas opuestas y debaten intentando mostrar a las organizaciones las ventajas y desventajas de modelos formales y juicio de expertos [3]. Jørgensen apoya el uso de este último ya que sostiene que hay evidencia suficiente de que este método arroja resultados más precisos que los modelos formales y es la más utilizada en la actualidad por las organizaciones. Además, alienta a las organizaciones a invertir en estudios e investigaciones para mejorar los procesos de estimación basados en juicio experto. También Jørgensen propone el uso del método Wideband Delphi[1] de Bohem para mejorar las estimaciones y evitar posibles pujas de intereses entre los stakeholders.

Bohem difiere con Jørgensen en que las estimaciones expertas producidas en estudios empíricos no son representativas de las estimaciones producidas en la práctica. Además, sostiene que ante la incertidumbre, las organizaciones van a optar por llevar a cabo exten-

[1] En la sección 3.2 se detalla el método Wideband Delphi.

sos análisis de sensibilidad, riesgo y compensación a través de modelos formales ejecutados de manera rápida y con muy poco esfuerzo humano. Boehm recomienda combinar ambos enfoques, almacenando los resultados al finalizar el desarrollo (o fases del mismo) y utilizar estos valores en el proceso de "close-loop feedback" donde se comparan las entradas de las estimaciones, las salidas y los resultados finales de manera de aprender de eso y poder calibrar estimaciones de futuros proyectos.

Respecto de la evaluación de la incertidumbre de las estimaciones realizadas, nos planteamos qué rol cumple el feedback sobre la discrepancia existente entre las horas estimadas versus las trabajadas. Existe evidencia suficiente [29], que indica que la mayoría de los desarrolladores son, inicialmente, demasiado optimistas sobre la precisión de sus estimaciones, manteniéndose así aun cuando el feedback provisto indica lo contrario. El autor sugiere que una condición importante que se tendría que dar para mejorar las estimaciones sobre la base del feedback provisto luego de la finalización de la(s) tarea(s), sería el uso de una estrategia explícita de evaluar la incertidumbre. Esta condición mejora mientras mayor es la cantidad de información histórica de la que se dispone.

Según [30], las estimaciones producidas por desarrolladores de software se ven afectadas no solamente por el esfuerzo. Destaca que la causa de las desviaciones es el nivel de interdependencia (haciendo énfasis en las relaciones, el contexto social y las interconexiones). Mientras más interdependiente es quien estima, más optimistas (bajas) son las estimaciones. Cabe aclarar que las desviaciones en las estimaciones se dan en todas las circunstancias.

Varios estudios [31] han revelado que las estimaciones de esfuerzo pueden verse altamente afectadas por información irrelevante y desconcertante. Por ejemplo: el poco esfuerzo dedicado al desarrollo del sistema que se está migrando; información sobre las expectativas irreales de bajo costo de los clientes; restricciones en cuanto a un periodo corto de desarrollo, etc. El autor aconseja evitar estos efectos eliminando o neutralizando esta información irrelevante en la especificación de requerimientos antes de ser conocida por los estimadores. Es difícil volver a los desarrolladores al estado mental anterior luego de haber sido expuesto a esta tipo de información.

En el formato de la ingeniería de software se suelen usar Analogías para realizar las estimaciones de desfuerzo. Éste es el proceso por el cual se estima un proyecto de desarrollo de software tomando como referencia las estimaciones realizadas en un proyecto similar llevado a cabo anteriormente. Si dicho proyecto referente tiene una productividad muy alta o muy baja, entonces se tienen que realizar ajustes. Uno de los ajustes recomendados por Jørgensen es el RTM (Regression Towards the Mean), Regresion hacia la Media. Se ha verificado que este enfoque mejora la precisión de las estimaciones [38]. Además, un análisis realizado sobre varios proyectos de desarrollo de software y mantenimiento ha demostrado que las estimaciones realizadas por expertos están, en cierta medida, ajustadas por RTM.

Las estimaciones suelen resultar problemáticas en ambientes basados en metodologías ágiles de desarrollo de software. Los pioneros del SCRUM consideran aceptable que la técnica de estimación "Planning Poker" arroje resultados con una tasa de error promedio de 20%. Sin embargo, deberían admitir que esto depende del nivel de madurez de los desarrolladores [43]. En el estudio "Diseño e implementación de un programa de medición para equipos

SCRUM", los autores indican que no solo esta tasa no se dio durante la investigación, si no que peor aún, los desarrolladores usaban las estimaciones como objetivos (tal como lo indica la Ley de Parkinson) y el desarrollo se vuelve estresante. En conclusión, la "Deuda Técnica" (Technical Debt) se acumula con cada sprint a una tasa muy grande y la motivación del equipo se ve afectada.

En síntesis, los estudios realizados por Jørgensen han resultado ser de gran valor para el desarrollo de la presente tesis, especialmente el artículo que escribe con Halkjelsvik [35]. "Los efectos de los formatos de requerimientos en las estimaciones de esfuerzo basadas en juicio de expertos" que, como veremos más adelante, es un punto crítico que resulta ser una parte importante de nuestra propuesta.

El foco lo hemos puesto en el tamaño de los proyectos ya que éstos al ser pequeños presentan características que los exponen a caer en el caos en el proceso de desarrollo, a la incertidumbre y a la imprevisibilidad [50], lo cual motiva nuestro estudio.

Según Russ et.al. [57], entre las características y factores del entorno que determinan esta clasificación de tamaño podemos encontrar:

- Madurez y tamaño de la organización de desarrollo: en organizaciones donde se desarrollan cientos de proyectos, probablemente tengan procesos de soporte ya establecidos. En general, la organización que desarrolla solo unos pocos proyectos, carecen de procesos de soporte. Esto también da un índice de madurez de dicha organización.

- La complejidad del proyecto: la manera de determinar la complejidad de un proyecto viene dada por lo sofisticado que puede ser el conocimiento del dominio requerido para el desarrollo de dicho proyecto. La clasificación puede ir desde procesos simples de negocio hasta aplicaciones en tiempo real. Mientras más complejo es el dominio, mayor es la necesidad de estructurar formalmente las actividades del proyecto.

- Los atributos de calidad: los proyectos pequeños suelen tener menos código, lo que significa que pueden alcanzar las metas de calidad más fácilmente

- Las interacciones del personal: los proyectos pequeños en general tiene la ventaja de que al tener un personal más reducido, de manera que las interacciones son usualmente menores y más informales. También, los proyectos de estas características suelen tener una estructura gerencial mucho más chata, lo cual hace que el proyecto pueda reaccionar más rápidamente ante ciertos problemas que requieran la intervención de la gerencia.

Típicamente, este tipo de proyectos incluyen actividades con presiones de tiempo y clientes que reaccionan solo cuando detectan la necesidad de una solución mediante un sistema de software. Por dicha razón, estos proyectos presentan un alto grado de volatilidad en los requerimientos [50] [58]. Al no haber elicitado adecuadamente los requerimientos desde el comienzo, hace que se produzcan permanentes cambios en los mismos. Los clientes sienten

que pueden cambiar dichos requerimientos sin asumir el costo ya que los desarrolladores no entienden lo que los clientes quieren [50]

Dado este escenario "reactivo" y la falta de reglas claras para enfrentar el proceso de desarrollo, lleva a los desarrolladores a seguir procesos "ad-hoc" [50] De esta manera estos proyectos no pueden asegurar el esfuerzo que les demandará dicho desarrollo.

La duración del proyecto y la funcionalidad inicial suelen estar establecidas pero la estimación del esfuerzo se reduce a un problema de dinero. El tiempo disponible para desarrollar el proyecto está directamente relacionado con la urgencia del cliente de tener el producto [51] y del dinero disponible.

Este tipo de proyectos cuya duración va de 1-6 meses se ha visto que se dan mucho en América Latina [50]. Nuestro estudio confirma dicho aspecto y lo amplía hacia las multinacionales con sucursales en América Latina. Como sucursales que se desempeñan bajo el formato de "outsourcing" observamos que es muy común recibir proyectos pequeños quedando los proyectos de mayor envergadura para ser desarrollados en casa central.

3. Marco teórico de los métodos de estimación más utilizados en la industria. Combinación de Juicio de Expertos

Se estudiaron los métodos de estimación basados en el Juicio de Expertos más usados en la industria del software en la actualidad [25][39] junto con Delphi y Wideband Delphi [56]. Se vieron los métodos de estimación usados en las metodologías de desarrollo de software ágiles: Planing Poker [47].

3.1. Juicio de Expertos

Según un estudio realizado por [39], el método de estimación dominante es Juicio de Expertos. Los autores sostienen que no hay suficiente evidencia disponible que sugiera que las estimaciones mejoran con el uso de métodos formales de estimación.

3.1.1. Evidencias de aplicación en empresas

Según el estudio de revisión de casos de estudio sobre estimación experta realizado por [25], el juicio de expertos es el método de estimación usado más ampliamente no solo en el área de desarrollo de software sino también en otras áreas tales como negocios, salud, educación, etc. [25] presenta los siguientes ejemplos entre otros: según el estudio realizado en Jet Propulsion Laboratory reportado en [20], encontraron que el 83% de los estimadores aplicaron "analogía informal" como estrategia primaria de estimación, el 4% usaron "analogía formal" (definida como juicio de expertos basado en proyectos documentados), el 6% usó "normas generales2" y el 7% usó modelos formales. Otro ejemplo es una investigación realizada en compañías holandesas descriptas en [19] concluye que el 62% de las organizaciones que desarrollan software basan sus estimaciones en "intuición y experiencia" y solo el 16% sobre modelos formales de estimación. [24] reporta que el 84% de las estimaciones de esfuerzo de desarrollo de software realizadas en una gran compañía de Telecom, se ba-

[2] Traducción del inglés de "rules of thumb"

saron en juicio de expertos. [41] asegura que el 72% de las estimaciones realizadas en proyectos en una compañía de desarrollo de software se basaron en juicio de expertos. [25] afirma que en su estudio de revisión no lograron encontrar ningún estudio que indicara que la mayoría de las estimaciones realizadas se basaran en métodos formales de estimación.

3.1.2. Descripción del método

Según [25] para poder categorizar una estimación como "experta" debe darse que dicha estimación sea realizada por una persona reconocida como experta en dicha tarea y que gran parte del proceso que se siguió para llegar a esa estimación esté basado en un proceso de razonamiento no explícito ni recuperable, es decir está basado en la "intuición". Tal como se reportó en un estudio de predicciones de negocios [6], la mayoría de los métodos de estimación tienen ambos elementos: la intuición y el razonamiento explícito. De hecho, aún los métodos formales de estimación de esfuerzo en el desarrollo de sistemas de software necesitan del juicio de expertos como parámetro de entrada [52].

Work Breakdown Structure

Un enfoque usado ampliamente para dar soporte a los procesos de estimación es la WBS (Work Breakdown Structure[3]) [65] que consiste en dividir el proyecto en paquetes discretos más pequeños que resultan más fáciles de estimar. Nos permite ver el trabajo requerido para completar el proyecto. Según el PMBOK® (Project Management Body of Knowledge) [1] la WBS se puede usar para descomponer efectivamente el alcance del proyecto, para mejorar las estimaciones, para controlar mejor la ejecución del proyecto y para verificar más precisamente la finalización del proyecto. Además este enfoque resume la información del proyecto y permite contar con información histórica, lo cual resulta beneficioso en términos de precisión y velocidad para proyectos futuros.

Este enfoque se utiliza para aplicar las estrategias de descomposición Top-Down y Bottom-Up donde los expertos estiman el esfuerzo tanto en paquetes pequeños como en paquetes de más alto nivel.

Estimación Top-Down

Esta estrategia de estimación experta es usada durante la fase de iniciación/planificación de los proyectos. Al comienzo del proyecto, cuando aún no se conoce mucho sobre los requerimientos del mismo, se puede utilizar la WBS a muy alto nivel. Esta estrategia también suele llamarse Top-Down Analogy-based [25] ya que se usa información histórica de proyectos anteriores similares que se pueden usar como punto de partida para las estimaciones de esfuerzo.

Estimación Bottom-Up

Esta estrategia de estimación experta es también conocida como descomposición aditiva. Aquí se aplica mucho mejor el enfoque de WBS a bajo nivel, descomponiendo el proyecto

[3] Despiece de tareas

en paquetes lo más pequeños posible. Este enfoque lleva mucho más tiempo pero también resulta ser más preciso que el Top-Down.

Los 12 principios de la Estimación Experta

A continuación se presentan los 12 principios de estimación expertas validados empíricamente por [25].

Los siguientes seis principios favorecen a la reducción de la influencia humana y situacional:

1) Evaluar la precisión en las estimaciones pero a la vez evitar la alta presión de evaluación.

   Jørgensen sostiene que la precisión en las estimaciones tienen que formar parte de los criterios de evaluación de los proyectos pero aconseja que las fuertes presiones sobre la responsabilidad que los estimadores tiene sobre la precisión de las estimaciones así como también el sistema beneficio/castigo sean evitados.

   Los proyectos pequeños se ven fuertemente impactados por este principio dado que al ser pequeños también suelen conllevan una alta presión en términos de tiempo y dinero.

2) Evitar metas de estimación en conflicto

   Las metas en conflicto se dan cuando el proceso de estimación se ve impactado por otras metas (o evaluaciones) diferentes de la meta de precisión en sí. Jørgensen recomienda que los profesionales de software aprendan a identificar cuando una persona tiene un profundo interés por los resultados en cuyo caso no se puede esperar a que la persona produzca estimaciones realistas, aun cuando esta persona sea la que más experiencia tenga.

   En este caso, en los proyectos pequeños, este principio puede ser más perceptible como así también puede ser más fácilmente pasado por alto dada la cantidad reducida de stakeholders que presenta.

3) Pedir a los estimadores que justifiquen y critiquen sus estimaciones

   La confianza que los estimadores tengan sobre sus estimaciones depende más de la cantidad de tiempo que hayan pasado trabajando en las mismas que en la precisión de las estimaciones. Según Jørgensen la justificación y la crítica de las estimaciones propias puede tener varias ventajas, entre ellas:

   - Incrementar la precisión de las estimaciones, particularmente en situaciones donde la incertidumbre es alta
   - Llevar a un proceso de estimación más analítico y reducir el riesgo de usar estrategias de estimación demasiado simples
   - Mejorar el nivel de confianza en las estimaciones

- Mejorar la compensación ante la falta de información.

En este principio lo vemos implementado como uno de los principios básicos del Planning Poker que veremos más adelante.

4) Evitar información de estimación irrelevante y poco confiable

Este parece ser un punto de fácil entendimiento pero según Jørgensen ha habido numerosos estudios que sugieren que no siempre es el caso, que las estimaciones expertas se vean fuertemente impactadas por información irrelevante, aún cuando el estimador sabe que la información es irrelevante. En este caso, se recomienda cambiar de estimador. Se debería buscar un nuevo estimador que no conozca dicho trasfondo.

Este principio encuentra un sustento muy importante en el ámbito de proyectos pequeños ya que dado el acotado alcance y tiempos que tienen, se debe tratar de evitar por todos los medios este tipo de información y/o poco confiable. También debemos tratar de evitar análisis parálisis que suele ser causado por este tipo de información irrelevante.

5) Usar datos documentados de tareas de desarrollo previas

Según Jørgensen, el uso de documentación previa significa que los estimadores expertos tienen la oportunidad de aplicar una estrategia de estimación más analítica y consecuentemente menos propensa a influencias humanas y situacionales. Los beneficios derivados del uso de datos de proyectos de software documentados fue reportado por [46], quienes sostuvieron que los desfasajes en los costos de los proyectos estaban relacionados con la falta de documentación de proyectos anteriores, es decir que había una gran dependencia de la memoria del personal.

Jørgensen considera que los beneficios potenciales del uso de documentación son similares a los potenciales beneficios del uso de modelos formales de estimación, es decir, se evitan la falta de precisión de las estimaciones y se reduce la influencia tendenciosa humana.

Este principio es altamente aplicable en proyectos pequeños; así fue como se obtuvieron los datos de base del presente caso de estudio.

6) Buscar estimadores expertos con conocimientos relevantes del dominio y buenos antecedentes de estimaciones.

Existe un supuesto subyacente sobre la selección de los estimadores expertos que dice que las personas más competentes para resolver la tarea son quienes deberían estimarlas. Aunque esta suposición puede ser válida según [59], Jørgensen sostiene que es necesario hacer algunos ajustes a la misma:

- La relevancia de la experiencia algunas veces es acotada, es decir solo aplicable a situaciones similares

- Jørgensen et.al. [40] reportaron que quienes realizaban mantenimiento de software y tenían experiencia específica en la aplicación; si bien tenían menos problemas de mantenimiento, esto no significaba que podían predecir su trabajo de manera más acertada.
- Klayman et.al. [42] reportaron que la gente se torna demasiado confiada en sus estimaciones cuando reciben un set de tareas a estimar más difíciles que las que reciben a menudo.
- Stone et.al. [64] reportaron que ser competentes a la hora de estimar no es lo mismo que tener habilidad para darse cuenta de la incertidumbre de una estimación.

Los siguientes cuatro principios dan soporte al proceso de estimación:

7) Estimar usando la estrategia de descomposición Top-Down y la de descomposición aditiva Bottom-Up independientemente

   Jørgensen recomienda en su estudio, que el enfoque Bottom-up (donde obtenemos las estimaciones de las actividades de la WBS a bajo nivel) se combine con el enfoque Top-Down (donde se estima el proyecto a alto nivel, comparándolo con otros proyectos históricos similares). Considera que estos enfoques se tienen que implementar de manera independiente para evitar el efecto de anclaje[4], es decir, que unas estimaciones se vean altamente influenciadas por las otras [40], especialmente por las de proyectos anteriores.

   El enfoque Bottom-Up ayuda a mejorar las estimaciones si la incertidumbre de la tarea completa es alta, es decir, que la tarea es demasiado compleja para estimarla como un todo de manera que la descomposición activa el conocimiento relevante. La validez de estas dos condiciones es típicamente imposible de conocer de antemano, de manera que aplicar ambos procesos de estimación top-down y bottom-up reduce el riesgo de estimaciones altamente inexactas.

   Este principio puede ser fácilmente aplicado en proyectos pequeños dado el acotado alcance de los mismos.

8) Usar checklists de estimaciones

   Según Jørgensen, los beneficios de usar check-lists están basados en cuatro observaciones:

   - Los expertos usualmente olvidan actividades y subestiman el esfuerzo requerido para resolver eventos inesperados.

   - Las estimaciones de expertos son inconsistentes, es decir, que para la misma entrada pueden resultar distintas estimaciones. Los checklists

[4] Efecto de anclaje: "tendencia de las estimaciones de expertos de ser influenciadas cuando comienzan con una estimación 'conveniente'. Esta estimación inicial (o ancla) puede estar basada en tradición, historia previa o datos disponibles" [3]

pueden incrementar la consistencia y por ende la precisión de las estimaciones expertas.

- La gente tiende a usar estrategias de estimación que requieren un mínimo de esfuerzo computacional a expensas de la precisión. Los checklists pueden llevar a los expertos a usar una estrategia de estimación más precisa.

- La gente tiene la tendencia a considerar solo las opciones que fueron presentadas y a subestimar la probabilidad de otras opciones.

Este principio quizás provea mayores beneficios en proyectos grandes, pero esto no quita que los proyectos pequeños no se puedan ver beneficiados por el uso de estos checklists. Creemos que en estos casos sería altamente beneficioso para tareas/eventos que creemos que se pueden volver a dar en el futuro.

9) Combinar estimaciones de diferentes expertos y estrategias de estimación

Según Jørgensen, hay muchas alternativas para combinar las estimaciones. El líder de proyecto puede, por ejemplo, recolectar las estimaciones de una misma tarea de diferentes expertos y luego ponderar dichas estimaciones de acuerdo con el nivel de competencia de los expertos. Otra opción es que el líder de proyecto pida a los diferentes expertos que discutan sus estimaciones y se pongan de acuerdo en un valor final de estimación. La elección de la estrategia de combinación y los beneficios de las estimaciones combinadas dependen de un número de variables: 1) el número de expertos; 2) la precisión de las estimaciones de cada individuo; 3) el grado de influencia entre los expertos y 4) la inter-correlación entre las estimaciones de los expertos.

Lo más importante es combinar estimaciones de diferentes orígenes, (preferentemente con alta precisión y baja inter-correlación) y no precisamente cómo se lleva a cabo dicha combinación.

Este principio es difícil de aplicar en proyectos pequeños pero no imposible. Según nuestra experiencia, aún cuando existe un solo desarrollador, su experiencia puede combinarse con la experiencia de líder de proyectos y llegar así entre ambos a una estimación en común. De todas maneras, en un caso como el expresado aquí, nosotros preferimos tomar las estimaciones del desarrollador quien es el que va a llevar la tarea a cabo. En caso de que las estimaciones del desarrollador difieran ampliamente con las estimaciones del líder de proyecto, se solicita la justificación a dichos valores.

En este principio también lo vemos implementado como uno de los principios básicos del Planning Poker.

10) Evaluar la incertidumbre en las estimaciones

Una evaluación de la incertidumbre es importante para aprender de las estimaciones ya que una precisión baja en las estimaciones no es necesariamente un

indicador de una baja habilidad en las estimaciones cuando el proyecto presenta un alto grado de incertidumbre.

Jørgensen recomienda que la incertidumbre de una estimación sea evaluada a través de un intervalo de predicción. Por ejemplo: el líder de proyecto puede estimar que el esfuerzo de desarrollo será de 10000 horas persona y que hay un 90% de certeza (nivel de confianza) de que el esfuerzo real caerá entre 5000 y 20000 horas persona. Luego el intervalo de horas persona [5000, 20000] es el intervalo de predicción del 90% de la estimación del esfuerzo de 10000 horas hombre.

Existe un proceso para elicitar la incertidumbre propuesto por [38].

1) Estimar el esfuerzo

2) Calcular el mínimo y el máximo esfuerzo como proporciones fijas del esfuerzo. Por ejemplo se pueden basar dichas proporciones poniendo un mínimo de 50% y un máximo de 200% del esfuerzo [49]

3) Decidir el nivel de confianza, es decir, evaluar la probabilidad de que el esfuerzo caiga entre el esfuerzo mínimo y el máximo

Una alternativa al método de elicitación que todavía no ha sido evaluado en contextos de software, es pedir intervalos de predicción basados en niveles bajos de confianza, por ejemplo: pedirle a un desarrollador de software que provea un intervalo de predicción de un 60% en vez de un 90%. Esto puede reducir los niveles de sobreestimación.

Este principio es altamente aplicable no solo para grandes proyectos si no para pequeños proyectos también. En los pequeños proyectos suelen darse situaciones donde el tiempo con el que se cuenta para la estimación es muy corto y la incertidumbre es mucha.

Estos últimos dos principios están orientados a proveer feedback y oportunidades de entrenamiento:

11) Proveer feedback sobre la precisión en las estimaciones y las relaciones entre tareas

El problema con la mayor parte del feedback que se da sobre las estimaciones del esfuerzo de desarrollo de software es que toma mucho tiempo entre el punto de estimación hasta el punto en el que se brinda el feedback. Esto es desafortunado ya que se ha demostrado que el feedback inmediato mejora fuertemente el aprendizaje sobre las estimaciones y la precisión.

Este principio es fácilmente aplicable en el caso de proyectos pequeños dado que no debería haber pasado más de 6 meses entre el “punto de estimación” y el “punto de feedback” del que hablaremos más abajo

12) Proveer oportunidades de entrenamiento en estimaciones

Jørgensen sugiere que las compañías de software provean oportunidades de entrenamiento a través de sus bases de proyectos completos. Una sesión de entrenamiento debería incluir estimaciones de proyectos completos basados en la información disponible al momento de la estimación ("punto de estimación") aplicando diferentes procesos de estimación. Las ventajas de este tipo de entrenamientos son:

- El feedback individual puede ser recibido justo después de que se completaron las estimaciones.
- El efecto de no aplicar los checklists y otras herramientas de estimación puede ser investigado sobre los métodos propios de estimación.
- La validez de la experiencia propia en estimaciones puede ser examinada en diferentes tipos de proyectos.
- Las razones para olvidar actividades o subestimar riesgos puede ser analizada inmediatamente, mientras que el sesgo de la retrospectiva es débil.
- La tendencia al exceso de confianza puede entenderse con el coaching y entrenamiento adecuados.

En este principio vemos la misma aplicabilidad para proyectos pequeños tal como describimos en el principio anterior.

3.1.3. Juicio de Expertos vs Modelos formales de estimación

Existen varias razones por las cuales el uso de los modelos formales de estimación del esfuerzo de desarrollo de software es bajo. [25] explica algunas de dichas razones, por ejemplo: las organizaciones no se sienten cómodas usando modelos que no entienden completamente; no existe suficiente evidencia que indique que el uso de modelos formales producirá estimaciones más precisas que lo que genera el juicio de expertos.

En su estudio, [25] detalla sus hallazgos más importantes como una guía para saber cuándo podemos esperar que las estimaciones expertas resulten ser más precisas que las obtenidas a partir de modelos formales. Para llegar a los siguientes puntos se encontró mucha evidencia de soporte:

- Las estimaciones expertas son más precisas que los modelos de estimaciones cuando los expertos poseen (y aplican eficientemente) conocimiento del dominio no incluido en los modelos. Los modelos de estimaciones son más precisos cuando el experto no posee (o aplica eficientemente) conocimiento del dominio no incluido en los modelos.
- Los expertos usan estrategias de estimación simples (heurísticas) que funcionan tan bien o mejor que los modelos de estimación cuando estas estrategias simples de es-

timación (heurísticas) son válidas. De otra manera, las estrategias pueden llevar a estimaciones sesgadas.

- Los expertos pueden estar fuertemente influenciados y confundidos por información irrelevante, por ejemplo: optimismo excesivo. Los modelos de estimación son más imparciales.

Para llegar al siguiente último punto, se encontró cierta evidencia de soporte:

- Las estimaciones expertas son más precisas que los modelos de estimaciones cuando el nivel de incertidumbre es bajo. Los modelos de estimaciones son más precisos cuando la incertidumbre es alta, por ejemplo, cuando el proyecto es mucho más grande que los proyectos anteriores.

3.2. Delphi

Delphi es la fonética inglesa de Delfos y su intención es eludir al "Oráculo de Delfos". La técnica Delphi fue creado para ayudar a aunar las opiniones de los expertos y obtener consenso. Esto se logra a través de una serie de cuestionarios intercalados con realimentación controlada de opiniones [13].

3.2.1. Evidencias de aplicación en empresas

Fue desarrollada por la Corporación RAND y surgió a partir de la necesidad de aunar opiniones de expertos en un proyecto de las Fuerzas Armadas de los Estados Unidos [55]

Desde 1950, Delphi pasó a ser utilizada por una amplia variedad de áreas en numerosos países. Sus aplicaciones se extendieron desde la predicción de un amplio rango de tendencias en ciencia y tecnología hasta aplicaciones que van desde políticas de formación hasta toma de decisiones.

