Necesidad del empleo de herramientas estándares en XP

Lic. Fernando Pinciroli

Centro de Investigaciones del
Departamento de Sistemas de Información
Universidad de Congreso
Mendoza – Argentina
pincirolif@profesores.ucongreso.edu.ar

II Workshop de Ingeniería de Software y Bases de Datos
(WISBD)

I. Resumen

Más allá de todo lo que se discutió y se seguirá discutiendo entre los seguidores de las metodologías de desarrollo ágiles y los que sostienen propuestas con el empleo de más herramientas y de más documentación, hay un aspecto que no se puede soslayar, y se trata del beneficio del empleo de herramientas estándares en XP en lugar de herramientas ad hoc. Quisimos realizar este análisis consultando la opinión de los principales involucrados en el tema, como lo son Kent Beck, Ron Jeffries, Rebecca Wirfs-Brock, Grady Booch, James Rumbaugh, Martin Fowler, con quienes mantuvimos correspondencia vía correo electrónico. Así, con todas estas opiniones y nuestro análisis, llegamos a la conclusión de que nuestro enfoque es factible y, aun más, que XP posee ciertas deficiencias que las herramientas estándares del UML permiten salvar. De este modo presentamos una propuesta para seguir trabajando con XP sin salirse de su espíritu, aplicando las herramientas estándares del UML donde sea factible y donde sea necesario hacerlo.

II. Palabras clave

Programación extrema, XP, metodologías ágiles, historias de usuario, casos de uso, diagramas de clases, diagramas de secuencias, tarjetas CRC.

III. Introducción

La Programación eXtrema, eXtreme Programming o XP surgió presentándose como una alternativa simple para el desarrollo de software en contraposición a las metodologías complejas y sobrecargadas de técnicas y herramientas. Logró adeptos, inició una corriente filosófica dentro del concierto de metodologías para el desarrollo de sistemas de software, conformó un corpus documental de sus principios y de sus prácticas y obtuvo éxitos resonantes en proyectos de envergadura[1].

Curiosamente, todo esto sucedió en un momento muy particular de la ingeniería de software: se estaban unificando principios y prácticas por primera vez en la historia y se había logrado el primer estándar mundial para el modelado y el diseño de software, concretamente, el UML. De esta forma, como protagonistas encontrados entre sí, rápidamente se vieron las caras al proponer soluciones desde veredas opuestas.

Sin dudas la disputa fue enorme. Como miembros del OTUG[2], fuimos espectadores de lujo de los enfrentamientos entre quienes sostenían una y otra postura. Booch, Rumbaugh, Jacobson, Martin, Beck, Jeffries y muchos otros eran los permanentes animadores de las largas discusiones que, sin lugar a dudas, nunca dieron un ganador absoluto, salvo de alguna pequeña escaramuza.

Pero antes que nada debemos pensar que, más allá del empleo de XP, UP[3] o cualquier otra propuesta, nos encontramos en presencia de una tecnología, que como toda técnica se debe juzgar -además de hacerse desde el punto de vista moral de la intencionalidad de quien la ponga en práctica como así también del resultado moral del efecto entre quienes fueran sus receptores- por factores vinculados a la eficacia, a la velocidad, a la reducción de costos, etc. De esta manera, podríamos resumir diciendo que nuestra perspectiva debe ser netamente *pragmática*.

Convencidos entonces de que el contar con un estándar mundial para el modelado y diseño de sistemas fue un paso fundamental en nuestra disciplina y, por otra parte, de que la conveniencia del empleo de una técnica debe medirse por sus resultados concretos, es que hemos querido enfocar el estudio de XP desde una perspectiva que, por un lado, nos permita explotar el estándar con el que hoy contamos -reemplazando las herramientas de XP por las estándar del UML en los casos en los que sea posible y enfocar con el espíritu del UML las técnicas de XP que no fueran materializadas con herramientas concretas-, y por el otro, dando solución a los inconvenientes en la aplicabilidad de las herramientas de XP tal como se encuentran planteadas hoy en día.

Desarrollamos nuestra propuesta, contrastada contra las opiniones de diversos autores -Beck, Jeffries, Wirfs-Brock, Booch y Fowler- con los que mantuvimos diálogos muy enriquecedores que nos permitieron obtener las conclusiones que presentamos.

IV. Estandarización y aplicabilidad de XP

Dado que se trata de una metodología de desarrollo liviana, XP cuenta con pocas herramientas de modelado y se cuida bastante de incorporar otras adicionales. De hecho, muchas veces se ha escuchado decir: "*en XP no existe más documentación que el código fuente*". Por lo tanto, y con el fin de no alterar el espíritu de XP, trataremos de centrar nuestra discusión solamente en las herramientas existentes en el método.

---

[1] Aunque el proyecto emblema de XP, el "C3" realizado en la empresa Chrysler, aparentemente tuvo un resultado catastrófico. La gente de XP echa la culpa a los usuarios afirmando que la metodología no falló.

[2] OTUG es el acrónimo de Object Technology Users Group, la lista de correo electrónico abierta de Rational Software Corporation en la que se realizaban las discusiones durante el establecimiento del UML como estándar mundial y que era frecuentada por los principales metodologistas del mundo.

[3] Acrónimo de *Unified Process*, proceso de modelado de sistemas iterativo e incremental, dirigido por casos de uso, centrado en la arquitectura y en camino de convertirse en estándar, hasta hace poco conocido como RUP.

