Universidad Central de Venezuela
Facultad de Ciencias
Escuela de Computación

Lecturas en Ciencias de la Computación
ISSN 1316-6239

Moviendo Modelos Mentales de Usuario

Nora Montaño

RT 2012-05

Centro de Ingeniería de Software y Sistemas
ISYS-UCV
Caracas, Agosto 2012.

Moviendo Modelos Mentales de Usuario

Un método para estimar el costo de aprendizaje del usuario en procesos de migración de software

Nora Montaño
Centro de Ingeniería de Software y Sistemas (ISYS)
Escuela de Computación, Universidad Central de Venezuela
Caracas-Venezuela
nora.montano@ciens.ucv.ve

***Resumen*—Uno de los riesgos en la migración de software surge cuando el usuario final no logra los mismos niveles de productividad que tenía en el ambiente anterior. Se presenta un método que tienen por objetivo estimar el costo de aprendizaje, basado en una relación de compatibilidad que valora el nivel de aprendizaje que enfrentaría el usuario con el nuevo software. Se proyecta el modelo mental del usuario a través de los modelos conceptuales de los productos de software involucrados. La proyección se establece a partir de un conjunto de tareas representativas homóloga entre los dos productos y las metáforas de interacción. Los resultados la aplicación del método, en el caso de la migración de software, aportan información útil para la toma de decisión, esto conduce a valorar la factibilidad de la migración desde la perspectiva del usuario.**

Palabras claves: modelo mental, modelo conceptual, aprendizaje, metáfora de interacción

***Abstract*—One of the risks in the migration of software arises when the end user does not get the same levels of productivity that had in the previous environment. A method is presented that aim to estimate the cost of learning, based on a relation of compatibility that values learning level would face the user with new software. It projects the user's mental model through conceptual models of software products involved. The projection is established from a set of representative tasks homologous between the two products and interaction metaphors. The results applying the method in the case of migration of software, provide useful information for decision making, this leads to assess the feasibility of migration from the perspective of the user.**

Keywords-component: mental model, conceptual model, learning, interaction metaphor

I. INTRODUCCIÓN

Las migraciones de software son procesos de cambio que pueden completarse satisfactoriamente o no, es por eso que deben ser conducidos de forma planificada para evitar posibles inconvenientes en el proceso, asegurando que el nuevo software sea tan productivo como el anterior.

Por lo general, en un proceso de migración se hace énfasis en los aspectos de la plataforma de hardware y software, buscando el mayor beneficio económico para la organización. Los costos asociados al aprendizaje de la nueva tecnología usualmente inciden en la contratación de cursos de entrenamiento para el personal pero no existe garantía que el usuario asimile el nuevo conocimiento al final de estos, como tampoco que los cursos estén orientados a solventar las deficiencias específicas de conocimiento, que pueden tener los usuario con el nuevo producto de software.

Existe una estrecha relación entre el proceso de migración y el modelo mental del usuario, así como la importancia de la usabilidad en este tema y cómo estos factores pueden ser aprovechados para garantizar el éxito del proceso de migración. El problema de fondo es no poder predecir qué tan fácil o difícil puede ser un proceso de migración de software cuando los usuarios están arraigados o acostumbrados al uso de una herramienta y sólo conocen el funcionamiento de la misma.

El objetivo de esta propuesta es medir el impacto y la viabilidad de un proceso de migración de software desde la perspectiva del usuario. Se presenta un método que permite estimar la asimilación tecnológica por parte del usuario, de forma sistemática y efectiva, permitiendo planificar de manera responsable cualquier proceso de migración.

Para lograr la estimación, se plantea un análisis de compatibilidad de los productos software cuya finalidad es, mover el modelo mental que posee un usuario, de un software a otro, a partir de la proyección de los modelos conceptuales de los productos de software implicados en la migración. Mover modelos mentales significa, determinar que tan reutilizables son las estructuras cognitivas que posee el usuario cuando se produce un cambio en el ambiente.

