Prototipado del Software

Guillermo Choque Aspiazu
gchoque@inf.umsanet.edu.bo

Resumen
En este articulo se presentan los fundamentos sobre los cuales se edifica la teoría del prototipado, se define un prototipo del software de manera inicial y se indican los resultados a los cuales puede conducir el prototipado, reconociendo que existen algunos términos que colaboran en la búsqueda de la definición adecuada del termino prototipado, incluyéndose la revisión de los prototipos de diseño, evolucionario, maqueta, armazón, entre otros. Luego se identifican las estrategias del prototipado y una clasificación primaria de su topología, para finalmente mostrar un modelo de construcción de prototipos y los métodos y herramientas necesarios para este cometido.
Palabras clave. Prototipado, software, evolucionario, descartable, horizontal, vertical.

1. INTRODUCCION

Un prototipo del software es una versión preliminar o un modelo de todo o parte de un sistema después de que un acuerdo completo es realizado para desarrollarlo [IT-STARTS, 1989]. Un prototipo del software puede también ser parte o el todo de un sistema que es desarrollado y entregado utilizando un enfoque iterativo en el cual los usuarios se encuentran involucrados. El prototipado es el proceso de creación de un prototipo.

El objetivo de crear prototipos es asistir, en alguna forma, al desarrollo de los objetivos o la entrega de los sistemas. Los resultados principales del desarrollo del software (Fig. 1.) que pueden ser direccionados por el prototipado son la elicitación, la demostración y la evaluación de lo siguiente:

a) Requerimiento de datos y estructuras
b) Requerimientos de funciones
c) Operación y rendimiento
d) Necesidades y resultados organizacionales

En usos normales, un prototipo es un modelo tentativo o una versión preliminar de un producto [Skyes, 1996]. En la ingeniería convencional, el prototipado a escala reducida, con versiones simplificadas de productos o con un producto de pre-producción, es una tradición largamente establecida. La idea de producir una construcción, un puente, un automóvil, o un aeroplano sin un prototipo o modelo es casi inconcebible [Jones, 1986]. Prototipar en una etapa anterior al desarrollo de un producto permite la evaluación y el ajuste antes de que el diseño sea finalizado.

La analogía entre el prototipado en ingeniería y el prototipado en software aparenta contar con un criterio de validez certero. La analogía puede ser fácil, sin embargo, debido a la naturaleza del software es diferente de los productos de la ingeniería convencional [Floyd, 1984], aún en alguno de sus productos tales como la construcción de puentes. Las diferencias son las siguientes:

a) La mutabilidad de un prototipo de software es mucho mayor que la de un prototipo de ingeniería.
b) A diferencia de muchos productos de ingeniería, el software no es fabricado, pero es un producto único. La replicación no es un problema como lo es en la ingeniería. El problema en el software es la producción del prototipo.
c) Los productos del software frecuentemente tienen características que no se encuentran claramente detalladas al inicio del desarrollo del prototipo.
d) La relación entre el producto software y su prototipo frecuentemente no es clara. El prototipo puede llegar a ser en algunos casos el producto mismo.

El desarrollo iterativo o incremental puede frecuentemente ser un termino más adecuado que el prototipado del software [Graham, 1989]. Sin embargo, el termino “prototipado” continua siendo ampliamente utilizado.

2. DEFINICIONES

Se puede decir que la definición de prototipado es una que ha sido extraviada en el tiempo. Sin embargo, de acuerdo a Tanik (1989) existe un numero de otros términos asociados de manera común con el sujeto de estudio, algunos de los cuales son relevantes y algunos de los cuales no lo son. La siguiente lista proporciona algunos de dichos términos:

a) Amplio. Este es un prototipo con un alto grado de funcionalidad, pero una limitada interface de usuario. Estos prototipos son valorados para estimar el rendimiento o eficiencia, o también para desarrollar de manera experimental funciones o estructuras de datos.
b) Diseño. Esta es la etapa en el desarrollo del software donde se planifica la forma del sistema. El prototipado no es una técnica de diseño. Un buen diseño es esencial para el prototipado efectivo y al mismo tiempo un prototipado puede ser un medio efectivo para la evaluación del diseño.
c) Desarrollo evolucionario[1]. Es un prototipo que llega a ser el sistema para entrega con una cantidad substancial de trabajo. El desarrollo evolucionario puede ser empleado para conducir todos los resultados del desarrollo (Fig. 1).
d) Desarrollo incremental. Esta es una estrategia en la cual el sistema para entrega es desarrollado en pasos pequeños. Cada paso esta contenido de manera propia en su propio software y documentación [Graham, 1989]. Las entregas al usuario pueden ser también incrementales. No existe implicación en el sentido de que el prototipado deba ser empleado en el desarrollo incremental y liberado luego como producto. Esta estrategia puede, sin embargo, ser una forma de manejar el prototipado.
e) Maqueta. Este es un prototipo estático y sub funcional que demuestra la apariencia externa del sistema a construir. Las maquetas típicamente se dirigen a cubrir los resultados cosméticos en el desarrollo del software, aunque pueden ser también valorados como elementos para elicitar los requerimientos funcionales de los usuarios y para el desarrollo de interfaces de usuario (Fig. 1).
f) Modelo. Esta es una abstracción del software de alcance limitado. Puede ser inanimado. Los modelos son frecuentemente herramientas de software de propósito general utilizados para probar casos específicos. Los modelos normalmente no llegan a ser parte del sistema para entrega, aunque sus mecanismos internos deben ser siempre utilizados. Los modelos tienden a ser útiles para probar las funciones y determinar las estructuras de datos.
g) Simulación. Esta es una representación dinámica, matemática o algorítmica de una actividad. Los simuladores, al igual que los modelos, son frecuentemente herramientas de software de propósito general aplicadas a problemas específicos. Los simuladores normalmente no podrían ser parte de un sistema para entrega. La simulación es valuable principalmente para la estimación del rendimiento (Fig. 1).
h) Armazón. Este es un prototipo de amplio alcance que es utilizado para evaluar o esclarecer reglas generales y restricciones para el sistema para entrega. Las funciones y datos son mas o menos completas, pero rudimentarias. El armazón puede llegar a ser la estructura del sistema a ser entregado. Los prototipos de armazón son utilizados para elicitar los requerimientos funcionales y de datos (Fig. 1)
i) Prueba y evaluación. Estos términos son similares, excepto que la prueba tiene una connotación mas limitada para determinar la correctitud de algo. La evaluación implica determinación de valor, como también la correctitud. Por ejemplo, una solución computacional puede estar correcta en términos técnicos, pero puede no tener un valor debido a que el usuario no conoce como utilizarlo. El prototipado no es una técnica de prueba; sin embargo puede ser considerado como un asistente de evaluación.
j) Descartable[2]. Este es un prototipo que es redesarrollado utilizando un enfoque convencional de desarrollo del software. Este tipo de prototipo es típicamente utilizado para elicitar requerimientos de datos y funciones (Fig.1).

[1] Prototipo de enfoque abierto o evolutivo.

[2] Prototipo desechable o de enfoque cerrado.

k) Desarrollado por usuario. Este prototipo se produce cuando los usuarios producen su propio software. Los desarrolladores usuario tienden a utilizar enfoques informales para el desarrollo del software, no representando este uso el prototipado. Los desarrolladores usuario pueden ser prototipadores bastante eficientes, esto debido a su conocimiento especializado respecto al problema.