A medida que XP fue apareciendo como una alternativa de desarrollo muy fuerte, y sucediendo esto durante la etapa en la que el UML estaba dando sus primeros pasos como estándar mundial, la discusión en el ámbito de los creadores y los usuarios de metodologías comenzó a centrarse en forma de planteo dialéctico: XP o UML.
Pero esta discusión era desacertada. En primer lugar, porque el UML es un lenguaje de modelado y XP es una metodología de desarrollo. La verdadera disputa debería haber estado centrada entre métodos o procesos de desarrollo, por ejemplo, entre XP y UP o entre XP y OBA/D[4], entre XP y OPEN[5], etc. En segundo lugar porque el UML ya había ganado un lugar preponderante en el mundo: es un estándar para el modelado y diseño de sistemas, principalmente orientados a objetos, desde noviembre de 1997, mientras que XP todavía tiene mucho camino por recorrer.
De esta manera, y teniendo en cuenta entonces que el UML es un lenguaje y es un estándar, y que en XP es necesario "expresar cosas", queda latente la pregunta: ¿no sería acaso posible expresar esas "cosas" de XP en un lenguaje estándar? Creemos que esta posibilidad cabría en el empleo de casos de uso en lugar de las historias de usuario, en la planificación de versiones e iteraciones y en la medición de la velocidad del proyecto en base a los casos de uso, en el empleo de UML como apoyo a la reutilización que promueve la simplicidad del código, en el empleo de diagramas de clases y de interacción como complemento de las tarjetas CRC. Finalmente daremos unas breves consideraciones acerca de la aplicabilidad de XP.

V. Historias de usuario reemplazadas por casos de uso

En primer término debemos partir de la discusión acerca de la necesidad de reemplazar las historias de usuario por casos de uso. Sin dudas, la razón principal pasa por el hecho de contar con una herramienta estándar para modelar los requisitos, cosa que de ninguna manera altera el sueño de los usuarios de XP. Según nos comentaba Ron Jeffries, "los iniciados en XP deberían ser más extremos que XP, no menos extremos. El aprendizaje es en el minimalismo"[6]. Pero sin dudas la necesidad e importancia de un estándar no es un tema menor, y que no precisa ser demostrado en este trabajo.
Entonces, ¿sería posible reemplazar las historias de usuario por casos de uso? La factibilidad estaría dada en caso de que fueran herramientas equivalentes o que pudieran adecuarse una a la otra. En este caso, y al tratar de preservar el espíritu de XP, serían los casos de uso los que deberían amoldarse a las historias de usuario. Más aún si consideramos que "las historias de usuario son parte central del método"[7].
Una primera comparación entre estas herramientas estaría dada porque ambas son herramientas para la captura de requisitos, porque se expresan en lenguaje natural y porque intentan modelar esos requisitos desde la perspectiva de los usuarios, tratando de darles la mayor satisfacción al incluir lo que ellos realmente quieren y en la forma en la que lo quieren.
Quienes den una mirada un poco más profunda podrán destacar otros aspectos comunes entre las dos herramientas, como el que ambas son orientadas a procesos, el que permiten dividir el dominio del problema en pequeñas porciones coincidentes con los diferentes objetivos y expectativas de los usuarios con el sistema, y el que son las bases sobre las que se funda el proceso de desarrollo subsiguiente. Incluso podríamos agregar que, así como cada historia de usuario posee sus pruebas de aceptación, los casos de uso también poseen los casos de prueba.
Pero si hacemos un análisis más exhaustivo veremos que la naturaleza de estas herramientas es absolutamente distinta: "el factor clave de diferenciación entre los casos de uso y las historias de XP

4 Acrónimo de Object Behavior Analysis and Design, cuyos creadores son Adele Goldberg y Kenneth Rubin.
5 Propuesto por una coalisión de autores -Brian Henderson-Sellers y Ian Graham entre otros- compitió infructuosamente con el UML ante el OMG para ser considerado el estándar de modelado.
6 JEFFRIES, Ron. Carta personal a Fernando Pinciroli (8/VII/2002).
7 Íbidem.

es la forma en la que está determinado su alcance"[8]. Este alcance abarca la formalidad de ambas herramientas y como modelan la funcionalidad y a los afectados por esta última.

A. Formalidad de las herramientas

Kent Beck plantea que las historias de usuario son más poderosas que los casos de uso debido a que con las primeras se puede tener un diálogo más sincero, distendido, amigable, participativo y, por consiguiente, fructífero entre los analistas y los usuarios. Como contrapartida, frente a los casos de uso los usuarios se sienten más temerosos y retraídos, quedando toda la responsabilidad acerca del contenido de los requisitos en las manos de los analistas, en lugar de que los usuarios sean quienes tengan la iniciativa.[9]

En una discusión que mantuvimos con Beck vía correo electrónico, el autor hizo hincapié en que las ventajas de las historias de usuario sobre los casos de uso son de dos tipos: prácticas y políticas. Prácticas porque no poseen un formato definido, no son tan formales, lo que evita tener que adiestrar a los que participen del relevamiento en el empleo de la herramienta; y políticas porque pasa el protagonismo de manos de los analistas a los usuarios, ya que no hay herramientas formales de por medio.[10]

Pero nuestra opinión es que no necesariamente hay que plantear el relevamiento en términos formales, poniendo al usuario en una situación que lo cohíba. Por el contrario, se puede mantener un diálogo perfectamente abierto, participativo y ameno con él mientras se van escribiendo o dibujando los casos de uso en una servilleta[11]. Todo es cuestión de hasta qué punto uno quiere ser formal. Pero el aspecto formal es el menos importante y la discusión realmente pasa por otra parte, como veremos a continuación.

