Modelo Concurrente de Ciclo de Vida - 1205

Alejandro Estayno
estayno@fibertel.com.ar
GIS - Universidad Nacional de la Matanza
Argentina

Abstract

The currents life cycle models tend to exhibit, certain alignement in its cojoint dynamics. The goal of this work is -linking different areas of knowledge - to show that it is highly improbable that the *continuum* could correspond with the real world. The *Modelo Concurrente de Ciclo de Vida* (MCCV-1205) is an abstraction that shows the lack of determinism and the concurrence of those activities required by the deliverables indicating the end of each phase of the software life cycle.

Keywords: life cycle, models, concurrence, determinism, semaphore, deliverables, software engineering.

Resumen

Los modelos de ciclo de vida actuales tienden a presentar, normalmente, cierta linealidad en su dinámica asociada. El presente trabajo tiene por objetivo, a través de interrelacionar diversas áreas del conocimiento, mostrar que el *continuum*, casi siempre, no se corresponde con el mundo real. El Modelo Concurrente de Ciclo de Vida (MCCV-1205) es una abstracción que muestra el indeterminismo y la concurrencia del derrotero de las actividades para concretar los entregables que señalan la finalización de cada fase de un ciclo de vida de software.

Palabras clave: ciclo de vida, modelos, concurrencia, determinismo, semáforos, entregables, ingeniería del software.

EL *CONTINUUM* ARTIFICIAL Y LOS CICLOS DE VIDA

Si consideramos la dinámica de un ciclo de vida (C de V), podemos claramente distinguir una evolución en el continuo en cada modelo existente desde los primeros intentos en la década del ´70 para obtener una abstracción que modelice este concepto.

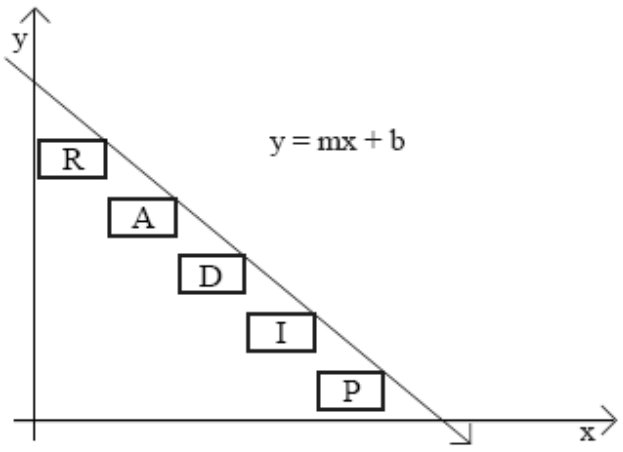
La exposición de estos modelos tiene como factor común la siguiente característica distintiva: el expositor, como interfase entre una representación del modelo y el auditorio, recorre con el puntero, en un sólo gesto, un derrotero continuo entre las distintas fases del ciclo de vida del proceso de construcción de un sistema software. Este derrotero puede revelarse en un segmento de recta, en el modelo de cascada [ROY 70], en una espiral [BOE 85] [BOE 97], en una "V" [MDRFA 92], pero siempre es continuo, siempre es secuencial... Siempre el dedo recorre el modelo sin saltos *cuánticos* o impredecibles.

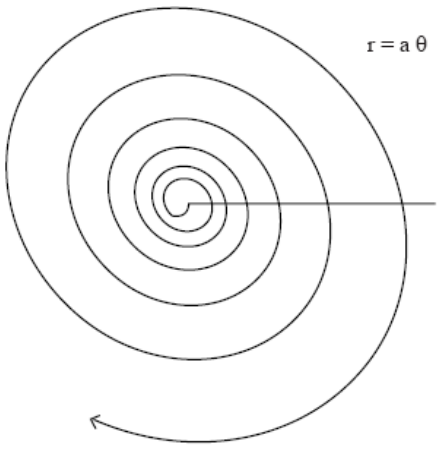
Las ciencias naturales describen fenómenos expresándolos en términos comprensibles, manejando abstracciones en modelos representativos. La modelización, connatural al hombre, se ha extraviado, en el terreno de las abstracciones de los C de V, de aquella ductilidad básica que la distingue cada vez que se exploran otras regiones del conocimiento. Jesús ejerció la defensa de la mujer apedreada modelizando los pecados de cada agresor con el ícono de sus faltas representadas en el suelo, Da Vinci modeló su paracaídas, extrapolando en su esquema el comportamiento de un fenómeno natural pero en control del hombre, Tony Hoare proclamó sus ideas predicando ... *la herramienta más poderosa que posee el hombre para resolver problemas complejos es la abstracción* [DAH DIJ HOA 72].