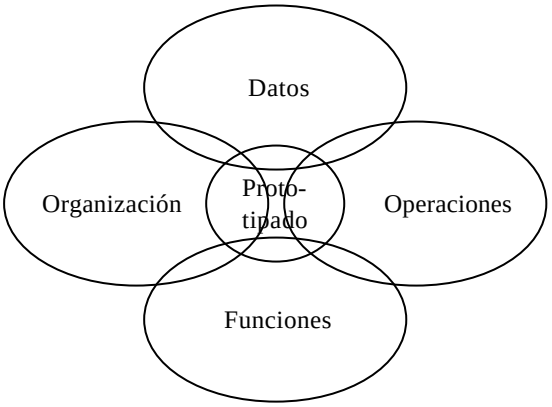

Fig. 1. Resultados dirigidos por el prototipado

3. PROTOTIPO BASICO

Es preferible restringir el uso de la palabra prototipado a las actividades que necesitan elicitación (Fig. 2), implementación del software, y evaluación de usuario de la implementación, lo cual resulta en la creación del prototipo final. El termino entrega es reservado para las actividades asociadas con la creación del objetivo o resultado a partir del prototipo implementado. Las actividades típicas de entrega son la documentación y el entrenamiento[3].

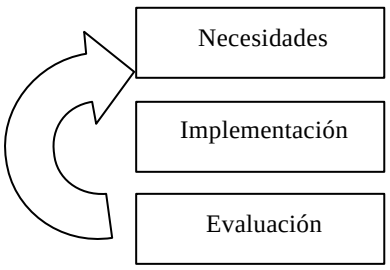

Fig. 2. Prototipo simple

Los enfoques tradicionales y los de ingeniería del software siguen el esquema de prototipado en una manera bastante general. La principal diferencia entre el prototipado y los otros enfoques es que la iteración o retroalimentación[4] puede ocurrir en cualquier punto del proceso de prototipado. No se supone que la retroalimentación sea necesaria en el enfoque tradicional o en el de la ingeniería del software. En el prototipado, existe también una implicación que el prototipo sirva como la sentencia formal de las necesidades del sistema. La característica que puede ayudar a distinguirla de los enfoques de desarrollo anárquicos es que la retroalimentación surge de la elicitación de las necesidades en acuerdo con los usuarios.

4. ESTRATEGIAS DE PROTOTIPADO

Existen dos estrategias grandes de prototipado: el prototipo descartable y el evolucionario [Parbst, 1984].

Los prototipos descartables (Fig. 3) son aquellos en los cuales el prototipo es redesarrollado o trasladado en otra forma antes de que el sistema objetivo sea entregado [Hetmakpour & Ince, 1986]. El proceso de traslación puede ser considerado como parte del prototipado o como parte de la entrega. De manera típica, el diseñador del software utiliza el prototipo descartable como un sustituto parcial o total de la etapa de análisis en el ciclo de vida clásico [Lugi, 1988]. Hablando de manera estricta, ninguna parte del prototipo descartable puede ser reutilizada.

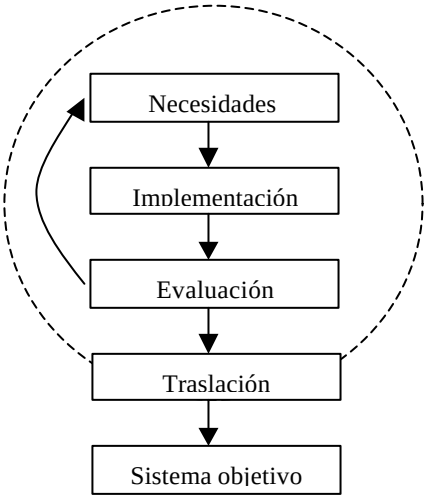

Fig. 3. Esquema de prototipo descartable

[3] También conocido como actividad de capacitación.

[4] Se refiere al feedback de la teoría general de sistemas.

Los prototipos evolucionarios (Fig. 4) son prototipos que por si mismos llegan a entregar los productos finales o sistemas objetivo. Ince (1987) analiza una diferencia entre prototipos evolutivos e incrementales, paro esa distinción aparente ser solamente académica.

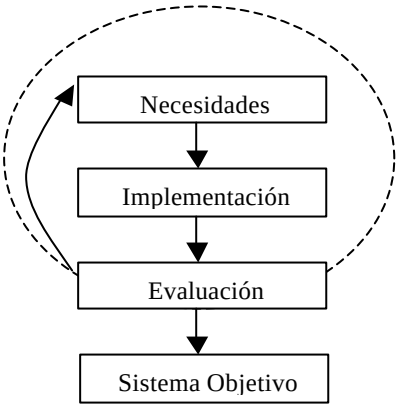

Fig. 4. Esquema de prototipo evolucionario

El prototipado evolucionario puede ser apropiadamente denominado desarrollo incremental [Graham, 1989]. El prototipo evolucionario es removido en las etapas finales de la ingeniería del prototipado, a diferencia del prototipo descartable.

