Métodos y herramientas de estimación, gestión cuantitativa de proyectos, trazabilidad de requerimientos y entrega continúa, orientados a la mejora de la calidad del software

Dapozo, Gladys N.; Greiner, Cristina;Irrazabal, Emanuel; Medina, Yanina; Ferraro, María; Lencina, Andrea; Chiapello, Jorge; Mascheroni, Agustín

Departamento de Informática. Facultad de Ciencias Exactas y Naturales y Agrimensura. Universidad Nacional del Nordeste
{gndapozo; cgreiner, ferraro, yanina}@exa.unne.edu.ar
gaspagu.3@gmail.com; emanuelirrazabal@gmail.com

Resumen

Este proyecto de investigación busca contribuir a la mejora en la calidad del desarrollo y del producto software. La automatización de los métodos de estimación permitirá a las empresas mejorar la precisión de la duración y costo de los proyectos de desarrollo, generando además información histórica que retroalimente a los métodos de estimación. La gestión cuantitativa, en particular, la gestión de incidentes y bugs, contribuirá a la eficiencia en los procesos de desarrollo. La sistematización de mecanismos de trazabilidad de los requerimientos y la incorporación de procedimientos que agilicen la puesta en producción de productos software seguros, contribuirán a mejorar la mantenibilidad del software, siendo este atributo uno de los más crítico en los ámbitos reales de producción de software.

Palabras clave: Estimación software. Incidencias. Entrega Continua. Gestión de Requerimientos.

Contexto

Las líneas de I/D corresponden al proyecto PI-F10-2013 “Métodos y herramientas para la calidad del software”, acreditado por la Secretaría de Ciencia y Técnica de la Universidad Nacional del Nordeste (UNNE) para el periodo 2014-2017.

Introducción

Estimación

El desarrollo de software requiere de la estimación para controlar y administrar los recursos antes y durante el proyecto. Existen diferentes modelos de estimación que van desde las técnicas orientadas a los procesos hasta métodos paramétricos o algorítmicos, basados en datos históricos utilizados para definir y calibrar el modelo[1] [2].La técnica (o conjunto de técnicas) que se utiliza debe adaptarse a los datos disponibles y la naturaleza del problema en la estimación. En [3] se concluye que a mayor especificidad del método en cuanto al contexto de desarrollo, mejor es la precisión de la estimación de esfuerzo y duración.

Asimismo, la determinación de costos de un producto es un aspecto clave para su comercialización. Un cálculo adecuado permite ganar clientes asegurando la expansión de las empresas. Considerando las características del proceso, del equipo, del producto y de la empresa, el Centro de Estudios de Ingeniería de Software de la Universidad de la Frontera (CEIS-UFRO) desarrolló un método que partiendo de una especificación de requerimientos basada en casos de uso y la productividad, llega a definir el costo de un producto de software transaccional web [4].

La realidad de las empresas de la región muestra, según un estudio cuyos resultados se detallan en [5], que las áreas

o empresas de sistemas, en gran medida, desconocen los métodos y herramientas orientados a mejorar la calidad del desarrollo y del producto software, siendo necesario realizar desde las universidades o los polos tecnológicos actividades de divulgación o capacitación sobre estos temas.

Entrega continua

La Entrega Continua de Software, en inglés Continuous Delivery (CD), se define como un enfoque en el cual los equipos mantienen la producción de software en ciclos cortos de tiempo, asegurando que el producto pueda ser lanzado de manera fiable en cualquier momento [6]. La idea es poder lanzar a producción un producto software libre de defectos con solo apretar un botón [7].

Existe un concepto similar al CD, que es el de Despliegue Continuo, en inglés Continous Deployment (DC). El DC es una actividad que consiste en lanzar cambios continuamente al ambiente de producción [8]. La principal diferencia se encuentra en la fiabilidad a la hora de lanzar una nueva versión del producto: el DC busca integrar código a producción una, dos, y muchas más veces en el mismo día, en cambio, el CD se centra en hacerlo con la certeza de que el producto que se está lanzando a producción tiene un alto grado de calidad y se encuentre libre de defectos.