Un examen realizado sobre la literatura que hay relacionada a este tema revela el uso de Delphi en áreas tales como: la industria de la salud, educación, sistemas de información, transporte e ingeniería.

Este crecimiento explosivo del interés por Delphi preocupó a algunos autores como Linstone y Turoff ya que éstos consideran que se había puesto mucho énfasis en estudios de la aplicabilidad en vez de otorgarle la importancia que merece la evaluación del método en sí. Sostienen que hasta parece incompatible en cuanto a la cantidad limitada de experimentación controlada o investigación académica existente, exceptuando los estudios realizados por RAND.

La gran mayoría de los artículos que se encontraron sobre la aplicación de Delphi son de distintas disciplinas que no se relacionan con la Ingeniería de Software.

3.2.2. Descripción del método

Esta técnica es útil en situaciones donde se necesita aprovechar y combinar las opiniones individuales para hacer frente a la falta de acuerdo o falta de conocimiento [14].

No hemos encontrado estudios de aplicación de Delphi en estimaciones de proyectos de desarrollo de software específicamente. Según [56] la literatura tiene cientos de papers sobre los procedimientos Delphi pero la mayoría tienen que ver con las aplicaciones en donde Delphi es solo usada como una herramienta para acumular juicio de expertos focalizándose en el juicio final o estimación. Las evidencias de evaluaciones experimentales son escazas. A pesar de que muchas investigaciones recientes han sido conducidas usando procedimientos experimentales estándares, hay poca evidencia acumulada acerca de cómo aplicar Delphi y cuándo usarla. Dado que esta técnica es usada ampliamente en una gran variedad de disciplinas, tales como planeamiento corporativo, salud, educación y crecimiento urbanístico [13], explicaremos a nivel general el método de aplicación de la misma. Es importante aclarar que este método sirve como base para la aplicación de un método mucho más utilizado, que explicaremos más adelante, Wideband Delphi, de cual sí hemos encontrado estudios de aplicación del mismo en estimaciones de proyectos de desarrollo de software.

Para que la aplicación del método sea más efectiva es necesario que se garanticen: el anonimato, las iteraciones o rondas, la retroalimentación controlada y la respuesta estadística del grupo [56]

El anonimato se logra a través del uso de cuestionarios permitiendo que los miembros del grupo expresen sus opiniones y juicios sin modo alguno de identificación de la identidad del opinante. Esto logra que se reduzcan los efectos de la presión social de parte de individuos dominantes o de la mayoría.

A través de las rondas de cuestionarios se le da la oportunidad a los panelistas de que cambien de idea sin miedo a perder prestigio frente al grupo.

Primero se explica el proyecto y la actividad al grupo de estimadores. Luego se le pide a cada experto que haga su mejor estimación de cuánto puede durar la actividad individualmente sin consultar a los otros participantes. Los resultados son tabulados y presentados al grupo en un histograma denominado “Primera Pasada” como se muestra en la **Figura 1**. A aquellos participantes cuyas estimaciones cayeron en los cuartiles externos, se les pide que expliquen las razones de sus estimaciones. Entre cada iteración, el facilitador informa al grupo las opiniones de sus colegas anónimos. Esta información contiene las opiniones de todos los miembros del grupo no solo de los más elocuentes o vocales.

Luego de escuchar las argumentaciones, se le pide a cada miembro que estime nuevamente, ahora ya sabiendo cuáles son las estimaciones de los demás estimadores. No hay lugar a discusiones. Se supone que en esta nueva ronda se va a estrechar la difusión de los números de la ronda anterior ya que algunas inquietudes son consideradas y analizadas más detenidamente. Los resultados son presentados y mostrados en un histograma denominado “Segunda Pasada” y otra vez se deben defender los cuartiles externos. Se lleva a cabo una tercera ronda de estimaciones y se arma un nuevo histograma llamado “Tercera Pasada”. Se

pueden realizar tantas rondas como se necesiten para estabilizar los resultados pero según [56] con tres sesiones ya se logra dicha estabilidad. El promedio de esta tercera ronda es lo que se toma como la "estimación del grupo".

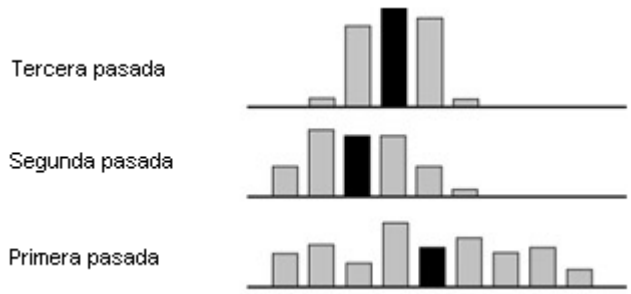

Figura 1 - Iteraciones o pasadas usadas en la aplicación de Delphi[5]

3.3. Wideband Delphi

Barry Boehm creó Wideband Delphi y fue popularizado en el libro de Boehm Software Engineering Economics 1998. Es una técnica de estimación de esfuerzo basada en el consenso logrado en un grupo de expertos. Deriva de la técnica Delphi. Fue llamada "Wideband" porque incluye mucha más interacción y comunicación entre los participantes distinto de la técnica Delphi original donde se evitan las discusiones de grupo [7][8]

3.3.1. Evidencias de aplicación en empresas

Boehm et.al.[8] solicitaron a un grupo de expertos profesionales de desarrollo de sistemas que estimaran los parámetros iniciales para los Factores de Ajuste del Esfuerzo que aparecen en el componente de estimaciones de esfuerzo del modelo de costos de integración COCOTS (COnstructive COTS).

Boehm et.al. [8] usaron esta técnica para estimar la introducción de defectos de software y las tasas de extracción durante las diferentes fases del ciclo de vida de desarrollo de software. Estos factores aparecen en COQUALMO (COnstructive QUALity MOdel) que predice la densidad de defecto residual en términos del número de defectos/unidad de medida. Además, estos dos autores aplicaron Wideband Delphi para especificar la información requerida para la calibración Bayesiana de COCOMO II.

Wiegers [72] aplicó el método para estimar el esfuerzo que le iba a demandar llevar a una organización a alcanzar el Nivel 2 de CMM.

Steve McConnell en su libro Sofware Estimation – Demystifying the Black Art 2006 realizó un experimento que llevó a cabo con 25 grupos de estimaciones. Estos grupos, primero estimaron promediando sus estimaciones y luego estimaron aplicando Wideband Delphi. McCon-

[5] [73]

nell muestra la tasa de error que obtuvo al comparar las estimaciones iniciales promediadas de estos 25 grupos con las estimaciones obtenidas luego de aplicar Wideband Delphi (**Figura 2**). Su experiencia sugiere que la aplicación de Wideband Delphi reduce el error en las estimaciones en un promedio de 40 % aproximadamente.

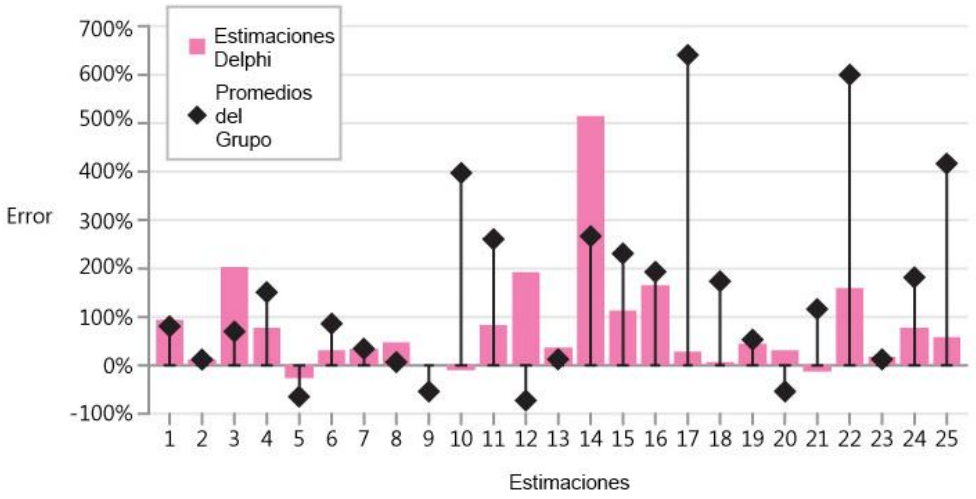

Figura 2 - Precisión de las estimaciones de un simple promedio comparado con la estimación Wideband Delphi. Este último reduce el error en aproximadamente dos tercios de los casos[6]

De los 10 grupos con los que trabajó que produjeron las peores estimaciones iniciales, Wideband Delphi mejoró la precisión de las estimaciones en 8 de 10 de los casos con un promedio de reducción del error del 60% (**Figura 3**)

[6] [48]

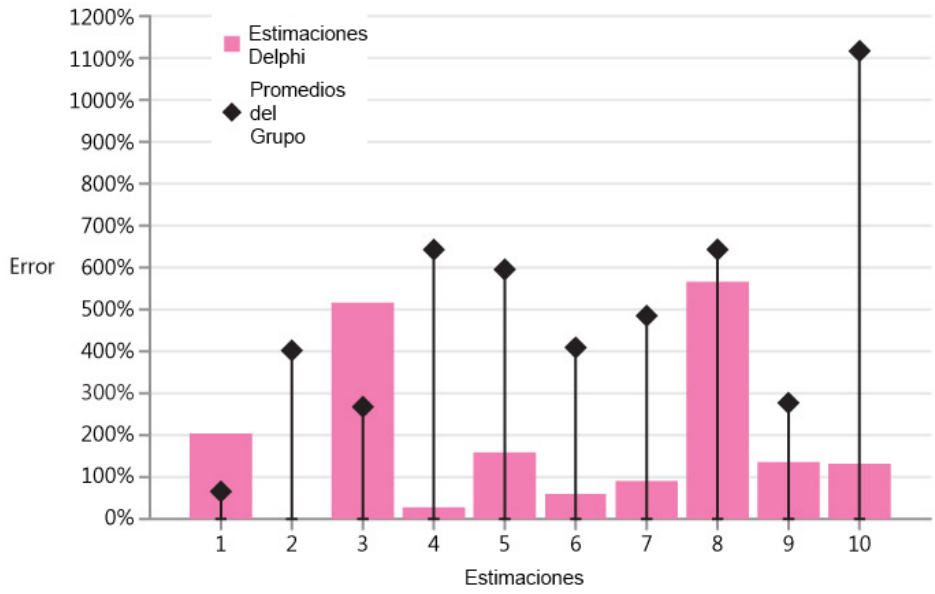

Figura 3 - Wideband Delphi vs las peores estimaciones. En este conjunto de datos, Wideband Delphi mejoró las estimaciones en 8 de los 10 casos[7]

Según las experiencias de [48] las técnicas de estimación que promedian las estimaciones individuales, confían en que la respuesta correcta va a estar dentro del rango con que se maneja el grupo de estimadores. La experiencia del autor con Wideband Delphi indica lo contrario. El 20% de los rangos de estimaciones iniciales de los grupos, no incluyó la respuesta correcta; lo que implica que promediar estimaciones individuales posiblemente puede no producir resultados precisos.

El fenómeno más interesante asociado con Wideband Delphi es que un tercio de los grupos cuyos rangos iniciales no incluían la respuesta correcta al final establecieron una estimación que se encontró afuera de su rango inicial y más cercano a la respuesta correcta (**Figura 4**). En estos casos, Wideband Delphi resultó con una estimación mejor que la mejor de las estimaciones individuales y ninguno de los grupos estableció una estimación final peor que la más mala de todas las estimaciones.

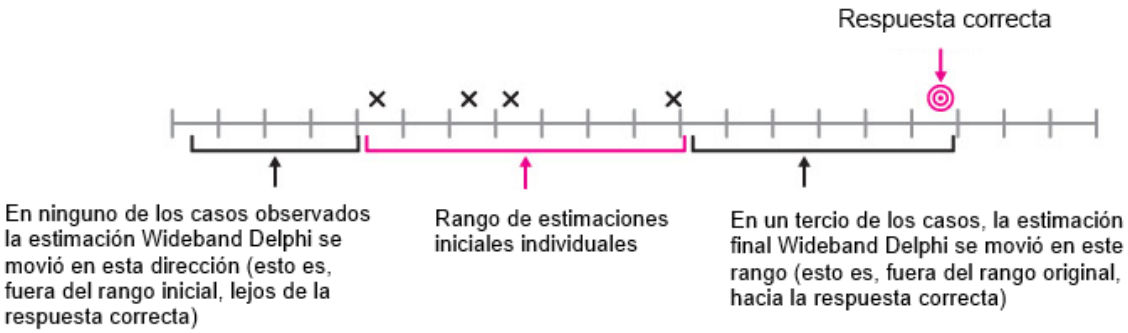

Figura 4 - En un tercio de los casos, Wideband Delphi ayudó a los grupos que no hab-

[7] [48]

ían incluido la respuesta correcta, a moverse fuera del rango inicial y más cerca de la respuesta correcta[8]

3.3.2. Descripción del método

Fases del proceso según Wiegers *[72]* (**Figura 5**)

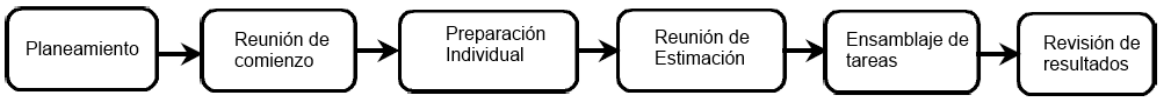

Figura 5 - Proceso de estimación Wideband Delphi[9]

Pasos del método adaptado de Boehm *[1998]*

1- El coordinador le presenta a cada experto la especificación de la actividad/tarea y un formulario de estimación.

2- Los estimadores preparan sus estimaciones individualmente (este paso se podría realizar luego del paso 3)

3- El coordinador llama a reunión donde los expertos discuten los problemas de estimación relacionados con el proyecto.

4- Los expertos llenan los formularios (**Figura 6**) anónimamente y se los entregan al coordinador.

5- El coordinador prepara un resumen de las estimaciones en un formulario por iteración y lo distribuye entre los estimadores para que vean como sus estimaciones difieren de las estimaciones de los otros estimadores.

6- El coordinador llama a reunión donde se focalizará en tener a los expertos discutiendo los puntos donde las estimaciones variaron ampliamente.

7- Los expertos votan anónimamente si aceptan el promedio de estimación. Si algunos de los estimadores vota que no, entonces se vuelve al paso 3.

8- La estimación final es un solo valor que se obtiene del ejercicio de Delphi.

Los pasos del 3 al 7 se pueden realizar personalmente, en una reunión de grupo, o electrónicamente vía email, chat, etc. Esto asegura la anonimidad. Dichos pasos se pueden realizar

[8] [48]

[9] [72]

inmediatamente o por partes. Todo dependerá de la criticidad de la situación y de la disponibilidad de los estimadores.

Tarea	Estimación #1	Cambio #1	Cambio #2	Cambio #3	Final
Cambio	--				--
Total					

Figura 6 - Formulario de estimación[10]

El formulario tiene que tener un rango de 1 a 20 meses. El rango que se muestra inicialmente debe ser tres veces más grandes que el rango que se espera que los estimadores usen para que no se sientan restringidos a un rango predefinido (**Figura 7**).

Figura 7 - Esfuerzo estimados[11]

El coordinador debe prevenir que las personalidades dominantes influyan en las estimaciones. Los desarrolladores de software no suelen tener personalidades asertivas y quizás las personas más reservadas son quienes tienen las mejores ideas.

Es muy útil mantener siempre la misma escala en todos los formularios que se muestran a los estimadores para que éstos vean cómo sus estimaciones van convergiendo y en algunos casos divergiendo en las diferentes rondas (**Figura 8**).

[10] [72]

[11] [48]

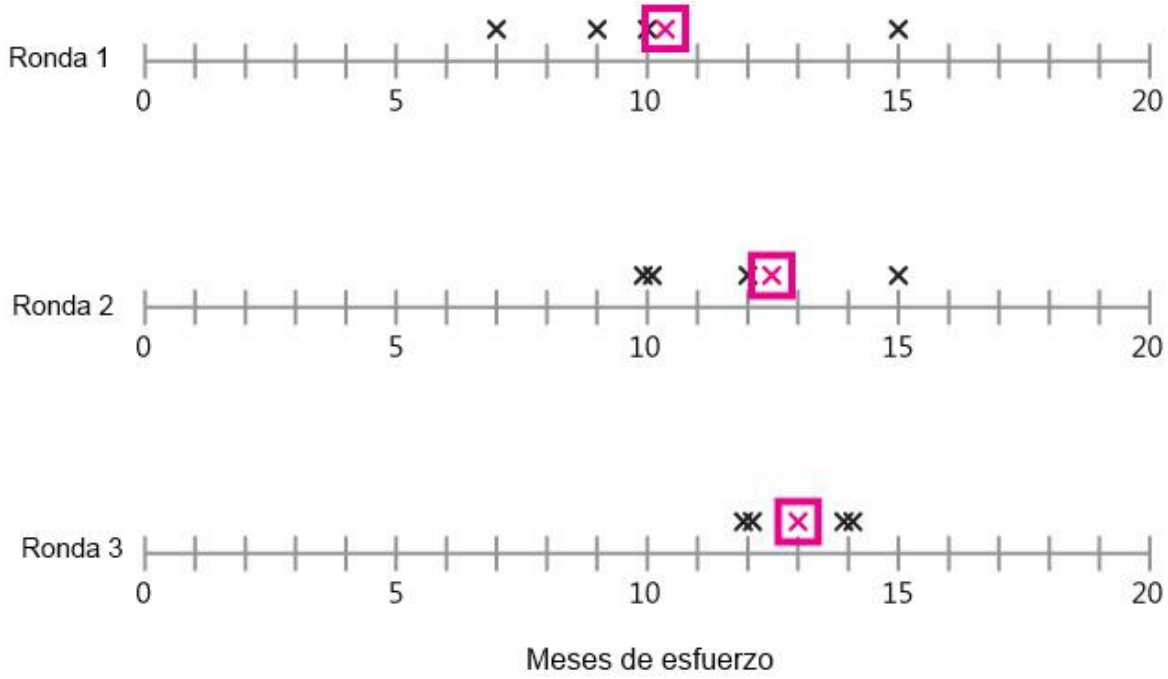

Figura 8 - Convergencia de esfuerzo estimado. En este caso, luego de la ronda 3, el grupo puede decidir quedarse con un rango de 12 a 14 meses de trabajo con un valor esperado de 13 meses de trabajo[12]

Cuándo usar Wideband Delphi

Aun cuando los datos de [48] aprueban el uso de esta técnica, otros estudios han encontrado que usando un simple promedio funciona mejor [38].

De todas maneras esta técnica requiere de una reunión lo que puede consumir mucho tiempo del equipo haciendo que ésta sea una manera cara de estimar. No es adecuada para estimar tareas detalladas [48]

Sí resulta útil para cuando se está estimando el trabajo en un área nueva de negocio, una nueva tecnología o se está trabajando para un nuevo producto de software. También resulta de utilidad para crear estimaciones de "órdenes de magnitud" durante la definición del producto o diseño de software antes de que se hayan determinado muchos de los requerimientos. También es útil si un proyecto que demande habilidades de diferentes especialidades, tales como: usabilidad poco común, complejidad algorítmica, performance excepcional, reglas de negocios complicadas, etc. Tiende a mejorar el alcance del trabajo y sirve para eliminar las diferentes suposiciones que se dan al estimar. En resumen, Wideband Delphi es muy útil para estimar aquellos proyectos con gran incertidumbre.

[12] [48]

3.4. Analogías

Según Shepperd et al. [61], la idea de usar analogías como base para estimar el esfuerzo en proyectos de software no es algo nuevo. [7] sugirió el uso informal de analogías como una técnica posible hace aproximadamente 30 años atrás. La idea fue reiterada por Cowderoy y Jenkins en 1988 pero nuevamente, sin un mecanismo formal de selección de analogías. El próximo desarrollo fue realizado por Vicinanza et al. [66][67][68] quien sugirió que los desarrollos de la comunidad de la máquina de aprendizaje en forma de CBR[13] podrían ser útilmente adaptados para ayudar a obtener mejores predicciones.

3.4.1. Evidencias de aplicación en empresas

Shepperd y Schofield [61] validaron este método usando el aplicativo ANGEL. La validación se realizó usando un total de 9 conjuntos de datos industriales diferentes con un total de 275 proyectos. En todos los casos Analogías arrojó mejores resultados que los métodos algorítmicos de regresión.

Atkinson y Shepperd realizaron un estudio sobre 21 proyectos reales donde demostraron que la aplicación de Analogías es como mínimo tan efectivo como Juicio de Expertos y/o Modelos Algorítmicos tradicionales derivados de Análisis de Regresión.

3.4.2. Descripción del método

Según [61], el principio de base es caracterizar proyectos en términos de características como por ejemplo: el número de interfaces, el método de desarrollo o el tamaño de los requerimientos funcionales. Existe una base de proyectos terminados que se usa para buscar aquellos que más se asemejen al proyecto que se quiere estimar.

Se dice [44] que el método de estimación por analogías es un ejemplo de la estrategia de CBR (Razonamiento Basado en Casos). CBR es una forma de razonamiento análogo donde los potenciales proyectos (en nuestro caso) análogos y el proyecto en cuestión (también llamado "target" en inglés) son ejemplos de una misma cosa, por ejemplo: proyectos de software [69].

CBR tiene cuatro aspectos:

- Caracterización de casos
- Almacenamiento de casos pasados
- Recuperación de casos similares para usar analogías
- Utilización del caso recuperado para resolver el caso del problema en cuestión conocido como caso de adaptación.

Se tienen "*p*" proyectos o casos, cada uno de los cuales tienen que ser caracterizados en términos de un set de "*n*" características. Se cuenta con una base de datos históricos de

[13] CBR del inglés Case Based Reasoning: Razonamiento Basado en Casos

otros proyectos ya finalizados. El nuevo proyecto a estimar es llamado "*target*". Dicho "*target"* es caracterizado en términos de las "*n"* dimensiones. Esto impone restricciones en el conjunto de características de manera que solo puede contener aquellas características para las cuales se conocerán sus valores al momento de realizar la predicción. El próximo paso es medir similitudes entre el "*target*" y los otros casos en el espacio n-dimensional [44].

La similitud puede ser definida de diversas formas pero la mayoría de los investigadores la definen como lo hacen [61][44], como la distancia Euclidiana en un espacio n-dimensional donde "*n"* es el número de características del proyecto. Se estandariza cada dimensión de modo tal que todas las dimensiones tengan el mismo peso. Los valores de esfuerzo conocidos del caso más cercano al nuevo proyecto son luego usados como base de predicción.

Una vez que el proyecto en cuestión llega a su fin, se guarda en el repositorio común de donde se sacó el proyecto que se usó para comparar.

Este proceso es automatizado usando herramientas de software tales como ANGEL. Según el estudio realizado por [44], las decisiones sobre cómo configurar un sistema CBR pueden tener un impacto sustancial en el nivel de precisión obtenido.

A su vez, los sistemas CBR forman parte de lo que se llaman máquinas de aprendizaje junto con Redes Neuronales y Reglas de Inducción. En su estudio, luego de comparar los tres métodos de predicción a través de tres factores: precisión, valor explicativo y configuración, llegaron a la conclusión de que las Redes Neuronales presentan una precisión superior frente a las Reglas de Inducción. Sin embargo esta técnica presenta problemas de configuración, de manera que en este caso, CBR se ve favorecido.

Jørgensen et. al. [37] encontraron que en general los modelos de estimación por Analogías actuales no incluyen ajustes relacionados a los análogos extremos. En su estudios, Jørgensen aplicó un enfoque de ajuste llamado Regresión hacia la Media (Regression Toward the Mean - RTM) y resultó en una mejora en la precisión de estimaciones. También realizaron un análisis sobre varios proyectos de desarrollo y mantenimiento de software industriales y llegaron a la conclusión que las estimaciones realizadas por expertos poseen una Regresión hacia la Media inherentemente.

3.5. Planning Poker

Este método fue descrito por primera vez por James Greening [16]y luego fue popularizado por Mike Cohn en su libro "Agile Estimating and Planning" [12]. Siendo esta una técnica relativamente nueva, solo se ha escrito sobre un estudio empírico sobre Planning Poker [18]

3.5.1. Evidencias de aplicación en empresas

Møløkken et. al. [48] realizaron un estudio donde aplicaron esta técnica de estimación en una empresa de software Noruega de tamaño medio que produce soluciones a medida para clientes del sector privado y público. Los autores comentaron que en algunos casos resulta ser más precisa que la combinación desestructurada de estimaciones en un grupo. Además,

la técnica resultó ser bien recibida por los integrantes del equipo quienes la encontraron muy útil a la hora de discutir estrategias de implementación de cada tarea así como también otorgó una mejor visión del trabajo de cada desarrollador.

Existe otro estudio [18] realizado en una empresa de desarrollo de software donde se estimaron 101 user stories de 4 proyectos, desarrollados por el mismo equipo usando metodologías ágiles. Dos proyectos fueron estimados usando Planning Poker mientras que los otros dos fueron estimados por medio de "grupo desestructurado". Dicho estudio sugiere que Planning Poker puede mejorar la eficiencia en las estimaciones sobre grupos desestructurados pero que posee las siguientes limitaciones:

1) Planning Poker es más preciso cuando el equipo cuenta con experiencia previa en tareas similares

2) Planning Poker incrementa los valores extremos.

Aparte de estas limitaciones el autor considera que usando Planning Poker para la estimación de user stories puede ayudar a obtener un planeamiento de entregables más realistas y obtener una asignación de recursos óptima.

3.5.2. Descripción del método

Es una técnica sencilla de estimación de esfuerzo de desarrollo de software basada en el consenso al que llega el equipo de expertos que está estimando. Es un enfoque que combina opiniones de expertos, analogías y desagregación que resulta en estimaciones rápidas y confiables [12]. Utilizada mayormente en el desarrollo ágil de software, en particular en Extreme Programming [18]

Resulta especialmente útil cuando las estimaciones están tomando mucho tiempo y cuando no se está involucrando al equipo completo [16]

El equipo de estimación está formado por todos los desarrolladores del equipo. Entendiéndose por desarrolladores a todos los programadores, testers, analistas, diseñadores, DBAs, etc. En proyectos ágiles no deberían exceder las diez personas [12]

A cada estimador se le entrega un mazo donde cada carta tiene escrito un número que representa una estimación válida, tales como: 0, 1, 2, 3, 5, 8, 13, 20, 40, and 100. Cada mazo se prepara antes de la reunión de Planning Poker [12]

La escala de estimación presentada más arriba tiene su razón de ser. Hay estudios que han demostrado que somos mejores estimando cosas que caen dentro de un rango de un orden de magnitud. [12] avala esta afirmación y sostiene que nuestras estimaciones además deberían caer dentro tales escalas. Una escala que da buenos resultados es la siguiente: 1, 2, 3, 5, and 8

Hay toda una lógica por detrás de cada escala. La primera es la Serie de Fibonacci. [12] considera que esta escala es una secuencia de estimación muy útil ya que las brechas entre los números se hacen lo suficientemente grandes a medida que crecen los números. Una brecha de un punto de 1 a 2 y de 2 a 3 parece ser adecuada, así como también lo es la brecha

de 3 a 5 y de 5 a 8. Esta secuencia no linear funcionan bien porque reflejan la gran incertidumbre asociada con grandes unidades de trabajo.