B. Modelado de la funcionalidad

Un *modelo de casos de uso* se puede definir como *la funcionalidad completa del sistema desde la perspectiva de los actores que interactúan con él*[12]. Discutiremos las dos partes esenciales de la definición.

Al considerar la primera parte de la definición, cuando decimos que "el modelo de casos de uso debe modelar la funcionalidad completa del sistema", estamos expresando que, si existe alguna funcionalidad en el sistema, y por menor que ésta sea, debe quedar representada dentro del modelo de casos de uso. Por ejemplo, habiendo realizado un modelo de casos de uso en las etapas iniciales del desarrollo, si se detectara una nueva funcionalidad en etapas más avanzadas debería actualizarse el modelo de casos de uso con ella.

Por el contrario, las historias de usuario no sólo no deben contener la funcionalidad completa del sistema, sino que sólo deben plantear los diferentes objetivos de los usuarios en un nivel de generalidad tal que permita cumplir con ciertas restricciones básicas -que consideraremos enseguida- y el detalle se obtendrá en discusiones cara a cara entre los programadores y los usuarios. De ninguna manera retornaremos a completar las historias de usuario escritas inicialmente

---

8 DAVIES, Rachel. "The Power of Stories". Cap. 11. Londres, Connextra, s/f. Pág. 46.

9 Cfr. http://c2.com/cgi/wiki?UserStoryAndUseCaseComparison

10 BECK, Kent. Carta personal a Fernando Pinciroli (8/VII/2002).

11 Planteamos esta situación de la servilleta debido a que Ron Jeffries nos comentaba "vos y yo no tendríamos necesidad de una herramienta: la conversación y una servilleta serían suficientes". Cfr. Ron Jeffries, carta personal a Fernando Pinciroli (8/VII/2002).

12 Cfr. PINCIROLI, Fernando. "Introducción al Unified Modeling Language: un nuevo estándar mundial para el análisis orientado a objetos". En: AA.VV. **Ingeniería del software y reutilización: aspectos dinámicos y generación automática**. Orense, Tórculo, 1998.

con el mayor detalle obtenido después. Justamente porque se trata de una herramienta destinada a facilitar "el desarrollo incremental de software... en un entorno en el que se esperan cambios en los requisitos".[13]

Pero por otra parte, y como ya dijimos antes, las historias de usuario deben ser especificadas con mayor detalle como listas de tareas concretas redactadas en lenguaje técnico, dirigido a los desarrolladores. Esto implica que, en algún momento posterior del desarrollo, los requisitos tendrán que poseer un grado de precisión mayor.

Nuestra idea consiste en utilizar los casos de uso de la siguiente forma. Por una parte, plantear el caso de uso con sus actores, colocándole un nombre apropiado, como podría ser perfectamente el de la historia de usuario correspondiente. Por otra parte, el caso de uso podría tener una descripción inicialmente "liviana", expresando los dos o tres renglones -a veces los autores de XP dicen *parágrafos*, lo que extendería bastante a esos dos o tres renglones- de las historias de usuario. Y la explicación detallada en términos técnicos orientada a los desarrolladores podría acompañar al caso de uso.

Si bien esta propuesta surgió de las discusiones de nuestro grupo mientras desarrollábamos este trabajo, contamos con el apoyo de varios autores que nos permiten sostener la idea. Por ejemplo, Alistair Cockburn plantea que las historias de usuario podrían ser considerados casos de uso de menor precisión.[14]

Rebecca Wirfs-Brock nos comentaba su opinión acerca de que, además de que las historias de usuario poseen un muy bajo nivel de detalle, las descripciones de los requisitos funcionales por medio de casos de uso o de scripts de OBA/D pueden cumplir exactamente el mismo objetivo que las historias de usuario de XP -incluso existe un trabajo en el que se aplicaron los beneficios de los scripts de OBA/D a los casos de uso del UML, y que expusimos en EE.UU. hace unos años atrás[15]-. Además considera a todas ellas como herramientas equivalentes, de entre las que las historias de usuario serían las más minimalistas. Por último, agrega que los casos de uso pueden expresar perfectamente la interacción entre los usuarios y el sistema, de la misma forma que en una conversación[16], como se desea representar con las historias de usuario.

Martín Fowler[17] plantea que la situación depende de cómo se utilicen los casos de uso y que él los emplea en la misma manera en la que Kent Beck usa las historias de usuario. Además, insiste en utilizar el nombre de "casos de uso" a fin de lograr una mejor comunicación con otros autores y para influenciarlos a que utilicen un enfoque más liviano.[18]

Si bien Ron Jeffries fue un poco más escéptico cuando nos respondió al respecto, Kent Beck fue muy claro, concreto y contundente al respondernos: "No me importa si usan historias de usuario u otro formato, siempre y cuando se satisfagan las restricciones".[19]

Estas restricciones básicas que deben cumplir las historias de usuario a las que Beck hace referencia también nos fueron descriptas en su correspondencia personal:

- "Deben poseer la suficiente información como para poder realizar estimaciones razonables
- Deben poseer suficiente información como para poder comparar las diferentes historias de usuario entre ellas, a fin de determinar cuál se implementará primero

13 DAVIES, Rachel. Op.cit.

14 Cfr. http://c2.com/cgi/wiki?UserStoryAndUseCaseComparison

15 Cfr. PINCIROLI, Fernando et al. "The use of scripting for the transition between the use cases and the interaction models". En: World Multiconference on Systemics, Cybernetics and Informatics SCI 2000. Proceedings, vol. II. Orlando, 2000.