Los modelos son parte de la vida y dominio del hombre en las capas superiores de este planeta, así han evolucionado tanto como el control o conciencia sobre el fenómeno o proceso que representan.

EL RECORRIDO INCIERTO, CONNATURAL AL HOMBRE

El proceso mental en la resolución de problemas rara vez se presenta en una evolución continua descriptible por una representación genérica de una función del tipo $\boldsymbol{y = mx + b}$, en coordenadas cartesianas para el modelo de cascada o, en sus correspondientes coordenadas polares, $\boldsymbol{r = a}\ q$, para el modelo de espiral.





Un modelo del ciclo de vida de software indica el recorrido por las distintas etapas de la construcción de un software, tomando desde el primer choque con los requisitos, razón de su existencia, hasta la implantación definitiva de la solución que aporta.

Para que los niveles de interpretación de un problema y representación de su solución, por un autómata, tengan las potencialidades necesarias para intentar minimizar la incertidumbre de fracaso, la metáfora del problema deberá ser construida, por medio de ciertas reglas de formación, en correspondencia uno a uno entre el problema -mundo real- y el modelo -mundo de las representaciones-.

Cada vez que se expone un modelo de ciclo de vida de software –cascada, espiral- la característica, mencionada anteriormente, de señalar de principio a fin sin levantar el puntero de su representación gráfica, connota que una idea de continuidad está fuertemente arraigada en las publicaciones académicas referentes al tema.

Estas figuras son quizás tan míticas como el pentáculo inexorable del rito de la invocación al *Demonio de Laplace*:

En el siglo XVIII, la aparición de la Ley de Gravitación Universal, contagió de confianza a la población científica del globo. La naturaleza parecía poder ser abstraída a modelos generales, fundamentales e inmutables. Algunos cientos de fórmulas matemáticas revelaban todo.

Pierre Simon de Laplace, un genio de amplio espectro, hizo un aporte revelador del estado de ánimo de su época [LAP 1814], afirmando que:

Una inteligencia que en un momento dado conociera el estado y todas las fuerzas que animan a la totalidad de las partículas del Universo tendría frente a sus ojos el pasado, el presente y el futuro. Había nacido el demonio de Laplace[1].

Laplace, sin acuñar el término *determinismo* presentaba una de sus posibles definiciones, poética y sintéticamente, expresando: *el estado presente del Universo como el efecto de su estado anterior, y como causa de su estado futuro.* Un poco más formalmente: dado un estado de un sistema para un instante t_i dado, para cada t_{i-1} o t_{i+1} existe un y sólo un estado posible.

Un poco más de cien años después Niels Bohr diría: *la predicción es difícil, especialmente sobre el futuro.*

Por su parte, Ilya Prigogine, extraordinario investigador del caos y el tiempo, [PRI 93] haría un repaso histórico observando: *La ciencia moderna se basa en la noción de leyes de la naturaleza. Estamos tan acostumbrados a ella, que ha llegado a ser como una perogrullada, y sin embargo posee implicaciones muy profundas... Siempre he pensado que en esta eliminación* el tiempo *tuvo una influencia importante el elemento teológico. Para Dios todo está dado. La novedad, la elección o la acción es-*

[1] *El mismo Laplace tuvo consideraciones al respecto cuando creó a su demonio, no dejó de cuestionar la imposibilidad de aquella mente superior para albergar en sus actividades deterministas a la misma realidad humana.* Gastón E. Giribet. *De símbolos y demonios.* BArtes II. Buenos Aires. 1996.

pontánea dependen de nuestro punto de vista humano. En los ojos de Dios el presente contiene el futuro y el pasado...

QUANTUM Y EXPERIENCIA

Cuanto más claramente dominamos el análisis intelectual de un modo de regular procedimientos en favor de nuestros intereses, tanto más decididamente rechazamos la inclusión de una evidencia que se niegue a armonizarse inmediatamente con el método ante nosotros. Algunos de los mayores desastres de la humanidad han sido producidos por la estrechez de miras de hombres poseedores de una buena metodología [WHI 29]

Según Prigogine, anteriormente se buscaban simetrías en el universo [PRI 93], el caso más fácilmente distinguible es el de la mecánica cuántica relativa a las partículas y antipartículas.