5. ALCANCE DEL PROTOTIPADO

Los prototipos pueden ser utilizados para el desarrollo de sistemas completos o solo para subconjuntos del mismo[5]. Los prototipos pueden ser internos o externos desde el punto de vista de los usuarios, aunque tradicionalmente ellos tienden a enfocarse sobre las superficies externas de los sistemas. Los prototipos también pueden exhibir funcionalidad horizontal o vertical (Fig. 5).

Los prototipos horizontales no pueden ejecutar una aplicación de manera completa. En otras palabras, los usuarios no pueden utilizar estos prototipos para ejecutar una tarea completa. Los prototipos horizontales típicos son los amplios y los de maqueta. Los prototipos horizontales, especialmente los que son parciales, fueron ampliamente utilizados por los desarrolladores de software, probando que de manera general son benéficos. Estos tienen un efecto no profundo en el desarrollo del software, debido a que no permite a los usuarios evaluar sus necesidades de software de manera completa.

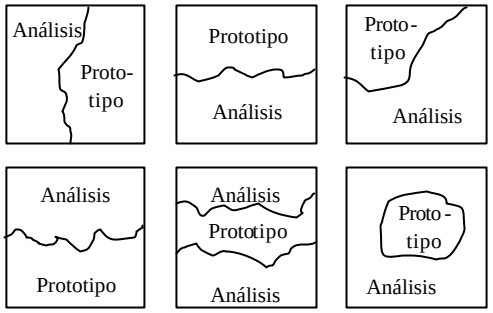

Fig. 5. Alcance del prototipado

Los prototipos verticales, tanto parciales y completos, son de mucho mas interés que los prototipos horizontales. Los prototipos verticales exhiben funcionalidad completa, aunque la extensión de sus aplicaciones sea limitada. Pueden también ser deficientes en rendimiento o apariencia final. Sin embargo, un usuario puede ser capaz de llevar a cabo, de manera realística, una tarea completa. Los prototipos verticales, por consiguiente, normalmente tienen los siguientes componentes:

- ?? Interface de usuario
- ?? Algoritmos y manejo de datos
- ?? Estructura de datos y almacenamiento

Los prototipos verticales son interesantes debido a que permiten a los usuarios evaluar sus necesidades de modo realístico. El efecto en el desarrollo del software al ser capaz de evaluar las necesidades del usuario en este sentido es tanto benéfico y profundo. Los prototipos verticales son mas adecuados a ser incluidos en un sistema para entrega [Graham, 1989].

6. MODELO DE CONSTRUCCIÓN DE PROTOTIPOS

En el proceso de aplicar el paradigma de la ingeniería del software a la construcción de prototipos se identifica al cliente que define, en primera instancia, un conjunto de objetivos generales para el desarrollo del software, pero no identifica de manera clara los requerimientos detallados de entrada, proceso o salida. Es precisamente en este entorno donde el paradigma para la construcción de prototipos se constituye en una respuesta adecuada.

[5] Es decir con prototipos parciales.

Según Pressman (2002), este paradigma comienza con la recolección de requerimientos de usuario, luego le sucede un diseño rápido; este diseño toma como núcleo la representación de aspectos del producto software que deben ser

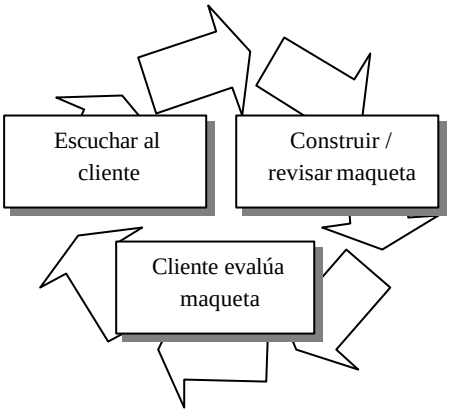

Fig. 6. Paradigma de construcción de prototipos.

visibles para el usuario. El diseño rápido conduce a la construcción del prototipo, luego es evaluado por el usuario y su utilidad principal radica en que ayuda a refinar los requerimientos del producto software a ser desarrollado. La iteración se produce cuando el prototipo se pone a punto para satisfacer las necesidades del cliente, permitiendo al mismo tiempo que el desarrollador comprenda mejor lo que se necesita hacer (Fig. 6).