16 WIRFS-BROCK, Rebecca. Carta personal a Fernando Pinciroli (8/VII/2002).

17 Curiosamente, Martín Fowler es a la vez miembro del equipo que realizó el primer y más resonante desarrollo con XP -el proyecto C3 de Chrysler- y autor de UML Distilled.

18 Cfr. http://c2.com/cgi/wiki?UserStoryAndUseCaseComparison

19 BECK, Kent. Íbidem.

- Deben ser lo suficientemente simples como para poder escribir muchas de ellas rápidamente
- Deben ser lo suficientemente sencillas de entender por parte de los usuarios como para aceptar las responsabilidades que se incluyeron en las historias"[20]

En definitiva, creemos que, desde el punto de vista de nuestras consideraciones, nuestra experiencia y la opinión de los autores, en particular del mismísimo creador de XP, estas restricciones no sólo se pueden cumplir, sino que es natural trabajar de esta forma con los casos de uso. De esta manera, nuestra hipótesis de emplear casos de uso para "hacer lo que hacen" las historias de usuario estaría probada.

C. Modelado de los actores

Pero ahora veamos algo que las historias de usuario no permiten hacer. Consideremos la segunda parte de la definición del modelo de casos de uso que planteamos antes. La funcionalidad del sistema debe ser modelada "desde la perspectiva de los actores que interactúan con él", lo que lleva a considerar a los actores como parte fundamental de esta etapa para el desarrollo del sistema. De ninguna manera XP desentona con esta postura, dado que dentro del método la presencia de los usuarios es un requisito fundamental, tanto para que actúen como *documentación viva* de los requisitos, como así también para que ellos mismos vean satisfechas sus expectativas a medida que ven cómo se desarrolla el sistema.
Pero no podemos olvidar dos aspectos: las historias de usuario no ponen de manifiesto claramente quiénes son los usuarios de cada uno de los requisitos, y no sólo los usuarios sino también los *stakeholders*: todo aquel involucrado en mayor o menor grado, o con más o menos expectativas acerca del sistema. Esto implicaría una pérdida de información importante para las etapas restantes del desarrollo del sistema y se debe a que para la gente de XP lo que importa es el contenido de la historia de usuario. Beck confirma esta afirmación: "las historias se escriben desde la perspectiva de sólo aquel que las cuenta, generalmente hablando. Si vos querés otras perspectivas, escribí otras historias. La combinación de las historias te da la cobertura (si se hace bien), y si no, escribí más historias"[21]. Esto refuerza un poco más nuestra idea de utilizar la perspectiva de casos de uso con la finalidad de lograr y reflejar la combinación de historias a la que Beck hace referencia.
Pero por otra parte, estaríamos en un serio problema desde el punto de vista del cumplimiento de una de las premisas fundamentales de XP: el usuario debe estar *siempre* presente, y sabemos positivamente que nunca tendremos la posibilidad de cumplir con esto si quisiéramos contar con la totalidad de los stakeholders del sistema.
Por este motivo, además de que el equipo de XP no sólo fuerza la realidad cuando no puede tener a los usuarios siempre presentes -tratamos este tema en el punto VII, "Aplicabilidad de XP"- sino que también corta por lo sano eliminando a los stakeholders del sistema. Rebecca Wirfs-Brock nos comenta que debido a que muchas veces "los usuarios y stakeholders no se ponen de acuerdo o piensan cosas que son de distinta importancia... en orden de no 'gastar tiempo' acomodando esto con el equipo de desarrollo, las prácticas de XP tratan de eliminar este problema al designar al usuario representativo que simule todo y guíe al equipo. Se busca usar interacciones usuario-desarrollo a su máxima efectividad. Pero esto puede resultar en un sistema sesgado (sesgado en función de lo que este representativo considera importante)".[22]

---

20 Ídibem.
21 BECK, Kent. Carta personal a Fernando Pinciroli (15/VII/2002).
22 WIRFS-BROCK, Rebecca. Carta personal a Fernando Pinciroli (15/VII/2002).

Creemos por consiguiente que, al modelar con casos de uso en lugar de hacerlo con historias de usuario, estaríamos aportando este factor tan importante y quizás tan crucial como es el de considerar a la totalidad de los involucrados con el sistema, además de dejarlo perfectamente modelado con esa herramienta. Wirfs-Brock coincide con nuestro punto de vista al decirnos: "yo creo que tener un punto de vista que cierre bien es muy importante (esto es por qué yo uso casos de uso y desarrollo centrado en el uso--- no necesariamente sólo métodos XP), necesita ser manejado en un modo que respete el tiempo y el talento de todos los involucrados".[23]
Como argumento final, Ron Jeffries reconoce la necesidad de emplear casos de uso para capturar este aspecto que las historias de usuario no contemplan. Él finalmente nos reconoció: "Si, ciertamente necesitás el punto de vista de todos los stakeholders" mientras que siguió sosteniendo este punto que nosotros consideramos un tanto débil "en XP, el rol del Usuario tiene la responsabilidad de representar a todos los stakeholders", para terminar aceptando la solución que proponemos para poder capturar la visión de todos los involucrados: "No se especifica en XP cómo se tiene que hacer esto. Los casos de uso, o los analistas de negocios o los analistas de productos... todos estos son ejemplos de lo que el usuario de XP podría hacer para estar seguro de tener todos los puntos de vista"[24]. No quedan dudas, entonces, que la falencia de las historias de usuario en cuanto a la consideración de los stakeholders puede ser corregida por los casos de uso.

