Criterio para Seleccionar Modelos de Proceso de Software

Introducción

En un esfuerzo para mejorar la calidad del software y su producción, los investigadores han definido algunas variaciones del proceso de desarrollo de software, que van desde el tradicional proceso en cascada hasta el prototipo desechable. Este artículo presenta unas pautas para seleccionar el modelo de proceso mas apropiado para un proyecto particular. Para facilitar esta selección, hemos organizado los modelos de proceso de software en una jerarquía de tres niveles. Después de aplicar todos los criterios, una clasificación relativa de los modelos de proceso es generada para guiar la selección.

1 El Problema de Selección del Modelo de Proceso

Muchos factores influyen en el éxito de un proyecto de software. Éstos incluyen las aptitudes de los desarrolladores de software, el talento de los directores de proyecto, la disponibilidad de herramientas CASE, el entorno informatico, la dificultad del problema a resolver, los recursos disponibles, y el modelo de proceso que está siendo utilizado. En la última categoría hay una gran variedad de modelos de procesos a escoger.

La selección de un modelo de proceso inapropiado puede resultar en (1) un sistema que no llega a satisfacer las necesidades de los usuarios, (2) un aumento del coste, y (3) largos tiempos de desarrollo. Por ejemplo, si las necesidades de el usuario son mal entendidas o cambiantes, el uso de un modelo de proceso convencional puede resultar en un sistema que no satisface las necesidades esperadas. Al contrario, si las necesidades de los usuarios son bien comprendidas y estables, entonces la construcción de un prototipo puede innecesariamente aumentar el precio del desarrollo y del programa.

Hoy en dia los jefes de proyectos eligen un modelo de proceso de software empleando criterios *ad hoc*, injustificados y con frecuencia indocumentados. El modelo espiral de Boehm reconoce la necesidad de una selección del modelo de proceso y basa la selección en un solo criterio: riesgo. Otros autores tambien han identificado esta necesidad. No fue hasta 1990 que se intentó generar un conjunto de criterios y valores para tales selecciones. Este articulo describe 20 criterios y un método para aplicarlos. Usando este método el director evalúa un proyecto con respecto a estos criterios, para desarrollar una lista de prioridades de los modelos de proceso.

Usaremos las siguientes notaciones:

- *C* es el conjunto de 20 criterios, con elementos c_i, siendo el rango *i* de 1 a 20 (c_i es la experiencia del usuario, c_g es la frecuencia de cambios).
- V_i es el vector ordenador de los elementos v_{ij} (los valores que cada criterio c_i puede tomar; por ejemplo V_{11}, V_{12} y V_{13} tienen los valores *novato*, *experimentado* y *experto*, respectivamente, mientras que V_{81}, V_{82} y V_{83} poseen valores de *raro*, *lento* y *rápido*).
- S_{ij} es un indicador binario de qué valor de criterio c_i se aplica a este proyecto particular (así, s_{11}, s_{12}, s_{13}, s_{81}, s_{82} y s_{83} son 0,0,1,1,0 y 0 respectivamente). Los s_{ij} forman una matriz de características de proyecto que refleja los atributos de un proyecto, y
- a_{ij} representa la aplicabilidad de un modelo de proceso para un v_{ij} (por ejemplo, a_{13} =1 en el modelo de proceso convencional indica que el proceso es apropiado para los proyectos con usuarios expertos). Los a_{ij} para cada modelo de proceso crean una matriz que se emplea para determinar la valoración de dicho modelo con respecto a un proyecto particular.

Este artículo aporta el conjunto de criterios, *C*, sus posibles valores, V_i, y una matriz de modelo de proceso para cada modelo que contiene los a_{ij}. Los s_{ij} son proyectos dependientes y se determinan por el director de proyecto.

Para determinar el adecuado modelo para un proyecto, el director sigue un método de tres pasos:

1. Examinar los atributos del proyecto y determinar los s_{ij} para los criterios que mejor describen el proyecto.
2. Para cada modelo de proyecto, se calcula:

$$RATING = \sum_{i=1}^{20} \sum_{j=1}^{3} (s_{ij} * a_{ij})$$

3. El modelo de proceso con el índice más alto es la mejor opción, el segundo más alto es la segunda mejor opción, y asi...