Como las partículas que se estudian son parte de un mundo artificial creado especialmente para la experimentación, se concluye que su analogía no es necesaria en nuestro medio cosmológico [PRI 85], *dominado* por simetrías, analogías *plenas* de coherencia y una representación diáfana.

Pero la realidad indica, cada vez más irreversiblemente que nunca, que el universo no es tan simétrico y cada vez más extraño a la armonía geométrica ideal de la física clásica.

El tiempo tal como se expresa hoy se presenta con particularidades más propias de su situación en su mismo y clásico eje, irreversible desde los átomos hasta las más alambicadas especulaciones de la filosofía natural. La era en que la mecánica clásica era la vedette, atravesando irreverentemente toda disciplina, con sus criterios filosóficos del *nivel fundamental*, ha muerto.

Hoy las ciencias y sus disciplinas se interpenetran una tanto más promiscuamente pero, paradójicamente, un tanto más reversiblemente. Ninguno de los nuevos niveles interconectados puede ser considerado fundamental.

QUANTUM Y CONOCIMIENTO

Ninguna entidad física, en términos de la física cuántica, se encuentra en un estado definitivo. A lo sumo, existe un conocimiento relativo de las probabilidades de que se encuentre en cierto estado.

Una ecuación de física cuántica puede expresar la *probabilidad*, en alguna situación particular, de que un electrón tenga un spin (medida del momentum angular intrínseco) positivo o negativo, por ejemplo: 40% de posibilidades de ser negativo y 60% de ser positivo.

Heisenberg, afirmó que es imposible reducir una situación como la descripta a un estado bien definido[2]. En ciertas situaciones un físico puede conocer una probabilidad cercana a uno, pero nunca es

[2] Principio de Incertidumbre de Heisenberg.

uno. Nunca existe el 100% de certeza. Nunca se puede definir un estado. El conocimiento certero no existe en términos de la cuántica.

Si se conoce con menor margen de incertidumbre el spin de una partícula atómica, se conocerá con mayor nivel de incertidumbre su posición y viceversa. Para predecir qué hará la partícula en el futuro se necesitan conocer su posición y su momentum, y esto no es posible de manera inequívoca, sólo de modo probabilístico, por consiguiente... *Dios (no) juega a los dados*[3]

SCHROEDINGER Y LA CAJA INFAME

De la intrincada y original mente de Erwin Schrödinger surgió una particular experiencia[4]: se mete un gato dentro de una caja insonora en cuyo interior se coloca una probeta sellada de vidrio que contiene un venero volátil y letal. Además, dentro de la caja se encuentra un artefacto con un martillo capaz de romper la probeta. El martillo se comanda eléctricamente y está conectado con un detector de partículas alfa. Por último se coloca, también dentro de la caja, un átomo que tiene una probabilidad de 0.5 de desintegrarse en una hora. De acuerdo a ecuaciones de la física cuántica, todo lo que se conoce sobre el átomo es que tiene la mitad de las chances de desintegrarse en una hora y no hay posibilidades de tener información definitiva.

Ahora bien, si el átomo se desintegra, el detector accionará el martillo, se romperá la probeta y el gato indefectiblemente morirá. De otro modo no hay radiación, el detector permanecerá suspendido y el gato sobrevivirá a esta instancia.

Evidentemente, este esquema resulta estrafalario aunque no inverosímil. La cuestión urgente es ¿cuál es el estado del gato después de una hora? De acuerdo con las bases de la teoría cuántica, el gato no está definitivamente vivo ni definitivamente muerto. Está algo así como más o menos muerto o más o menos vivo.... No se puede determinar si el átomo -en el contexto de una caja insonora y opaca-, está *definitivamente* desintegrado, dado que no es posible calcular su estado con un *grado no nulo de incertidumbre*.

Antes de un obvio final, que derriba con las repetidas especulaciones probabilísticas, veamos dos autores y un concepto.

El postulado de proyección [NEU 32] de John Von Neumann, en su tratado clásico de teoría cuántica, enuncia que cuando una entidad es *medida sin incertidumbre* se reduce a un estado bien definido.

Von Neumann considera la medición completa cuando se *registra en la conciencia* del observador.