D. Impacto de los casos de uso en las planificaciones y en la medición de la velocidad del proyecto

En XP existen otras dos actividades fuertemente ligadas a las historias de usuario y que si las reemplazamos por casos de uso, pasarían ahora a depender de éstos. Por otra parte, y siguiendo con el propósito de no desviarnos ni un ápice –siempre y cuando sea posible- de la línea filosófica que da fundamento a XP, veremos si el reemplazo de herramientas que propusimos tendría algún impacto en las planificaciones de versiones y en la medición de la velocidad del proyecto.
Ya comentamos que a fin de poder realizar las planificaciones es necesario poder establecer el esfuerzo que se requiere para poder implementar cada una de las historias de usuario. Si hemos cumplido con aquella restricción de Beck acerca de que la herramienta que empleemos para capturar los requisitos debería permitirnos realizar estimaciones razonables, podríamos decir que estamos en condiciones de estimar tiempos y recursos con absoluta tranquilidad en base a nuestros casos de uso.
Por otra parte, recordemos que hemos reducido el nivel de complejidad y detalle de los casos de uso a fin de reemplazar con ellos las historias de usuario, de modo que podemos estar tranquilos de contar con una herramienta que nos permitiría llegar tan lejos como quisiéramos en el detalle y así poder estimar con tanta exactitud como fuera necesario.
En función de estas estimaciones se harán las planificaciones de las versiones e iteraciones, y como hemos efectuado el reemplazo en una relación de uno a uno y a la vez hemos mantenido el estilo de las historias de usuario, no vemos la razón de que algo impidiera planificar en función de los casos de uso.
La velocidad de proyecto se mide en XP en función de la *cantidad de historias de usuario por unidad de tiempo*, pero ahora debería hacerse en función de *los casos de uso implementados por unidad de tiempo*.
Un aspecto importante a tener en cuenta en este punto consiste en que las historias de usuario normalmente se dividen a fin de adaptarlas a las planificaciones, por ejemplo, porque una historia de usuario podría tener una duración mayor que la de una iteración. Mientras hay autores que

23 Íbidem.
24 JEFFRIES, Ron. Carta personal a Fernando Pinciroli (16/VII/2002).

consideran que esta situación no se podría salvar si se emplearan casos de uso debido a que ellos no se explotan; a que constituyen requisitos completos, y por lo tanto no es posible subdividirlos sin dejar de lado el concepto de caso de uso; y a que las únicas partes en las que se podrían dividir sería en sus propios pasos, nosotros tenemos otra opinión.[25]

Aceptamos por completo todos los argumentos recién expuestos, pero creemos que existe la posibilidad de subdividir un caso de uso o de tomar una parte de él: el empleo de casos de uso vinculados por medio de la relación de *inclusión,* la división en cursos alternativos (subflujos) y la división en cursos excepcionales son técnicas normales para la planificación de versiones. Sería impensado que los miembros del equipo de XP dividan una historia de usuario en una parte que no tenga sentido en si misma, es decir, que no posea un grado de cohesión suficiente, que no se pueda implementar como un módulo. Y si esa porción de requisito cumple con esas condiciones, bien puede ser modelada en cualquiera de las formas que proponemos.

Con nuestra alternativa de modelado no sólo estaríamos obteniendo un beneficio concreto y real para el problema en mano, esto es, la planificación, sino que estaríamos haciendo el aporte de una herramienta de modelado poderosísima desde el punto de vista de la *refactorización*, concepto clave de XP.

Por último, también sería posible modelar los agrupamientos lógicos de requisitos de estas planificaciones -versiones e iteraciones- mediante diagramas de paquetes de UML, pero a fuerza de verdad, creemos que ya nos estamos entusiasmando demasiado y quizás nos podamos estar alejando del minimalismo exigido por XP. Pero en caso de que fuera necesario presentar de algún modo estos agrupamientos de elementos a implementar, esta alternativa sería sin dudas más que válida.

Nuestra hipótesis inicial de reemplazar las historias de usuario con casos de uso quedaría fundamentada con todas estas reflexiones y ese fundamento, expresado en los siguientes términos:

a) la formalidad de los casos de uso no necesariamente se debe trasladar al usuario final
b) la funcionalidad que presentan los casos de uso puede ser la que poseen las historias de usuario y el detalle técnico de tareas para los programadores en XP
c) la falta de consideración de todos los participantes de cada historia de usuario puede ser corregida por los casos de uso
d) las planificaciones de versiones e iteraciones se pueden realizar en función de los casos de uso
e) la medición de la velocidad del proyecto se puede realizar en función de los casos de uso.