Este artículo está estructurado en tres partes, en la primera se presenta los diferentes aspectos que motivaron esta investigación, así como los requerimientos de asistencias en forma de escenarios posibles para planificar la migración. La segunda parte, se dedica a explicar de forma detallada cómo se puede estimar el costo de aprendizaje. Por último, se presenta variantes en la aplicación del método que derivan en otras formar de medición.

II. IDENTIFICANDO NECESIDADES EN EL PROCESO DE MIGRACIÓN

A. Motivación de la investigación

La migración de software no es algo nuevo, las organizaciones siempre han tenido que migrar bien sea para mejorar su plataforma por razones técnicas o económicas.

Esta investigación surge a raíz del Decreto N° 3.390 [1] promulgado por el Gobierno de la República Bolivariana de Venezuela, donde se toma la iniciativa para promover el uso y desarrollo de Software Libre y, así garantizar la apropiada Soberanía Tecnológica [2], pero mirando más allá de la política gubernamental, la migración de plataformas propietarias hacia las libres tiene un gran impacto social en la comunidad de usuarios finales.

El problema que se vislumbró en ese momento, era que el uso de software libre no puede ser decretado, tiene que ser planificado en base a la apropiada selección de las herramientas libres, que garanticen en cierto grado la asimilación tecnológica por parte del usuario final, de forma fácil y segura. Para esto se requiere saber con exactitud qué elementos pueden incidir en el RE-Aprendizaje del usuario, puesto que re-aprender significa deshacer estructuras cognitivas ya conocidas para incorporar nuevas formas de actuar, y esto representa una dificultad para el usuario [3].

Una propuesta de solución se tiene cuando en el diseño del software a dónde se quiere migrar, se le incorpora como ancla conceptual [4] las estructuras cognitivas ya conocidas por el usuario, logrando que el usuario identifique que lo nuevo es igual a lo anterior. En esta propuesta se aplica la teoría del aprendizaje significativo propuesta por Ausubel [4] como lineamiento de diseño.

Estudios de usabilidad entre el *producto de software origen* (ya conocida por el usuario) y el *producto de software destino* (desconocido por el usuario) deben ser orientados a resolver las siguientes preguntas: ¿cuánto peso tendría en la productividad del usuario el RE-Aprendizaje de nuevas formas de actuar? ¿qué objetos y acciones tendría que incorporar a su estructura cognitiva la cual está minada de las estructuras que adquirió utilizando la herramienta propietaria?

Es claro que en la usabilidad es la cualidad clave y esperada en el producto de software destino además está muy sesgada por el contexto organizacional donde ocurre el proceso de migración.

Esta investigación inicia con la propuesta del proyecto de investigación ProLibre [5], dentro del mismo se realizó un trabajo especial de grado titulado "Una Propuesta para la Migración de Software basada en un Análisis de Compatibilidad" [6] que permitió probar la propuesta en el contexto de migración de herramientas propietarias a libres. Actualmente, la propuesta ha evolucionado pudiendo ser aplicada en otros contextos de desarrollo, teniendo como supuesto que todo desarrollo de software implica una migración de modelos metales y esto, incide con el aprendizaje del usuario en el nuevo software.

B. Requerimientos de asistencia

Toda organización involucrada en un proceso de migración debe considerar como riesgo el impacto del producto de software destino en el usuario final, puesto que esto tiene incidencia en la productividad de la organización. También hay que considerar que los cursos de entrenamiento en el nuevo producto no son la solución "mágica" para aminorar ese riesgo.

La propuesta se centra en introducir nuevas variantes al tipo asistencia requeridas por el usuario. Se comienza por entender que el usuario no necesita ser "entrenado", más bien ser "asistido y motivado" para enfrentar un cambio en su forma de trabajar. Para lograr esto, es necesario fijar requerimientos cognitivos dentro de los escenarios posibles del proceso de migración.