En el intento de implementar el CD, muchas organizaciones terminaron alcanzando solamente el DC. Por ejemplo, algunas empresas como Facebook, Atlassian, IBM, Adobe, Microsoft y Flickr, han tenido éxito en la implementación de diferentes enfoques para realizar entregas en periodos cortos de tiempo. Sin embargo, diferentes autores señalan que el proceso de lanzar rápidamente productos software a producción, aún presentan inconvenientes por resolver [9].

Uno de los principales problemas estaría en la calidad del producto software. Ésta puede disminuir, dado que, al realizarse los despliegues del sistema con mayor frecuencia, aparecen más defectos en el producto [7]. Por tanto, es esencial desarrollar un enfoque de priorización de los diferentes aspectos en la calidad del producto software, teniendo en cuenta la forma de trabajo actual de las empresas de desarrollo software.

Para adoptar este enfoque, en la literatura se utiliza lo que se conoce como Tubería de Despliegue (DP - Deployment Pipeline). El DP es un estándar para automatizar el proceso de CD [7]. A pesar de que cada organización puede variar en la implementación de este estándar, el mismo se conforma de las siguientes actividades: Instalación Compilación, Pruebas de Aceptación, Pruebas de Capacidad, Pruebas Manuales, Liberación a Producción.
Con este tipo de solución se logra un entorno automatizado de compilación y pruebas de código dividido en etapas, que generan resultados rápida y eficientemente. Sin embargo, aún no se han encontrado evidencias de un proceso formal y estandarizado, ni se han evidenciado diferentes infraestructuras teniendo en cuenta el tipo de sistemas o la prioridad en el tipo de pruebas [10].

Gestión cuantitativa - Incidencias

La gestión cuantitativa en el desarrollo de software proporciona una visión del grado de cumplimiento de metas, así como las causas que explican desviaciones significativas en procesos o productos [11]. La gestión de proyectos en base al conocimiento cuantitativo contribuye a la determinación de los aspectos de mayor relevancia, cuyo rendimiento afecta en forma significativa al logro de los objetivos del proyecto y la satisfacción de los clientes, obteniendo productos con un mayor nivel de calidad. La medición de código fuente aporta a este conocimiento, por ejemplo estableciendo relaciones en cuanto a la complejidad del software y la propensión a fallos, lo cual ofrece indicios sobre qué

clases o porciones de código debe enfocarse la prueba de software.

Vinculado con el código fuente aparece el concepto de deuda técnica, la que se contrae cuando las actividades relacionadas con el desarrollo de software no se realizan con los niveles de calidad adecuados y por tanto puede disminuir la mantenibilidad del código u ocasionar mayores costos en el proyecto [12].

A medida que el proyecto crece en cantidad de requerimientos, tamaño del código o personas involucradas, el éxito resulta cada vez más difícil de alcanzar. Es importante que los equipos se mantengan dentro de presupuestos y tiempos planificados. Una forma efectiva para alcanzar las metas es que los equipos cuenten con herramientas que les permitan sistematizar aspectos tales como peticiones de cambios, incidencias y bugs. SWEBOK define como petición de cambio (ChangeRequest, CR) a: "una solicitud para ampliar o reducir el alcance del proyecto; modificar las políticas, procesos, planes o procedimientos; modificar los costos o presupuestos; o revisar los horarios. Una fuente de CR es el inicio de medidas correctivas en respuesta a informes de problemas [13].

La International Software Testing Qualifications Board (ISTQB) señala que durante la ejecución de una prueba de software puede ocurrir que los resultados reales varíen de los esperados. En este caso se dice que existen incidentes, bugs, defectos o problemas. Sin embargo, marca diferencia entre incidentes y defectos o errores. Un incidente es cualquier situación en la que el sistema muestra un comportamiento cuestionable, y se denomina bug sólo cuando es la causa de algún problema que se está probando [14].

SWEBOK define un bug como "un defecto en el código fuente. Un paso, proceso o definición de datos incorrectos en el programa. La codificación de un error humano en código fuente" [13].