Hemos encontrado que existen sistemas simples diseñados en formato web que permiten llevar a cabo este tipo de reuniones on-line. De manera que se mantiene la privacidad de las estimaciones en la primera ronda y además permite que los estimadores asistan a la reunión con sus laptops, sin necesidad de usar cartas físicas, o mejor aún, facilita la aplicación de esta técnica en un Equipo Virtual.

Para cada historia (story/user story) a ser estimada, el moderador lee la descripción. Cualquiera puede ser moderador ya que no hay ningún privilegio asociado a ese rol. El Product Owner responde cualquier pregunta que los estimadores puedan tener.

Luego de que todas las preguntas han sido respondidas, cada estimador elige la carta que representa su estimación. Las cartas no se muestran hasta que todos los estimadores hayan hecho su elección. Luego, todas las cartas son dadas vuelta a la vez. Los estimadores que poseen las estimaciones más bajas y más altas deben explicar sus estimaciones.

El grupo discute la historia. Mientras tanto, el moderador toma notas que pueden resultar útiles a la hora de programar la tarea. Luego de la discusión se vuelve a estimar. Hasta que todas las estimaciones queden emparejadas. En general, las estimaciones convergen en la segunda ronda. Muy rara vez se llega a tres rondas o más.

Una de las preocupaciones al aplicar Planning Poker es que podrían no darse discusiones. Las estimaciones podrían carecer de sentido sin estas discusiones. Existe evidencia de que los equipos se vuelven rápidos al estimar funciones que resultan familiares. Esta rapidez tiende a disminuir cuando se tiene que lidiar con historias nuevas. Las discusiones parecen darse según sea necesario.

Discusiones

Algunas discusiones de diseño preliminares son apropiadas durante las estimaciones pero nos preguntamos cuánto es la cantidad correcta. Según [12] demasiada discusión llevará al equipo muy arriba de la curva de precisión/esfuerzo. Una manera de controlar esto dinámicamente mientras se realiza la sesión de Planning Poker es contar con un cronómetro y programar 2 minutos. Entonces cuando se cumplen los 2 minutos de discusión, se estima y se dan vuelta las cartas. El equipo puede focalizar sus energías en las diferencias de opiniones y no perder tiempo en lo que ya se está de acuerdo [16].

Sesiones

No es necesario aplicar Planning Poker con todos los miembros del equipo sino con parte del mismo. Esto es especialmente útil cuando hay muchas tareas por estimar. Si el equipo tiene más de 5 personas, entonces se podría dividir en 2 equipos más pequeños con, como mínimo, 3 personas, donde cada uno va a estimar por separado. Lo ideal es que todos los equipos estimen consistentemente. Por ejemplo, se debe acordar previamente qué se va a considerar, si 3 puntos son 3 horas o días, etc. Se debería tener una sesión todos juntos para asegurarnos que todos entienden y manejan los mismos parámetros [12].

En qué momento usar Planning Poker

Primero se necesitarán estimar una gran cantidad de tareas antes de que comience oficialmente el proyecto o durante sus primeras iteraciones.

Segundo, se necesitará estimar las tareas que vayan surgiendo en cada iteración a medida que avanza el proyecto. Una forma puede ser tener pequeñas reuniones para estimar al final de cada iteración. Esto sirve para ir midiendo estas tareas que van surgiendo y permite priorizarla para la próxima iteración en caso que fuera necesario incluirla [12].

Propiedades de Planning Poker que lo hacen adecuado para las estimaciones. Combinación de conocimiento experto

Según [12], Planning Poker reúne opiniones de múltiples expertos de manera que, al ser los éstos parte de un equipo multifuncional, están mucho mejor preparados para la estimación de tareas. [25] concluye que las personas más competentes para resolver la tarea son quienes deberían estimarla.

Impacto Social, precisión

Que las estimaciones sean reveladas simultáneamente hace que se reduzca el impacto social de comparación [9]. A su vez, cada estimador debe justificar sus estimaciones ante sus pares. De esta manera, todos participan, no solo los más elocuentes. Ahora todos los participantes son jugadores. Pueden surgir muy buenas ideas de la gente más callada del equipo. Las interacciones cara a cara hace que los participantes se sientan más comprometidos con las decisiones [47]

Precisión y optimismo

Según [12], se aprecia una mejora en la precisión de las estimaciones, especialmente en aquellos ítems donde la incertidumbre es grande. Otro punto importante que se da ante el hecho de tener que justificar las estimaciones, es que se logra compensar la falta de información.

Hay estudios que han demostrado que promediar estimaciones individuales lleva a conseguir mejores resultados [19], ayudan a reducir la tendencia hacia el optimismo.

Anclaje

Planning Poker reduce el problema potencial denominado “anclaje” al revelar de manera simultánea todas las estimaciones del grupo

3.6. Comparación de métodos de estimación basados en Juicio de Expertos

En la **Tabla 1** se muestra una comparación de métodos de estimación basados en Juicio de Expertos más usados en la industria del software en la actualidad estudiados en el presente trabajo.

Tabla 1 - Tabla comparativa de métodos de estimación basados en Juicio de Expertos

Métodos de Estimación	Descripción del método	Anclaje	¿Mejora la Precisión?	Tipos de proyectos en los que puede ser usado	Evidencias de aplicación en empresas	¿Los estimadores tienen la posibilidad de discutir y fundamentar sus estimaciones?	¿Se usan datos históricos de proyectos anteriores?	¿Acepta combinación con otras estrategias?	Autores destacados
Juicio de Expertos	Las estimaciones son realizadas por personas reconocidas como expertas en dicha tarea. Se hace uso de WBS bajo los enfoques Top-Down y Bottom-Up combinados.	Bajo. Es mucho menos imparcial que los modelos formales.	Sí	Grandes, medianos y pequeños. Esto es cierto si se usan los enfoques Top-Down y Bottom-Up combinados. De lo contrario, esto no se puede garantizar.	Método de estimación dominante, más ampliamente usado no solo en el área de desarrollo de software sino también en otras áreas tales como negocios, salud, educación, etc Ejemplos de casos de aplicación: Jet Propulsion Laboratory; compañías holandesas, una compañía de Telecom; otras compañías de desarrollo de software.	Sí	Sí	Sí	Jørgensen Shepperd

Delphi	Ayuda a aunar las opiniones de los expertos y obtener consenso. Esto se logra a través de una serie de cuestionarios intercalados con realimentación controlada de opiniones.	Nulo	No se encontró evidencia.	Grandes	No se encontraron evidencias de aplicación en empresas de desarrollo de software pero sí en otra áreas, tales como: la industria de la salud), educación, sistemas de información, transporte e ingeniería y las Fuerzas Armadas de los Estados Unidos.	Sí	No. Solo se accede a este tipo de información si el estimador estuvo involucrado en proyectos similares anteriores.	No	Rowe Wrigth Delbecq
Wideband Delphi	Técnica de estimación de esfuerzo basada en el consenso logrado en un grupo de expertos. Fue llamada "Wideband" porque incluye mucha más interacción y comunicación entre los participantes distinto de la técnica Delphi original donde se evitan las discusiones de grupo	Nulo	Sí	Grandes	Esta técnica fue usada para estimar la introducción de defectos de software y las tasas de extracción durante las diferentes fases del ciclo de vida de desarrollo de software. Fue aplicado para estimar el esfuerzo que iba a demandar llevar a una organización a alcanzar el Nivel 2 de CMM	Sí	No. Solo se accede a este tipo de información si el estimador estuvo involucrado en proyectos similares anteriores.	No	Boehm McConnell
Analogías	Caracterizar pro-	Alto	Sí	Grandes, medianos y	Método usando el	No	Sí	Sí	Shepper

	yectos en términos de características como por ejemplo: el número de interfaces, el método de desarrollo o el tamaño de los requerimientos funcionales. Existe una base de proyectos terminados que se usa para buscar aquellos que más se asemejen al proyecto que se quiere estimar			pequeños	aplicativo ANGEL Estudio realizado sobre 21 proyectos reales				d Schofield Boehm Cowdero y Jenkins Vicinanza
Planning Poker	Técnica sencilla de estimación de esfuerzo de desarrollo de software basada en el consenso al que llega el equipo de expertos que está estimando. Es un enfoque que combina opiniones de expertos y analogías.	Bajo	Sí	Grandes, medianos y pequeños.	Técnica aplicada en una empresa de software Noruega	Sí	No. Solo se accede a este tipo de información si el estimador estuvo involucrado en proyectos similares anteriores.	Sí	Greening Cohn

4. Descripción del problema

Una empresa necesita mejorar su método de estimación de esfuerzo de los proyectos de software. En las siguientes secciones se describirá el contexto del problema, las características de los proyectos involucrados en el problema y el problema propiamente dicho.

4.1. Contexto

La empresa donde se plantea el problema es una multinacional con sede en Argentina. Al momento del estudio la empresa contaba con 400 empleados. La empresa no pertenece al rubro del desarrollo de software pero cuenta con un área de IT que da soporte a sus actividades comerciales. Dentro del área de IT, existe un área de desarrollo y soporte de aplicaciones desarrolladas por dicha área[14].

4.2. Pequeños Proyectos de Software

La categorización de tamaño "Pequeño Proyecto" está determinada por características que ya expusimos en el apartado *2. Estado del arte*. En este apartado nos interesa poner el foco en el tema de tiempos. Este tipo de proyectos cuya duración va de 1-6 meses se ha visto que se dan mucho en América Latina [49]. Nuestro estudio confirma dicho aspecto y lo amplía hacia las multinacionales con sucursales en América Latina. Como sucursales que se desempeñan bajo el formato de "outsourcing" observamos que es muy común recibir proyectos pequeños, quedando los proyectos de mayor envergadura para ser desarrollados en casa central.

Si las estimaciones indican que el proyecto requerirá más de tres días de esfuerzo y menos de seis meses de esfuerzo , entonces este proyecto es considerado pequeño, de manera que el equipo se hará responsable no solamente de llevarlo a cabo, sino también de que se cumpla con lo pactado en tiempo y dentro del presupuesto. En caso contrario, si el proyecto demandara más de seis meses de esfuerzo, entonces, se asignará un Gerente de Proyectos[15] que se hará cargo de dicho proyecto. Esto implica una demora significativa ya que entra en juego otro proceso de manejo de proyectos completamente diferente que demanda mucho más esfuerzo y dinero.

Los proyectos trabajados en el presente Caso de Estudio podían presentar todas o algunas de las siguientes características en lo referido a funcionalidad del sistema en cuestión:

- Desarrollo de nuevas funcionalidades en el sistema existente

[14] También conocido como desarrollo "in-house"

[15] Traducción de "Project Manager"

- Mantenimiento de funcionalidades existentes; esto es, mejora de ciertas funcionalidades o corrección de errores[16]en el sistema existente

En general, los sistemas desarrollados en esta área son de tipo administrativos. Pueden ser desarrollados para plataformas web o de escritorio y cuentan con bases de datos relacionales e interfaces con otros sistemas de los cuales se toman datos necesarios para el negocio.

4.3. Problema

La gerencia solicita que se revea el método de estimación que estaba siendo usado en la empresa en ese momento. Dicho método de estimación lo llamamos MEBDHP (Método de Estimación Basado en la Disponibilidad de Horas persona) y se detalla en el **ANEXO III**.

Se requirió una revisión del MEBDHP debido a que las solicitudes de revisión del cambio de alcance eran muy frecuentes y las estimaciones se consideraban poco precisas ya que los errores siempre superaban el 10%. Los cambios de alcance podían darse por cambios en los requerimientos o por estimaciones de esfuerzo demasiado optimistas. Al producirse un cambio de alcance era necesario pasar por un proceso de autorizaciones que demoraban la continuidad del proyecto. Esto generaba malestar e insatisfacción en los clientes generada por no solamente dichas demoras sino también por el incumplimiento en las entregas de requerimientos; ya que se corría el riesgo de que hubiera que dejar requerimientos afuera o en el peor de los casos, dar el proyecto por terminado por falta de fondos.

Se intenta que el esfuerzo requerido para desarrollar el producto de software sea lo más preciso posible, de manera que no supere más del 10% a las estimaciones en las que se basa el presupuesto solicitado. Además es necesario partir de requerimientos completos y acordados con los clientes para evitar sorpresas y malos entendidos al momento de encarar una revisión del cambio de alcance.

5. Descripción de la propuesta: Método de Estimación Basado en la Especificación de Requerimientos (MEBER)

Considerando las limitaciones descriptas en la sección anterior, se definió un nuevo método de estimación basado en la especificación de requerimientos (MEBER). A continuación se describen las características diferenciales del nuevo método MEBER con respecto al método usado anteriormente, el MEBDHP.

5.1. Características del MEBER

- Las estimaciones se realizan sobre requerimientos completos y consensuados con el cliente. El método se basa en el estándar: IEEE Std 830-1998 – “Guía para la especificación de requerimientos de software”

[16] Traducción de la frase “bug fixing”

- Se realizan rondas de estimación basadas en Juicio de Expertos, donde participan todos los miembros del equipo de desarrollo y se realizan ajustes de extremos, similares a los métodos Planning Poker, WideBand Delphi y Analogías.

Definición de requerimientos completos

A continuación se detalla el workflow de la propuesta que comienza con los requerimientos de alto nivel.

1. El DP[17] entrega un listado de requerimientos de alto nivel.
2. El LP[18] y el DP se reúnen para revisar juntos dichos requerimientos.
3. El LP genera una lista de requerimientos detallados completos, de bajo nivel.
4. El LP analiza los requerimientos, uno a uno. De ser posible, el LP genera un prototipo con interfaces, sin código de base, que reflejan un posible diseño para los requerimientos detallados.
5. El LP confecciona el ERS[19] donde ingresa los requerimientos de bajo nivel. El LP puede adjuntar capturas de pantallas del prototipo en el caso de haber generado uno.
6. El LP valida el ERS con un desarrollador Sr (o Srr) quien, de ser posible, conozca el sistema o tenga experiencia en el desarrollo de sistemas similares.
7. El LP se reúne con el DP donde analizan el ERS. Es muy probable, que luego de esta reunión, surjan nuevos requerimientos, algunos requerimientos sean modificados y otros sean descartados.
8. El DP aprueba el ERS y prioriza los requerimientos.
9. El LP genera una WBS a partir de los requerimientos detallados.
10. Una vez que se tiene la WBS, el LP la ingresa en un GANTT que se usará como base para ir cargando las estimaciones.

Sesiones de estimación

El equipo de desarrollo estima cada tarea de la WBS. La dinámica del proceso de estimación es similar al Planning Poker dado que se requieren que todos los desarrolladores estimen en privado y luego muestran sus estimaciones al resto del equipo llegado el momento; de esta manera se pueden discutir dichas estimaciones entre todos en el equipo hasta obtener consenso. La diferencia con Planning Poker es que no se utilizan las barajas de cartas para seleccionar el valor de estimación para cada tarea. Tampoco se utilizan puntos de historia

[17] Dueño del Proyecto

[18] Líder del Proyecto

[19] Ver ANEXO III

[20]como en Planning Poker. En nuestro caso directamente las estimaciones se realizan en horas persona.

Para este momento todos los invitados a la sesión de estimaciones tienen que haber leído el ERS. Algunos de los desarrolladores ya fueron consultados y/o ellos mismos propusieron la solución al requerimiento a estimar.

En la reunión, se va a trabajar sobre el GANTT que contiene la WBS. Éste se proyecta (a través de un proyector en una sala de reuniones o se comparte pantalla si la reunión es virtual) de manera que todos en la reunión sean testigos de la carga de estimaciones y asignación de los recursos que trabajarán cada tarea/actividad.

El LP facilita esta reunión.

1. El LP lee al resto de los participantes el primer requerimiento enunciado en el GANTT (documentado en MS Microsoft Project 2013) y lo explica haciendo referencia al ERS donde se encuentra todo el detalle. También explica la solución correspondiente. La explicación de cada requerimiento y sus respectivas soluciones no puede tomar más de 5 minutos.

2. Antes de pedir las estimaciones a cada desarrollador, se aclaran todas las preguntas y/o comentarios sobre el requerimiento y su respectiva solución.

3. El LP solicita al/los desarrolladores que estimen el esfuerzo en horas persona necesario para completar la tarea. Cada uno piensa, en privado -es decir, sin compartir con el resto de los participantes- el esfuerzo que considera que llevará la tarea en cuestión. El LP les da un minuto para que piensen sus estimaciones. Cuando se cumple el tiempo (1 min) el LP pide por turnos que cada participante comunique su estimación en horas persona. El LP va anotando en un documento temporal (planilla de cálculo) las estimaciones de cada uno de los participantes para la tarea en cuestión. De esta manera queda asentado para cada participante, cuál fue su valor de estimación. Este documento también se comparte en pantalla.

4. Una vez que se tienen todas las estimaciones de la tarea en cuestión en pantalla, el LP revisa las estimaciones e identifica los valores “extremos” (es decir, los valores más alto y más bajo). Si la diferencia entre ambos valores extremos es mayor a 16 horas persona aproximadamente, pide a los estimadores de cada uno de los llamados “extremos” que expliquen sus razones para haber elegido dichos valores. Esto abre la discusión para el resto de los participantes también. De esta discusión pueden surgir aspectos que no se tenían en cuenta, así como también puede darse que el estimador del valor máximo se dé cuenta que su estimación fue muy pesimista o que el estimador del valor más bajo se dé cuenta de que su estimación fue demasiado optimista, entre otros factores que pueden ayudar a ajustar la precisión de las estimaciones. Esta discusión no puede durar más de 5 minutos. Una vez que se cumple el tiempo, el LP abre a una nueva ronda de estimaciones donde ahora los desarrolladores cuentan con más información. En general, y en particular para proyectos pe-

[20] Traducción del Inglés: Story Points

queños donde no se tienen más de 2 desarrolladores, no se dan más de 2 rondas de estimaciones. Una vez que las diferencias son menores a 16 hs persona, el LP saca un promedio de todas las estimaciones y anota ese valor en el GANTT.

El valor de estimación del esfuerzo requerido para cada requerimiento lo referenciaremos como: HE_{ReqN} (Horas Estimadas para el Requerimiento "N")

5. Una vez que la tarea en cuestión ya tiene su valor cargado en el GANTT el LP consulta a los desarrolladores, quién quiere trabajar dicha tarea y asienta el nombre del desarrollador voluntario en el GANTT.

Una vez terminada la sesión, se tienen todas las estimaciones cargadas en el GANTT. De manera que se calculan las horas estimadas para todo el proyecto, haciendo la sumatoria de las estimaciones para cada requerimiento.

$$HE_{ProjN} = \Sigma\ HE_{Req1} + HE_{Req2} + \ldots + HE_{ReqN}$$

Acto seguido, el LP carga el porcentaje de contingencia (Ca%) sobre el total de horas estimadas para el proyecto (HE_{ProjN}) definida por el rango de errores que existe en la estimación de pequeños proyectos en el contexto real de esta empresa. Esta contingencia oscila entre el 10% y el 25%. La decisión final sobre el porcentaje de contingencia a utilizar la tiene el gerente.

$$HE_{ProjN} + Ca\% = (\Sigma\ HE_{Req1} + HE_{Req2} + \ldots + HE_{ReqN}) + Ca\%$$

Por último, el LP confecciona el Contrato[21] y lo somete a aprobaciones para la asignación del presupuesto solicitado.

5.2. MEBER. Diagrama de estados

La **Figura 9** muestra el diagrama de los estados del proyecto que va desde la desde los requerimientos de alto nivel, estimación del esfuerzo hasta la aprobación del contrato para la liberación de fondos.

[21] Es el mismo documento estándar usado en MEBDHH.

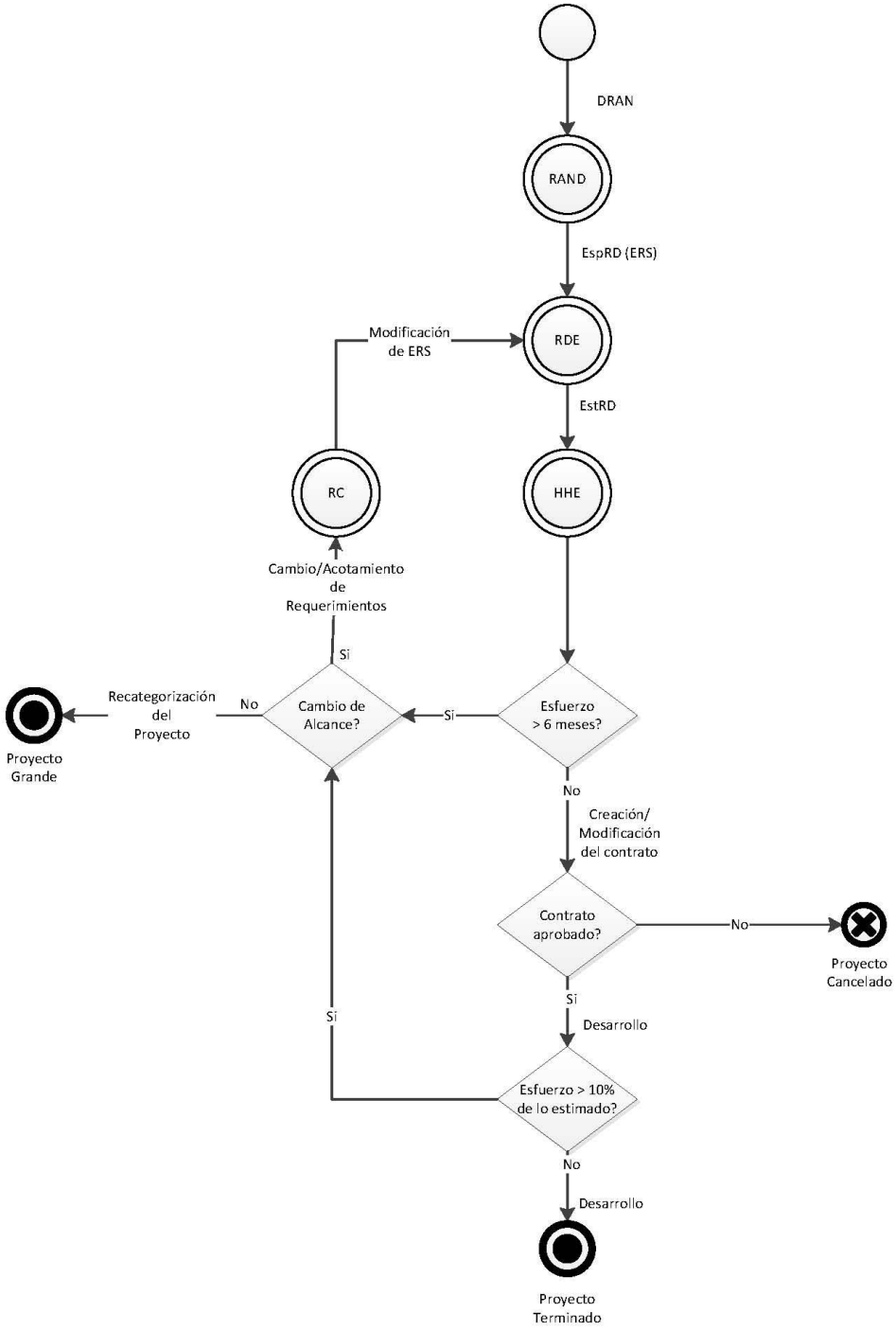

Figura 9 - Diagrama de Transición de estados del MEBER

Referencias pertenecientes a la Figura 9

Estados	Acciones
RAND: Requerimientos de Alto Nivel Definidos **RDE**: Requerimientos Detallados Especificados **HHE**: Horas persona Estimadas **RC**: Requerimientos cambiados	**DRAN**: Definición de Requerimientos de Alto Nivel. **EspRD(ERS)**: Especificación de Requerimientos Detallados. Creación de documento ERS (Especificación de Requerimientos de Software que se adjuntará al contrato) **EstRD**: Estimación de Requerimientos Detallados. **Esfuerzo >6 meses?**: Si el esfuerzo requerido para completar el proyecto es mayor a 6 meses, entonces evaluar el cambio de alcance. Si no, Creación/Modificación del contrato. **Cambio de alcance?**: Si hay cambio de alcance, entonces se realiza el Cambio/Acotamiento de Requerimientos. Si no, Re-categorización del Proyecto. **Creación/Modificación del contrato**: se crea el Contrato. Si el proyecto sufrió un cambio de alcance, entonces se modifica el Contrato acorde a los cambios realizados. El Contrato se somete a aprobación. **Cambio/Acotamiento de Requerimientos**: cambiamos y/o sacamos requerimientos de la lista. **Modificación del ERS**: a partir del cambio/acotamiento de requerimientos, se realiza una modificación del documento ERS. **Contrato Aprobado?**: Si se aprueba el Contrato, entonces se comienza con el Desarrollo. Si el proyecto sufrió un cambio de alcance, entonces se vuelve a aprobar el contrato y se continúa con el desarrollo. Si no se aprueba el contrato, entonces se cancela el Proyecto. **Recategorización del Proyecto**: el proyecto requiere más de 6 meses de esfuerzo en horas persona y no va a haber cambio de alcance, por lo tanto, se debe recategorizar el proyecto a Proyecto Grande. **Esfuerzo > 10% de lo estimado?**: Si se ha realizado un esfuerzo mayor al 10% de lo estimado, entonces se hará una revisión del cambio de alcance. Si no, se continúa con el desarrollo.

5.3. ERS (Especificación de Requerimientos de Software)

Para documentar los requerimientos, fue necesaria la creación y uso de un documento que se pudiera adjuntar al contrato de manera que fuera público y de consulta permanente para el equipo de desarrollo y el cliente.

Es un documento creado por el LP pero trabajado por el mismo y el cliente. Luego es revisado y utilizado por el equipo de desarrollo.

Este documento fue una herramienta utilizada por el LP para abrir los canales de comunicación con el cliente y para que quedaran plasmados los requerimientos acordados y sus posibles soluciones como parte del contrato. Esto evitaba confusiones entre ambas partes y cumplía con las expectativas de los clientes, a la vez que otorgaba transparencia al proceso.

El contexto de las políticas de la empresa obliga a seleccionar un estándar internacional. Por lo tanto, la tesista generó una plantilla (ver ANEXO III) adaptada a las necesidades de la empresa, a partir del estándar internacional de la IEEE, 830-1998. Esta plantilla lleva el nombre estándar de Especificación de Requerimientos de Software (ERS)[22].

Esta plantilla fue presentada a la gerencia, quien la adoptó y solicitó al resto de los equipos de desarrollo del área para la aprobación de los contratos de cada pequeño proyecto.