En general, los modelos de proceso de desarrollo de software definen la partición, ordenación, y relaciones temporales que existen entre las actividades del desarrollo de software. Sin embargo, no todos los modelos de proceso dirigen estos al mismo nivel de abstracción. De esta manera algunos modelos de proceso trazan amplios grupos de actividades sobre el ciclo de vida del software mientras que otros están detallados hasta el punto de especificar los métodos particulares y las herramientas usadas para llevar a cabo las actividades descritas. El método descrito arriba debe ser aplicado a cada nivel de abstracción para seleccionar el modelo apropiado en ese nivel.

La Sección 2 de este articulo describe los criterios del proceso de seleccion (*C*) y los rangos de valores para estos criterios (V_{ij}). La Sección 3 describe algunos modelos de procesos en uso hoy día y los organiza dentro de una jerarquía. La Sección 4 establece la correspondencia entre cada valor de criterio y cada modelo de proceso (a_{ij}). La Sección 5 proporciona un breve ejemplo. La Sección 6 resume nuestras conclusiones.

2 Criterios iniciales de Proyecto

Los criterios de proyecto se dividen en las siguentes categorías generales: *personal, problema, producto, recurso, y organización*. La Tabla 1 muestra el rango de valores (V_i) para cada uno de los criterios (c_i).

2.1 Criterio Personal

El criterio personal interesa tanto a desarrolladores como a usuarios, y sus valores están descritos a continuación.

La experiencia de los usuarios en el Dominio de la Aplicación (c_1): Este criterio evalúa el conocimiento del usuario sobre el dominio del problema. Los valores son: V_1= (*Novato, conocedor, experto*). Cada resultado de esta escala puede funcionar al mismo tiempo a favor y en contra del esfuerzo. Los usuarios novatos pueden no conocer mucho sobre el área de problema, pero ellos estarán mas dispuestos a aceptar nuevas o diferentes soluciones. Los usuarios expertos sabrán mucho sobre el problema pero pueden ser reticentes a los cambios.

Habilidad de los usuarios para expresar necesidades (c_2): Este criterio evalúa como los usuarios son capaces de comunicar sus necesidades. Los valores son V_2= (*callado, comunicativo, expresivo*). En el extremo *callado* o *silencioso* están los usuarios que no pueden comunicar lo que necesitan pero lo reconocerán en cuando lo vean. Los usuarios *expresivos* son capaces de decir exactamente lo que piensan que necesitan.

La experiencia de los desarrolladores en el Dominio de la Aplicación (c_3): Este criterio mide el conocimiento de los desarrolladores del dominio del problema. El conocimento del los desarrolladores puede ser el resultado de ser un usuario en el dominio de la aplicación, o por desarrollar otras aplicaciones en el dominio. Los valores son: V_3= (*Novato, conocedor, experto*).

Experiencia de los desarrolladores de Ingenieria de Software (c_4): Este criterio trata la experiencia de los desarrolladores y el conocimento de las herramientas de software, métodos, técnicas, y lenguages necesarios para un buen resultado. Los valores son: V_4= (*Novato, conocedor, experto*).

2.2 Criterio del Problema

El Criterio del Problema trata el problema que debe ser resuelto por el desarrollo de software. Este criterio y sus valores son explicados a continuación:

La madurez de la aplicación (c_5) : Este criterio evalúa la cantidad de conocimiento general del problema. El desarrollo de software en areas de aplicación maduras pueden beneficiarse de resultados de desarrollo previos, aunque hay menos antecedentes de conocimiento en las aplicaciones de domino más recientes. Los valores son: V_5= (*Nuevo, standar, bien establecido*). Un ejemplo del valor *bien establecido* es el área de aplicación de nóminas.
Gran parte del software de inteligencia artificial representa en cambio una nueva aplicación.

Complejidad del problema (c_6): Este criterio mide la complejidad del problema que debe ser resuelto. Los valores son: V_6=(*sencillo, dificil, complejo*). Los grandes sistemas paralelos en tiempo real suelen ser complejos,mientras los mas pequeños sistemas secuenciales suelen ser sencillos.