En la literatura especializada surge que la asignación de bugs es un aspecto crítico. En [15] se describe cómo la tarea de asignar un CR puede ser costosa en tiempo, y la reasignación doblemente costosa, señalando la falta de uso de una herramienta automatizada que asista en esta tarea, y la necesidad de considerar la información histórica para tomar decisiones efectivas, tales como la carga de trabajo actual y conocimiento de habilidades de sus desarrolladores. En [16] se describe la asignación de bug como un proceso social debido a la complejidad que implica determinar la persona más adecuada para resolver el problema, señalando la importancia de tener una base de conocimiento sobre la experticia de los desarrolladores y herramientas que abarquen información socio-técnica de la organización. En [17] se menciona el conocimiento de los desarrolladores y la repercusión en la carga de trabajo del equipo como desafíos claves en la asignación de un bug.

Trazabilidad

Los procesos de evolución del software varían considerablemente dependiendo del tipo de software a mantener, los procesos de desarrollo utilizados en una organización y el personal implicado en el proceso. Por tanto, es imprescindible contar con una herramienta de gestión de cambios que acompañe este proceso, para minimizar el impacto de los cambios en el sistema y lograr que los mismos se realicen de manera planificada y controlada. Se reconoce que realizar el seguimiento a los requisitos a lo largo del proceso de desarrollo de software no es tarea fácil. Todo artefacto de software cambia en el tiempo por la evolución en las necesidades de los usuarios. Para minimizar el impacto causado por dicha evolución, la práctica de la trazabilidad ha sido estudiada e implementada con diferentes modelos y técnicas que permiten lograr mayor calidad en los productos de software [18]. El desafío es sin duda, acompañar la gestión de

cambios con el suficiente nivel de trazabilidad que logre el equilibrio entre calidad de software y tiempo dedicado a la gestión.

Líneas de investigación y desarrollo

En la línea de Estimación se propone:
- En base a los resultados obtenidos en [3] se desarrollará una herramienta para automatizar la estimación de costos en proyectos web aplicando el método de estimación CEIS-UFRO.

En la línea de Entrega Continua:
- Estudiar, a partir de encuestas, el grado de importancia que los líderes de proyectos le confieren a requerimientos no funcionales.
- Realizar una revisión sistemática sobre modelos de entrega continua, haciendo hincapié en las estrategias, en el formato y tipo de pruebas y su automatización, y en los tipos de sistemas.
- Analizar características de las herramientas de entrega continua existentes, y contrastarlas con las necesidades de los profesionales, teniendo en cuenta el grado de importancia dado a los requerimientos no funcionales y los tipos de pruebas que se incorporan en los proyectos.

En la línea de Gestión Cuantitativa e Incidencias se propone:
- Un estudio detallado de herramientas de gestión de incidentes que se utilizan en empresas de software de la región NEA, identificando sus dificultades y aportes cuantificables en el desarrollo de software.
- Diseñar y construir una herramienta que permita realizar el seguimiento de las incidencias de los proyectos de software, aportando información de las características socio técnicas de los equipos involucrados. La herramienta permitirá gestionar un histórico de las incidencias, desde su reporte inicial hasta su corrección, siendo configurable el ciclo de vida de las mismas para que se adapte a las necesidades de la organización. A su vez, tendrá en cuenta la experiencia, carga de trabajo actual e involucramiento de los desarrolladores.

En la línea de Trazabilidad, se propone:
- Un estudio detallado de herramientas de gestión de requerimientos que se utilizan en las empresas de software de la región NEA, desde el aporte de trazabilidad visible y cuantificable en el desarrollo de software.
- Proponer técnicas de trazabilidad como un conjunto de buenas prácticas en la gestión de requerimientos que contribuyan a la calidad del software sin sobrecargar las tareas de gestión.