VI. Empleo de diagramas de clases y de secuencia como complemento de las tarjetas CRC

Además de los casos de uso, existen otras herramientas que no pueden ser dejadas de lado. Sabemos que los puntos más salientes del empleo de tarjetas CRC en XP consisten en la simplicidad de su uso, en la poca -o casi nula- instrucción que se requiere para poder comenzar a emplearlas, en la facilidad de su manejo e intercambio entre las personas involucradas en el proyecto, en la rapidez con la que se puede escribir -e incluso garabatear- en ellas y, fundamentalmente, en otro aspecto menos objetivo pero no menos importante: en que son coherentes con la simplicidad de XP -dado que XP y las tarjetas CRC fueron creadas por Beck-.

Como su nombre lo indica, las tarjetas CRC (*Class-Responsibility-Collaboration*) poseen información suficiente y clara acerca de las responsabilidades de los objetos, poseen información suficiente y no tan clara acerca de las colaboraciones y no poseen información suficiente -por lo que no podemos juzgar, entonces, su claridad- acerca de la estructura estática entre las clases.

25 DAVIES, Rachel. Op.cit. Pág. 49.

Rebecca Wirfs-Brock utilizó las tarjetas CRC en su método *Responsibility-Driven-Design*[26] -de ahora en más RDD- casi como herramienta principal a fines de los '80 y comienzos de los '90, pero las complementó con otras herramientas. Utilizó principalmente los grafos de estructuras y los diagramas de Venn para el modelado de la estructura de las clases -aunque para poner de manifiesto sólo las relaciones de generalización y no de toda la estructura estática- y los grafos de colaboraciones, una suerte de diagrama de interacción en la que expresaba gráficamente las colaboraciones -o la interacción- entre los objetos del sistema incluidos en las tarjetas CRC.[27]

Este mismo planteo, aunque ampliando el modelado a la totalidad de la estructura estática de las clases, es el que proponemos como complemento de las tarjetas CRC. La necesidad radica principalmente en lo expuesto en el primer párrafo de este punto: pretendemos modelar los aspectos fundamentales que las tarjetas CRC dejan de lado y los que describe sin la claridad suficiente.

Adicionalmente, las tarjetas CRC deberán ser traducidas en implementaciones concretas, debiendo pasar de una herramienta bastante informal a otra u otras más formales. Rebecca Wirfs-Brock nos comenta al respecto que "nosotros, también, experimentamos un cambio cuando nos mudamos desde un modelado informal con tarjetas CRC a descripciones de secuencias de colaboraciones e interacciones más formales. A veces necesitamos pintar un cuadro amplio de colaboraciones; otras veces necesitamos ofrecer explicaciones muy exactas".[28]

De esta manera, deberíamos contar con el diagrama de clases y con los diagramas de interacción del UML -diagramas de secuencias y de colaboraciones- a fin de modelar los aspectos antes citados, con el mismo objetivo que el buscado por Wirfs-Brock en RDD, aunque reemplazando sus herramientas.

La importancia de esta propuesta no necesita ser demostrada, sino tan sólo la oportunidad de hacerlo dentro del contexto de XP. Como era de esperar, Grady Booch nos decía: "me gustan las tarjetas CRC, y yo mismo las utilizo en sesiones de tormenta de ideas. No obstante, las cosas como los diagramas de clases y sus diagramas de secuencia asociados te fuerzan a considerar semánticas más profundas".[29]

La opinión de Rebecca Wirfs-Brock coincide exactamente con la de Booch, y nos manifestaba: "veo a las tarjetas CRC como la vista de primer nivel de un modelo de objetos. Y, por supuesto, se pueden suplementar por más vistas del modelo que muestren colaboraciones (diagramas de secuencia y/o de colaboraciones) y las relaciones estructurales y de herencia entre objetos. Nuevamente, veo al UML como el siguiente nivel de detalle. A veces ese detalle ilumina las cosas que no podés ver en las tarjetas CRC. Muchas veces, yo pienso que te perdés una visión amplia de tu diseño, sus objetos claves y sus colaboraciones cuando comenzás demasiado detalladamente. De esta manera, me gustan ambas vistas".[30]

Pero, por supuesto, estas voces no son exactamente las de los creadores de XP, y si bien sus palabras apoyan nuestra hipótesis, habría que escuchar a los principales "afectados" por nuestra propuesta. Justamente esto es lo que haremos a continuación.

Si bien Ron Jeffries nos planteaba, como buen exponente de XP: "no sé para qué querría esa información" haciendo referencia al aporte adicional de los diagramas de clases y de interacción a la información de las tarjetas CRC, termina reconociendo -¿quizás un poco a su pesar?- que "el equipo

---

26 WIRFS-BROCK, Rebecca et al. **Designing Object-Oriented Software**. Englewood Cliffs, Prentice-Hall, 1990.

27 Cfr. PINCIROLI, Fernando. "El Análisis de Sistemas Orientado a Objetos: evolución, principales métodos y tendencias". En: AA.VV. **Desarrollo de Sistemas de Información Complejos: Una visión Global**. Prólogo de Kenneth Rubin. A ser publicado por la Universidad de Vigo, España

28 WIRFS-BROCK, Rebecca. "Describing Collaborations". Cap. 7 enviado por la autora a Fernando Pinciroli en julio de 2002 antes de su publicación en WIRFS-BROCK, Rebecca y Alan McKean. **Object Design. Roles, Responsibilities and Collaborations**. Addison-Wesley, 2003.