Requisito para funcionalidad parcial (c_7): Este criterio mide la viabilidad y/o necesidad para entregar productos intermedios que proporcionan sólo una parte de la total funcionalidad del producto final. Los valores son: V_7= (*urgente, deseable*, *no deseable*).
El sistema de sostenimiento de vida en una nave espacial es un ejemplo en donde la funcionalidad parcial no es deseable. La automatización de funciones de oficina es un ejemplo donde la entrega rápida de funciones de tratamiento de textos puede ser deseable.

Frecuencia de cambios (c_8): Este criterio indica la frecuencia con la que cambia el problema. Los valores son: V_8=(*rápido, lento, raramente*). Un ejemplo que cambia rapidamente es un sistema de armamento porque debe responder con rapidez a las cambiantes amenazas del entorno .Una compilación para un lenguaje standar es un ejemplo de un problema cambiante más lento.

Ci	CRITERIOS	*vi1*	*vi2*	*vi3*
c1	Experiencia del Usuario	Novato	Conocedor	Experto
c2	Expresión del Usuario	Callado	Comunicativo	Expresivo
c3	Exper.Desarr. con Aplicación	Novato	Conocedor	Experto
c4	Exper.Desarr. con Software	Novato	Conocedor	Experto
c5	Madurez de Aplicación	Nuevo	Estandar	Bien Establecido
c6	Complejidad del Problema	Sencillo	Difícil	Complejo

Ci	CRITERIOS	*vi1*	*vi2*	*vi3*
c7	Funcionalidad Parcial	No deseable	Deseable	Urgente
c8	Frecuencia de Cambios	Raramente	Lento	Rápido
c9	Magnitud de Cambios	Menor	Moderado	Extremo
c10	Tamaño del Producto	Pequeño	Mediano	Grande
c11	Complejidad del Producto	Sencillo	Difícil	Complejo
c12	Requisitos "-Ilidad"	Flexible	Moderado	Exigente
c13	Requisitos de Interfaz Humano	Menor	Significativo	Crítico
c14	Perfil de Financiación	Bajo-Alto	Estable	Alto-Bajo
c15	Disponibilidad de Fondos	Escaso	Suficiente	Amplio
c16	Perfil de Personal	Bajo-Alto	Estable	Alto-Bajo
c17	Disponibilidad de Personal	Escaso	Suficiente	Amplio
c18	Accesibilidad de Usuarios	Sin Acceso	Acceso Limitado	Acceso Libre
c19	Compatibilidad de Dirección	Sólo Directrices	Flexible	Imposición Estricta
c20	Compatibilidad GC/CC	Básico	Intermedio	Avanzado

Magnitud de cambios (c_9): Este criterio valora el tamaño relativo de los cambios esperados en el problema.
Los valores son: $V_{9=}$(*menor, moderado, extremo*). Un nuevo protocolo de comunicaciones puede desembocar en un pequeño cambio para un sistema de telefonía. La construcción de un nuevo sensor puede requerir un rediseñado completo de un sistema de defensa.

2.3 Criterio de producto

El criterio de producto incumbe al producto actual de software que va a ser desarrollado. El criterio de producto y sus valores están descritos a continuación:

Tamaño de producto (c_{10}): Este criterio mide el tamano supuesto del producto final. Como esta medida esta hecha al comienzo del intento del desarrollo, es sólo una estimación. Usando las definiciones de tamaño de producto de Fairley (indicadas entre paréntesis), los valores son: V_{10}=(*grande* (muy grande y extremadamente grande), *mediano* (mediano y grande), *pequeño* (insignificante y pequeño)).

Complejidad de producto (c_{11}): Este criterio calibra la complejidad del software a desarrollar. Los valores son: V_{11}= (*complejo, difícil, sencillo*).

Requisitos "-ilidad" *("ility" en el original)* (c_{12}): Los "ión" representan los requerimientos de no comportamiento que tienen lugar en un producto de software, como fiabilidad, adaptabilidad, estipulación, utilización, reutilización, y mantenimiento. Este criterio mide el peso relativo de tales requisitos. Los valores son: V_{12}= (*exigente, moderado, flexible*). El software cuyos fallos pueden hacer peligrar la vida es un ejemplo de requisito de fiabilidad exigente y estipulación. El software que se espera usar por largo tiempo puede tener requisitos de adaptabilidad exigente y mantenimiento.

Requisitos de interfaz humano: este criterio mide la importancia del interfaz humano. Los valores son: V_{13}= (*crítico, significativo, menor*).