Se consideran los siguientes escenarios:

- Ideal: el usuario se identifica totalmente con el nuevo ambiente, logrando realizar las tareas de la misma forma que lo hacía con el producto de software origen.
- Familiar: el usuario se identifica con el nuevo ambiente, asocia una cantidad considerable de elementos con los existentes en su memoria de largo plazo, logrando realizar las tareas de la misma forma que lo hacía con el producto de software origen, pero desconoce algunas tareas.
- Transporte: el usuario se identifica parcialmente con el nuevo ambiente, asocia pocos elementos con los existentes en su memoria de largo plazo, logra realizar algunas tareas de igual forma que en el producto de software origen. Desconoce gran parte del producto de software destino.
- Catastrófico: El usuario no logra identificarse con el nuevo ambiente. No reconoce elementos con los existentes en su memoria de largo plazo.

En una escala de valoración de riesgo, el mayor se encuentra en escenario catastrófico y el menor en el ideal. El determinar tempranamente el tipo de escenario al cual se enfrenta en el proceso, permitiría planificar qué tipo de asistencia debe ser contemplada para el usuario. Análogo a los escenarios se establecen los siguientes requerimientos:

a) En un escenario ideal, el usuario no requiere ningún tipo de asistencia.

b) En un escenario familiar, el usuario requiere asistencia en el momento cuando surja algun incoveniente. La asistencia en línea o telefónica es apropiada puesto que se prevee poca incidencia de problemas.

c) En un escenario de transporte, la asistencia debe ser planificada como curso, preferiblemente sobre el trabajo activo para mejorar los tiempos de asimilación del nuevo producto de software. Algunas alternativas como la asistencia in situ resulta de gran interés y provecho.

d) En un escenario catastrófico, requiere planificar totalmente la forma cómo el usuario debe desaprender para iniciar la asimilación del nuevo conocimiento.

Los requerimientos se ubican en estimar la cantidad de elementos que tiene presente el usuario en sus estructuras cognitivas, es decir hurgar en el conocimiento previo, para determinar si estos elementos son suficientes para manipular el nuevo ambiente.

III. ESTIMANDO EL COSTO DE APRENDIZAJE

Es usual asociar al término estimación un valor económico, en nuestro contexto, el costo de aprendizaje se entiende como el esfuerzo realizado por el usuario para ejecutar una tarea con éxito, dentro de un ambiente de trabajo.

A continuación se presentan las actividades propuesta en el método, donde inicialmente se define los términos utilizados así como la estrategia que apoya el método.

A. Terminología utilizada

Un *modelo mental* es una representación de un estado de cosas del mundo exterior almacenada en la memoria de largo plazo. Se trata de una forma de representación de los conocimientos reconocida por numerosos investigadores en ciencias cognitivas por ser la manera natural por la cual la mente humana construye la realidad, concibe sus alternativas y verifica hipótesis cuando entra en un proceso de simulación mental.

Un modelo mental según Norman [7], tiene las siguientes características:

- Es incompleto.
- Es ejecutable mentalmente, es decir, el usuario puede mentalmente simular su funcionamiento.
- Es inestable, es decir, el usuario olvida sus detalles fácilmente.
- No tiene unos límites claros y se confunde con los modelos mentales de sistemas físicos similares.
- Es acientífico e incluye supersticiones y creencias erróneas sobre la conducta del sistema.
- Es parsimonioso porque los usuarios prefieren reducir su complejidad.

Por consiguiente, el modelo mental no es ni completo, ni consistente, ni exacto, representa más bien un conocimiento general, que puede ser vago, acerca del sistema.

Cuando un usuario se enfrenta por primera vez a un sistema, se forma un modelo mental para tratar de entender su funcionamiento. El modelo mental puede no ser acertado en principio, sin embargo, a través de la continua interacción con el sistema y mediante un proceso de aprendizaje, se va refinando para convertirse en un conocimiento específico. Este proceso hace que el modelo mental del usuario se vaya acercando a la definición del modelo conceptual del sistema.