7. METODOS Y HERRAMIENTAS

Según Tanik (1989), para crear prototipos rápidos, se encuentran disponibles tres clases genéricas de métodos y herramientas:

a) **Técnicas de cuarta generación (T4G)**. Comprenden una amplia gama de lenguajes de consulta e informes de bases de datos, generadores de programas y aplicaciones y de otros lenguajes no procedimentales de muy alto nivel. Estas técnicas permiten generar código ejecutable de manera rápida y son ideales para la creación rápida de prototipos.

b) **Componentes reutilizables**. Otro enfoque para crear prototipos rápidos consiste en ensamblar, mas que construir, el prototipo mediante un conjunto de componentes software existentes. La combinación de prototipos con la reutilización de componentes de programa solamente funcionará si se desarrolla un sistema de biblioteca de componentes catalogados y de libre disposición. Debe resaltarse que se puede utilizar un producto software existente como prototipo de uno nuevo y mejorado.

c) **Especificaciones y entornos formales**. Durante las pasadas dos décadas, se desarrollaron varios lenguajes formales de especificación y herramientas como sustitutos de las técnicas de especificación con lenguaje natural. Actualmente, los desarrolladores de estos lenguajes se encuentran desarrollando entornos interactivos que: (1) permitan al analista crear interactivamente una especificación basada en un lenguaje, (2) puedan invocar herramientas automáticas que traducen la especificación basada en el lenguaje en código ejecutable, y (3) permitan al cliente utilizar el código ejecutable del prototipo para refinar los requisitos formales.

8. CONCLUSIONES

En este articulo se presentaron los fundamentos sobre los cuales se edifica la teoría del prototipado, se procedió a la definición de prototipado desde el reconocimiento de algunos términos como los de prototipo de diseño, prototipo evolucionario, prototipo maqueta, prototipo armazón, entre otros. También se logro identificar las estrategias de prototipado, reconociéndose el papel de los prototipos descartables y evolucionarios, posteriormente se intento una clasificación primaria de su funcionalidad recurriendo a los prototipos horizontales y verticales. Finalmente se presentó un modelo para la construcción de prototipos además de los métodos y herramientas necesarios para el logro de este cometido.

BIBLIOGRAFÍA

Floyd, C; A systematic look at prototyping, in *Approaches to prototyping*. Springer Verlag, 1984.

Graham, D.R.; Incremental development: review of nonmonolithic life-cycle development models. *Information and software technology*, 31, No 1, January, 1989, pp 7-20.

Hetmakpour, S. & D.C. Ince; *Rapid software prototyping*. Open University, Milton Keynes. 1986.

Ince, D.; Model answers. *Informatics*. September 1987, pp. 61-72.

IT-STARTS; *Developers Guide*. National Computing Centre, Manchester, EE.UU. 1989.

Jones, C.; Programming Productivity. McGraw-Hill. New York. 1986

Lugi; Software evolution through rapid prototyping. *Computer*, May 1988, pp. 13-25.

Parbst, F; Experience with prototyping in an IBM-based installation. in *Approaches to prototyping*. Springer Verlag, 1984.

Pressman, R.; *Ingeniería del Software: Un enfoque práctico*. Quinta edición, McGraw-Hill, 2002.

Skyes, J.B.; *The concise Oxford Dictionary of Current English.* Oxford University Press. EE.UU. 1996

Tanik, M.M. y R.T. Yeh (eds); Rapid Prototyping in Software Development. *IEEE Computer*, vol 22, No 5, Mayo 1989.