2.4 Criterios de recurso.

Los criterios de recurso conciernen a los recursos disponibles para el desarrollo. A diferencia de otros criterios, los cuales están evaluados seleccionando un valor de una escala, los criterios de financiación y dotación de personal son evaluados por el perfil del recurso a lo largo del tiempo. Los criterios de recurso y su evaluación son lo siguientes:

Perfil de financiación (c_{14}): este criterio mide la cantidad y disponibilidad de fondos para el desarrollo. Los gráficos mostrados en la figura 1 muestran la forma o configuración de la financiación de la mayoría de los proyectos. El *Perfil (a) Bajo-Alto* muestra la cantidad de aumento de fondos al principio del desarrollo. El *Perfil (b) Alto-Bajo* muestra la mayoria de los fondos disponibles al comienzo del desarrollo. El *Perfil (c) Estable* muestra un nivel estable de los fondos sobre el desarrollo. Así, V_4=(*Bajo-Alto, Alto-Bajo, Estable*).

Disponibilidad de fondos (c_{15}) : este criterio mide la adecuación de fondos disponibles para el desarrollo. Los valores son V_{15}= (*escaso, suficiente, amplio*).

Perfil de dotación de personal (c_{16}): este criterio valora el número de personas disponibles a lo largo del tiempo para el desarrollo. Los perfiles para el personal son los mismos que para los fondos. Como se muestra en la figura 1, V_{16}= (Bajo-Alto, Alto-Bajo, Estable).

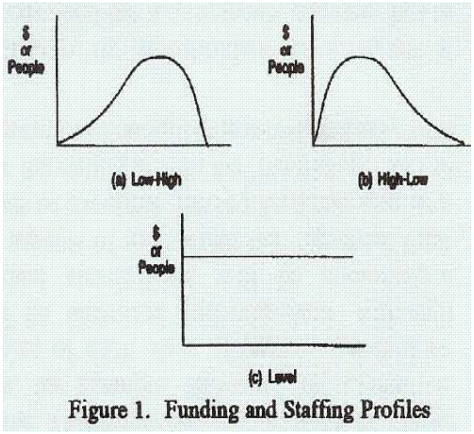

Figure 1. Funding and Staffing Profiles

Disponibilidad de personal (c_{17}): este criterio calibra la suficiencia de personal disponible para el proyecto. Los valores son V_{17}= (*escaso,suficiente,amplio*).

Accesibilidad de los usuarios (c_{18}): este criterio mide la cantidad de accesos que los desarrolladores tienen para los ususarios. Los valores son V_{18}= (sin acceso, acceso limitado, acceso libre).

2.5 Criterio de organización.

El criterio de organización relaciona como las normas de organización impactan un desarrollo. El criterio de organización se trata a continuación.

Compatibilidad de la direccion (c_{19}): este criterio mide el grado en el cual un modelo de proceso de software es compatible con todos los requisitos de desarrollo de la organizacion. Los valores son V_{19}= (*imposición extricta, flexible, solo directrices*).

Garantía de calidad y Capacidad de la configuración de dirección (c_{20}): Este criterio valora la compatibilidad entre un modelo de proceso particular y la garantía de calidad de la organización, asi como los procedimientos de configuración de dirección. Los valores son V_{20} =(*básico, intermedio, avanzado*).

3 Una jerarquía de Modelos de Proceso.

La variedad de modelos de proceso en uso hoy día es asombroso, con el nivel de abstraccion al cual cada uno trata el proceso de software como una variante significativa.

El nivel más alto de la jerarquía contiene modelos de proceso que trazan amplios grupos de actividades. El nivel siguiente contiene modelos de proceso que mas adelante refinan el agrupamiento y las relaciones entre las actividades prescritas. Al siguiente nivel están los modelos de proceso que especifican herramientas particulares y metodos de llevar a cabo las actividades. La figura (2) muestra un ejemplo de esta jerarquía . En esta figura el nivel alto (nivel 3) muestra un modelo de proceso particular:

determina el problema , desarrollo, operaciones y mantenimiento, asi como actuación secuencial. El nivel 2 muestra el proceso particular para llevar a cabo el desarrollo. El nivel 1 muestra un conjunto de herramientas particular y técnicas para llevar a cabo las actividades requeridas en el nivel 2. La siguiente sección describe algunos modelos de proceso al nivel 3 y 2 . En el nivel 1, los métodos y herramientas están fuera del alcance de este artículo. La selección de los metodos y herramientas para los requisitos el nivel 1 han sido tratados en (9).