6. Caso de estudio

Nuestro caso de estudio está basado en los datos recolectados de 8 proyectos que se ejecutaron de principio a fin bajo los lineamientos del método propuesto, MEBER. Estos son proyectos típicos representativos de la realidad.

6.1. Características de los proyectos

Los datos fueron recolectados a medida que se iban desarrollando los proyectos y fueron provistos por el LP quien, en todos los casos, resultó ser la misma persona. El grupo de trabajo también se mantuvo igual.

Para realizar este estudio, se tomaron dos sistemas distintos. Uno de ellos, es un sistema que da soporte al área de recursos humanos de la empresa. Es un sitio web de uso exclusivamente interno desarrollado con tecnología ASP.NET usando VB.NET como lenguaje de programación. La base de datos relacional corría sobre SQL Server 2005. Se encontraba en el ambiente de Producción desde el año 2008 y no planteaba grandes desafíos respecto a la precisión en las estimaciones. Igualmente, agrega valor al presente trabajo ya que nos muestra un ambiente estable.

El otro sistema también se usaba para dar soporte al área de recursos humanos pero tenía un propósito diferente del primero. Igualmente se encontraba relacionado con el primero, siendo éste alimentado por dicho sistema. Recién había salido a Producción al momento del desarrollo del presente caso de estudio. Es un sistema de escritorio desarrollado sobre la plataforma .NET Framework usando el lenguaje de programación VB.NET. Su base de datos relacional corría sobre SQL Server 2005. Este sistema experimentó grandes falencias en el ciclo de vida de desarrollo que obviamente tuvieron un gran impacto en las estimaciones. Por ende, este escenario se presenta con un cliente insatisfecho y una relación no ideal para

[22] Traducido de los términos en inglés: Software Requirements Specification (SRS)

con el equipo de desarrollo. Al equipo se le planteó el desafío de hacerse cargo del mantenimiento de este sistema, a la vez que tuvo la compleja tarea de afianzar la relación con el cliente que hasta el momento se encontraba desgastada. El equipo debió tomar las lecciones aprendidas con este sistema y plantear un cambio de estrategia para poder lograr sus objetivos que eran: lograr una mayor precisión en las estimaciones a partir de requerimientos completos y mantener una buena relación con los clientes. El cumplimiento del primer objetivo devengará claramente en el cumplimiento del segundo.

El mantenimiento y desarrollo de los sistemas se llevó a cabo a través de la creación de pequeños proyectos. Mantenimiento incluye: el desarrollo de nuevas funcionalidades, corrección de errores, cambio de funcionalidades existentes, cambio/nuevos procesos de batch e interfaces con otros sistemas

Se tomaron ocho proyectos ejecutados en la realidad. En la **Tabla 2** se enumeran dichos proyectos y se provee una breve descripción de los cambios que se realizaron en dicho proyecto (por políticas de confidencialidad de la empresa no es posible brindar más detalles al respecto). También se detalla el número de requerimientos para dar una idea del tamaño de la lista de requerimientos. Ejemplos de requerimientos de bajo nivel podrían ser: "Implementar un cuadro de diálogo que permita la carga de personas dentro del grupo X"; "Implementar una funcionalidad que permita cargar datos, tales como: organización, fecha de ingreso y fecha de egreso de la planilla de cálculo X ", etc.

Además se incluyen las horas estimadas (**HE_{ProjN}**) por proyecto. Cada HE_{ProjN} es la sumatoria de horas originales estimadas de cada requerimiento:

$$HE_{ProjN} = \Sigma\ HE_{Req1} + HE_{Req2} + \ldots + HE_{ReqN}$$

A este valor se le suma el porcentaje de contingencia aplicada (**Ca%**):

$$HE_{ProjN} + Ca\%$$

$HE_{ProjN} + Ca\%$ no sufrió ajustes durante el desarrollo. Estas horas no representan trabajo neto, es decir que se considera que el desarrollo productivo de 8 hs es en realidad de 6 hs. La contingencia aplicada (*Ca %*) a cada proyecto, como explicamos anteriormente, es una decisión de la gerencia.

También se incluyen las horas reales (**HR $_{ProjN}$**) por proyecto. Cada HR $_{ProjN}$ es la sumatoria de horas originales reales que tomó cada requerimiento. Estas horas no representan trabajo neto, es decir que se considera que el desarrollo productivo de 8 hs es en realidad de 6 hs.

$$HR_{ProjN} = \Sigma\ HR_{Req1} + HR_{Req2} + \ldots + HR_{ReqN}$$

Tabla 2 - Proyectos seleccionados

Proyecto	**Breve descripción de cambios realizados como parte del proyecto**	**Nro. de Requerimientos**	**Horas estimadas + Contingencia (HE $_{ProjN}$ + Ca%)**	**Contingencia Ca%**	**Horas Reales (HR $_{ProjN}$)**
1	Se implementaron cambios en la interfaz de usuario y se corrigieron errores funcionales en 1 de los 5 módulos del sistema.	15	116	20%	130
2	Se implementaron nuevos cambios en la interfaz de usuario y se corrigieron errores funcionales.	15	128	20%	159
3	Se realizó la corrección de errores funcionales y de interfaz de usuario en 4 de los 5 módulos del sistema. Esto también incluye generación de reportes nuevos y correcciones en otros existentes.	25	560	20%	667.5
4	Mayormente se implementaron nuevas funcionalidades críticas para el Cliente que involucraban importantes cambios en la base de datos y cambio en la interfaz con otro sistema. Esto último implicaba cambios y mejoras en el rendimiento del proceso de batch. Además se corrigieron algunos de los errores más críticos de funcionalidades existentes.	25	415	0%	478
5	Se realizaron correcciones de errores en funcionalidades existentes e interfaz de usuario.	12	120	25%	99.5
6	Se implementaron dos nuevas funcionalidades y se realizaron correcciones de errores en funcionalidades existentes y errores en las interfaces de usuario.	5	66	25%	76.5
7	Se implementaron nuevas funcionalidades en uno de los módulos centrales del sistema. Se corrigieron errores en reportes, funcionales y cosméticos. Además se implementó un nuevo sistema de batch	25	435	25%	478

	para tomar datos que alimentarían al sistema por única vez luego de una restructuración organizacional.				
8	Se implementó un nuevo módulo de control con el que el usuario daba mantenimiento a las tablas maestro del sistema, así como también se usaba para la resolución de conflictos dados luego de recibir la alimentación del sistema con el que se tenía interfaz. También se realizaron algunas mejoras y corrección de errores en funcionalidades existentes.	26	490	25%	539.5

6.2. Aplicación del método MEBER

Este método fue aplicado a lo largo de aproximadamente 2 años. Los proyectos tuvieron la ventaja de tener a la tesista como LP.

En la **Tabla 3** se enumeran los proyectos trabajados y se considera la contingencia (**Ca%**) ya carga en las horas estimadas (**HE $_{ProjN}$**).

Igual que en la **Tabla 3**, se incluyen las horas reales (**HR $_{ProjN}$**) por proyecto. Cada HR $_{ProjN}$ es la sumatoria del esfuerzo real en horas persona que tomó cada requerimiento. Estas horas no representan trabajo neto, es decir que se considera que el desarrollo productivo de 8 hs es en realidad de 6 hs.

$$HR_{ProjN} = \Sigma\ HR_{Req1} + HR_{Req2} + \ldots + HR_{ReqN}$$

También se incluyen las horas estimadas (**HE $_{ProjN}$**) por proyecto. Cada HE $_{ProjN}$ es la sumatoria de horas originales estimadas de cada requerimiento:

$$HE_{ProjN} = \Sigma\ HE_{Req1} + HE_{Req2} + \ldots + HE_{ReqN}$$

A este valor se le suma el porcentaje de contingencia aplicada:

$$HE_{ProjN} + Ca\%.$$

HE $_{ProjN}$ + Ca% no sufrió ajustes durante el desarrollo. Estas horas no representan trabajo neto, es decir que se considera que el desarrollo productivo de 8 hs es en realidad de 6 hs. La contingencia aplicada (***Ca %***) a cada proyecto.

Además se muestra, el error relativo (**ER**) entre las horas estimadas y las horas reales:

$$ER = [HR_{ProjN} - (HE_{ProjN} + Ca\%)] / HR_{ProjN}$$

También se muestra la magnitud del error relativo (**MER**) que es el valor absoluto del error relativo, de manera que muestra las desviaciones que se produjeron respecto de las estimaciones, hayan sido estas pesimistas donde vemos valores negativos en la columna ER u optimistas que nos muestran valores positivos en la columna ER.

$$MER = ABS(ER)$$

$$MER = ABS\{[HR_{ProjN} - (HE_{ProjN} + Ca\%)] / HR_{ProjN}\}$$

Tabla 3 - Estimación de proyectos considerando la contingencia (Ca%) dentro de las horas estimadas

Proyecto	**Horas Reales** (HR ProjN)	**Horas Estimadas** (HE_{ProjN} + Ca%)	**Contingencia** (Ca%)	**Error Relativo** ER=[HR_{ProjN} – (HE_{ProjN} + Ca%)] / HR_{ProjN}	**Magnitud del Error Relativo** MER = ABS(ER)	**Error Relativo – 10%** (10% implica revisión del cambio de alcance para la empresa)
1	**130**	116	20%	11%	11%	1%
2	**159**	128	20%	19%	19%	9%
3	**667,5**	560	20%	16%	16%	6%
4	**478**	415	0%	13%	13%	3%
5	**99,5**	120	25%	-21%	21%	-31%
6	**76,5**	66	25%	14%	14%	4%
7	**498**	435	25%	13%	13%	3%
8	**539,5**	490	25%	9%	9%	-1%
		Medias	**20%**	**9%**	**14%**	**4%**

Aún considerando la contingencia como parte de las horas estimadas, vemos que los errores superan el 10% lo que implica que no se ha evitado un cambio de alcance.

Según este planteo, para evitar un cambio de alcance, sería necesario corregir la contingencia con la cual se trabaja. Se podría usar una contingencia de un 34%, siendo la contingencia promedio usada anteriormente de un 20% a lo que podríamos sumarle una corrección de un 14% que es la media de la magnitud del error relativo.

Ahora en la **Tabla 4** presentamos un escenario más claro donde quitamos la contingencia de las horas estimadas ya que queremos medir puramente la precisión en las estimaciones de esfuerzo. Vemos que los valores de MRE están en un rango de [10%-36%]. Aplicamos porcentajes de Ca en un rango de 34% al 39% (cfr. **Tabla 5**), observamos que un valor del 39%, nos asegura que todos los MRE no superan el 10%, asegurando que no va a existir un cambio de alcance, en ninguno de los proyectos. Al mismo tiempo un 39% de Ca aumenta los valores de sobreestimación, aspecto que también tiene sus consecuencias negativas al provocar horas ociosas o una disminución de la productividad del grupo de trabajo.

La recomendación es tomar un valor de contingencia del 39% para evitar la revisión de cambio de alcance. Al mismo tiempo el líder de proyectos tiene que balancear el riesgo de una sobrestimación.

Tabla 4 - Estimación de proyectos SIN considerar la contingencia (Ca%)

Proyecto	**Horas Reales** (HR ProjN)	**Horas Estimadas** (HE_{ProjN})	**Error Relativo** $ER=(HR_{ProjN} – HE_{ProjN}) / HR_{ProjN}$	**Magnitud del Error Relativo** MER = ABS(ER)
1	130	92,8	29%	29%
2	159	102,4	36%	36%
3	667,5	448	33%	33%
4	478	415	13%	13%
5	99,5	90	-10%	10%
6	76,5	49,5	35%	35%
7	498	326,25	34%	34%
8	539,5	367,5	32%	32%

Tabla 5 – MER calculados usando diferentes valores de Ca%

Proyecto	**34%**	**35%**	**36%**	**37%**	**38%**	**39%**	**40%**
1	4%	4%	3%	2%	1%	1%	0%
2	14%	13%	12%	12%	11%	10%	10%
3	10%	9%	9%	8%	7%	7%	6%
4	-16%	-17%	-18%	-19%	-20%	-21%	-22%
5	-21%	-22%	-23%	-24%	-25%	-26%	-27%
6	13%	13%	12%	11%	11%	10%	9%
7	12%	12%	11%	10%	10%	9%	8%
8	9%	8%	7%	7%	6%	5%	5%

6.3. Análisis de los resultados

Para evitar un cambio de alcance sería necesario aplicar una contingencia de 39% para quedar dentro de los límites aceptables menores al 10% donde no se haría una revisión del alcance.

Se destaca:

1) No se evitaron los cambios de alcance ya que todos tuvieron errores mayores al 10%.

2) El cliente está más satisfecho debido a que se cumplió con la entrega de los requerimientos acordados para cada proyecto.

7. Discusión

En el presente caso de estudio planteamos un nuevo método de estimación (MEBER) sobre requerimientos completos y consensuados con los clientes. Fue una ventaja el haber podido probar dicho método durante aproximadamente 2 años, con ocho proyectos típicos reales de tamaños diferentes (según cantidad de horas personas) dentro del marco que los categoriza como proyectos pequeños ya que sus duraciones se encuentran dentro del rango de 1 a 6 meses. Tuvimos proyectos muy pequeños de unas 100 horas reales de esfuerzo hasta proyectos de más de 650 horas persona de esfuerzo. Se contó con las horas estimadas, con las horas reales y también con la contingencia que la gerencia de la empresa decidió agregar en la concepción de cada proyecto

Desde el punto de vista operativo, al aplicar el MEBER no se lograron estimaciones lo suficientemente precisas como para no exceder el límite de tolerancia (o revisión del cambio de alcance) del 10%. Pero sí se logró cumplir con las expectativas de los clientes y evitar conflicto con los mismos al establecer un acuerdo respecto de los requerimientos completos y las posibles soluciones que se iban a dar a cada uno de ellos.

Vimos que sería bueno hacer una corrección de la contingencia a aplicar sobre las estimaciones de esfuerzo de un 39%, dejando al líder de proyectos la evaluación del riesgo de sobreestimación, el cual es posible que ocurra.

No sería recomendable dejar las estimaciones libradas sin contingencia ya que vimos que en la mayoría de los casos, los estimadores fueron optimistas y subestimaron las tareas. Además se penaliza la revisión de cambio de alcance a nivel gerencial. Como contrapartida, tenemos que la sobreestimación puede generar estimadores pesimistas que hagan valer la Ley de Parkinson que sostiene que mientras más tiempo se tiene, más tiempo toma una tarea en realizarse. Entonces, por las características de la empresa y por el contexto donde se valora más el hecho de evitar el cambio de alcance, podríamos recomendar que el Líder de Proyecto sea el que maneje los valores de contingencia aplicados al proyecto en privado para evitar influenciar negativamente a los desarrolladores.

Según Jørgensen et. al. [35] uno de los efectos negativos que más influyen en el fracaso de los proyectos de software es el exceso de optimismo. Jørgensen et. al. [31]. descubren un punto importante que pueden estar llevando a los estimadores a ser demasiado optimistas, esto es el formato utilizado al formular la pregunta que solicita la estimación de esfuerzo. El **formato tradicional** sería: “¿Cuántas horas se necesitan para completar la tarea X?” y el alternativo sería: “¿Cuántas tareas se pueden completar en Y horas?” Al analizar la situación que se daba en la empresa antes de implementar MEBER vimos claramente que se optaba por el formato alternativo, donde los clientes planteaban al equipo de desarrollo cuántas tareas (o requerimientos) se podían cumplimentar dado un presupuesto anual fijo ya determinado. La recomendación de Jørgensen et. al. [34] es que siempre se opte por el formato tradicional ya que este no conlleva ninguna desviación impuesta por los clientes que quieren obtener el máximo con un presupuesto irreal. Este escenario logramos revertirlo, sacando el foco del presupuesto disponible y dirigiéndolo al trabajo conjunto con el cliente a través de la toma de requerimientos, estimaciones y priorización de tareas. El conocimiento

del presupuesto disponible solo establecía el límite sobre la ejecución sobre las tareas rankeadas.

Reconocemos que quizás el margen de error aplicando MEBER podría ser mayor si contáramos con otro equipo de trabajo [43], dado que el equipo varió muy poco de proyecto a proyecto y los desarrolladores eran personas calificadas como Ssr quienes además conocían los sistemas.

No sería recomendable trabajar los requerimientos con mucha profundidad (a tal punto de llegar a la construcción de un prototipo si el proyecto es muy pequeño (es decir que su tamaño indique que se puede desarrollar en un rango de uno a dos meses de esfuerzo). Pero sí recomendamos trabajar con los clientes en el entendimiento de cada requerimiento, planteo de posibles soluciones y estimación en equipo antes de la liberación presupuestaria. Dicho trabajo deber quedar plasmado en el contrato y en el ERS para evitar conflictos y cubrir las expectativas. Esto otorga transparencia al proceso.

Recomendamos que la toma de requerimientos, para la ejecución de estos pequeños proyectos de software, sea realizada por el LP para evitar que las estimaciones de esfuerzo se vean altamente afectadas por información irrelevante y desconcertante. [31] aconseja evitar estos efectos eliminando o neutralizando esta información irrelevante en la especificación de requerimientos antes de ser conocida por los estimadores. Es difícil volver al estado mental anterior luego de haber sido expuesto a este tipo de información.

También nos planteamos que si se aumenta la contingencia en MEBDHP[23] sería posible resolver el problema planteado. Lamentablemente al no tener datos históricos sobre la aplicación de este método no es posible afirmarlo ni negarlo. Al mismo tiempo destacamos que en el hipotético caso de que se pudiera disminuir la frecuencia de petición de cambios de alcance creemos que sería probable que se mejore la relación con el cliente, que era otro aspecto solicitado. Esta disminución en la revisión del cambio de alcance, haría que el proyecto pueda continuar sin sufrir interrupciones como cuando se queda a la espera de la liberación de presupuesto extra para continuar y/o terminar con dicho proyecto.

8. Conclusión

Para poder contestar la pregunta de investigación de si es posible mejorar la estimación de esfuerzo medido en horas persona de los pequeños proyectos de software, investigamos en profundidad los diferentes métodos de estimación de productos de software basados en el juicio de expertos (Capítulo 3). Estos conocimientos nos ayudaron a resolver el problema que se presentó en una empresa multinacional cuando se planteó la necesidad de disminuir el porcentaje de error de proyectos pequeños de software para evitar la revisión del alcance (Capítulo 4). Para esto se diseñamos un método llamado MEBER (Capítulo 5) que se aplicó a un conjunto de ocho proyectos típicos reales de la empresa, durante aproximadamente dos años. De esta manera podemos decir que el método es aplicable (Capítulo 6).

La aplicación de MEBER fue positiva, aunque para proyectos futuros resta ajustar el valor de la contingencia a aplicar para lograr la reducción del error según el estándar definido por la

[23] Método usado anteriormente por la empresa. Se detalla en el Anexo II.

empresa. Hicimos una prueba aplicando una progresión de contingencias sobre los valores de horas persona estimadas hasta obtener errores menores al límite permitido en la empresa. Esto nos dio como resultado que siempre debemos considerar una contingencia del 39%.

El ejercicio de tomar datos históricos y analizar las diferencias ha sido un entrenamiento necesario para resolver el problema planteado y lograr una mayor eficiencia en la estimación futura de proyectos de software.

A modo de resumen:

1) Sugerimos la incorporación de una contingencia de 39%. También recomendamos que el líder de proyectos realice una evaluación del riesgo de sobreestimación; dado que es posible que esto ocurra.

2) La diferencia a destacar entre la aplicación de un método o el otro es la estimación basada en expertos y la satisfacción del cliente, ya que el contrato con el cliente incluye tanto un presupuesto otorgado como la entrega de funcionalidades acordadas sobre la base de contar con requerimientos completos. Ambos conceptos no se pueden desligar.

Como conclusión final de la tesis se estableció que es posible evitar los cambios de alcance aumentando la contingencia y se pudo establecer un entorno de trabajo en equipo saludable y de amplia colaboración con el cliente, de allí que recomendamos su implementación

9. Trabajos de investigación futuros

Es necesario investigar la aplicabilidad (o modificaciones a realizar) del método MEBER en proyectos grandes con mayor número de desarrolladores.

También sería útil investigar el impacto en las estimaciones al aplicar MEBER sobre proyectos trabajados por equipos heterogéneos. Es decir, equipos constituidos por desarrolladores con distintos seniorities y por desarrolladores que tengan distintas experiencias previas en el desarrollo del sistema en cuestión.

Aportaría un gran valor investigar la aplicabilidad de este método para mejorar las estimaciones y la satisfacción de los clientes en entornos donde actualmente se están usando metodologías ágiles. Evaluar la aplicabilidad del MEBER, especialmente considerando el uso del ERS como herramienta para documentar los requerimientos completos y acordados con los clientes puede resultar desafiante.

Bibliografía básica relacionada

[1] A Guide to the Project Management Body of Knowledge. PMBOK Guide. Third Edition. 2004

[2] Aiello, G., Alessi, M., Cossentino, M., Urso, A., Vella, G. RTDWD: Real-Time Distributed

[3] Armstrong, J. S. 2001. The forecasting dictionary. Principles of forecasting: A handbook for researchers and practitioners. Ed. J. S. Armstrong. Boston, Kluwer Academic Publishers: 761-824.

[4] Beck, K.1999. Embracing change with Extreme Programming. IEEE Computer Society.

[5] Beck, K, Fowler, M, 2000. Planning Extreme Programming. Addison Wesley. October, 12 2000. ISBN 0-201-710961-9, 160 pages. http://agilemanifesto.org/

[6] Blattberg, R. C., S. J. Hoch. 1990. Database models and managerial intuition: 50% model + 50% manager.Management Science 36: 887-899.

[7] Boehm, B., 1981. Software Engineering Economics, Prentice Hall

[8] Boehm, B., C. Abts, and Chulani, S., 2000. Software development cost estimation Approaches - A survey. Annals of Software Engineering, 10: p. 177-205.

[9] Brown, R., 2000. Group Processes, second ed. Blackwell Publishers.

[10] Casier, K., Verbrugge, S., Meersman, R., Ooteghem, JV., Colle, D., Pickavet, M. and Demeester, P. 2005. A fair cost allocation scheme for CapEx and OpEx for a network service provider.

[11] Charette R., 2005. Why software fails? IEEE Spectrum (September): 42-49

[12] Cohn, M. 2005. Agile Estimating and Planning. Robert C. Martin Series. Prentice Hall.

[13] Dalkey, 1969. The Delphi Method. An experimental study of group opinion. Rand. Santa Mónica. CA 90406

[14] Delbecq A.L., Van de Ven A.H, Gustafson D.H., 1975 Group Techniques for Program Planning: A Guide to Nominal and Delphi Processes. Scott, Foresman and Co., Glenview, IL.

[15] Foss, T. Stensrud, E. , Kitchenham, B., Myrtveit I., 2003. A simulation study of the model evaluation criterion MMRE. IEEE Transactions on Software Engineering, 2003. 29(11): p. 985-995.

[16] Grenning, J., 2002.Planning Poker.

[17] Grenning, J., 2002. Planning Poker or How to avoid analysis paralysis while release planning.

[18] Haugen N.C. 2006. An empirical study of using Planning Poker for User Story estimation. Proceedings of AGILE 2006 Conference. Computer Society. IEEE

[19] Heemstra, F. J., R. J. Kusters 1991. Function point analysis: Evaluation of a software cost estimation model. European Journal of Information Systems 1(4): 223-237.

[20] Hihn, J., H. Habib-Agahi 1991. Cost estimation of software intensive projects: A survey of current practices. International Conference on Software Engineering, IEEE Comput. Soc. Press, Los Alamitos, CA, USA: 276-287. IEEE Std 830-1998 - Guide to Software Requirements Specifications

[21] ISO/IEC 19761:2003 COSMIC-FFP - A Functional Size Measurement Method. DOI = www.iso.org

[22] ISO/IEC 20926:2003 IFPUG 4.1 Unadjusted functional size measurement method - Counting practices manual. DOI = www.iso.org

[23] Jacobson, I., Grady, B., Rumbaugh, J.,1999. The Unified Software Development Process, Addison Wesley.

[24] Jørgensen, M. 1997. An empirical evaluation of the MkII FPA estimation model. Norwegian Informatics Conference, Voss, Norway, Tapir, Oslo: 7-18.

[25] Jørgensen, M. 2004. A review of studies on expert estimation of software development effort. Journal on System and Software, Vol. 70, No. 1-2, 37-60.

[26] Jørgensen, M. 2009. How to Avoid Selecting Bids Based on Overoptimistic Cost Estimates IEEE Software Vol.: 26 Issue:3, 79 - 84. May-June 2009.

[27] Jørgensen, M. 2010. Selection of strategies in judgment-based effort estimation. Journal of Systems and Software. June 2010. Vol 83, Issue 6, 1039-1050, Software Architecture and Mobility.

[28] Jørgensen, M. 2010. How to avoid selecting bids based on overoptimistic cost estimates. Information and Software Technology, Vol.: 52, Issue: 5, 506-516. May 2010.

[29] Jørgensen, M. and Boehm, B 2008. Software Development Effort Estimation: Formal Models or Expert Judgement? IEEE Software(March/April):14-19.

[30] Jørgensen, M. and Grimstad 2010. Software Development Estimation Biases: The Role of Interdependence. Transactions on Software Engineering.

[31] Jørgensen, M. and Grimstad, S. 2010. The Impact of Irrelevant and Misleading Information on Software Development Effort Estimates: A Randomized Controlled Field Experiment. IEEE Transactions on Software Engineering, 06 Aug. 2010. IEEE computer Society Digital Library. IEEE.

[32] Jørgensen, M. and Grimstad, S. 2010. Software Development Effort Estimation — Demystifying and Improving Expert Estimation, Simula Research Laboratory. Springer Berlin Heidelberg, Isbn: 978-3-642-01156-6, 381- 403.

[33] Jørgensen, M and Gruschke, Tanja M. 2008. The role of outcome feedback in improving the uncertainty assessment of software development effort estimates. ACM Trans. Softw. Eng. Methodol. 17, 4, Article 20 (August 2008), 35 pages.

[34] Jørgensen, M. and Gruschke, Tanja M. 2009. The Impact of Lessons-Learned Sessions on Effort Estimation and Uncertainty Assessments, IEEE Transactions on Software Engineering, Jan. 2009. IEEE computer Society Digital Library. IEEE Computer Society.

[35] Jørgensen, M. and Halkjelsvik, T. 2010. The effects of request formats on judgment-based effort estimation, (2010) Journal of Systems and Software, 83 (1), 29-36.

[36] Jørgensen, M. and Sjøberg, D. 2001. Impact of effort estimates on software project work. Information and Software Technology 43(15): 939-948.

[37] Jørgensen, M., Indahl, U and Sjøberg, D. 2003. Software Effort Estimation by Analogy and "Regression Toward the Mean". Journal of Systems and Software 68(3): 253-262.

[38] Jørgensen, M. and K. H. Teigen 2002. Uncertainty Intervals versus Interval Uncertainty: An Alternative Method for Eliciting Effort Prediction Intervals in Software Development Projects. Proceedings of: International conference on Project Management (ProMAC), Singapore, 343-352.