Resultados y Objetivos

Los principales resultados de las actividades desarrolladas en estas líneas son:

En la línea de Estimación:
- Se evaluaron diferentes métodos de estimación, Puntos de Casos de Uso, CWADEE, Webmo y RESC, para estimar duración y esfuerzo, y se concluye que a mayor especificidad del método (en cuanto al tipo de desarrollo), mejor es el ajuste a la duración real. Como así también, a mayor disponibilidad y pertinencia de datos históricos de proyectos, mejor es la precisión de la estimación [3].
- Basado en los resultados obtenidos en [3], se desarrolló una herramienta que automatiza la estimación en los proyectos web aplicando los métodos estudiados. Este desarrollo fue parte del Proyecto Final de un alumno de la carrera.
- Analizados los datos de una encuesta realizada a empresas de software y áreas de sistemas de las organizaciones, respecto de la estimación se concluye que en la mayoría de las áreas/empresas las prácticas de estimación más utilizadas son la de juicio de experto y por analogía. Se conocen, pero no se

utilizan, o se utilizan poco, las técnicas paramétricas que pueden aportar mayor precisión en la estimación. Así también es bajo el porcentaje de las áreas/empresas que utilizan datos históricos para la estimación [19].

En la línea de Entrega Continua:

- Se realizó una Revisión Sistemática de la Literatura sobre la refactorización de software basada en valor para obtener conclusiones acerca de la manera en la cual se priorizan las mejoras del código fuente y del tipo de herramientas utilizadas. Los resultados se publicaron en [20].
- Asimismo, se realizó un estudio de la mejora de la mantenibilidad. En el artículo se presentaron los resultados de la utilización del modelo de medición y de las herramientas durante diez meses en una empresa española. Se detalla la evolución de las mediciones, la satisfacción de los desarrolladores y cómo han ido evolucionando las incidencias encontradas en las pruebas de aceptación [21].
- Se realizó una investigación preliminar para determinar el grado de importancia conferido a los atributos de calidad, mediante una encuesta a los responsables de aseguramiento de la calidad de los proyectos en centros de la Universidad de Ciencias Informáticas en Cuba. Se concluyó que la funcionalidad es la característica percibida como más importante. Esto tiene relación directa con los tipos de pruebas que se realizan de manera habitual antes de una entrega de software: las pruebas funcionales [22].

En gestión cuantitativa:

- Se realizó una Revisión Sistemática de la Literatura (RSL) sobre herramientas para el apoyo a la gestión de proyectos, la que permitió observar que dentro de los aspectos tratados con mayor frecuencia aparece muy fuerte la administración de los equipos de desarrollo, a través de herramientas de coordinación de actividades [23].
- Se profundizó el estudio de deuda técnica, con el propósito de proponer buenas prácticas para prevenir, medir y gestionar la deuda técnica en el desarrollo de software [24].
- Se abordó el tema de la priorización de las mejoras aplicadas al código fuente y el tipo de herramientas utilizadas, a través de una RSL [21].

En la línea de trazabilidad:

- En la encuesta realizada para determinar el estado de situación del desarrollo de software en la ciudad de Corrientes, considerando las empresas de software y las áreas de sistemas de las organizaciones, se determinó que es escasa o casi nula la presencia de trazabilidad de requerimientos, sobre las distintas herramientas de gestión utilizadas [19].

Formación de recursos humanos

En el Grupo de Investigación sobre Calidad de Software (GICS) están involucrados 5 docentes investigadores, 2 becarios de investigación de pregrado y 1 tesista de doctorado. Un alumno de la carrera finalizó la misma con un proyecto vinculado con estos temas.

Referencias

[1] J. Andrés, D. Fernandez-Lanvin, P. Lorca. "Cost estimation in software engineering projects with web components development".2015, vol.82, n.192

[2] E. Mendes, "Using knowledge elicitation to improve Web effort estimation: Lessons from six industrial case studies," Software Engineering (ICSE), 2012.

[3] G. Dapozo, Y. Medina, A. Lencina, G. Pedrozo Petrazzini. “Métodos de estimación de esfuerzo y duración en proyectos web pequeños". Revista Eletrônica Argentina-Brasil de Tecnologias da Informação e da Comunicação. v. 1, n. 2 (2014)

[4] J. E. Díaz Villegas, G. Robiolo. “Método de estimación de costos de un producto de software Web”. 43 JAIIO - ASSE 2014 - ISSN: 1850-2792

[5] G. N. Dapozo; C. L. Greiner, E. Irrazábal, Y. Medina, M. A. Ferraro, A. B.