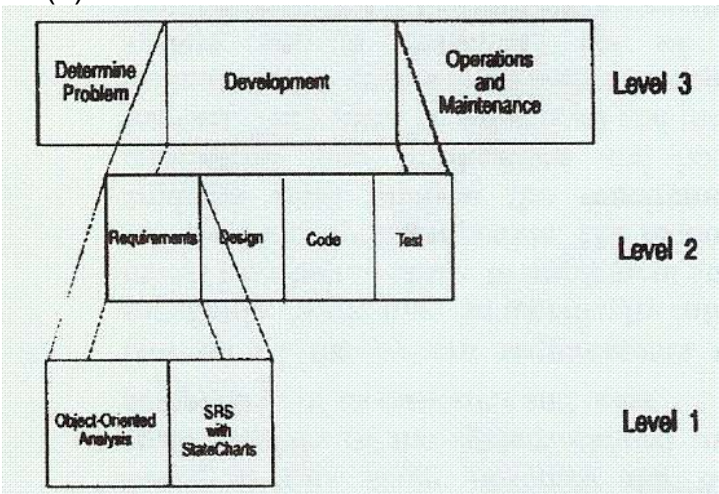

Figure 2. The Hierarchy of Software Process Models

3.1 Modelo de proceso nivel 3.

El modelo de proceso del nivel 3 adjunta etiquetas a segmentos de tiempo para caracterizar actividades en curso. Estos modelos también dirigen cualquier partición de las necesidades de los usuarios. Este trabajo considera tres ejemplos : *convencional, incremental y evolutivo* .
Bajo el modelo de proceso de software convencional el problema no está subdividido. Un producto se desarrolla para dirigir el programa entero . El ciclo de vida del software está dividido en tres segmentos: determinar el problema que hay que resolver, desarrollar el producto, y utilizarlo y mantenerlo. Ver figura 3.

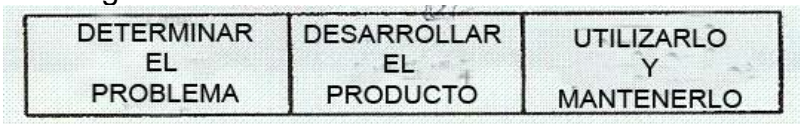

Figure 3. The Conventional Software Process Model

Bajo el modelo de proceso de software incremental hay una decisión consciente para retrasar el desarrollo de una parte del objetivo del producto de software. El problema entero es subdividido en grupos antes de que cualquier producto provisional sea desarrollado. El producto de cada desarrollo provisional será usado en el entorno operativo, y se acercará lo

máximo posible a no tener ningún mantenimiento. La figura 4 muestra el modelo de proceso de software incremental.

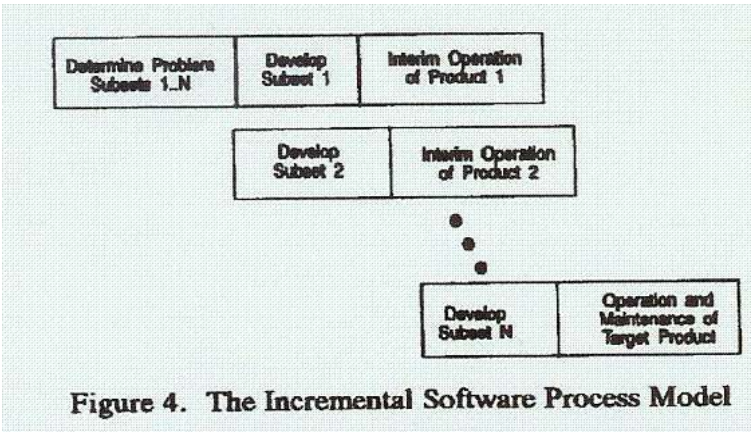

Figure 4. The Incremental Software Process Model

El modelo de proceso de software evolutivo es similar al incremental en que hay una decisión consciente en retrasar el desarrollo de una porción del producto de software final. A diferencia del desarrollo incremental, sin embargo, los productos provisionales se desarrollan antes de que la siguiente subdivisión se determine. Ver figura 5.