---

[3] "*La mecánica cuántica es muy impresionante. Pero una voz interna me dice que esto no es todavía lo auténtico. La teoría da mucho, pero difícilmente nos acerca al secreto del Viejo. De todas maneras estoy convencido que Él no juega a los dados*". Carta de Einstein a Bohr escrita en diciembre de 1926.

[4] E. Schröedinger. Experiencia imaginaria presentada en la Universidad de Berlín. 1937

Por su lado, Eugene Paul Wigner [WIG 60] coincide y pone en evidencia este concepto en sus escritos al considerarlo como la clave en la definición de una medida, expresando textualmente:

El observador. . . tiene a su disposición una característica y muy familiar facultad, a la que podemos llamar la "facultad de introspección", él puede así tomar en cuenta de su propio estado de manera inmediata. Esto es en virtud de su "inmanente conocimiento" así reclama el derecho a crear para si mismo su propia objetividad, es decir, cortar la cadena de coordinaciones estadísticas . . . certificando: "Estoy en el estado wk' " o más simplemente, veo que G = gk' o directamente "F=fk' "

Ahora sí, claramente, el hecho es que si miramos en la caja al cumplirse la hora, fenomenológicamente tomamos conocimiento de la muerte o la supervivencia del gato. Cumpliéndose el tiempo de observación, se selecciona definitivamente una y sólo una de las dos posibilidades, obteniendo un resultado inequívoco.

RETEJIENDO UNA TRAMA METADISCIPLINARIA DE ANALOGÍAS

Ante los claros estímulos de las experiencias intelectuales anteriores, se evidencia la necesidad de un nuevo planteo, como para garantizar que, a pesar de la evidencia de sus reflejos lentos, la ingeniería del software es joven, está en pleno desarrollo y aún cuenta con pulso.

Los C de V conocidos revisten en su composición características genéticas del demonio de Laplace. Representan exactamente el pasado, el presente y el futuro del ciclo de vida en cada instante, lo que parece ideal pero no real, conveniente pero no lógico.

A los fines prácticos de la ingeniería del software, la medida, aludida en el marco de referencia de los conceptos físicos y filosóficos arriba expuestos, es *conocimiento*. El obtener conocimiento, en términos del proceso de adquisición descrito en el párrafo anterior, es un estado definitivo.

Está claro que no se puede pasar de un estado a otro, ni siquiera ejecutar un proceso por simple y mínimo que fuera dentro del C de V sin *conocimiento*. El proceso de adquisición del conocimiento no es secuencial y menos aún lineal. El planteo básico de este trabajo es el siguiente: cada porción del conocimiento necesaria puede ser transmitido de un proceso -subproceso- a otro, para lograr pasar de un estado -intermedio- a otro. El C de V no puede ser secuencial, necesariamente debe ser concurrente y no hay posibilidades de determinación para las coordenadas temporales de los pasajes de conocimiento.

Por ejemplo, si alguna de las subetapas de la etapa de análisis comienza, es porque se tiene el conocimiento, concreto, necesario para darle inicio. El conocimiento es portado, en términos de *mejores prácticas,* por el entregable, input del proceso asociado a la subetapa. Este entregable es la caja de Schröedinger y el conocimiento es el que el analista obtiene al abrirla concluyendo fenomenológicamente en la definición del estado que refleja la vida o la muerte del pobre gato, creando objetividad por el derecho otorgado por su conocimiento inmanente. Es recién en esta instancia cuando están dadas las condiciones para comenzar a ejecutar el proceso, por medio del cual se cambia de estado, progresando en la dinámica del C de V.

CÓMO ES EL PROCESO

A esta altura del planteo propuesto, pareciera transparente exponer un proceso de construcción de la solución -distancia que separa al estado *actual* del *deseado*- en términos de los recorridos hacia hitos intermedios -sin conocer estrictamente su orden- que anticipan la solución definitiva del problema. Esta dinámica no luce naturalmente como un proceso de derrotero continuo, de manera contrapuesta a la concepción, hasta ahora clásica, de cómo debe ser un ciclo de vida de un proyecto de software.

Los ciclos de vida secuenciales, observando la evolución epistemológica de la ingeniería del software, son a la mecánica clásica como este modelo es a la mecánica cuántica. El modelo 1205 no revela una solución sino un defecto y propone una alternativa.