29 BOOCH, Grady. Carta personal a Fernando Pinciroli (11/VII/2002).

30 WIRFS-BROCK, Rebecca. Carta personal a Fernando Pinciroli (8/VII/2002).

de XP podría dibujar un diagrama de clases en el pizarrón, o raramente, un diagrama de secuencia", pero no puede con su genio y finaliza diciendo "ellos (el equipo de XP) harían esto informalmente, sin herramientas, en la forma en la que lo haríamos vos y yo en el café".[31]
Pero es nuevamente Kent Beck quien, quizás por su experiencia, es más sensato y da mayor fundamento a nuestra propuesta. Él nos comentaba que, además de las tarjetas CRC, "las otras herramientas que utilizamos son borradores en la pizarra, pruebas y código. Yo recomiendo... aprender a utilizar estas herramientas bien, y entonces utilizar otras herramientas donde éstas sean inadecuadas. Nuevamente, no estoy en contra de los diagramas o de las herramientas de diagramación, sino que nosotros deberíamos tener cuidado del riesgo costo/beneficio cuando decidamos...".
A partir nuestros razonamientos y de los testimonios directos de los autores acerca del tema, podemos concluir en varios puntos fundamentales:

a) los diagramas de clases y los de interacción son las herramientas más adecuadas que se podrían utilizar como complemento de las tarjetas CRC
b) las tarjetas CRC necesitan ser complementadas (siempre o a veces, según la tendencia de los autores)
c) no contradice en absoluto el espíritu de XP el hacerlo

VII. Aplicabilidad de XP

Finalmente, sólo queremos dedicar unos pocos párrafos con el objeto de aportar algunas consideraciones acerca de la aplicabilidad de XP, vista desde el espíritu que exige, el tipo de desarrollos en los que se aplica y la disponibilidad de los usuarios, de modo de dar más fuerza a nuestras conclusiones.
El espíritu de XP: sólo se puede lograr en aquellas personas que, además de haber sido entrenadas en las prácticas del método, estén absolutamente convencidos de todos sus principios y reglas y sean absolutamente disciplinadas. Son muchos quienes hablan del "evangelismo de XP".[32]
La programación en XP: las reglas de XP son aplicables principalmente a proyectos en donde la programación sea orientada a objetos y la mayoría de ellas surgieron de entre los programadores de Smalltalk -de hecho Kent Beck es un referente-. Por ejemplo, la programación de a pares se puede dar con un estilo de programación en donde los programas son cortos -las funciones de Smalltalk no superan las siete líneas en promedio- y por consiguiente las responsabilidades se deben distribuir uniformemente en varias clases del sistema, lo que incrementa el "diálogo" entre las clases y a la vez posibilita el diálogo entre los programadores, para quienes es mejor discutir entre varios la forma en la que estas responsabilidades se deberían distribuir. Esta idea se difundió entre los programadores de Smalltalk antes del nacimiento formal de XP[33]. Si se tuvieran que escribir programas largos y monolíticos la cosa sería bastante diferente.
Los usuarios en XP: debemos ser realistas y sabemos que los usuarios no siempre podrán o querrán estar presentes. Debido al "fanatismo" de muchos practicantes de XP, en varias oportunidades se fuerza la aplicabilidad del método, haciéndolo contraproducente. Rebecca Wirfs-Brock nos comentaba que conoce un caso en donde se le entregó un teléfono celular al usuario a fin de tenerlo "siempre presente", como es uno de los principios fundamentales de XP, al menos virtualmente[34]. Incluso, si el usuario está muy ocupado o no tiene la predisposición para estar

31 JEFFRIES, Ron. Íbidem.
32 Cfr. http://c2.com/cgi/wiki?XpEvangelism
33 Cfr. SHARP, Alec. "Getting Started with Object-Oriented Thinking". En: **Smalltalk by Example. The Developer's Guide**. S.l., McGraw-Hill, 1997. Págs. 72 y 73.
34 WIRFS-BROCK, Rebecca. Carta personal a Fernando Pinciroli (15/VII/2002).

siempre presente, la autora sostiene con muchísimo sentido común que "si el proyecto está organizado de esta forma... yo te recomendaría que modifiques tus prácticas de desarrollo", cosa un poco difícil que reconozcan los seguidores de XP. Y concluye: "si estás tratando de ser ágil, hay maneras de reunir los requisitos de uso y prueba que cumplan con las necesidades de usuarios y stakeholders sin tener al cliente sentado todo el tiempo con el equipo de desarrollo. En algunos sentidos, la forma de XP no es muy amigable para un usuario ocupado. Así... es cuestión de balancear las necesidades y tiempo de todos los involucrados".[35]
El mismo Beck reconoce que "no sentarse juntos tiene un impacto enorme", pero no dice que por esa razón dejaría de utilizar XP. Agrega además: "cuando no es posible sentarse juntos, generalmente trato de encontrar aun una manera de tener algunas personas de negocio y técnicas sentados juntos en cada sitio. La separación estricta de personas de negocio y técnicas es simplemente estúpido, aunque...", afirma con gran sinceridad, "juntarlos puede ser extremadamente, incluso imposible, inconfortable".[36]
Ron Jeffries es concluyente al respecto. La aplicabilidad de XP tal como está planteado, se termina cuando no existe la disponibilidad de los usuarios: "Estrictamente hablando, XP se define solamente en las circunstancias descriptas: usuario (rol) presente, programadores juntos". La alternativa a esto pasa a ser una adaptación de XP, pero ya pierde su espíritu, principalmente porque la documentación pasa a ser el elemento que cobra el protagonismo al resolver este inconveniente de la presencia de los usuario: "Muchos equipos del mundo real están utilizando prácticas y principios de XP sin que se sigan todas las restricciones. Usualmente, el resultado es un proceso menos efectivo -pero no menos efectivo que cualquier otro proceso que tenga clientes remotos o programadores distribuidos- y el equipo debe compensar produciendo más documentación, realizando más llamadas telefónicas, más viajes, más correo electrónico, etc".[37]