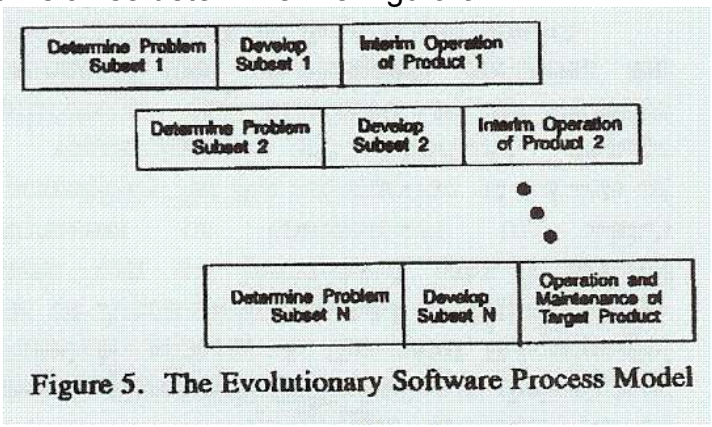

Figure 5. The Evolutionary Software Process Model

3.2 Modelo de proceso nivel 2.

El modelo de proceso del nivel 2 especifica una serie de actividades a realizar en cada segmento de tiempo de un modelo de proceso tipo nivel 3. Estos modelos también dirigen las diferentes representaciones del producto que será construido. Este trabajo considera 3 ejemplos: *cascada, prototipo híbrido, y especificación operativa*.

El *modelo en cascada* fue introducido por primera vez por Royce, en 1970. La Figura 6 muestra un típico modelo de proceso de software en cascada.

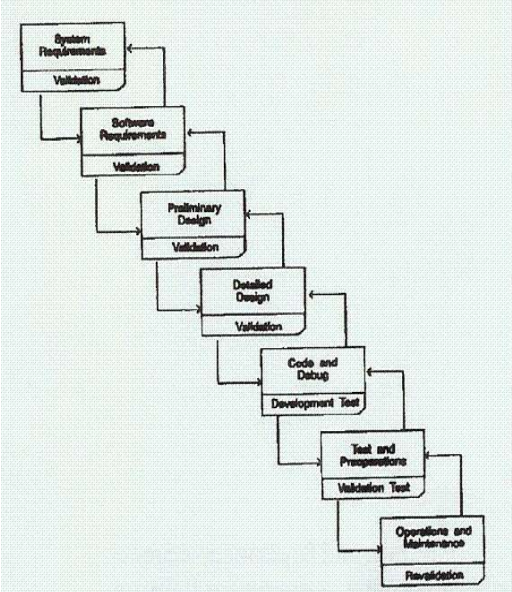

Figure 6. The Waterfall Software Process Model
© 1976 IEEE

En el *prototipo híbrido*, los prototipos se construyen en el inicio. En etapas posteriores, partes de dichos prototipos se descartan selectivamente, mientras que otras partes (por ejemplo, objetos, o codigo reutilizable) se incorporan al producto (Luqi emplea el término *prototipo rápido,* pero esto también se puede referir a la construcción rápida de prototipos desechables). No debe confundirse con prototipos desechables, se que emplean a menudo como método en el proceso de modelo del nivel 1, para realizar algunas de las actividades prescritas. La figura 7 muestra el proceso de modelo de software nivel 2.