[39] Jørgensen, M. and Shepperd. 2007. A systematic review of software development cost estimation studies, IEEE Transactions on Software Engineering, Vol. 33, No. 1, 3-53, January.

[40] Jørgensen, M. and D. I. K. Sjøberg 2002. Impact of experience on maintenance skills. Journal of Software Maintenance and Evolution: Research and practice 14(2): 123-146.

[41] Kitchenham, B., S. L. Pfleeger, B. McColl, S. Eagan 2002. A case study of maintenance estimation accuracy. To appear in: Journal of Systems and Software.

[42] Klayman, J., J. B. Soll, V. C. Gonzalez and S. Barlas 1999. Overconfidence: It depends on how, what and whom you ask. Organizational Behaviour and Human Decision Processes 79(3): 216-247.

[43] Ktata, O and Lévesque, G. 2010. Designing and implementing a Measurement Program for Scrum Teams: what do agile developers really need and want? B.C.Desai. ACM

[44] G. Kadoda, M. Cartwright, L. Chen, and M. Shepperd. 2000. Experiences using Case-Based Reasoning to predict software project effort. Empirical Software Engineering Research Group Department of Computing Bournemouth University Talbot Campus Poole, BH12 5BB, UK

[45] Larman, Craig. 2003. Agile and Iterative Development: A Manager's guide. Addison Wesley.

[46] Lederer, A. L., J. Prasad. 1992. Nine management guidelines for better cost estimating. Communications of the ACM 35(2): 51-59.

[47] Moløkken-Ostvøld K., Haugen N.C., Benestad H.C. 2008. Using planning poker for combining expert estimates in software projects,(2008) Journal of Systems and Software, 81 (12), 2106-2117.

[48] McConnell, S. 2006. Software Estimation – Demystifying the Black Art.

[49] NASA 1990. Manager's handbook for software development. Goddard Space Flight Center, Greenbelt, MD, NASA Software Engineering Laboratory.

[50] Ochoa, S., Pino, L., Guerrero, Luis A., Collazos, C., 2006. SSP: A Simple Software Process for Small Size Development Projects.In S. Ochoa and G.-C. Roman (Eds.): Advanced Software Engineering: Expanding the Frontiers of Software Technology, IFIP International Federation for Information Procesing, Vol. 219, Boston:Springer, 2006, pp. 94-107

[51] Ochoa, S., Bastarrica, S., 2003. CWADEE: A Chilean Web Application Development Effort Estimation Process. In Proceedings of LA-Web 2003 Conference. IEEE Press. Santiago, Chile. 10-12 November, 2003.

[52] Pengelly, A. 1995. Performance of effort estimating techniques in current development environments. Software Engineering Journal 10(5): 162-170.

[53] Robiolo, G., Castillo, O., Rossi, B., Santos, S., 2013. ¿Es posible superar la precisión del juicio de expertos de la estimación de esfuerzo de proyectos de software? Experimental Software Engineering Latin American Workshop (ESELAW 2013), Uruguay, 2013.

[54] Robiolo, G. Santos, S. and Rossi, B., 2013. Expert Estimation and Historical Data: an Empirical Study, in proceedings, The Eighth International Conference on Software Engineering Advances, ICSEA 2013, October 27 - November 1, 2013 - Venice, Italy.

[55] Rowe , G and Wright, G, 1999. The Delphi technique as a forecasting tool: issues and analysis.

[56] Rowe, G and Wright, 2002.Expert Opinions in Forecasting: The role of the Delphi Technique.

[57] Russ, M., McGregor, J. 2000. A Software Development Process for Small Projects. IEEE Software. September-October 2000 – P.96-101

[58] Sacre, E, 2003. A Methodology To Develop Web Applications in Small and Medium Size Enterprises. Master Thesis. Computer Science Department, University of Chile.

[59] Sanders, N. R., L. P. Ritzman. 2001. Judgmental adjustment of statistical forecasts. Principles of forecasting: A handbook for researchers and practitioners. Ed. J. S. Armstrong. Boston, Kluwer Academic Publishers: 405-416.

[60] Schatz B., Abdelshafi I. 2005. Primavera Gets Agile: A Successful Transition to Agile Development. IEEE Software. – Vol. 22. – No. 3. – P. 36–42.

[61] Shepperd, M., Schofield, C., 1997. Estimating Software Project Effort Using Analogies. IEEE Transactions on Software Engineering, Vol. 23, N° 12, November 1997.

[62] Schwaber, K, 2004. Agile Project Management with Scrum. Microsoft Press.

[63] Smith, G., Sidky, A. 2009. Becoming Agile ... in an imperfect world. Manning Publications Co.

[64] Stone, E., R and R. B. Opel 2000. Training to improve calibration and discrimination: The effects of performance and environmental feedback. Organizational Behaviour and Human Decision Processes 83(2): 282-309.

[65] Tausworthe, R 1980. The Work Breakdown Structure in Software Project Management. Jet Propulsion Laboratory.

[66] Vicinanza, S., Mukhopadhyay, T., Prietula, M. 1992. Examining the feasibility of a case-based reasoning model for software effort estimation, MIS Quarterly, 16 (June), pp155-71.

[67] Vicinanza, S., Mukhopadhyay, T., Prietula, M. 1996. Software Effort Estimation With a Case-Based Reasoner, J. Experimental & Theoretical Artificial Intelligence, 8, pp341 – 363.

[68] Vicinanza, S., Mukhopadhyay, T., Prietula, M. 1990. Case-based reasoning in effort estimation, in Proc. 11th Intl. Conf. on Info. Syst.

[69] Walkerden, F., Jeffery, R. 1999. An Empirical Study of Analogy-based Software Effort Estimation. Empirical Software Engineering, 4, 135–158 (1999) Kluwer Academic Publishers, Boston. Manufactured in The Netherlands.

[70] Walpole, R. 1999. Probabilidad y Estadísitica para Ingenieros. Sexta Edición. Pearson Education.

[71] Wideband-Delphi for user stories estimation. Engisud S.p.A. - Research and Development Lab. - Palermo, Italy. Engineering Ingegneria Informatica S.p.A. - Research and Development Lab. - Palermo, Italy. ICAR-CNR Istituto di Calcolo e Reti ad Alte Prestazioni Consiglio Nazionale delle Ricerche, Palermo, Italy

[72] Wiegers , Karl E 2000. Stop Promising Miracles. Software Development magazine.

[73] Wysocki, Robert K. and Rudd McGary. 2003. Effective Project Management, 3rd Edition.John Wiley & Sons ©

Glosario de siglas y términos

CAPEX: CapEx: Capital Expenditures. Gastos de Capital.

DP: Dueño del Producto. Es el stakeholder clave quien tiene una clara visión de qué es lo que se quiere construir. Tiene como responsabilidad la generación de los requerimientos de alto nivel y la priorización de los mismos.

ER: Error Relativo. Es el cociente de la división que resulta de la diferencia entre las Horas Reales (HR) y las Horas Estimadas (HE), y las Horas Reales. Ver fórmula (1)

$$ER = (HR - HE)/HR \quad (1)$$

ERS: Especificación de Requerimientos de Software (*o SRS en inglés*) (Ver ANEXO IV). Se creó una plantilla a partir del estándar (IEEE Std 830-1998) recomendado por la IEEE para la toma de requerimientos de software. Esta se adaptó para el uso interno en la compañía.

HE: Horas Estimadas. Esfuerzo estimado en horas persona.

HR: Horas reales. Esfuerzo real en horas persona.

LP: Líder de Proyecto. Es el encargado de custodiar el presupuesto asignado y de llevar a cabo el proyecto de principio a fin. Es el principal punto de contacto con el PO.

MEBDHP: Método de Estimación Basado en la Disponibilidad de Horas persona. Este era el método que se usaba anteriormente en la empresa en la que se basó el caso de estudio trabajado en la presente tesis. Ver ANEXO III

MEBER: Método de Estimación Basado en la Especificación de Requerimientos. Método nuevo diseñado y aplicado en el caso de estudio trabajado en la presente tesis.

MER: Magnitud del Error Relativo. Es el valor absoluto del ER. Ver fórmula (2)

$$MER = ABS((HR - HE)/HR) \quad (2)$$

MMRE: Media de la Magnitud del Error. Es la media de los MER aplicado a *"n"* proyectos.

Negocio: área encargada de llevar adelante las actividades comerciales de la empresa. Dicha área recibe el soporte del área de IT para el desarrollo y mantenimiento de las aplicaciones utilizadas por los mismos. Quienes pertenecen a dicha área son considerados clientes.

OPEX: OpEx: Operational Expenditures. Gastos Operativos.

PO: Product Owner. *(Cohn, 2004)* En las metodologías ágiles, es un rol que representa a los Stakeholders y es la voz del cliente. Se encarga de garantizar que el equipo entregue valor al cliente. Asigna prioridades de manera que siempre se trabaje en los ítems de mayor valor

para el cliente. Toma decisiones de manera que conduzcan a un buen retorno sobre la inversión en el proyecto.

Sr: desarrollador senior.

Ssr: desarrollador semi senior.

WBS: del inglés Work Break Down Structure, despiece de tareas. Es una descomposición del trabajo a realizarse en un proyecto para cumplir con los objetivos del mismo y crear los entregables requeridos. La WBS organiza y define el alcance total del proyecto.

ANEXOS

ANEXO I - ¿Es posible superar la precisión basada en el juicio de expertos de la estimación de esfuerzo de productos de software?

Por Gabriela Robiolo, Oscar Castillo, Bibiana Rossi y Silvana Santos

¿Es posible superar la precisión basada en el juicio de expertos de la estimación de esfuerzo de productos de software?

Gabriela Robiolo
Universidad Austral
Buenos Aires, Argentina
grobiolo@austral.edu.ar

Oscar Castillo y Bibiana Rossi
Universidad Argentina de la Empresa
Buenos Aires, Argentina
oscar.alexander@gmail.com,
birossi@uade.edu.ar

Silvana Santos
Universidad Nacional de La Plata
La Plata, Argentina
silvanasantos@gmail.com

Abstract. La estimación de esfuerzo de productos de software basada en el juicio de expertos es el método más difundido en la industria del software y existen evidencias que puede tener la misma o mejor precisión cuando éste es comparado con modelos de estimaciones formales. Con la finalidad de brindar una mayor evidencia de esta afirmación se plantea un caso de estudio de una aplicación compleja desarrollada en el contexto de una empresa pública. Se compara el método de estimación de expertos versus los métodos formales de Regresión lineal y Analogías, utilizando modelos de una sola variable Tamaño (medido en COSMIC) o Complejidad (medida en Paths). Los resultados muestran que no fue posible superar la precisión del juicio de expertos debido a su nivel de experiencia medio-alto.

Keywords: estimación de expertos, estimación de esfuerzo, regresión lineal, analogías

1 Introducción

Jorgensen [1] afirma que la estimación de esfuerzo de proyectos de software basada en el juicio de expertos es el método más difundido en la industria del software. Si bien esta afirmación no invalida el uso de métodos formales de estimación, pone en evidencia las limitaciones de dichos métodos, los cuáles no han podido superar la capacidad humana de sintetizar diversas variables complejas. También el autor observa que la estimación de expertos puede tener la misma o mejor precisión al ser comparada con un modelo estimación formal. Encuentra una fuerte evidencia de que la estimación de expertos es más precisa cuando el experto posee un importante conocimiento del dominio.

Se presenta en este artículo un caso de estudio con la finalidad de aportar una mayor evidencia a estas afirmaciones e investigar si es posible sostenerlas en un contexto de una empresa del ámbito público.

Las preguntas de investigación planteadas son:

- ¿Es posible reducir el error en la estimación de esfuerzo basada en expertos de un producto de software complejo aplicando un método de estimación formal que utiliza como única variable Tamaño o Complejidad?
- ¿Es posible reducir el error de estimación de esfuerzo de un producto de software complejo si a una historia sucesiva de estimaciones se aplica analogía evaluada por expertos vs analogía basada en Tamaño o Complejidad?

En segunda instancia, se optó por una sucesión de estimaciones para comprender cómo es la evaluación de analogías que realiza un experto en comparación con las analogías basadas en medidas objetivas.

Con el objetivo de responder a estas preguntas se seleccionó una aplicación compleja, la cual ha sido desarrollada en sucesivas versiones, de la cual se obtuvo la especificación de requerimientos y el Esfuerzo Real (ER). También fue posible obtener la estimación de esfuerzo de un experto de la empresa para comparar sus estimaciones con los resultados obtenidos usando métodos formales de estimación.

Los métodos de estimación formales usados en la comparación son métodos frecuentemente utilizados [2]: regresión lineal [3] y analogías [4]. Se seleccionaron las métricas COSMIC [5], [6] y Paths [7], dado que la primera es un estándar internacional y la segunda una métrica de complejidad adecuada para las características de la aplicación a analizar. Además, resulta interesante que en [1] no se han incluido artículos que usen COSMIC o Paths.

COSMIC (Common Software Measurement International Consortium) function points es un estándar de medición cada vez más aceptado que mide Tamaño de requerimientos funcionales. Los requerimientos de usuarios funcionales pueden ser mapeados en procesos funcionales. Cada proceso funcional consiste en subprocesos que envuelven movimientos de datos. La cantidad de los movimientos de datos de entrada, salida, lectura y escritura es el Tamaño funcional expresado en CFP.

Paths es una medida introducida por Robiolo [7], [8] que captura la complejidad de los requerimientos. Es una aplicación de la métrica de MacCabe [9] a la descripción de requerimientos. La complejidad de los requerimientos está expresada en términos de caminos, donde cada camino corresponde a un escenario alternativo de un caso de uso y es expresado en P.

El presente artículo presenta qué se entiende por juicio de expertos y algunos de los últimos estudios empíricos en torno a este tema y una discusión bibliográfica de la conveniencia del uso de métodos de estimación basado en expertos versus métodos formales de estimación. Desarrolla un caso de estudio y finalmente detalla las conclusiones junto con la descripción de posibles trabajos futuros.

2 El juicio de expertos

Jorgensen [1], uno de los autores con mayor cantidad de publicaciones en torno a este tema en los últimos años, define una estrategia de estimación como juicio de expertos a un trabajo de estimación realizado por una persona reconocida como un experto en esta tarea, donde una parte significativa del proceso de estimación es realizada en forma intuitiva, ejecutando un proceso no explícito e inconstruible. Realizo una revisión de estudios detallada en torno a este tema. Los resultados arrojados por dicho proceso de revisión sugieren que las estimaciones basadas en juicio de expertos es la más utilizada para proyectos de software. Afirma que no existe evidencia sustancial en favor del uso de modelos de estimación y que hay situaciones donde se puede esperar que las estimaciones expertas sean mucho más precisas que los métodos formales de estimación. Además propone una lista de 12 “best practices” o principios de estimación experta validados empíricamente y provee sugerencias sobre cómo implementar estas guías en las organizaciones. Una de las mejores prácticas es buscar expertos con conocimiento del dominio y capacidad de realizar buenas estimaciones, aspecto que se destaca en este artículo.

Respecto de la evaluación de la incertidumbre de las estimaciones realizadas, se planteó el rol que cumple el feedback sobre la discrepancia existente entre las horas estimadas versus las trabajadas. Existe evidencia suficiente [10], que indica que la mayoría de los desarrolladores son, inicialmente, demasiado optimistas sobre la precisión de sus estimaciones, manteniéndose así aun cuando el feedback provisto indica lo contrario. El autor sugiere que una condición importante que se tendría que dar para mejorar las estimaciones sobre la base del feedback provisto luego de la finalización de la(s) tarea(s), sería el uso de una estrategia explícita de evaluar la incertidumbre. Esta condición mejora mientras mayor es la cantidad de información histórica de la que se dispone. Circunstancia que es confirmada en éste artículo.

Uno de los efectos negativos que más influyen en el fracaso de los proyectos de software es el exceso de optimismo. Jørgensen y Halkjelsvik [11] descubren un punto importante que pueden estar llevando a los estimadores a ser demasiado optimistas, esto es el formato utilizado al formular la pregunta que solicita la estimación de esfuerzo. El formato tradicional sería: "¿Cuántas horas se necesitan para completar la tarea X?" y el alternativo sería: "¿Cuántas tareas se pueden completar en Y horas?" Cualquiera de estos dos formatos deberían, en teoría, arrojar los mismos resultados. Según Jørgensen, cuando se utiliza el formato alternativo, se obtienen sorprendentemente estimaciones mucho más bajas y por ende mucho más optimistas que si se usara el formato tradicional. La recomendación final de dicho estudio es que siempre se opte por el formato tradicional ya que este no conlleva ninguna desviación impuesta por los clientes que quieren obtener el máximo con un presupuesto irreal. En nuestro caso de estudio se usó el formato tradicional.

Mendes [12] investigó –en el contexto de proyectos financiados por el gobierno de Nueva Zelanda y más tarde en Brasil- el uso de un enfoque centrado en el experto en combinación con una técnica que permite la inclusión explícita de la incertidumbre y las relaciones causales como medio para mejorar la estimación del esfuerzo de proyectos de software. El artículo presenta una visión general del proceso de estimación de esfuerzo, seguido por la discusión de cómo un enfoque centrado en el experto mejora dicho procedimiento y puede ser ventajoso para las compañías de software.

3 El juicio de expertos vs los métodos formales

Jorgensen [1] presenta quince estudios que comparan la estimación de expertos con estimaciones basadas en modelos formales de estimación. Cinco de los artículos están a favor de la estimación de expertos, cinco no encuentran diferencia y cinco están a favor de los modelos formales de estimación. Resalta que el diseño de los experimentos tiene un fuerte impacto en los resultados. Además, destaca que los experimentos que no usaban modelos formales calibrados mostraban que la estimación de expertos era más precisa. Posterior al survey citado anteriormente sólo se ha hallado un artículo que compara juicio de expertos vs métodos formales, en particular analiza las ventajas y desventajas de juicio de expertos y aprendizaje automático [13].

En otro artículo, Jorgensen [14], remarca las mismas ideas resaltando que los expertos tienen una natural ventaja dado que típicamente procesan más información (o falta de ésta) en una forma más flexible y que podría ser difícil construir modelos de estimación más precisos. En el caso de estudio presentado en este artículo, se pone en evidencia la capacidad del experto anteriormente destacada.

También Jørgensen [1] reconoce que el juicio de expertos tiene sus aspectos negativos, como por ejemplo: el grado de inconsistencia y la ponderación de variables. Destaca que si estos efectos negativos se pudieran reducir, la mejora en la precisión de las estimaciones sería mucho mayor.

Jørgensen y Boehm [15], toman dos posturas opuestas y debaten intentando mostrar a las organizaciones las ventajas y desventajas de modelos formales y juicio de expertos. Boehm discierne con Jørgensen en que las estimaciones expertas producidas en estudios empíricos no son representativas de las estimaciones producidas en la práctica. Además, sostiene que ante la incertidumbre, las organizaciones van a optar por llevar a cabo extensos análisis de sensibilidad, riesgo y compensación a través de modelos formales ejecutados de manera rápida y con muy poco esfuerzo humano. Boehm recomienda combinar ambos enfoques, almacenando los resultados al finalizar el desarrollo (o fases del mismo) y utilizar estos valores en el proceso de "close-loop feedback" donde se comparan las entradas de las estimaciones, las salidas y los resultados finales de manera de aprender de eso y poder calibrar estimaciones de futuros proyectos. Jørgensen alienta a las organizaciones a invertir en estudios e investigaciones para mejorar los procesos de estimación basados en juicio experto. Esto mismo afirman [12][16].También Jørgensen propone el uso del método Wideband Delphi de Bohem para mejorar las estimaciones y evitar posibles pujas de intereses entre los stakeholders.

MacDonell and Shepperd [17], afirman que hay un alto grado de interdependencia entre las estimaciones basadas en los modelos comunes de estimación y estimación de expertos, y es difícil derivar reglas para seleccionar el modelo de estimación más preciso, por lo tanto la solución parece ser usar la combinación de modelos.

4 Caso de Estudio

La empresa donde se desarrolló el caso de estudio es una empresa grande y compleja del ámbito público argentino. La aplicación desarrollada es un sistema de infracciones de tránsito. Por motivos de confidencialidad no se describen más aspectos. La especificación de requerimientos de la aplicación seleccionada se basó en casos de uso (CU). Fueron seleccionados los casos de uso de cinco versiones que estaban claramente documentados, la codificación había sido finalizada y tenían registrado el ER. Se realizó la medición de Tamaño funcional y Complejidad de los casos de uso aplicando las métrica COSMIC y Paths respectivamente.

La Tabla 1 muestra para cada caso de uso el ER medido en horas hombre, el Tamaño medido en COSMIC, la Complejidad medido en Paths y el valor de las horas estimadas por un experto expresado en horas hombre para los CU de la aplicación y para las estimaciones sucesivas de las versiones.

Table 1. Datos de la aplicación del ámbito público

Versión	**ID CU**	**ER**	**CFP**	**P**	**Experto (CU)**	**Experto (versiones sucesivas)**
7	1	264	20	3	240	240
7	2	32	5	3	24	24
7	3	248	15	23	208	208
7	4	112	37	15	88	88
9	5	104	16	5	80	80
9	6	136	15	9	96	96
10	7	56	10	3	64	64
10	8	184	7	25	112	120
10	9	416	93	63	328	344
10	10	8	11	2	8	16
10	11	16	4	2	8	16
11	12	208	90	58	176	200
11	13	24	5	2	16	16
11	14	144	149	16	120	144
11	15	96	54	5	96	88

12	16	520	46	43	504	504
12	17	112	97	6	136	144
12	18	40	71	30	40	40

En la Tabla 2 y 3 se muestran las características del experto que realizó las estimaciones, las cuales describen el perfil, el nivel de experiencia y el grado de conocimiento del entorno del experto. Se le pidió al experto que clasificara sus capacidades en uno de los siguientes niveles: alto, medio o bajo. El experto es una persona que tiene un nivel alto de experiencia, tanto en el desarrollo de software como en el liderazgo, estimación de proyectos y en la tecnología usada en el proyecto. En el momento que realizó las estimaciones el experto estaba a cargo del sector donde se desarrolló la aplicación, pero no trabajaba en este sector cuando se desarrollaron las versiones que se muestran en este caso de estudio, por lo tanto no tenía conocimiento del ER. A solicitud de los autores del artículo realizó la estimación, basándose en la especificación de requerimientos, sin conocer las horas reales de la aplicación.

Table 2. Perfil del experto

Aptitudes	**Descripción**
Título de grado	Ingeniero
Años de experiencia en el desarrollo de software	12
Años de experiencia en liderazgo	8
Especialidad, Conocimientos	Desarrollo .NET, Java patrones, liderazgo de equipos, metodologías de desarrollo

Table 3. Nivel de experiencia y grado de conocimiento del entorno del experto

Capacidades	**Alto**	**Medio**	**Bajo**
Nivel Experiencia	x		
Conocimiento del rendimiento de los perfiles del grupo de desarrollo		x	
Conocimiento de la Tecnología	x		
Conocimiento del Dominio		x	

Para el cálculo de los errores se utilizara la Magnitud Relativa del Error (MRE) y el error relativo (ErR) siguiendo las fórmulas 1 y 2, respectivamente.

$$MRE = abs\ (ER-EE)\ /ER \qquad (1)$$

$$ErR = (ER-EE) /ER \quad (2)$$

Además, la calidad de la predicción (PQ) se calculará aplicando la siguiente fórmula (3)

$$PQ(0.25)= k/n \quad (3)$$

Siendo k la cantidad de CU cuyo error es menor a 0.25 y n el total de los casos de uso [18].

Con el objetivo de testear estadísticamente los resultados se plantean las siguientes hipótesis alternativas:

- $H1_a$: La media del MRE de la estimación de experto es menor que la media del MRE de la estimación usando el modelo de regresión lineal y la variable independiente P.
- $H1_b$: La media del MRE de la estimación de experto es menor que la media del MRE de la estimación por Analogía medido el Tamaño en CFP.
- $H1_c$: La media del MRE de la estimación de experto es menor que la media del MRE de la estimación por Analogía medida la Complejidad en P.
- $H1_d$: La media del MRE de la estimación de experto es menor que la media del MRE de la estimación por Analogía medido el Tamaño en CFP, en una historia sucesiva de estimaciones.
- $H1_e$: La media del MRE de la estimación de experto es menor que la media del MRE de la estimación por Analogía medida la Complejidad en P, en una historia sucesiva de estimaciones.

4.1 Estimación de CU de una aplicación

En primer lugar se analiza la aplicación en su conjunto, por lo tanto se consideran todos los CU para comparar los métodos formales de estimación con la estimación del experto.

Métodos formales de estimación.

Se consideraron dos métodos formales: Regresión lineal y Analogía.

Regresión lineal.

Se planteó el modelo de regresión lineal Y= a + b X donde Y es el Esfuerzo Real y X es CFP o P. La Tabla 4 muestra los resultados de la regresión lineal. No se obtuvo un modelo significativo usando como variable independiente CFP, como se puede observar en la Tabla 4 el valor de R^2 Ajustado es muy bajo y p-value es mayor a 0.05. Por el contrario fue posible obtener el valor de la recta de regresión usando como variable independiente a P obteniendo un R^2 Ajustado igual a 0.50 y un p-value igual a 0.001, eliminando dos outliers. También se testeo la condición de normalidad de los residuos aplicando el test de normalidad Shapiro-Wilk obteniendo un p-value igual a 0.21.

Table 4. Método de regresión lineal

EE= a + b X	**Outliers**	**R^2**	**R^2 Ajustado**	**p-value**
--	--	0.07	0.02	0.26
EE=58.78 +5 * P	CU_{12} y CU_{16}	0.52	0.50	0.001

Para el cálculo de los errores en la estimación se usó el método de "cross-validation", eliminando del modelo cada caso de uso a estimar. La Tabla 5 muestra los valores de la media (MMRE) y mediana (MeMRE) de los MRE, la calidad de la predicción (PQ) y los ErR.

Table 5. Comparación de los métodos de estimación

Método de Estimación	**MMRE**	**MeMRE**	**PQ(0.25)**	**ErR (min..max)**
Regresión Lineal (P)	1.45	0.34	0.38	-8.38..0.79
Analogía (CFP)	1.53	0.83	0.06	-6.7..0.87
Analogía (P)	0.94	0.46	0.39	-4.52..0.89
Experto	0.19	0.18	0.72	-0.21..0.5

Analogía.

Este método de estimación consiste en encontrar un proyecto similar al proyecto a estimar basándose en una característica. En forma independiente se usó $Tamaño_{[CFP]}$ o $Complejidad_{[P]}$. Se considera la productividad del CU "más similar" en Tamaño o Complejidad para la obtención del EE. Entonces, para el cálculo de la Productividad (PR) se utiliza la fórmula 4 [19],

$$PR = ER / X \quad (4)$$

Siendo X $Tamaño_{[CFP]}$ o $Complejidad_{[P]}$. Se obtiene el EE utilizando la fórmula 5,

$$EE = X^* PR_{PA} \quad (5)$$

Siendo el PR_{PA} la productividad del proyecto análogo, la productividad del proyecto que tiene un valor más cercano en $Tamaño_{[CFP]}$ o $Complejidad_{[P]}$. En caso de existir más de un proyecto análogo se toma el valor promedio de la productividad de los proyectos análogos.