Comencemos haciendo una analogía con el proceso mental disparado ante un problema emergente a solucionar. Al instante de conocer el problema que se resolverá con la implementación de una solución concretada en un sistema software, se establecen una confluencia caótica de escenarios y actos mentales. Una alternativa para alguien con cierta experiencia en el desarrollo de aplicaciones software podría ser:

- primera versión psíquica de la interfase software
- conjunto mínimo de funcionalidades básicas
- ciertas funcionalidades esenciales
- algunos problemas de las funcionalidades
- algunas soluciones a un subconjunto de los problemas anteriores
- algún integrante del team a quien consultar problemas a resolver
- analogías con sistemas conocidos
- posibles reutilizaciones de soluciones implementadas
- primera visión de la estrategia tecnológica
- universo de usuarios
- imagen de un layout posible del team project
- alguna circunstancia en el escenario de desarrollo de la solución
- estimación bruta del tiempo
- ...

Aunque en términos clásicos se esté en medio de un estadio típico de captura de requisitos, natural y concurrentemente se han establecido conexiones y se ha logrado un avance superior a una distancia nula, en los espacios de la solución del problema, en el devenir de las diversas fases del ciclo de vida del proyecto. En estos caminos alternativos, se han disparado procesos que anticipan ciertas fases del ciclo de vida de desarrollo de un proyecto software como análisis, prototipación, diseño, test y codificación. Aún más, este proceso se hace constante y tiene como único límite la finalización del C de V.

Dejando a un lado, por el momento, la concurrencia de avances sobre las fases del C de V, surge la cuestión sobre cuáles son las condiciones para que cada uno de ellos avance o se detenga.

Por lo tanto, antes de continuar se presentará un overview sobre el mecanismo de *semáforos*, herramienta que utilizaremos para facilitar la manifestación de esta idea.

LOS SEMÁFOROS – REFRESCANDO UN VIEJO CONCEPTO

Los *semáforos*, concebidos por Edsger Wybe Dijkstra [DIJ 65] [DIJ 69], constituyen una herramienta de sincronización que restringe o permite el acceso a recursos, por ejemplo, y, en nuestro caso, al conocimiento. Presentaremos las definiciones y la versión implementada en Operating System Concepts [SIL GAL 98].

Un semáforo S es una variable entera a la que, una vez que se le ha asignado un valor inicial, sólo puede accederse a través de dos operaciones atómicas estándar: espera (wait) y señal (signal). Estas operaciones se llamaban originalmente P (para espera; del holandés *proberen*, probar) y V (para señal; del holandés *verhogen*, incrementar).

El valor de una variable del tipo semáforo es el número de unidades del recurso (en nuestro caso conocimiento/s) que está/n disponible/s. La operación P detiene la ejecución hasta que hay un recurso disponible, en cuyo caso lo reclama inmediatamente y sigue su ejecución normal. V es la operación inversa; hace disponible un recurso común, y permite que uno de los procesos suspendidos por la falta de ese recurso, prosiga con su ejecución. La operación *Inicia* se utiliza para inicializar el semáforo antes de que se hagan peticiones sobre él.

Las definiciones clásicas de espera y señal son:

```
Espera(S): while S = 0 do nada;
           S:= S - 1;

Señal(S): S:= S + 1;
```

Las modificaciones del valor entero de los semáforos en las operaciones Espera y Señal se deben ejecutar de forma indivisible, es decir, mientras un proceso modifica el valor del semáforo, ningún otro proceso puede modificar simultáneamente el valor de ese mismo semáforo. En el caso de Espera(S) la prueba del valor entero de S (S = 0) y su posible modificación (S := S - 1), también deben ejecutarse sin interrupción.

LOS SEMÁFOROS EN EL MODELO

Cada fase de un ciclo de vida se comporta como un dispositivo receptor y transmisor de conocimiento sobre *cómo* debe/n seguir otra/s fase/s.

Por ejemplo:

Es sabido que, bajo las concepciones de un método secuencial, la tecnología a aplicar en el desarrollo de una solución software es una decisión de diseño. Pero si en la etapa de requisitos, precisamente en la primera entrevista -más allá de las técnicas empleadas para su captura- se obtiene información básica sobre la tecnología del sistema de gestión de base de datos implantada en las instalaciones sobre las que se va a realizar el deployment del sistema a desarrollar, estamos en presencia de una posible concurrencia en la evolución de las fases.