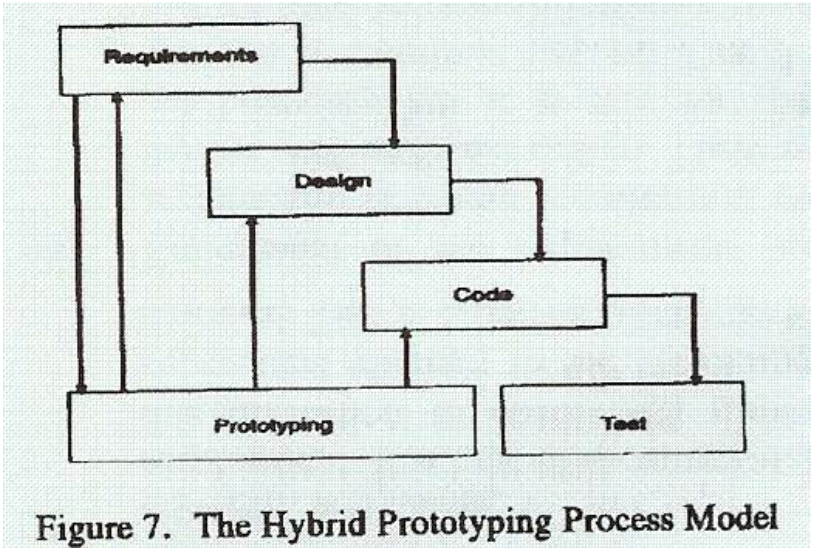

Figure 7. The Hybrid Prototyping Process Model

La *especificación operativa* ofrece una representación ejecutable del producto, directamente desde sus especificaciones. Ver la figura 8. La primera fase describe el comportamiento externo de un sistema de

soluciones. Las fases posteriores lo transforman en un sistema operativo más consistente y fuerte.

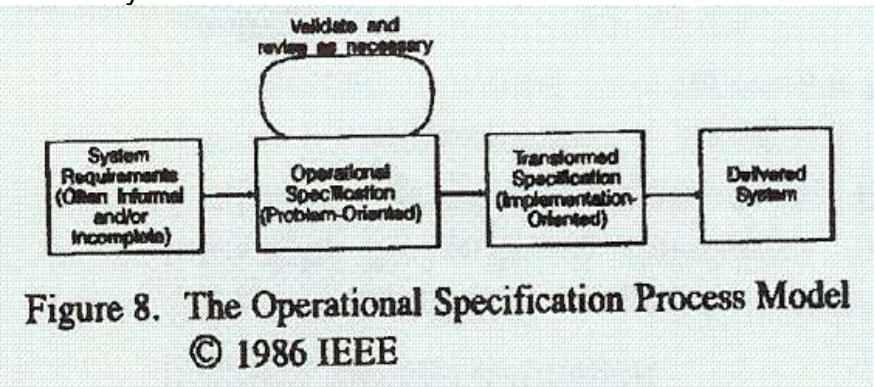

Figure 8. The Operational Specification Process Model

3.3 Resumen.

Los modelos de proceso que se han presentado anteriormente no son los únicos existentes. Por ejemplo, los prototipos operativos y los modelos de software simultáneos ofrecen modelos de nivel 3 alternativos. El modelo de proceso transformador es de hecho un modelo tipo nivel 2 alternativo.

4 Tablas de Selección del Modelo de Proceso.

Este apartado contiene unas tablas que indican la conveniencia de un modelo de proceso para los proyectos que lucen determinados valores particulares v_{ij} para los criterios c_i. Éstos se han desarrollado atendiendo a las características de los modelos de proceso, y a los rasgos que se necesitan en situaciones de proyecto particulares. Las características claves que se consideraron son:

1. La oportunidad para modificar, validar y/o verificar el software.
2. El grado de funcionalidad entregado y cuándo se hizo.
3. El nivel de actividad con respecto al tiempo.

Un valor de 1 en una tabla indica que el modelo de proceso es apropiado para los objetos que exhiben ese valor en un criterio. Un valor de 0 indica que el modelo de proceso es inadecuado. Así, en la Tabla (?) muestra que el modelo de proceso convencional es apropiado para proyectos con usuarios expertos, usuarios expresivos, aplicaciones establecidad, etc. La Tabla (?) a través de (?) muestra la misma información para los modelos de proceso restantes.

Table 2. Conventional Selection Matrix

Criteria	v_{i1}	v_{i2}	v_{i3}
c_1	0	0	1
c_2	0	0	1
c_3	0	0	1
c_4	1	1	1
c_5	0	0	1
c_6	1	0	0
c_7	1	0	0
c_8	1	0	0
c_9	1	0	0
c_{10}	1	0	0
c_{11}	1	0	0
c_{12}	1	1	1
c_{13}	1	0	0
c_{14}	1	0	0
c_{15}	0	0	1
c_{16}	1	0	0
c_{17}	0	0	1
c_{18}	0	1	1
c_{19}	1	1	1
c_{20}	1	1	1