El *modelo conceptual del sistema* debe suministrar información al usuario acerca de lo que hace el sistema y los mecanismos para llevarlo a cabo. Su importancia radica en que debe favorecer el aprendizaje del sistema, es una guía para predecir el comportamiento del sistema además, el usuario utilizará este modelo para establecer estrategias encaminadas a resolver sus problemas.

El modelo conceptual es la imagen del sistema tal como fue concebida por sus desarrolladores. En el desarrollo de la interfaz, intervienen varios modelos mentales, el del diseñador, programador, etc., el modelo conceptual es el resultado de ese proceso.

Un modelo conceptual correcto permite al usuario predecir los efectos de sus actos [8] Un usuario experto posee un modelo mental muy cercano al modelo conceptual del sistema, mientras más parecidos, aumenta la usabilidad de la interfaz.

El modelo de la acción o tarea propuesto por Norman [8] explica como el usuario transita desde su modelo mental hasta el modelo conceptual en la interacción con el sistema. El usuario busca realizar una actividad específica para cumplir su meta planteada, realiza acciones individuales que identifica en el modelo conceptual, el conjunto de estas acciones se les denominan *tarea*.

Las tareas tienen un objetivo y pueden ejecutarse siguiendo las opciones de un menú desplegable, a través del reconocimiento de íconos o utilizando los atajos de combinaciones de teclas conocidos como shortcuts.

Los productos de software que han sido diseñados para el mismo propósito, no sólo deberían compartir funcionalidades, sino también la forma de ejecutar las tareas. Se puede definir una tarea t_i como una secuencia finita de funcionalidad y se expresa como:

$$t_i = f_1, f_2, .., f_n \quad (1)$$

El orden de la secuencia es de importancia, puesto que puede cambiar el efecto de la tarea. En algunos contextos, se trabaja la tarea como la funcionalidad, en este trabajo se busca generalizar este concepto para poder trabajar los dos tipos de sistema, el primero viene dado como un conjunto de funcionalidades y es la responsabilidad del usuario componerlas para construir las tares; en el segundo, la tarea es el elemento con el cual interactúa el usuario.

El termino *metáfora* es muy utilizado en diferentes contextos, particularmente en el área de la interacción humano-computador tiene mucha relevancia, basta señalar el cambio de paradigma iniciado con la introducción de la metáfora escritorio en la década de los ochenta.

La *metáfora de interacción* es la representación de un objeto o de una acción en la interfaz, para soportar talprincipio; es decir, es lo que el usuario percibe de la interfaz, y debe permitir que él comprenda el nuevo dominio por analogías con el dominio fuente [9].

Las metáforas pueden variar de pequeñas imágenes puestas en botones de barras hasta pantallas completas, es por eso que se suele referir a éstas como metáforas de interacción. Lo realmente importante de una metáfora es su silueta, pues al ser observada perdura por más tiempo que los detalles o colores de la misma, por tanto la silueta o forma externa de una metáfora transmite la mayor información.

La importancia de las metáforas es que como instrumentos permiten recuperar rápidamente un modelo mental que posibilita la comprensión de los objetos del sistema. Al ser utilizadas en diversos productos de software, las metáforas deberían ser consistentes a nivel de aspecto y funcionalidad, para provocar en la mente del usuario los mismos modelos conceptuales, lo cual cooperaría de forma significativa en un proceso de migración.

Si el usuario puede identificar las metáforas en un software y encontrar sus imágenes o equivalentes en el software al que desea migrar, para luego descubrir que el significado de ambas metáforas es similar, entonces no tiene que desechar su modelo mental previo, sino simplemente reutilizarlo.

Se puede concluir, que el conocimiento adquirido por el usuario acerca de metáforas y tareas con un cierto sistema, constituyen las estructuras cognitivas necesarias para construir el modelo conceptual de otro software.

B. Estrategia del método: Mover Modelos

El método propuesto está basado en aprovechar la afinidad entre las herramientas de software involucradas en el proceso de migración. Esto es, que los modelos conceptuales de tales herramientas puedan ser proyectados entre sí, es decir, que cada funcionalidad, tarea y metáfora de interacción del producto de software origen tenga su correspondiente imagen en el software destino. Si es posible establecer una función inyectiva o biyectiva entre ambos sistemas, se puede afirmar que estas herramientas son homólogas y no debería representar un problema el uso de una u otra.