Este fenómeno, en términos de modelización, puede ser asociado a un arreglo unidimensional de semáforos KT (Knowledge Transferor), compuesto de semáforos KT_f donde $f \in N$ en el rango [1..n], siendo n la cantidad de fases distinguibles en el C de V y f un identificador de una de sus fases.

El valor de cada semáforo es 0 (cero) si la fase no tiene los *conocimientos* suficientes para continuar y su valor varía de 1 a n dependiendo de la *carga de conocimiento* transferida por la fase *transmisora*, permitiéndole avanzar en la solución. En principio el vector está inicializado a 0 -al tratarse de un problema desconocido, no hay conocimiento básico para transferir-.

Una abstracción posible sobre este modelo puede expresarse en términos del siguiente *pseudocódigo*:

/** Declaración e iniciación de un arreglo global de semáforos de transferencia de conocimientos a fases del ciclo de vida **/

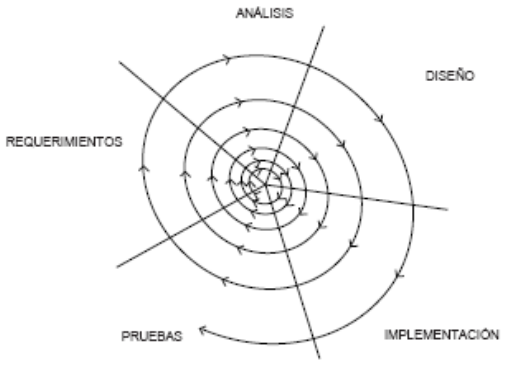
var

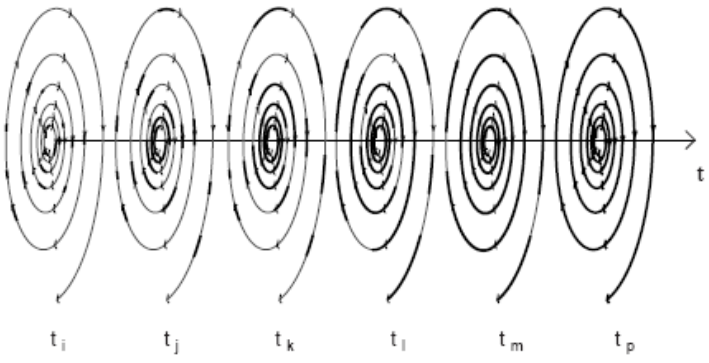
KT: array [1.. n] of *semaphores*

Requisitos := 1;
Análisis := 2;
.....
Pruebas := n;

```
begin
        for i := 1 to n do
                inicia(KT [i], 0)
/** Cómo cada fase es sincronizada dentro de la dinámica del C de V **/
Requisitos : : Fase de C de V
begin
while .t. begin
        P(KT [Requisitos]);
                                /** si tiene conocimientos, o sea, si KT [Requisitos] > 0,
                                avanza, si no espera por conocimiento transferido desde alguna
                                otra fase **/
        ... evolución de la fase ...

        case conocimientos para transferir a Análisis
                V (KT [Análisis]);
                                /** si tiene conocimientos para transferir a Análisis, ejecu-
                                ta un V sobre el semáforo de avance de Análisis habilitando a la
                                fase a continuar avanzando en su evolución **/

        case conocimientos para transferir a Diseño
                V (KT [Diseño]);
                                /** Ídem anterior, pero con respecto a Diseño **/
        .....
        case conocimientos para transferir a Pruebas
                V (KT [Pruebas]);

                                /** Ídem anterior, pero con respecto a Pruebas **/
end.
```

Si suponemos un conjunto ***extremadamente clásico de fases***: Requisitos, Análisis, Diseño, Implementación y Pruebas, la representación gráfica correspondiente al MCCV-1205 será:





donde $i < j < k < l < m < p$, y t es tiempo

Dado que el modelo es concurrente y las actividades de cada fase son disparadas de manera no determinística, no se puede predecir cómo se ordenarán las fases del C de V, lo que revela un nuevo planteo.

- Toda fase existe desde el instante $T_i = 0$;
- Se instancian de manera no determinística, de acuerdo con una función de correspondencia estocástica, con una distribución de probabilidades[5], generando un posible derrotero del C de V;
- Sus finalizaciones se desencadenan ordenadamente, de acuerdo con el modelo clásico. A pesar de que el momento de nacimiento de cada fase tiene un grado no determinado de incertidumbre, el orden de finalización está claramente determinado. La probabilidad P de que la fase de requisitos finalice antes de la fase de implementación es 1, de otro modo, la fase de requisitos incluiría actividades que no tendrían impacto en la implementación, de lo que se desprendería un cuestionamiento sobre el sentido de estas actividades.