Table 3. Incremental Selection Matrix

Criteria	v_{i1}	v_{i2}	v_{i3}
c_1	0	1	1
c_2	0	1	1
c_3	0	1	1
c_4	0	1	1
c_5	0	1	1
c_6	1	1	0
c_7	0	1	0
c_8	1	1	0
c_9	1	1	0
c_{10}	1	1	1
c_{11}	1	1	1
c_{12}	1	1	0
c_{13}	1	1	0
c_{14}	0	1	0
c_{15}	0	1	1
c_{16}	0	1	0
c_{17}	0	1	1
c_{18}	0	1	1
c_{19}	1	1	0
c_{20}	0	1	1

Table 4. Evolutionary Selection Matrix

Criteria	v_{i1}	v_{i2}	v_{i3}
c_1	1	1	1
c_2	1	1	1
c_3	1	1	1
c_4	0	1	1
c_5	1	1	1
c_6	1	1	1
c_7	0	1	1
c_8	1	1	1
c_9	1	1	1
c_{10}	1	1	1
c_{11}	1	1	0
c_{12}	1	0	0
c_{13}	1	1	1
c_{14}	0	1	0
c_{15}	1	1	1
c_{16}	0	1	0
c_{17}	1	1	1
c_{18}	0	0	1
c_{19}	1	1	0
c_{20}	0	1	1

5 Ejemplo.

Este apartado contiene un ejemplo de como se usan las matrices de modelo para procesos. El ejemplo ha sido llevado a cabo usando un proyecto real, la Terminal de Vigilancia del Prototipo Oceánico (TVPO), cuyas características han sido definidas por un miembro del equipo de desarrollo.

Paso 1: Examinar el proyecto y determinar las s_{ij} para cada criterio c_i. La finalidad de este proyecto era situar estaciones para puestos de oficina en barcos, y asi poder analizar informaciónes de inteligencia estratética. La aplicación era algo nuevo tanto para usuarios como para desarrolladores, y suponía un problema complejo y cambiante a gran velocidad. Había asimismo una necesidad de entrega urgente de un producto sin ninguna capacidad. La parte de usuario representaba una parte crítica de los requerimientos. La Tabla 8 engloba las características de este proyecto, especificando sus s_{ij}.

Paso 2: Calcular el índice de cada modelo de proceso usando esta fórmula:

La tabla 9 muestra los resultados de llevar a cabo este paso para un modelo de proceso convencional. Cada entrada en la tabla es el resultado de la multiplicación (s_{ij} * a_{ij}). Los números en la columna de la derecha son el resultado de la sumatoria en *j*. El total al final de la columna de la derecha es el resultado de la sumatoria en *i*. Para este proyecto en concreto, el modelo de proceso convencional dio un índice de 8. Los resultados de todos estos modelos de proceso se muestran en la tabla 10.

Table 10. Process Model Ratings for Example Project

Process Model	Rating
Level 3	
Conventional	8
Incremental	8
Evolutionary	16
Level 2	
Waterfall	6
Hybrid Prototyping	10
Operational Specification	3

Paso 3: Seleccionar el modelo de proceso con el índice más alto. Para los modelos de proceso de nivel 3, los índices indican que el modelo de proceso evolutivo sería la opción más apropiada para este proyecto. Para los modelos de proceso de nivel 2, los modelos de proceso de prototipo híbrido son la mejor elección a este nivel de abstracción.

6 Conclusiones.

Los criterios de selección determinados en este trabajo no son ni mucho menos definitivos. Su definición viene tan solo de una búsqueda en literatura sobre el tema y de observaciones informales de muchos esfuerzos de desarrollo de software. Sólo se podrán hacer correlaciones válidas entre características de proyecto y modelos de proceso alternativos, después de ejecutar análisis correlativos, emplear varios modelos de proceso, y después de múltiples aciertos y fallos en un gran número de proyectos.

Se requiere un estudio más exhaustivo del significado actual del resultado de un modelo de proceso. Debe haber algún beneficio en calcular resultados *negativos* que indican el *castigo* por ignorar ciertos criterios o valores. Esto puede ser útil para romper situaciones sin salida o para evaluar el coste de no usar un modelo de proceso recomendado. Los criterios con una influencia *negativa* significativa también pueden motivar planes que eludan problemas, cuando el modelo de proceso recomendado no se emplea; o bien pueden adaptar un modelo de proceso que se ajusta a todos los cálculos.