Lencina “Características del desarrollo de software en la ciudad de Corrientes”.CACIC 2015.
[6] L. Chen, "Continuous Delivery: Huge Benefits, but Challenges Too" in IEEE Software 03/2015. V. 32(2).
[7] J. Humble and D. Farley. “Continuous delivery: reliable software releases through build, test, and deployment automation”, 1st ed. Boston, US: Pearson Education, 2010.
[8] H. H. Olsson, H. Alahyari, and J. Bosch. "Climbing the ‘Stairway to Heaven’-A Mulitiple-Case Study Exploring Barriers in the Transition from Agile Development towards Continuous Deployment of Software" in 38th EUROMICRO Conference, 2012, pp. 392-399.
[9] B. Fitzgerald and K. Stol, "Continuous Software Engineering and Beyond: Trends and Challenges," in 1st International Workshop on Rapid Continuous Software Engineering, 2014.
[10] O. Prusak."Continuous Testing: The Missing Link in the Continuous Delivery Process". Blaze Meter. 2015. https://blazemeter.com/blog/continuous-testing-missing-link-continuous-delivery-process
[11] Gou, L., Wang, Q., Yuan, J., Yang, Y., Li, M., & Jiang, N. “Quantitative defects management in iterative development with BiDefect”. Software Process Improvement and Practice, 14(4), 227-241. 2009.
[12] W. Cunningham, “The wycash portfolio management system,” in OOPSLA ’92: Addendum to the proceedings on Object-oriented programming systems, languages, and applications (Addendum), 1992.
[13] P. Bourque y R. E. Fairley, SWEBOK, Tercera ed., IEEE, 2014, pp. 82-87.
[14] International Software Testing Qualifications Board (ISTQB). “What is an Incident in software testing?”. [En línea].http://istqbexamcertification.com/what-is-an-incident-in-software-testing/
[15] P. Anselmo da Mota y Y. Cerqueira. “Towards Understanding Software Change Request”. Proceedings of the 17th International Conference on Evaluation and Assessment in Software Engineering, 2013.
[16] P. Guo y T. Zimmermann.“Not my bug! and other reasons for software bug report reassignments”. CSCW '11 Proceedings of the ACM 2011 Conference on, 2011.
[17] G. Bortis y A. van der Hoek. Teambugs: a collaborative bug. CHASE '11 Proceedings of the 4th International, pp. 69-71, 2011.
[18]M. Tabares, f. Arango, R. Anaya. Una Revisión de Modelos y Semánticas Para la Trazabilidad de Requisitos. Revista EIA, ISSN 1794-1237 Número 6, p. 33-42. Diciembre 2006. Escuela de Ingeniería de Antioquia, Medellín. Colombia.
[19] G.N. Dapozo, Y.Medina, A. Lencina. “La práctica de la estimación en empresas y áreas de Sistemas”. Revista Eletrônica Argentina-Brasil de Tecnologias da Informação e da Comunicação. v.1, n. 4.2015
[20] E. Irrazábal, C.Greiner, G. Dapozo. “La refactorización de software basada en valor: Revisión sistemática de la literatura”. JAIIO 2015, ASSE, pp. 130 -144.
[21] E. Irrazábal, “Mejora de la mantenibilidad con un modelo de medición de la calidad: resultados en una gran empresa”. XXI CACIC. 2015.
[22] M. Pérez García, E. Irrazábal, R. Carrasco-Velar, Y. Coca Bergolla. “Importancia de los requisitos no funcionales: estudio preliminar en una Universidad de Cuba”. VII taller internacional de calidad en las tecnologías de la información y las comunicaciones calidad 2016. La Habana, Cuba (pendiente de publicación).
[23] G. Dapozo, C. Greiner, J. Chiapello.. "Revisión Sistemática de Herramientas de Apoyo a la Gestión Cuantitativa de Proyectos de Software". III Jornadas de Investigación en Ingeniería del NEA y Países Limítrofes. 2014.
[24] G. Dapozo, C. Greiner, E. Irrazabal, J. Chiapello. “Análisis de código fuente orientado a la calidad del producto”. XVII Workshop de Investigadores en Ciencias de la Computación (WICC2015).