VIII. Conclusión

Quienes promueven XP afirman que esta disciplina surgió a partir de la necesidad de reducir los costos de desarrollo, de aportar una solución a los desvíos en las planificaciones de los proyectos de software, de la observación y compilación de las mejores prácticas de programación, de la extrapolación de soluciones que se utilizan normalmente en otros ámbitos y que dan resultados reales a la industria del software, de la necesidad de lograr la mayor satisfacción de los usuarios, entre otras razones. Pero curiosamente, todos estos puntos pueden encontrarse en boca de cualquier metodologista, sin importar la corriente filosófica a la que pertenece, el método o enfoque que propone, la época o el lugar. Entonces, ¿cuál es la verdad?
En principio debemos "creer" a todos, pero recién luego de llevar a la práctica las propuestas más razonables podremos obtener una conclusión. Por supuesto que esta conclusión estaría condicionada por la situación en la que se pusiera en práctica la técnica, las características de quienes la pusieran en práctica, y muchos otros factores.
Pero hay una realidad indiscutible: independientemente de las discusiones, planteos, vencedores y derrotados, existe un estándar mundial para el análisis de sistemas, el UML, y por esa razón no deberíamos escatimar esfuerzos en adoptarlo y en adaptarlo. Ese fue el tema central de este trabajo y el sujeto de la adopción y adaptación fue XP.
Creemos que además de discutir acerca de las bondades y aplicabilidad de esta técnica, un punto fundamental consistía en intentar utilizar el UML en el contexto de XP, tanto desde el punto de

35 Ídibem.
36 BECK, Kent. Carta personal a Fernando Pinciroli (15/VII/2002).
37 JEFFRIES, Ron. Carta personal a Fernando Pinciroli (16/VII/2002).

vista del empleo del lenguaje estándar como de los aportes concretos que éste puede hacer por medio de sus herramientas a las falencias que pudieran tener aquellas que fueron propuestas por XP. Hicimos un gran esfuerzo para poder contar con los protagonistas principales de estas disciplinas y consideramos que sus comentarios fueron muy importantes a fin de dar mayor claridad a las demostraciones de nuestras hipótesis. Sólo resta esperar que estas ideas sirvan como aporte a nuestra disciplina.

IX. Bibliografía

ARMOUR, Frank y Granville Miller. **Advanced Use Case Modeling**. s.l., Addison-Wesley, 2000.

BECK, Kent, carta personal a Fernando Pinciroli (8/VII/2002).

BECK, Kent, carta personal a Fernando Pinciroli (15/VII/2002).

BOOCH, Grady, carta personal a Fernando Pinciroli (11/VII/2002).

C3 TEAM. "Case Study: Chrysler Goes to 'Extremes'". En: Distributed Computing. Oct. de 1998.

COCKBURN, Alistair. **Writing Effective Use Cases**. s.l., Addison-Wesley, 2001.

DAVIES, Rachel. "The Power of Stories". Cap. 11. Londres, Connextra, s/f.

FOWLER, Martin, carta personal a Fernando Pinciroli (8/VII/2002).

JEFFRIES, Ron, carta personal a Fernando Pinciroli (8/VII/2002).

JEFFRIES, Ron, carta personal a Fernando Pinciroli (16/VII/2002).

PINCIROLI, Fernando. "Introducción al Unified Modeling Language: un nuevo estándar mundial para el análisis orientado a objetos". En: AA.VV. **Ingeniería del software y reutilización: aspectos dinámicos y generación automática**. Orense, Tórculo, 1998.

PINCIROLI, Fernando. "El Análisis de Sistemas Orientado a Objetos: evolución, principales métodos y tendencias". En: AA.VV. **Desarrollo de Sistemas de Información Complejos: Una visión Global**. Prólogo de Kenneth Rubin. A ser publicado por la Universidad de Vigo, España.

PINCIROLI, Fernando et al. "The use of scripting for the transition between the use cases and the interaction models". En: World Multiconference on Systemics, Cybernetics and Informatics SCI 2000. Proceedings, vol. II. Orlando, 2000.

WIRFS-BROCK, Rebecca, carta personal a Fernando Pinciroli (8/VII/2002).

WIRFS-BROCK, Rebecca, carta personal a Fernando Pinciroli (15/VII/2002).

WIRFS-BROCK, Rebecca. "Describing Collaborations". Cap. 7 enviado por la autora a F. Pinciroli en julio de 2002 antes de su publicación en WIRFS-BROCK, Rebecca y Alan McKean. **Object Design. Roles, Responsibilities and Collaborations**. Addison-Wesley, 2003.

WIRFS-BROCK, Rebecca et al. **Designing Object-Oriented Software**. Englewood Cliffs, Prentice-Hall, 1990.