Por ejemplo, CU_1 la Tabla 1 de tiene un esfuerzo de 264 HH y un tamaño de 20 CFP. El CU más cercano en Tamaño es CU_5 con un CFP de 16 y una ER de 104 HH, PR_{CU5} es igual al cociente entre 104 y 16, lo que equivale 6.5. Por lo tanto, EE_{CU1} es igual a 130 HH. La Tabla 5 muestra el análisis estadístico de los valores obtenidos usando CFP o P.

Comparación de los métodos formales con la estimación del experto.

La Tabla 5 compara los métodos de estimación formales con el método de estimación basado en el experto. Usando un método de estimación basado en expertos se obtuvo un resultado aceptado para una técnica de estimación: errores menores a un 25% [18].

En la Tabla 5 se observa que al emplear regresión lineal, solo se obtuvo valores significativos cuando se uso P como variable independiente. Solo el 38% de las estimaciones tuvieron un error menor al 25%, siendo este porcentaje bajo.

Al emplear estimación por analogía los resultados correspondiente a CFP fueron muy bajo, es decir, solo el 6% de las estimaciones realizadas por analogía tomando como variable dependiente a CFP pudieron tener un error menor al 25%. Por el contrario al emplear P como variable independiente se pudo mejorar el resultado: el 39% de las estimaciones tuvieron un error menor al 25%.

Si comparamos ambas técnicas podemos concluir que usar P como medida de Complejidad es mejor que usar CFP como medida de Tamaño. Al mismo tiempo, ninguna de estas técnicas para este caso de estudio es mejor que la estimación dada por el experto. Como se observa el 72% de las estimaciones del experto tiene un error menor al 25%.

Se puede concluir entonces que ninguna de las técnicas de estimación tradicionales pudo superar la estimación del experto. Existen algunas que son mejores que otras pero no son equiparables a la experiencia del experto.

4.2 Estimación sucesivas de versiones de una aplicación

En un escenario en el cual se estiman sucesivamente versiones de una aplicación se espera que el experto aprenda de las estimaciones anteriores. La Tabla 1 muestra los CU agrupados por versiones. Se compara la estimación del experto con un método formal basado en analogías.
El método de estimación por analogía usando como variables a CFP y P se aplicó en la misma forma que se explicó anteriormente, con la particularidad que el conjunto de CU usados para encontrar el análogo, es el subconjunto de los CU de la versión correspondiente más los CU de las versiones anteriores. Por ejemplo, cuando se estima la versión 10 se usan los CU de la versión 7 y 9. Dada esta limitación en el cálculo no fue posible aplicar regresión lineal puesto que en algunos casos la cantidad de puntos a considerar en el modelo era muy pequeña, obteniendo modelos no significativos.

Para obtener los resultados del experto, se realizó una segunda entrevista. Pero esta vez se le mostro solo los CU a estimar por versión y los errores cometidos al estimar las versiones anteriores. Por ejemplo, al estimar la versión 9, se le mostró los valores reales de la versión 7, además de la estimación realizada anteriormente de esta versión. La Tabla 1 muestra las estimaciones sucesivas realizadas por el experto. La Tabla 6 muestra los errores de las estimaciones sucesivas, aplicando el método por Analogías (CFP y P) y basado en expertos.

Como se ve en la tabla 6 las MMRE de la estimación del experto tiene un rango de [0.18-0.22], y los PQ [0.63-0.75], valores mejores que los restantes. En el caso de Paths el rango del MMRE es de [0.76-2.22] y el del PQ es de [0-0.39]. Los valores obtenidos usando COSMIC son MMRE [1.18-1.53] y PQ [0-0.25]. Si bien es mejor el resultado obtenido usando la métrica Paths y el método de estimación por Analogía no hay una mejora significativa si se lo compara con COSMIC.

Las estimaciones sucesivas realizadas por el experto fluctúan pero la media y mediana de MRE se mantienen dentro de valores aceptables para la industria, esto es menores al 25% y similares a la estimación del experto mostrada en Tabla 5. Los valores de PQ son los más altos de la Tabla 6 y si consideramos el PQ de la primera medición (72%), podemos concluir que la información más detallada no ha mejorado la precisión del experto. El experto al conocer su error introduce en la corrección de este una desviación mayor con respecto al valor real. Al mismo tiempo se observa en la Tabla 6 un aprendizaje por parte del experto, no logrando un valor mejor por las limitaciones de la mente humana.

Las estimaciones realizadas por Analogía y la métrica Paths también fluctúan, pero el PQ tiende a mejorar. Usando la métrica COSMIC también varía, reportando el PQ una mejora no sensible. En ambos casos el hecho de contar con una cantidad de CU menores para seleccionar el más análogo no mejora el error en la estimación.

Table 6. Comparación de la estimación sucesiva de versiones

Método de Estimación	**MMRE**	**MeMRE**	**PQ(0,25)**	**Errores relativos (min..max)**
COSMIC v7	1.35	0.92	0.25	-3.36.. 0.2
COSMIC v9	1.18	0.9	0	-3.36.. 0.5
COSMIC v10	1.43	0.75	0	-6.7..0.87
COSMIC v11	1.29	0.79	0	-6.7..0.87
COSMIC v12	1.53	0.83	0.06	-6.7..0.87
Paths v7	2.22	0.66	0	-7.25..0.88

Paths v9	1.86	0.95	0	-7.25..0.88
Paths v10	1.01	0.83	0.1	-4..0.83
Paths v11	0.76	0.46	0.33	-4..0.83
Paths v12	0.94	0.46	0.39	-4.52..0.89
Experto (V7)	0.18	0.19	0.75	0.09..0.25
Experto (V9)	0.20	0.22	0.67	0.09..0.29
Experto (V10)	0.26	0.21	0.64	-1..0.35
Experto (V11)	0.22	0.17	0.67	-1..0.35
Experto (V12)	0.20	0.17	0.63	-1..0.35

4.3 Amenazas de validez

La mayor amenaza de validez de este caso de estudio es la registración del ER realizado por la empresa. Es conocido que estos registros no son exactos, por lo que fueron validados por registros alternativos de horas trabajadas. Concluyendo que son registros correctos.

Puede llamar la atención la omisión de la versión 8. Esta fue descartada por falta de calidad de la descripción textual de los casos de usos.

También la cantidad de casos de usos utilizados es limitada y podría afectar los resultados. En la selección de los casos de uso se seleccionó un período de versiones que tuvieran registrado y disponible el ER. El tiempo de medición fue otra variable que fue necesario tener en cuenta considerando que era una aplicación compleja. Si bien la cantidad de datos es acotada la comprobación estadística de la hipótesis planteada es significativa.

Otro aspecto que puede influir es el conocimiento que tiene el experto de las horas reales de este caso de estudio. El experto se incorporó a la empresa en una etapa posterior al desarrollo real de las versiones presentadas en este caso de estudio y no tenía conocimiento de las horas reales. Además, el grupo que desarrollo las versiones incluidas ha variado con respecto al momento en que se consultó al experto.

4.4 Conclusiones del caso de estudio

Para realizar la verificación estadística se utiliza el test no-paramétrico Wilcoxon dado que las distribuciones no son normales. Se selecciona este test, puesto que para pruebas pareadas es posible aplicarlo a distribuciones continuas. Fue posible rechazar la hipótesis nula a favor de la hipótesis alternativa $H1_a$ aplicando el test no-paramétrico Wilcoxon p-value igual a 0.009. También fue posible rechazar la hipótesis nula a favor de la hipótesis alternativa $H1_b$ aplicando el test no-paramétrico Wilcoxon p-value igual a 0.000, $H1_c$ p-value igual a 0.006, $H1_{d,}$ p-value igual a 0.000 y $H1_e$ p-value igual a 0.000.

Por lo tanto se concluye que en este caso de estudio NO fue posible superar con las estimaciones de expertos utilizando métodos de estimación formales (Regresión lineal y Analogía), tanto en una estimación para todos los CU de la aplicación como para una estimación sucesiva de versiones.

La participación de un experto limita la conclusión del caso de estudio, pero no la descarta, destacando el valor de todo caso de estudio en un ámbito real. Al mismo tiempo es importante comprender que el experto fue tipificado, esperando obtener diferentes precisiones al variar el perfil del experto [20].

5 Conclusión final

Como ha sido anticipado por Jorgensen [1] existen situaciones donde se puede esperar que las estimaciones expertas sean más precisas que los métodos formales de estimación. En el caso de estudio presentado el experto tiene una alta experiencia en el desarrollo de software, un alto conocimiento de la tecnología y un conocimiento medio en el dominio y en el rendimiento del grupo. Estos aspectos favorecieron la precisión de sus estimaciones.

Sorprendió que en la historia sucesiva de estimaciones no se obtuviera una mejora en la precisión. Se cree que esto se debe a la imprecisión de los ajustes humanos, aunque se muestra un aprendizaje en la sucesión de estimaciones, logrando en la estimación final un valor similar obtenido en la estimación de todos los casos de uso de la aplicación.

En el caso de los métodos formales de estimación, se usaron modelos de una sola variable: Tamaño o Complejidad. Esto afecta la precisión de los modelos dado que si bien estos factores son los más importantes, sus variaciones no explican en un alto porcentaje la variación del esfuerzo.

El presente trabajo aporta un caso de estudio usando métricas no relevadas en Jorgensen [1] y pone en evidencia la obtención de unos resultados particulares destacando las características del experto en un proyecto complejo del ámbito industrial. Para poder generalizar sus resultados es necesario analizar otros productos de diversos dominios, incorporar expertos con perfiles diferentes y otros métodos formales.

Sería interesante en trabajos futuros trabajar con modelos que incluyan otras variables que afecten la estimación de esfuerzo y variar el perfil de los expertos, particularmente observar la variación de la precisión del experto al tener un conocimiento del rendimiento de los perfiles del grupo de desarrollo y del dominio alto.

Agradecimientos. El presente proyecto se ha realizado con el apoyo de la Universidad Austral y la Universidad Argentina de la Empresa.

Referencias

1. Jorgensen, M.: A review of studies on expert estimation of software development effort. Journal on System and Software, Vol. 70, No. 1-2, 37-60 (2004).
2. Jorgensen, M. y Shepperd.. A systematic review of software development cost estimation studies, IEEE Transactions on Software Engineering, Vol. 33, No. 1. p. 3-53, (2007).
3. Montgomery, D, Peck, E.A., Vining, G.G.: Introducción al análisis de regresión Lineal, Compañía Editorial Continental (2004)
4. Shepperd, M., Schofield, C.: Estimating Software Project Effort Using Analogies. IEEE Trans. on Software Eng., vol. 23, no. 11, pp. 736-743, Nov. (1997).
5. COSMIC – Common Software Measurement International Consortium, 2009: The COSMIC Functional Size Measurement Method - version 3.0.1 Measurement Manual (The COSMIC Implementation Guide for ISO/IEC 19761: 2003), May (2009).
6. COCOMO II Model Definition Manual. http:// csse.usc.edu/csse/research/COCOMOII/ cocomo_downloads.htm
7. Robiolo, G., Badano, C., Orosco, R.: Transactions and Paths: two use case based metrics which improve the early effort estimation. ACM-IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM '09), Lake Buena Vista, FL, October (2009).
8. Lavazza, L., Robiolo, G.: Introducing the Evaluation of Complexity in Functional Size Measurement: a UML-based Approach. Symposium on Empirical Software Engineering and Measurement (ESEM), Boszano-Bozen, Italia, Sept 16-17 (2010).
9. McCabe, T. A: Complexity measure, IEEE Transactions on software Engineering, Vol. SE-2, NO. 4. (1976)
10. Jorgensen, M and Gruschke, Tanja M.: The role of outcome feedback in improving the uncertainty assessment of software development effort estimates. ACM Trans. Softw. Eng. Methodol. 17, 4, Article 20 (August 2008), 35 pages. (2008)
11. Jorgensen, M. and Halkjelsvik, T: The effects of request formats on judgment-based effort estimation, Journal of Systems and Software, 83 (1), 29-36. (2010)

12. Mendes, E.: Improving software effort estimation using an expert-centred approach. In Proceedings of the 4th international conference on Human-Centered Software Engineering (HCSE'12), Winckler, M., Forbrig, P. and Bernhaupt R.(Eds.). Springer-Verlag, Berlin, Heidelberg, 18-33 (2012).
13. Cuadrado-Gallego, J.J., Rodríguez-Soria, P. and Martín-Herrera B.: Analogies and Differences between Machine Learning and Expert Based Software Project Effort Estimation. In Proceedings of the 2010 11th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD '10). IEEE Computer Society, Washington, DC, USA, 269-276 (2010).
14. Jørgensen, M.: Forecasting of software development work effort: Evidence on expert judgement and formal models , International Journal of Forecasting 23 449–462 (2007)
15. Jorgensen, M. and Boehm, B.: Software Development Effort Estimation: Formal Models or Expert Judgement? IEEE Software (March/April):14-19 (2008).
16. Hughes, R.T.: Expert judgement as an estimating method, Information and Software Technology, Volume 38, Issue 2, 67-75, (1996).
17. MacDonell, S. G., & Shepperd, M. J.: Combining techniques to optimize effort predictions in software project management. Journal of Systems and Software, 66(2), 91−98 (2003).
18. Fenton, N.E. and Pfleeger, S.L.: Software Metrics. PWS Publishing Company (1997)
19. Jørgensen, M., Indahl, U and Sjøberg, D.: Software Effort Estimation by Analogy and "Regression Toward the Mean". Journal of Systems and Software 68(3): 253-262 (2003)
20. Halstead, S., Ortiz, R., Córdova, M. and Seguí, M.: The impact of lack in domain or technology experience on the accuracy of expert effort estimates in software projects. In Proceedings of the 13th international conference on Product-Focused Software Process Improvement (PROFES'12), Dieste, O., Jedlitschka, A. and Juristo, N. (Eds.). Springer-Verlag, Berlin, Heidelberg, 248-259, (2012).

ANEXO II - Expert Estimation and Historical Data: An Empirical Study

Por Gabriela Robiolo, Silvana Santos y Bibiana Rossi

Expert Estimation and Historical Data: An Empirical Study

Gabriela Robiolo
Universidad Austral
Av. Juan de Garay 125
Buenos Aires, Argentina
grobiolo@austral.edu.ar

Silvana Santos
Universidad Nacional de La Plata
Calle 50 y 20
La Plata, Argentina
silvanasantos@gmail.com

Bibiana Rossi
Univ. Argentina de la Empresa
Lima 717
Buenos Aires, Argentina
birossi@uade.edu.ar

Abstract— **Expert estimation is the estimation strategy which is most frequently applied to software projects; however, this method is not very much reliable as the accuracy of the estimations thus obtained is always influenced by the level of experience of the expert. As part of the experts' experience is made up by the information they obtain from historical data, we wanted to learn about the value such historical data has for an expert estimator. To do so, we designed an empirical study. We compared the accuracy of the estimations made with several estimation methods based on productivity, size, and analogies which use historical data, to that obtained with expert estimation. We used two similar applications; one was used as the target application and the other one was used to obtain historical data. The results show that the accuracy of expert estimation is affected by the expert's work experience, the level of experience he/she has in the technologies to be used to develop the applications, and his/her level of experience in the domain of the applications. The use of historical data may improve the intuitive expert estimation method when the work experience, the experience in the technologies to be used to develop the application, and the experience in a given domain is low, as well as when the team velocity is unknown.**

Keywords—Expert; Expert Estimation; Effort Estimation; Empirical Study; Historical Data.

I. INTRODUCTION

Expert estimation is the estimation strategy which is most frequently applied today to estimate the effort involved in the development of software projects, and this is so because there is evidence in favor of using it [1]. However, the estimations thus obtained are far from being as accurate as we would like them to be so, if we expect to improve estimation accuracy, further research should be carried out in order to understand how the estimation process works.

With this goal in mind, we found out that the compilation of information about cost estimation made by Jørgensen and Shepperd [2] in 2004 is extremely valuable, since they systematically reviewed papers already written on cost estimation studies and they provided recommendations for future research. They found out that there are few researchers working in this field and that there is no adequate framework to develop high quality research projects that may lead to concluding evidence. Consequently, they suggested the following improvements in the field of research: (a) deepen the study of the basic aspects of software estimation, (b) widen the research on the current, most commonly used estimation methods in the software industry, (c) perform studies that support the estimation method based on expert judgment, instead of replacing it with other estimation methods and (d) apply cost estimation methods to real situations.

As we completely agree with their diagnosis, we believe research on expert estimation has become mandatory, if more accurate estimations are to be obtained.

As far as we know, expert estimation may be said to be based on both intuition, which is acquired by the developer through his daily work experience, and analogy. In fact, such analogy will be made by using both the information the estimator has in his memory and the historical data he may obtain [2]. Although all experts are expected to have some experience, the types of experience they have may be very different, and their estimation performances will surely be different too. Besides, even in cases in which the expert is supposed to have wide experience, there will be factors that will undoubtedly affect his estimations. For example, the domain where the software estimation must be made could be new to him, the team he would work with may have been recently created or the technological environment may not have been previously used.

In Agile contexts, in particular, there is another critical aspect to be dealt with: not knowing the velocity at which the developing team works. Actually, Cohn [3] suggested that one of the challenges when planning a release is estimating the velocity of the team. He mentioned three possible ways to estimate velocity. Firstly, estimators may use historical averages, if available. However, before using historical averages, they should consider whether there have been significant changes in the team, the nature of the present project, the technology to be used, and so on. Secondly, estimators may choose to delay estimating velocity until they have run a few itera-

tions. Cohn thinks that this is usually the best option. Thirdly, estimators may forecast velocity by breaking a few stories into tasks and calculating how many stories will fit into the iteration.

Bearing in mind the present working conditions, as described in the two previous paragraphs, and in order to deepen our knowledge about expert estimation, as recommended by Jørgensen and Shepperd [2], we decided to research on the importance of historical data when performing expert estimations in agile contexts in which the project domains and the technological environments are new to the team, and the teams -with little experience in Agile contexts- have recently been created, so the team velocity is unknown.

In this scenario, we tried to answer the following research question: *when may the accuracy of an expert estimation made in a context of agile software development be improved by using historical data?* The results we obtained through our empirical study have led us to conclude that historical data may improve the accuracy of an intuitive estimation made by an expert when the estimator has limited experience in the job to be performed, the technologies to be used and the domain to be dealt with, and when the team velocity is unknown.

In section two, we will introduce three estimation methods: Expert Estimation (ExE), Analogy-Based Method (AbM), and Historical Productivity (HP). In section three, we will describe an empirical study and analyze the results obtained. In section four, we will investigate related work to see if there is any other evidence of improvement in expert estimation accuracy when using historical data, and finally, in section five, we will draw conclusions as regards the evidence of the benefits of using historical data.

II. ESTIMATION METHODS

This section will describe the three estimation methods used in our empirical study: ExE, AbM and HP. However, before doing so, it is important to focus on the definition of certain expressions used to define such methods. For example, when defining expert, Jorgensen [1] used a broad definition of the phrase, as he included estimation strategies that ranged from unaided intuition ("gut feeling") to expert judgment supported by historical data, process guidelines, and checklists ("structured estimation"). In his view, for an estimation strategy to be included under the expert estimation category, it had to meet the following conditions: first, the estimation work must be conducted by a person recognized as an expert in the task, and second, a significant part of the estimation process must be based on a non-explicit and non-recoverable reasoning process, i.e., "intuition". In our study, however, a narrower definition of the concept of expert was used: that which refers only to intuition. This way, we made a difference between intuitive ExE, and the methods that involve the use of historical data: AbM and HP. It is important to note that in our study, when we used Planning Poker –an ExE method-, no historical data was taken into account.

To further clarify the terms used, we must say that by AbM we meant the estimation performed by an expert, who is aided by a database containing information about finished projects [4]. As regards HP, which is another way of using historical data, it is worth mentioning that in our empirical study we focused on the size characteristic of the products, as suggested by one of the authors that inspired this article [4].

A. *Expert Estimation Method (ExE)*

When estimating the effort of a software development task, an expert estimation may be obtained either by a single expert, whose intuitive prediction will be considered an expert judgment, or by a group of experts, whose estimation will combine several experts' judgments.

A very frequently used way to obtain group expert judgment is called Planning Poker, a technique that combines expert opinion, analogy, and disaggregation. It is based on the consensus that is reached by the group of experts who are performing an estimation; in fact, it is considered a manageable approach that produces fast and reliable estimations [3][5][6]. This method was first described by James Greening [8] and it was then popularized by Mike Cohn through his book "Agile Estimating and Planning" [3]. It is mainly used in agile software development, especially in Extreme Programming *[7]*. To apply Planning Poker, the estimation team should be made up of, ideally, all the developers within the team, that is, programmers, testers, analysts, designers, DBAs, etc. It is important to bear in mind that, as this will happen in Agile contexts, the teams will not exceed ten people [3]. In fact, Planning Poker becomes especially useful when estimations are taking too long and part of the team is not willing to get involved in the estimation process [8]. The basic steps of this technique, according to how Grenning described them, are:

"The client reads a story and there is a discussion in which the story is presented as necessary. Then, each programmer writes his estimation on a card, without discussing his estimation with anyone else. Once every programmer has written down his estimation, all the cards are flipped over. If everybody has estimated the same, there is no need for discussion; the estimate is registered and the next story is dealt with. If the estimates are different, the team members will discuss their estimates and try to come to an agreement" [8].

Mike Cohn further developed this technique: he added a pack of cards especially designed to apply this technique and he shaped the whole process: each estimator is given a pack in which there are cards that have numbers written on them Those numbers represent a valid estimation, such as 0, 1, 2, 3, 5, 8, 13, 20, 40, and 100. Each pack has to be prepared before the Planning Poker meeting and the numbers should be big enough to be seen from the other side of the table. There is a raison d'être for the estimation scale presented above. There are studies which have demonstrated that we are better at estimating things which fall within one order of magnitude [9][10], so these were the cards that were employed when Planning Poker was

used in the empirical study reported in this article. It should be noted that no historical data was used when estimating with Planning Poker for our study.

B. *Analogy-Based Method (AbM)*

The idea of using analogy as a basis to estimate effort in software projects is not new: in fact, Boehm [11] suggested the informal use of analogies as a possible technique thirty years ago. In *1988*, Cowderoy and Jenkins [12] also worked with analogies, but they did not find a formal mechanism to select the analogies. According to Shepperd and Schofield [13], the principle is based on the depicting of projects in terms of their characteristics, such as the number of interfaces, the development methodology, or the size of the functional requirements. There is a base of finished projects which is used to search for those that best resemble the project to be estimated.

So, when estimating by analogy, there are *p* projects or cases, each of which has to be characterized in terms of a set of *n* characteristics. There is a historical database of projects that have already been finished. The new Project, the one to be estimated, is called "target". Such target is characterized in terms of the previously mentioned *n* dimensions. This means that the set of characteristics will be restricted to include only those whose values will be known at the time of performing the prediction. The next step consists of measuring similarities between the "target" and the other cases in the *n*-dimensional space [14].

Such similarities may be defined in different ways, but most of the researchers define the measuring of similarities the way Shepperd & Schofield [13] and Kadoda, Cartwright, Chen & Shepperd [14] do: it is the Euclidean distance in an *n*-dimensional space, where *n* is the number of characteristics of the project. Each dimension is standardized so that all the dimensions may have the same weight. The known effort values of the case closest to the new project are then used as the basis for the prediction.

In our empirical study, we applied AbM in its simplest version. The participants compared the user stories of two projects: one considered "historical" and the other one "target". The Estimated Effort (EE) of the user story of the target project was, in fact, the Actual Effort (AE) of the "most similar" user story of the historical project. Actually, no specific characteristics of the user stories were specially taken into account.

C. *Historical Productivity*

Jørgensen, Indahl, and Sjøberg *[4]* defined Productivity as the quotient of Actual Effort (AE) and Size, and the EE as the product of Size and Productivity. In this empirical study, COSMIC [15] was used as a measure of Size, and EE was calculated as the product of Size and Historical Productivity (HP). The HP is the value of productivity of the project to be used as historical project, that is, the quotient of the AE and the Size of the historical project.

To measure size, COSMIC was selected because it is an international standard [16] that is widely recognized in the software industry, and also because there is a previous study that used it in an Agile context [17]. With the COSMIC software method, the Functional User Requirements can be mapped into unique functional processes, initiated by functional users; in fact, user stories are actually used in this paper. Each functional process consists of subprocesses that involve data movements. A data movement concerns a single data group, i.e., a unique set of data attributes that describe a single object of interest. There are four types of data movements: a. an Entry moves a data group into the software from a functional user, b. an Exit moves a data group out of the software to a functional user, c. a Read moves a data group from persistent storage to the software, and d. a Write moves a data group from the software to persistent storage.

In the COSMIC approach, the term "persistent storage" denotes data (including variables stored in central memory) whose value is preserved between two activations of a functional process.

The size expressed in CFP is given by the equation CFP = Entries + Exits + Reads + Writes, where each term in the formula denotes the number of corresponding data movements. So, there is no concept of "weighting" a data movement in COSMIC, or, equivalently, all data movements have the same unit weight.

III. Description of our Empirical Study

Our empirical study is described in this section, considering its conception, how it was planned, the particularities of its execution and the results obtained.

A. *Definition*

This empirical study was designed in order to establish when the accuracy of an expert estimation made in a context of agile development, under the circumstances that will be described below, may be improved by using historical data. Such circumstances are: the project domain and the technological environment must be new to the estimator, and the team would have recently been created, so that the team velocity will be unknown.

The development steps of this empirical study may be summarized as follows:

The study was developed in the context of graduate education for IT practitioners from different educational and work backgrounds. The participants attended a workshop which had two objectives, one oriented to the subjects and another one oriented to the development of this empirical study. The workshop gave the participants the opportunity to: a. understand both how a historical database is built, and under which circumstances such database will give value to the estimation process, b. estimate using three methods and c. compare their results with other participants' results. Later on, the same workshop was conducted for undergraduate students.

The workshop participants were asked to re-estimate the first spring of an application that had been previously developed by a group of undergraduate students who did not participate of the workshop. The selected application had been developed using a development language un-

known by the workshop participants and the application belonged to a domain the latter knew little of. The original team velocity was not reported to the participants, to simulate that it was unknown.

The re-estimations were made using three different estimation methods: ExE, based on the participants' intuition, and two other methods which use historical data. The historical data was obtained from a similar application that had been developed by a third undergraduate group – a group that had neither developed the original application nor participated of our empirical study-.

To guarantee the best results, we followed the recommendations of Juristo and Moreno [18] and Wohlin et al. [19] in order to develop this empirical study. To report it, we took into account Jedlitschka, Ciolkowoski and Pfahl's guidelines for reporting empirical research in software engineering [20].