CONCLUSIÓN

En toda disciplina científica, especialmente en las aplicadas -tal es el caso de la ingeniería del software-, el concepto de modelo se define como una idealización de la realidad, una abstracción que facilita, por su intrínseca simplificación, un planteo claro del problema.

Los modelos de ciclo de vida del software utilizados convencionalmente no explicitan la indeterminación del momento de inicio de las actividades asociadas a cada fase o subfase. *"Muestran" una secuencialidad arbitraria* que no contribuye a evidenciar la presencia de una clara incertidumbre acerca de *cuál* actividad se inicia y *cuándo*. Más aún, la falta de su tratamiento explícito genera un problema cuya identificación aparece ambigua, implícita o ausente en las metodologías aplicadas actualmente para el desarrollo de software.

El modelo MCCV-1205 facilita la interpretación de la dinámica asociada al ciclo de vida del software, modelando sus fases y la indeterminación de los instantes donde se produce el disparo de los procesos asociados a la dinámica de cada fase. El aporte del modelo es fundamentalmente *desnaturalizar*[6] estos constructos conceptuales y proponer la utilización de mecanismos abstractos de sincronización, como los semáforos, para ampliar el marco conceptual que subtiende a los procesos de planificación y ejecución en la administración de proyectos de software.

FUTUROS TRABAJOS

- Propuesta sobre la organización de las comunicaciones internas en un equipo de proyecto de desarrollo de software.
- Herramienta de administración que considere la concurrencia natural en el proceso de desarrollo de software, como una alternativa de fast-tracking, para la disminución de costos en recursos y disminución del la longitud del schedule.
- Metodología de project management, para desarrollo de software, a partir de un modelo de C de V concurrente.

[5] Esta función será objeto de estudios posteriores.

[6] Desnaturalizar, en el presente trabajo, se incluye como sinónimo de poner en términos obvios todo aquel concepto que subyace en el conocimiento pero no está explícito y por lo tanto no se controla en su totalidad.

REFERENCIAS

[**BOE 85**] B. W. Boehm . *A spiral model of software development and enhancement*. ISPW. Trabuco Canyon. 1985

[**BOE 85**] B. W. Boehm, A. Egyed, J. Kwan, R. J. Madachy:. *Developing Multimedia Applications with the WinWin Spiral Model*. ESEC/ SIGSOFT FSE. 1997

[**DAH DIJ HOA 72**] O. J. Dahl, E. W. Dijkstra, C. A. R. Hoare. *Structured Programming*. Londres. 1972

[**DIJ 65**] E. W. Dijkstra, *Solution of a problem in concurrent programming control*. Communications of ACM. NuevaYork. 1965

[**DIJ 69**] E. W. Dijkstra. *Cooperating sequential processes*. University of Texas. NuevaYork. 1968

[**LAP 1814**] P. S. Marqués de Laplace. *Essai philosophique sur les probabilités*. Paris. 1814

[**MDRFA 92**]. Ministerio de Defensa de la República Federal de Alemania. *The V-Model.* Bonn. 1992

[NEU 32] J. Von Neumann. *Mathematical Foundations of Quantum Mechanics*. Princeton University Press. Princeton, NJ. 1955. La primera publicación en alemán. *Mathematische Grundlagen der Quantenmechank*. Springer. Berlin: 1932]

[PRI 85] I. Prigogine. *Entrevista de Christian Dellacampagne.* Paris. 1985

[PRI 93] I. Prigogine. *Les lois du chaos*. Gius Laterza & Figli Spa. Roma-Bari. 1993

[**ROY 70**] W. W. Royce. WesIEEE. *Managing the Development of Large Software Systems: Concepts and Techniques*. WESCON, IEEE Computer Society Press. Los Alamitos. 1970

[SIL GAL 98] A. Siberschatz y P. B. Galvin. *Operating System Concepts*. Addison Wesley. Massachusetts. 1998

[WHI 29] A. N. Whitehead, *The Function of Reason*, Princeton. 1929

[WIG 60] E. Wigner. *The unreasonable effectiveness of mathematics in the natural sciences.* Communications in Pure and Applied Mathematics. New York. 1960.