As previously stated, the objective of this empirical study was to analyze when the accuracy of an estimation made by an expert, based on his personal intuition, may be improved by using historical data. This objective was achieved by comparing the estimation errors obtained by two different groups: undergraduate students and practitioners, when estimating using three different methods: ExE, AbM, and HP.

In fact, the hypotheses to be tested were:

H_0: The mean value of the MRE calculated with the ExE is equal to the mean value of the MRE obtained when calculating with AbM or HP.

H_1: The estimated mean value of the MRE calculated with the ExE is lower than the mean value of the MRE obtained when calculating with AbM or HP.

B. Planning

The *experimental subjects* were IT graduate students and undergraduate advanced students of Informatics Engineering. In fact, all of the graduate students were practitioners. So, in this paper, when we say "participants" we mean both the graduate and undergraduate students, and by "practitioners" we refer only to the graduate students.

The participants were asked to give some information about themselves regarding the following aspects:

- If graduate or undergraduate student
- Professional experience (they had to state the number of years they had worked in software development)
- Experience with COSMIC
- Experience with user stories (they had to inform the number of user stories that they had written/read (fewer than 20, 20-100, more than 100)
- Experience with Ruby [21] language.
- Experience in Database development
- Experience in working in Agile development contexts.
- Level of prior knowledge about the productivity of the teams that developed the experimental objects (high, medium, low)
- Level of experience in the technologies used to develop the experimental objects (high, medium, low)
- Level of experience in the domain of the experimental objects (high, medium, low)

The *experimental objects were* two similar applications (P1 and P2), which were social networks. The first application was a system through which users may conduct surveys. The system classifies users into several categories, builds different groups and instantly surveys those users who fall within the right categories. It was developed by a team of undergraduate students who registered the estimated and actual hours using the Scrumy tool [22], and who were supervised by two professors.

The second application, which we identified as the "target project", was a network where different types of events may be published. For example, an event may be a party, a meeting or a football game. Events are the core elements in this application, not people. It works with event and friend suggestion algorithms and gives the option of buying a ticket for an event online.

The data corresponding to the experimental objects are displayed below. Table 1 shows the user stories of P1 and the Actual Effort (AE) of each user story measured in man hours. As some user stories were not functional processes, they were discarded. Table 2 shows the user stories of P2, which are the user stories of only the first sprint, as it was the only sprint for which effort was estimated.

As regards the counting of the man-hours worked on P1 and P2, one of the tasks within the assignment the undergraduate students that developed the projects had to undertake was to register the hours worked. These two groups did not participate in the empirical study; in fact, they were undergraduate students from a university different from the one where the undergraduate participants studied. The applications were developed in an Agile context, as an assignment in a practical subject. They first estimated the work to be done and then compared their estimations to their real effort. Two professors supervised these tasks. This empirical study used the actual effort of P1 and P2 and the estimated effort of P2 (obtained by the original development group), so that they may be compared to the participants' results.

The aspects of the development process that were controlled to facilitate such comparison were:

- Similarity: Two similar applications that had been developed in Agile contexts were selected as experimental objects. They had been developed in an academic context by advanced undergraduate students, who had been requested to develop an application for an assignment in which a company environment was simulated.
- Experience in team velocity: Since in Agile contexts developers learn from previous estimations, and in this case the estimators were expected to

have no previous experience, only the first sprint of the target application could be estimated in order to be compared to the actual effort estimation of P2, as it was only for the first sprint that the original P2 estimators did not have experience in team velocity.

- Language experience: Participants with experience in Ruby language, in Agile contexts, and / or COSMIC were equally distributed.

In order to obtain comparable results in this study, man-hours had to be used to unify the unit of measurement of effort, as the historical values had been previously measured in man-hours, instead of in story points or ideal hours, which are the measures usually used to make effort estimations with Planning Poker in Agile contexts [3].

The workshop was run following these steps:

1) The participants were given a set of materials that included: Brief Vision Documents [23] of P1 and P2, the professor's slides explaining the empirical study, and an Excel file where each sheet was a step of the empirical study.

TABLE I. DATA OF THE APPLICATION TO BE USED AS HISTORICAL INFORMATION

P1	**Actual Effort [man-hours]**
Create survey	18
Sign up	15
See user's profile	9
Answer survey	9
Log in/Log out	6
Comment on survey	12
Search for survey	9
Eliminate user	3
Edit personal data	6
Search for user	9
Generate and publish statistics	30
Follow user	30
Select user segment	18
Sort the content according to date	18
Upload pictures	21
UPR (User Popularity Ranking)	36

TABLE II. USER STORIES OF THE TARGET APPLICATION

P2
Create, Modify and Eliminate User
Log in (Log out)
Create event
Search for event

2)Each one of the empirical study steps was explained to the participants. The participants were trained to perform each activity. Also, two examples of COSMIC measurement were included.

It is important to note that the participants worked with an Excel file that was designed to facilitate the understanding of the activities, and the sequence in which they had to do them. The following are the activities presented sequentially in each one of the sheets in the file:

a) *Perform the expert estimation*, based on their intuition, they estimated the man hours to be worked on the target application (P2). Based on the Vision Document of P2, the participants estimated the EE of each user story described in Table 2.

3)Build the historical database. The participants measured the size of the user stories of the historical application (P1) by using COSMIC, as shown in Table 1. The Excel sheet automatically calculated the Historical Productivity (HP) of P1 as the quotient of AEP1 and SizeP1, where AEP1 is equal to the sum of the AE of each user story of P1, and SizeP1 is equal to the sum of the Size of each user story of P1. The data movements of P1 were indentified for each user story, based on: the information included in the Vision Report, the name of the user story, and the explanation given by the leader of the workshop when asked for it. The measurement of the user stories, using COSMIC, was performed in a way similar to that of [17].

b) *Measure the size of the target application (P2), by using COSMIC to measure the size of the user stories.* These size values were automatically used to calculate $EE_{P1,}$ which was calculated as the product of $Size_{P1}$ and Historical Productivity (HP_{P2}).

c) *Estimate the effort for the target application (P2) using AbM.* The participants had to select for each one of the user stories in P2 the most similar user story from the set of user stories in P1 -though based on their characteristics, not on their size- and then assign to the Estimated

Effort (EE) of each user story in P2 the AE of the similar user story in P1.

d) *Individually compare and analyze the EE values obtained using ExE, AbM, and HP methods*. The Excel sheet automatically presents a Table which displays the three EE values –those obtained by applying the three different estimation methods- for each user story in P2.

3) The participants estimated the effort of the target application following the steps listed above, and completed the worksheets.

4) The data was collected and the results were analyzed with the participants. A rich discussion about the comparison of the MRE obtained by applying the three estimation methods (ExE, HP and AbM) was conducted by the leader of the empirical study.

C. Execution

The characteristics of the participants are described in Table 3.

Forty nine undergraduate students, who were distributed in fourteen groups of 3-4 students, participated in the two workshops. The median work experience of the students was three years. No one had experience using COSMIC, and they had little experience with user stories. All of them had approved the course "Database" and only 8 had experience in working in an Agile context, that is to say, a small proportion of them. The Level of experience of the development teams in the technologies to be used and in the domain of the experimental objects was low. In one of the workshops, fourteen practitioners worked on their own. The median work experience of the practitioners was fourteen years. No one had experience in using COSMIC, and five of them had experience with user stories.

Their median experience in "Database" was ten years and only three of them had experience in working in an Agile context, which is a small proportion. The Level of experience in the technologies and in the domain of the experimental objects was medium-low.

Table 4 shows the effort estimation values of the target project, obtained by the two groups applying the three estimations methods: ExE, HP, and AbM. Moreover, the AE of the student group that developed the target application (P2) was 35 man-hours.

Figure 1 shows the boxplots of the residuals and Figure 2 the boxplots of the MRE for the target project. To obtain the MRE, the actual value registered for the first sprint of P2 by the group that actually developed the project was used as AE.

TABLE III. WORKSHOP PARTICIPANTS

Type	**Number**	**Work Experience (Years)**	**Experience using COSMIC**	**Number of User Stories [<20, 20<US<100, >100]**	**Database Experience**	**Experience with Ruby Language**	**Work experience in Agile context**	**Experience in the technologies**	**Experience in the domain**
Under graduate	49 (14 groups)	[0-13] Median: 3	No one	<20: 44 20<US<100: 3 >100: 2	All had approved the Course "Database"	No one	Only 8	Low: 47 Average: 2 High: 0	Low: 43 Average: 4 High: 2
Practitioners	14	[4-36] Median: 14	No one	<20: 9 20<US<100: 3 >100: 2	Database experience measured in years [0-36] Median:10	Only one	Only 3	Low: 9 Average: 5 High: 0	Low: 11 Average: 3 High: 0

The boxsplots show the different results obtained by each group of participants. The undergraduate participants obtained better estimation results when applying the AbM, rather than the ExE and HP methods. Figure 2 shows the median values, but it must be noted that a more significant difference was observed when comparing the values obtained for the mean MRE in the undergraduate group: AbM: 69.80, ExE:151,43 and HP:175,04. On the other hand, the practitioners group obtained the best results when applying ExE, instead of HP and AbM, as shown by the boxplots. Also, their mean values were ExE: 29.09, HP: 205.16, and AbM: 87.14.

D. Threats to validity

The difference in background of the experimental subjects is the major weakness of this empirical study. However, this drawback may be transformed into a strength if we consider that in this empirical study the experience of the expert is stressed, showing that the accuracy of an expert estimation depends on the estimator's expertise, which is measured by his work experience, his level of experience in the technologies used to develop the experimental objects and his level of experience in the domain of the experimental objects.

Another threat is that the expert estimations were made in two different manners: either alone or in groups. The practitioners worked alone and the undergraduate students formed groups of three or four persons and used Planning Poker to obtain the expert values. However, we think that this combination of expert methods, that is, using Planning Poker or not, did not introduce bias in this study, in accordance with what was reported in [24].

Unfortunately, only a brief explanation about COSMIC was given to the undergraduate students, since it was not possible to give an extensive explanation, as there was not enough time to do so (the whole workshop was three hours long). Thus, the little available time was devoted to those COSMIC characteristics that were necessary for them to know in order to make a correct measurement. However, this did not seem to be a big problem, as the concept of data movement is quite intuitive for all the participants and the medians of the errors shown in both Figure 1 and 2 for the HP method are similar.

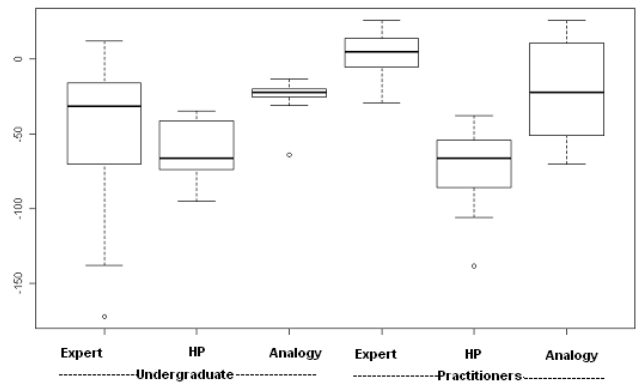

Fig. 1. Boxplots of the residuals of the target project

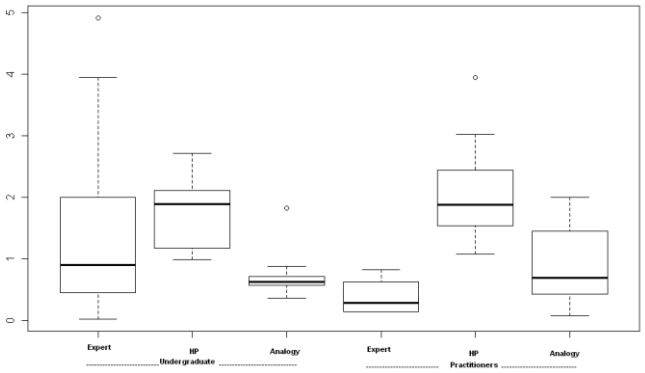

Fig. 2 Boxplots of the MRE of the target project

Also, the use of examples and previous training in Function Points made it easier for the participants to understand how to use this measuring method. On the other hand, the practitioners had been previously trained in COSMIC, so they presented no difficulty. Besides, if anybody had any doubts, the person who led the empirical study would give further explanations.

The order in which the estimations were performed could have introduced bias in the result, so it would have been more convenient if the participants had not performed the estimations in the same order, except for ExE, which must always be performed in the first place.

The selection of a similar application to make up the historical database is clearly an advantage in order to obtain a better estimation, but the problem is that sometimes the estimator does not have data about similar applications at hand, so she or he has to use an application from a different domain. This circumstance may vary the results obtained in this empirical study.

The experimental subjects were identified either as undergraduates or practitioners. However, it may be argued that more categories would have been necessary, as some of the practitioners had more experience in the domain or in the technologies than some others. Consequently, to obtain more evidence of the benefit of using historical data, it is necessary to have a bigger number of estimators, which would allow us to identify different levels of expertise, for example, three expertise levels for practitioners and three for undergraduates.

To conclude, as the experimental objects used in the empirical study came from a particular environment and the experts' experience did not cover the big spectrum of expertise that exists, general conclusions cannot be drawn because there may be different estimation problems in different environments and experts' performances.

IV. DATA ANALYSIS AND INTERPRETATION

To answer the research question posed above, it is important to understand the circumstances under which the use of historical data may improve the expert estimation accuracy. In this empirical study, two types of experts

were involved: we called them undergraduate and practitioner participants. Consequently, each group will be separately analyzed first, then the statistical significance of the results will be dealt with, and afterwards, the research question will be answered. Finally, this will be completed with the discussion of aspects omitted in the previous sections.

A. Result analysis

1) Undergraduate

We noticed that there were three aspects that affected the intuitive expert estimation: the work experience, the level of experience in the technologies used to develop the experimental objects, and the level of experience in the domain of the experimental objects. The undergraduate participants' work experience measured in years varied from 0 to 13, with a median of 3. This shows that the "experts" had little experience in estimations and also, that the level of experience in the technologies used and in the domain was low.

Their best result was obtained when using AbM: the MRE median was 63% within a [37%-183%] range. The lack of experience, in this case, was compensated for by the historical data.

By using HP, the MRE dispersion was increased: the MRE values ranged from [99%-272%]. The MRE of the 14 groups had a median of 189% and a standard deviation of 54%.

2) Practitioners

When compared to the undergraduate participants, the most significant difference was their work experience:

TABLE IV. EE OF THE TARGET PROJECT

Participants	**Number of estimations**	**ExE %**	**HP %**	**AbM %**
Undergraduates	14 (made by groups of 3-4 undergraduate students)	161.00	110.00	57.00
		61.00	74.70	57.00
		34.00	76.30	60.00
		65.00	69.72	48.00
		207.00	84.12	48.00
		85.00	106.13	55.00
		173.00	90.21	66.00
		68.00	102.84	57.00
		79.00	101.15	57.00
		56.00	101.44	51.00
		51.00	72.00	57.00
		32.00	130.15	57.00
		105.00	108.93	99.00
Practitioners	14	11.00	108.94	11.00
		30.00	173.22	24.00
		21.00	84.77	20.00
		30.00	122.15	60.00
		9.00	90.55	9.00
		64.00	85.96	39.00
		30.00	120.61	105.00
		29.00	111.87	86.00
		16.00	72.88	32.00
		30.00	105.05	95.00
		40.00	88.93	57.00
		40.00	97.07	94.00
		49.00	92.37	70.00
		57.00	140.94	57.00

measured in years, it varied from 4 to 36, with a median of 14. Ten practitioners were project leaders or managers, three were senior developers and only one was a junior developer. This shows that these "experts" had experience in project management and, of course, in estimations.

The practitioners' level of experience in the technologies used to develop the experimental objects and the level of experience in the domain of the experimental objects was medium-low. These characteristics justify the results obtained when using ExE.

During the study, three of them did not perform the expert estimation because they considered that they were no "experts", while two of them assigned to the expert estimation the same value they had assigned to the AbM estimation. Seven of the eleven practitioners that applied

pure expert estimation estimated with an MRE lower than 25%.

The estimation by AbM had a MRE median of 70 % in a range result of [8.57%-200%], which is a result similar to that obtained by the undergraduates.

By using HP, the MRE dispersion was increased: [108.22%-384.91%]. The MRE of the 14 practitioners had a median of 189% -similar to that of the undergraduate value-and a big standard deviation of 75%, which may have been caused by the subjectivity introduced by COSMIC, originated by the practitioners' different backgrounds.

3) The statistical significance of the results

The Wilcoxon rank test, at a significance level of 0.05, was used to analyze the statistical significance of the results. This non-parametric test was selected because the distributions of the variables were not normal. It was applied to test the accuracy of ExE versus that of HP or AbM, according to the results obtained by each group (practitioners and undergraduate participants). The MRE and the absolute residuals were used. Table 5 shows the p-value of each subset, when using the MRE. The results obtained when using the absolute residuals are not shown because there is no significant difference.

TABLE V. STATISTICAL SIGNIFICANCE

Groups	**ExE vs:**	**MRE**
Undergraduate	HP	0.162
	AbM	0.948
Practitioners	HP	0.000
	AbM	0.022

When analyzing the MRE obtained by:

- the practitioners, when comparing ExE to HP, it was possible to reject H0 in favor of H1.
- the practitioners, when comparing ExE to AbM, once again, it was possible to reject H0 in favor of H1.
- the undergraduates, when comparing the ExE method to HP, it was not possible to reject H0 in favor of H1.
- the undergraduates, when comparing the ExE method to AbM, it was not possible to reject H0 in favor of H1.

It should be noticed that the three practitioners who did not use the method, as they did not consider themselves to be "experts", were also included in the table. However, later on, the Wilcoxon rank test was also computed, but this time only for the eleven practitioners who made the estimations, and the results did not vary.

Now we can answer the research question: W*hen may the accuracy of an expert estimation made in a context of Agile software development be improved by using historical data?*

These results show that the expert estimation was not improved by the use of historical data when the expert had some work experience, and his level of experience in the technologies used to develop the application, and his level of experience in its domain were medium-low.

However, we found out that historical data may improve the expert estimation when the estimator's work experience, his level of experience in the technologies used to develop the application, and his level of experience in the domain of the application to be developed is low.

4) Discussion

There are some aspects that have not been mentioned yet, but it is worth doing so now. One of them is the little experience in Agile development contexts that the two groups had. We think that this fact did not affect the results obtained because, as the work experience of the undergraduate group was small, their experience in Agile contexts was small too. On the other hand, practitioners were experienced in project management and estimations, so this compensated for their little experience in Agile contexts. On top, as the empirical study was designed to only use the first sprint of a software product development, no estimations were made for the rest of the sprints -which would be usually done when using an Agile method- so their little experience in Agile contexts had no impact on our study.

Another interesting aspect is that most of the effort calculations proved to be underestimated, which may be seen in Figure 1. This could be explained by the fact that almost all the participants did not have previous experience with the Ruby language. On the contrary, the group that developed the target application had previous knowledge of the velocity that they could achieve because they had done a Ruby on Rails tutorial before. Consequently, the level of experience of this group in the technologies used to develop the target application and the level of experience in the target application domain was medium-high, which justifies the accuracy of the estimation: 3% MRE, which was high. At the same time the group that developed the target application had a higher velocity than the group that developed the historical application. Obviously, the bigger the difference in the velocity, the bigger the error in the effort estimation.

One question that may arise is: how would the participants be able to make meaningfully expert estimations if they did not have any knowledge about the developers? This condition was part of the scenario that we were simulating; as it was stated in the introduction of this paper, the team velocity was unknown.

Figure 2 shows that the medians obtained by the two groups when estimating with HP were similar, but their

standard deviations were not: the standard deviation of the MRE for the undergraduate group was 53.7 and 75 for the practitioners. This is a consequence of the subjectivity introduced by the COSMIC measurement of both the historical user stories and the user stories to be estimated. The estimation was affected by the subjectivity of the measurements and by the difference between the historical productivity of P1 and the actual productivity of P2.

Figure 2 shows that the MRE medians obtained when the two groups used the AbM method were similar but their MRE distributions were quite different. It was surprising to see that the results obtained by the practitioners using the AbM were worse than those obtained by the undergraduates. As the AbM is based on the selection of a "similar" user story, we may conclude that the undergraduate participants had a comparable concept of "similarity" to that of the original undergraduate group that developed the target application.

The estimation results obtained with the AbM and HP method would have been better if the historical data had been obtained from a similar project –one developed using Ruby on Rails- , but unfortunately, there was none available. Besides, the fact that the user stories that were not functional processes were discarded may have also influenced the results. In addition, another interesting factor that may have been considered is team size.

In our study, the empirical objects were two similar applications, but what would have happened if they had not been similar? Obviously, the results of the undergraduate group would have been affected, as their best results were obtained using the AbM. The reason is that such method is based on analogy, so if the degree of similarity between the application from where the historical data was to be obtained and that of the target application had been low, the accuracy of the estimation would have been poor too.

Moreover, although we only used the estimates of the first sprint of the target application this time, we believe the estimates of the following sprints could be used in future replications to evaluate if (and to what extent) expert estimations improve while participants gain knowledge of the projects (while AbM and HP are expected to yield constant accuracy throughout the sprints).

Finally, we may wonder about the participants' characteristics included in Table 3 and the reason why other characteristics were not included. To begin with, database experience is related to work experience, so it was necessary to check it because the COSMIC measurement would have been affected if experience in database had been small. In fact, the experience in using COSMIC was defined as a controlled variable. Moreover, the number of user stories the participants had written/read was included because it is related to their work experience in Agile contexts: in fact, there was a correlation between the number of user stories read/written and their experience in Agile contexts, which proved the consistency of the information. In addition, the level of experience with Rugby language and the level of experience in the technologies to be used had to be tested in order to verify if the participants fit our empirical study. Besides, the impact of the level of experience in the application domain was previously analyzed by [25]. We think that these characteristics have made the main differences between the two groups clear.

V. RELATED WORK

Apparently, this has been the first article to have been written about whether using historical data in an agile context improves expert estimation.

However, regarding expert estimation in general, there are some authors that have already reported evidence about the importance of the developers' level of maturity when evaluating the accuracy of estimations, which is in line with the conclusions of our study. For example, SCRUM pioneers believe it is acceptable to have an average error rate of 20% in their results when using the Planning Poker estimation technique, but they have admitted that this percentage depends on the level of maturity of the developers [25]. Another study [26] agrees with this statement, as it indicates that the optimism bias which is caused by the group discussion diminishes or even disappears as the expertise of the people involved in the group estimation process increases.

On the other hand, another study [27] has already examined the impact of the lack of experience of the estimators in the domain problem, as well as that in the technologies used in a software development project. In fact, what was studied was the accuracy with which the effort of a given task was estimated. Such estimation was performed by a single expert by comparing the estimated and the actual efforts. The reason for researching on this aspect is that, occasionally, organizations do not have in their staff experts that have relevant prior experience in some business or technology related aspect of the project they are working on. This research investigates the impact of such incomplete expertise on the reliability of estimates.

It is important to note that Jorgensen [1] has both defined a list of twelve "best practices", that is to say, empirically validated expert estimation principles, and suggested how to implement these guidelines in organizations. One of the best practices he proposed is to use documented data from previous development tasks and another one is to employ estimation experts with a relevant domain background and good estimation records. Actually, our article headed in the same direction; we focused on historical data and we analyzed the impact of the difference in experts' skills.

An aspect that should be taken into account when performing expert estimations is excessive optimism, as it is one of the negative effects that influences the most when a software project fails. Jørgensen and Halkjelsvik [28] have discovered something that seems to be important to understand what may be leading estimators to excessive optimism: the format used to word the question that asks about effort estimation. The usual way to ask about effort estimation would be: "How many hours will be used to complete task X?". However, there are people who would

say: "How many tasks could be completed in Y hours?". Theoretically, the same results should be obtained by using any of the two formats. Nevertheless, according to Jørgensen and Gruschke [29], when the second option is used, the estimations which are thus obtained are much lower than those obtained when the traditional format is used, that is to say, the time to fulfill a task will be shorter, and consequently, the estimation will be much more optimistic. Thus, in our study, the expert estimations were made using the usual question. In fact, the final recommendation of this study is that the traditional format should always be used, as this does not contain any deviation imposed by the clients who ask the developers for more than they can pay for.

VI. Conclusion and Future work

This paper specifically focuses on an agile context in which the project domain and the technological environments are new to the estimators, the teams have recently been created, and the team velocity is unknown. As under these circumstances historical data may become important, we tried to answer the following research question: *when may the accuracy of an expert estimation made in a context of agile software development be improved by using historical data?* To find out whether there is any advantage in using historical data when the historical velocity is unknown, an empirical study was developed in an Agile software development context.

Historical data seems to be valuable when the work experience, the level of experience in the technologies to be used to develop an application, and the level of experience in the domain of the application to be developed are low.

So, for estimators who have the restrictions described above, and who have no option but to work with them, we may suggest the following:

- Use intuitive expert estimations when your work experience, your level of experience in the technologies to be used to develop the application, and your level of experience in the domain of the application to be developed are not low.
- Use historical data when your work experience, your level of experience in the technologies to be used to develop the application, and your level of experience in the domain of the application to be developed are low.

In order to generalize this conclusion, a replication of this empirical study is recommended, especially if different software life cycle models [30], application domains, expert profiles, and levels of performance are included. Also, different estimation methods, such us linear regression or Analogies –next time, using the size characteristic- may be used. Finally, in order to enrich this empirical study, it would also be convenient to compare the estimation performed by an expert who has deep knowledge of this domain, and also knows the team velocity, to the estimations obtained by the participants of our study.

AKNOWLEDGMENTS

Our thanks to the Research Fund of Austral University, which made this study possible, and to Luigi Lavazza for his opportune comments.

REFERENCES

[1] M. Jorgensen, "A review of studies on Expert estimation of software development effort," Journal on System and Software, Vol. 70, No. 1-2, 2004, pp. 37-60.

[2] M. Jorgensen, and Shepperd, "A systematic review of software development cost estimation studies," IEEE Transactions on Software Engineering, Vol. 33, No. 1, January 2007, 2007, pp. 3-53.

[3] M. Cohn, Agile Estimating and Planning. Addison-Wesley, 2005.

[4] M. Jørgensen, U. Indahl, and D. Sjøberg, "Software effort estimation by analogy and regression toward the mean," Journal of Systems and Software, 68(3), 2003, pp. 253-262.

[5] T.J.Bang, "An Agile approach to requirement specification," Agile Processes in Software Engineering and Extreme Programming, SE:35, VL:4536, Lecture Notes in Computer Science, G. Concas,E. Damiani, M. Scotto, G.Succi, Eds., Springer Berlin Heidelberg, 2007, pp. 193-197.

[6] J. Choudhari and U. Suman, "Phase wise effort estimation for software maintenance: an extended SMEEM model," in Proceedings of the CUBE International Information Technology Conference, ACM, 2012, pp. 397-402,

[7] N.C. Haugen, "An empirical study of using Planning Poker for user Story estimation," Proceedings of AGILE 2006 Conference, Computer Society, IEEE, 2006, 9 pp. – 34.

[8] J. Grenning,. "Planning Poker or how to avoid analysis paralysis while release planning," 2002, DOI=, http://sewiki.iai.uni-bonn.de/_media/teaching/labs/xp/2005a/doc.planningpoker-v1.pdf: August, 2013.

[9] E. Miranda. "Improving Subjective estimates using paired comparisons," IEEE Software, 18(1), 2001, pp. 87–91.

[10] T. Saaty, Multicriteria decision making: the Analytic Hierarchy Process. RWS Publications, 1996.

[11] B. Boehm. Software Engineering Economics. Prentice Hall. 1981.

[12] A.J.C. Cowderoy and J.O. Jenkins, "Cost estimation by analogy as a good management practice," in Proc. Software Engineering 88. Liverpool: IEE/BCS, 1988, pp. 80-84,

[13] M. Shepperd, and C. Schofield, "Estimating Software Project Effort Using Analogies," IEEE Trans. on Software Eng., vol. 23, no. 11, 1997, pp. 736-743.

[14] G. Kadoda, M. Cartwright, L. Chen, and M. Shepperd. Experiences using Case-Based Reasoning to predict software project effort. Empirical Software Engineering Research Group Department of Compu-

ting Bournemouth University Talbot Campus Poole, BH12 5BB, UK. 2000.

[15] COSMIC – Common Software Measurement International Consortium, 2009, The COSMIC Functional Size Measurement Method - version 3.0.1. Measurement Manual (The COSMIC Implementation Guide for ISO/IEC 19761: 2003).

[16] ISO (2011) ISO/IEC19761:2011, Software Engineering -- COSMICFFP– A Functional Size Measurement Method, ISO and IEC.

[17] J. Desharnais, L. Buglione, and B. Kocatürk,. "Using the COSMIC method to estimate Agile user stories," *in Proceedings of the 12th International Conference on Product Focused Software Development and Process Improvement*, ACM, New York, 2011, pp. 68-73.

[18] N. Juristo and A.M. Moreno, Basics of Software Engineering Experimentation. Kluwer Academic Publishers., 2001.

[19] C. Wohlin, P. Runeson, M. Höst, M.C. Ohlsson, B. Regnell, and A. Wesslen, Experimentation in Software Engineering: an Introduction. Kluwer Academic Publisher, 2000.

[20] A. Jedlitschka, M. Ciolkowoski and D. Pfahl, "Reporting experiments in Software Engineering," in Guide to Advanced Empirical Software Engineering, Section II, 2008, pp. 201-228.

[21] Ruby on Rails. http://rubyonrails.org/: August, 2013.

[22] Scrumy, http://www.scrumy.com: August, 2013

[23] K. Bittener and I. Spence. Use case Modeling. Addison Wesley, 2003.

[24] K. Molokken-Ostvold, N.C. Haugen and Benestad, H.C., "Using planning poker for combining Expert estimates in software projects," Journal of Systems and Software, 81 (12), 2008, pp. 2106-2117.

[25] O. Ktata, and G. Lévesque, "Designing and implementing a measurement program for Scrum teams: what do agile developers really need and want?," in *Proceedings of the Third C* Conference on Computer Science and Software Engineering* (C3S2E '10), ACM, New York, NY, USA, 2010, pp. 101-107.

[26] V. Mahnič and T. Hovelja, "On using planning poker for estimating user stories," *J. Syst. Softw.* 85, 9 (September), 2012, pp. 2086-2095.

[27] S. Halstead, R. Ortiz, M. Córdova, and M. Seguí, "The impact of lack in domain or technology experience on the accuracy of Expert effort estimates in software projects," in *Proceedings of the 13th international conference on Product-Focused Software Process Improvement* (PROFES'12), Springer-Verlag, Berlin, Heidelberg, 2012, pp. 248-259.

[28] M. Jorgensen, and T. Halkjelsvik, "The effects of request formats on judgment-based effort estimation," Journal of Systems and Software, 83 (1), 2010, pp. 29-36.

[29] M. Jorgensen and M. Gruschke, "The Impact of lessons-learned sessions on effort estimation and uncertainty assessments," IEEE Transactions on Software Engineering, Jan. IEEE computer Society Digital Library. IEEE Computer Society, 2009, pp. 368 - 383.

[30] A. M Davis, E. H. Bersoff and E. R. Comer, "A strategy for comparing alternative software development life cycle models", Software Engineering, IEEE Transactions on (Volume: 14 , Issue: 10), 1988, pp. 1453 – 1461.

ANEXO III - Método de Estimación Basado en la Disponibilidad de Horas persona (MEBDHP)

Método de Estimación Basado en la Disponibilidad de Horas persona (MEBDHP)

Este método de estimación está basado en una tabla previamente definida donde se establece el valor de las horas estimadas de acuerdo con la duración prevista. Se desconoce el origen de dicha tabla pero es un proceso que se encuentra documentado en un sistema interno de la empresa a modo de guía para el manejo de pequeños proyectos.

1. Descripción de método

Como se comentó anteriormente, la empresa contaba con un método de estimación documentada. Bajo este método, el LP estimaba dos veces.

Las primeras estimaciones se realizaban antes de pedir las aprobaciones. Luego de haber obtenido las aprobaciones del presupuesto para llevar adelante el proyecto, se volvía a realizar una estimación, en este caso, más detallada. Si existía una diferencia mayor al 10% entre ambas estimaciones, entonces se debía pasar por el proceso de "Cambio del Alcance" donde se confeccionaba otro documento que se sometía a aprobación de todos los stakeholders que ya habían aprobado en primera instancia.

1.1 Estimación Inicial

Este proceso inicial no podía durar más de un día y respondía a las siguientes preguntas:

- ¿Cuándo? Luego del requerimiento inicial del negocio.
- ¿Qué? Guía de estimaciones Inicial (esfuerzo y capital).
- ¿Por qué? El negocio requería estimaciones de alto nivel para obtener la aprobación del presupuesto para el proyecto.

Pasos

1. El Dueño del Proyecto (DP) (quien pertenece al área que denominamos el negocio) entregaba una lista de requerimientos de alto nivel al LP.
2. El LP se reunía con el DP para revisar los requerimientos para lograr entender lo que se necesitaba en cada uno de ellos.
3. El LP realizaba una primera estimación de cuánto esfuerzo demandaría en general el proyecto.
4. El LP estimaba la duración del proyecto y basándose en la duración obtenía, de la *Tabla de Horas Disponibles* (**Tabla 1**), el máximo de horas como valor a presupuestar.

5. El LP confeccionaba un documento, llamado "**Contrato**", donde se detallaba lo siguiente:

    - Stakeholders involucrados (quién tomaría el liderazgo del proyecto, el DP y quienes aprobarían el presupuesto)
    - Fecha de solicitud del proyecto
    - Objetivos del negocio
    - Necesidades del negocio a alto nivel
    - Objetivos del proyecto
    - Alcance:
        - Qué se incluiría en el proyecto
        - Qué quedaría fuera del alcance
        - Entregables
        - Fecha de comienzo del proyecto
        - Fechas de entrega
        - Fecha de finalización
        - Presupuesto: aquí se indicaba el esfuerzo en horas persona que demandaría el proyecto y el costo de cada hora por perfil[24].
        - Contingencia: era un porcentaje que se agregaba al valor final y que sería usado en caso de desfasaje en las estimaciones. En general el criterio de selección del porcentaje de contingencia era evaluado por el LP y sometido a aprobación de la gerencia. Comúnmente, los criterios de asignación de contingencia eran bastante arbitrarios. Para dar una idea aproximada pero no taxativa se puede decir lo siguiente:

            Contingencia del 15%
            - Complejidad baja
            - Desarrolladores familiarizados con el sistema
            - Criticidad baja

            Contingencia del 20%
            - Complejidad media-alta
            - Desarrolladores familiarizados con el sistema
            - Criticidad media-alta

            Contingencia del 25% (o más)
            - Complejidad alta
            - Algunos desarrolladores no están familiarizados con el sistema
            - Criticidad alta

    - Aprobaciones: este documento debía ser aprobado por:
        - el gerente del equipo de desarrollo
        - el gerente del área que va a utilizar el sistema
        - el gerente del área de Planeamiento que manejaba los presupuestos
    - Apéndice: aquí es donde se adjuntaba la lista de requerimientos.

[24] Se omiten por restricciones de confidencialidad.

Tabla 1 - Tabla de Horas Disponibles en Horas persona

Esfuerzo	**Horas**	
	Min	**Max**
3 - 10 días (hasta 0,5 mes de esfuerzo)	24	80
10 - 20 días (hasta 1 mes de esfuerzo)	80	160
20 - 40 días (hasta 2 meses de esfuerzo)	160	320
40 - 60 días (hasta 3 meses de esfuerzo)	320	480
60 - 80 días (hasta 4 meses de esfuerzo)	480	640
80 - 100 días (hasta 5 meses de esfuerzo)	640	800
100 - 120 días (hasta 6 meses de esfuerzo)	800	960

1.2 Estimaciones detalladas

Cabe aclarar, que aun cuando a esta sección se llamaba "estimaciones detalladas" en realidad solo se llegaba a esta instancia en caso de que al comenzar la ejecución del proyecto, ya se preveía que el proyecto se iba a desfasar más de un 10% de las estimaciones iniciales explicadas en la sección anterior. De manera que esta segunda ronda de estimaciones se podría pensar como un cambio de alcance. Cambios de alcance que cambian la categorización de un proyecto de pequeño a grande, están fuera del alcance del presente trabajo de tesis.

También conocido como proceso de post-estimación. Respondía a las siguientes preguntas:

- ¿Cuándo? Después de la aprobación del proyecto y *durante la fase de diseño y desarrollo*.
- ¿Qué? En general se usaba un formato de plantilla similar a la que se muestra en la **Tabla 2** para obtener las estimaciones detalladas. Igualmente cada LP podía usar el formato que deseara.

Pasos

1. El LP definía todas las tareas y creaba la WBS.
2. El LP creaba la *Plantilla de estimaciones* (**Tabla 2**) ingresando las tareas obtenidas de la WBS a la misma.
3. El LP estimaba cuánto tiempo demandaría la realización de cada tarea y las ingresaba a la *Plantilla de estimaciones*.

   Si una tarea debía ser realizada por otro equipo, como por ejemplo los DBAs, entonces, este grupo pasaba sus estimaciones al LP quien ingresaba dichas estimaciones en dicha Plantilla.
4. El LP validaba las estimaciones contra la *Tabla de Horas Disponibles* **(Tabla 1)** de donde obtenía los nuevos valores para sus estimaciones.

5. Si existiera una **diferencia de más del <u>10%</u> entre las estimaciones iniciales y las estimaciones detalladas,** entonces el LP debía confeccionar un documento de cambio del alcance donde especificaba tal cambio.

La **Tabla 2** muestra, a modo de ejemplo, un formato de plantilla que se podía usar para documentar las estimaciones detalladas. Como se puede observar, el proyecto se dividía en fases: diseño, desarrollo, documentación, testing e implementación. Se tienen columnas para ingresar la cantidad de horas que va a llevar cada tarea por equipo involucrado (por ejemplo: las tareas relacionadas con la configuración de bases de datos son llevadas a cabo por el grupo de DBAs). Además se tiene una tarifa establecida por grupo. También se hace una diferenciación entre OPEX y CAPEX.

CapEx proviene de los términos en Inglés: "Capital Expenditures" que se traduce como "Gastos de Capital". Contribuye a la infraestructura fija de la compañía y son depreciadas con el tiempo. Es usado cuando se necesitan expandir el servicio de infraestructura.

OpEx proviene de los términos en Inglés: "Operational Expenditures" que se traduce como "Gastos Operativos". No contribuyen a la infraestructura en sí misma y en consecuencia no se deprecian con el tiempo. En nuestro caso, cuando el proyecto demandaba gastos de infraestructura tales como nuevos servidores, entonces, estos eran considerados parte de CAPEX. El resto de los gastos, especialmente el desarrollo de software era considerado parte del OPEX. En nuestro caso de estudio siempre se usó OPEX y no hubo necesidad de usar CAPEX.

Tabla 2 - Plantilla de estimaciones

Estimaciones

Nombre del Proyecto				**Total hrs**
Descripción del Proyecto				**OPEX**
Líder de Proyecto				**Contingencia %**
Cliente				
Fecha de Comienzo				**Total OPEX**
Fecha de Finalización				**CAPEX**
				TOTAL

		Equipo A	Equipo B	Equipo C
Tarifas				
TOTAL $				
Diesño	**HS**			
Desarrollar especificaciones de negocio				
Desarrollar especificaciones técnicas				
Desarrollar calendario				
Reuniones				
Desarrollo	**HS**			

Tarea 1				
Tarea 2				
Tarea N				
Documentación	**HS**			
Doumentación				
Testing	**HS**			
Pruebas				
Pruebas de usuario en ambiente de Testing				
Implementación	**HS**			
Desarrollar plan de implementación				
Implementación en Producción				
Pruebas de usuario en Producción				
TOTAL Hs				

1.3 Comparación de las estimaciones detalladas y los valores de la Tabla de Horas Disponibles

En los apartados anteriores, cfr 5.1.1 y 5.1.2, tanto para las Estimaciones Iniciales como para las Estimaciones Detalladas, se llevaba a cabo el proceso de Verificación de dichas estimaciones contra la *Tabla de Horas Disponibles* (**Tabla 1**) según el siguiente procedimiento.

Pasos

1. El LP tomaba la cantidad de horas de esfuerzo estimadas y localizaba en qué rango (significa: en qué fila) de la **Tabla 1** caía dicho valor.

2. De allí se tomaba el valor máximo del rango referencial que le correspondía, el cual indicaría el nuevo valor de estimación a considerar para los pasos siguientes.

   Por ejemplo: si las estimaciones resultaban ser de 200 hs de esfuerzo, entonces el LP se ubicaba en la tercera fila de la *Tabla de Horas Disponibles* y tomaba el valor máximo, que es 320 hs como su nuevo valor de estimación de esfuerzo.

Como se puede observar, la unidad de medida rectora es "*meses de esfuerzo*". La **Tabla 1** está dividida en meses de esfuerzo y sus equivalentes en *días y horas*. Comenzando con *medio mes de esfuerzo* (de 3 a 10 días) lo cual implica un mínimo de 24 horas y un máximo de 80 horas. Luego se tiene el siguiente valor correspondiente a *un mes de esfuerzo* (de 10 a 20 días) lo cual implica un mínimo de 80 horas persona y un máximo de 160 horas personas. Luego se indican los valores correspondientes a *2, 3, 4, 5 meses* hasta llegar al máximo posible de *6 meses de esfuerzo.*

1.4 MEBDHP. Diagrama de Estados

La **Figura 1** muestra el diagrama de los estados del proyecto durante el método de estimación MEBDHP.

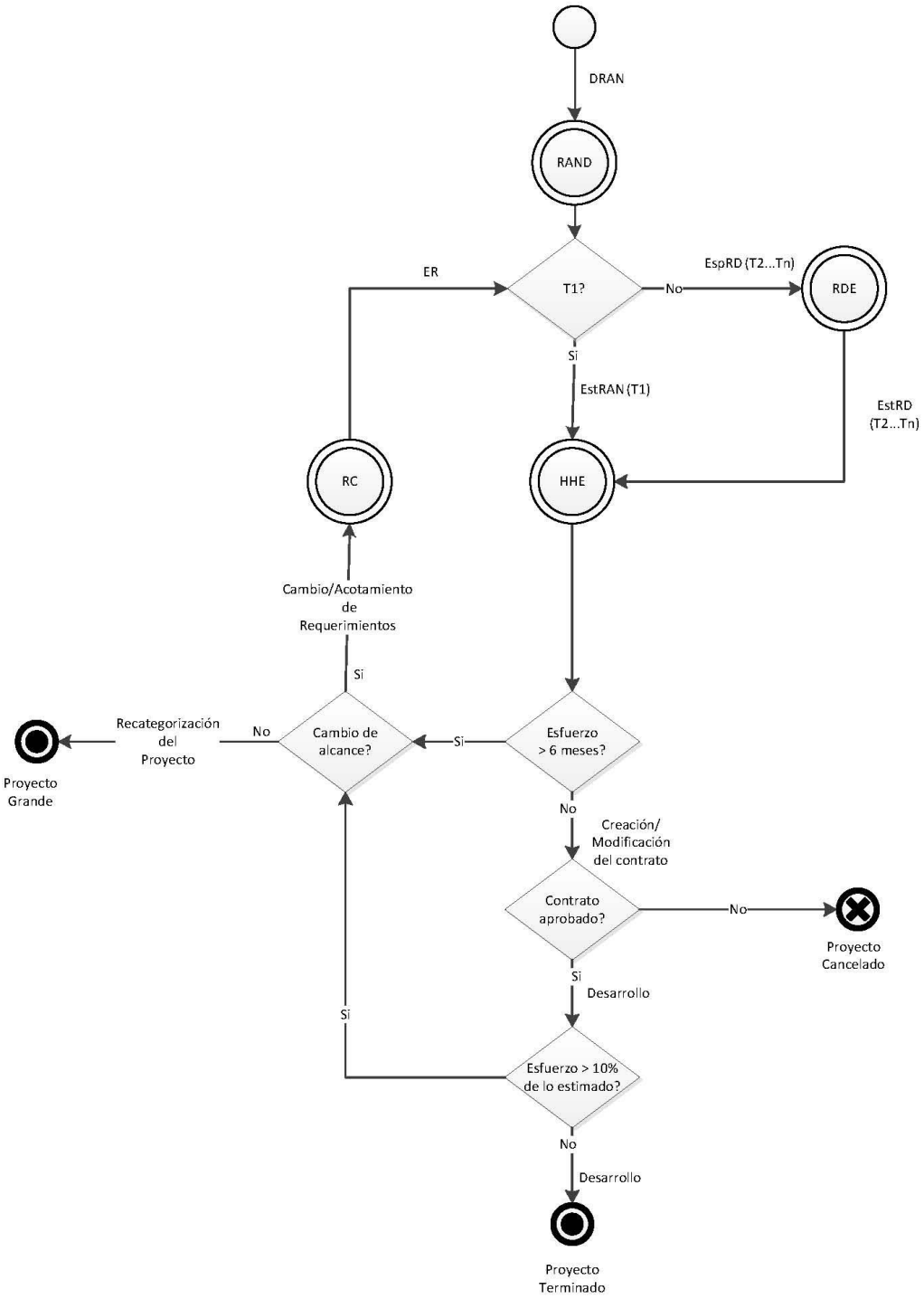

Figura 1 - Diagrama de Transición de estados del MEBDHP

Referencias pertenecientes a la Figura 1

Estados	Acciones:
RAND: Requerimientos de Alto Nivel Definidos **HHE**: Horas persona Estimadas **RC**: Requerimientos cambiados **RDE**: Requerimientos Detallados Especificados	**DRAN**: Definición de Requerimientos de Alto Nivel. **T1?**: Tabla 1? Si el proceso se encuentra en la Etapa Inicial (Tabla1), entonces EstRAN (Tabla1). Si no, entonces ERD (Tabla2...N) **EstRAN (T1)**: EstRAN (Tabla1). Estimación de Requerimientos de Alto Nivel en la Etapa Inicial. **EspRD (T2…N)**: EspRD (Tabla2…N). Especificación de Requerimientos Detallados en la Segunda Etapa y etapas subsiguientes. **EstRD (T2…N)**: EstRD (Tabla2…N). Estimación de Requerimientos Detallados en la Segunda Etapa y etapas subsiguientes. **Esfuerzo >6 meses?**: Si el esfuerzo requerido para completar el proyecto es mayor a 6 meses, entonces evaluar cambio de alcance. Si no, se crea o modifica el contrato. **Cambio de alcance?**: Si hay cambio de alcance, entonces Cambio/Acotamiento de Requerimientos. Si no, Recategorización del Proyecto. **Creación/Modificación del contrato**: Creación del Contrato si se encuentra en la Etapa Inicial (T1) o Modificación del Contrato si se encuentra en la Segunda Etapa o Etapas subsiguientes (T2…N) **Cambio/Acotamiento de Requerimientos**: cambiamos y/o sacamos requerimientos de la lista. **Contrato Aprobado?**: Si el Contrato es aprobado, entonces se comienza con el Desarrollo si se encontraba en la Etapa Inicial del proceso (T1) o se sigue con el desarrollo si se encontraba en la Segunda Etapa o Etapas Subsiguientes del proceso (T2…N). Si no, se cancela el Proyecto. **ER**: nueva especificación de requerimientos en T1 o T2…N. **Recategorización del Proyecto**: el proyecto requiere más de 6 meses de esfuerzo en horas persona y no va a haber cambio de alcance, por lo tanto, se debe recategorizar el proyecto a Proyecto Grande. **Esfuerzo > 10% de lo estimado?**: Si se ha realizado un esfuerzo mayor al 10% de lo estimado, entonces se evaluará Cambio de Alcance. Si no, se continúa con el desarrollo.

2. Debilidades del MEBDHP

Se sintetizan las debilidades del MEBDHP en los siguientes aspectos:

1. Constantes cambios de alcance. Ineficiencia en la realización de dos estimaciones, siendo que la primera se realiza con pocos datos pero que determina el tiempo y el presupuesto a utilizarse, mientras que la segunda, más detallada, se realiza ya durante la etapa de desarrollo con un presupuesto aprobado. De manera que cualquier cambio significativo implicaría un cambio de alcance. Lo cual genera un impacto negativo en el cliente. Esto es lo que motivó a la gerencia de IT a pedir más precisión en las estimaciones.

2. Rangos estrictos. La Tabla de Horas Disponibles (**Tabla 1**) no es equitativa en el caso de los extremos en cada rango: existe una variación significativa. Por ejemplo, supongamos dos proyectos (Proyecto 1 y Proyecto 2) de similares características, donde el Proyecto 1 se estime inicialmente en 39 días y el Proyecto 2 en 41 días. Luego de verificar ambas estimaciones contra la Tabla de Horas Disponibles (**Tabla 1**), el Proyecto 1 terminaría siendo estimado en 320 hs contra 480 hs que terminaría siendo estimado el Proyecto 2. En este caso, 2 días de diferencia entre ambos proyectos de características similares, implicaría una diferencia de 1 mes más esfuerzo.

3. Requerimientos incompletos a la hora de generar las estimaciones.

    a. El LP no se juntaba con el DP para entender los requerimientos y expectativas en detalle.

    b. No se profundizaba en el impacto de lo que se necesita desarrollar.

    c. No se documentaba la toma de requerimientos.

4. Generalmente las estimaciones eran realizadas por una única persona (generalmente el LP) y no por el equipo de trabajo.

ANEXO IV - Software Requirements Specification

Software Requirements Specification

Introduction

Purpose

<Specify software whose requirements are detailed in this document. Include version/release number. Describe the scope that is covered by this SRS, particularly if this SRS describes only part of the system or a single subsystem.>

Document Conventions

<Describe any standards or typographical conventions that were followed when writing this SRS, such as fonts or highlighting that have special significance.>

Intended Audience and Reading Suggestions

<Describe the stakeholders this document is intended for, such as analysts, developers, business custodians, owners, users, testers, project managers, etc. Suggest a sequence for reading the document, beginning with the overview sections and proceeding through the sections that are most pertinent to each reader type.>

Scope

<Provide a short description of the software being specified and its purpose, including relevant benefits and objectives. Relate the software to corporate goals or business strategies.>

References

<Provide any other documents or links to which this SRS refers to, e.g.: Security Model, Support Model, guidelines, coding practices, etc.>

Overall Description

Product Perspective

<State whether this product is an evolution, a replacement of an existent system, a new product or a follow-on member of a product family. If the SRS defines a component of another system(s), identify interfaces between the two.>

User Characteristics

<Identify the users of this product. Users may be differentiated based on frequency of use, subset of product functions used, technical expertise, security or privilege levels, educational level, or experience. Certain requirements may pertain only to certain user.>

Operating Environment

<Describe the environment in which the software will operate, including the hardware platform, operating system and versions, and any other software components or applications with which it must coexist.>

Design and Implementation constraints

<Describe any items or issues that will limit the options available to the developers. These might include: corporate or regulatory policies; hardware limitations (timing requirements, memory requirements); interfaces to other applications; specific technologies, tools, and databases to be used; parallel operations; language requirements; communications protocols; security considerations; design conventions or programming standards, e.g.: scalability, maintainability, code reviews, use of Coding style and conventions, Secure Coding guidelines, etc.>

User Documentation

<List the user documentation components (such as user manuals, on-line help, and tutorials) that will be delivered along with the software.>

Assumptions and Dependencies

<List any assumed factors (as opposed to known facts) that could affect the requirements stated in the SRS. These could include third-party or commercial components that you plan to use, issues around the development or operating environment, or constraints. The project could be affected if these assumptions are incorrect, are not shared, or change. Also identify any dependencies the project has on external factors, such as software components that you intend to reuse from another project.>

System Features

System feature 1

Descriptions and priorities

<Provide a short description of the feature and indicate whether it is of High, Medium, or Low priority.>

Functional Requirements

<Itemize the detailed functional requirements associated with this feature. Include snapshots, validations, and descriptive tables, how the product should respond to anticipated error conditions or invalid inputs, etc. Requirements should be concise, complete, unambiguous, verifiable, and necessary. Each requirement should be uniquely identified with a sequence number.>

REQ-1:

REQ-2:

REQ-3:

System feature 2

Descriptions and priorities

Functional Requirements

REQ-4:

REQ-5:

REQ-6:

….

External Interface Requirements

User Interface

<Describe the characteristics of each interface. This may include screen images, any UI standards or style guides that are to be followed, screen layout constraints, standard buttons and functions (e.g., help) that will appear on every screen, keyboard shortcuts, error message display standards, etc..>

Software Interfaces

Other Nonfunctional Requirements

Performance Requirements

<If there are performance requirements for the product under various circumstances, state them here and explain their rationale, to help the developers understand the intent. Specify

the timing. Make such requirements as specific as possible. You may need to state performance requirements for individual functional requirements or features.>

Safety Requirements

<Specify those requirements that are concerned with possible loss, damage, or harm that could result from the use of the product. Define any safeguards or actions that must be taken, as well as actions that must be prevented, e.g: ergonomic aspects like use of keyboard shortcuts (TAB index, ENTER, ESC, Alt+ First Label Letter, etc), standard buttons (OK, Cancel) especially when the system manages large amount of data.>

Security Requirements

<Specify any requirements regarding security or privacy issues surrounding use of the product or protection of the data used or created by the product. Define any user identity authentication requirements. Refer to any external policies or regulations containing security issues that affect the product.>

Software Quality

<Specify any additional quality characteristics for the product that will be important to either the customers or the developers. Some to consider are: adaptability, availability, correctness, flexibility, interoperability, maintainability, portability, reliability, reusability, robustness, testability, and usability..>

Other Requirements

<Define any other requirements not covered elsewhere in the SRS. This might include database requirements, internationalization requirements, legal requirements, documentation updates or generation, changes on processes and so on.>

Appendix – Glossary

<Define all the terms necessary to properly interpret the SRS, including acronyms and abbreviations.>