En la Figura 1. presenta las diferentes relaciones entre modelos consideradas en este trabajo. Se asume que existe una el modelo mental de A es similar al modelo conceptual Software A. En el caso del Software, dentro del contexto de migración, no tenemos la certeza de si el usuario experto A pueda comportase de igual forma con el software B. Una primera idea es proyectar de alguna forma el modelo mental de A en B, pero dada la naturaleza de inestabilidad y reducción, los resultados de esta proyección no serían confiables.

Una vía segura y confiable se encuentra en el análisis entre los modelos conceptuales, si se logra establecer una proyección desde el modelo conceptual del software A al B, entonces se puede proyectar el comportamiento del usuario experto A en el usuario final del producto de software B.

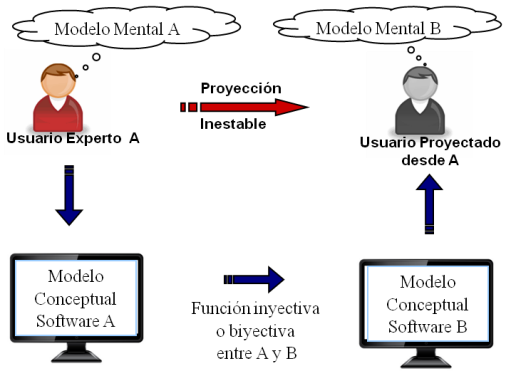

Figura 1. Relación entre modelos mentales y conceptuales

Se trata de ir moviendo el modelo mental A a través de de los modelos conceptuales hasta llegar al modelo mental B. El punto central de la propuesta es la definición de una función inyectiva o biyectiva desde el modelo conceptual software A al B.

La obtención de tal función, permite considerar los modelos conceptúales como homólogos, si la función es inyectiva, se garantiza que el usuario experto A tendrá el mismo comportamiento, pero una asistencia al usuario es necesaria puesta que algunas funcionalidades de B no son provista en A. Si se establece la biyección entre los modelos, implica que los productos de software son totalmente homólogos y las diferencias serian meramente sintáctica.

Ejemplos particulares de las situaciones descritas anteriormente son: en el caso de establecer una función inyectiva es muy común cuando se trata de la migración de versiones de un mismo producto de software bajo un enfoque de desarrollo iterativo e incremental, asimismo, establecer una biyección correspondería a versiones del desarrollo evolutivo de un producto de software. En ambos casos se apunta al mismo software y el interés se centra en establecer las relaciones entre software desarrollados para el mismo propósito.

C. Extraer el modelo mental del usuario a través del modelo conceptual del producto de software origen

Esta es la primera actividad prevista en el método. Se supone que el usuario domina un conjunto de funcionalidades del software para llevar a cabo su trabajo, de manera que le resulta familiar la estructuración de la imagen del sistema (metáfora de interacción), los iconos, la forma en que se combinan las funcionalidades para conformar las tareas representativas que le permiten ser productivo.

Todo esto representa su modelo mental, el cual está almacenado en su memoria de largo plazo y es recuperado cuando se le presenta algún estímulo (imagen o ícono) para traerlo a su memoria de corto plazo y comenzar a interactuar siguiendo el modelo de la tarea planteado por Norman [7]. Se plantea la siguiente fórmula para describir el modelo mental del usuario:

$$MMU = TareasRepresentativas(SoftwareOrigen) + MetáforaInteracción\ (softwareOrigen) \quad (2)$$

Donde *MMU* corresponde al Modelo Mental del Usuario expresado en términos de las tareas que domina y a la percepción que tiene del ambiente de trabajo a través de la Metáfora de Interacción del sistema.

Un conjunto T de tareas representativas viene dado por

$$T = \{t_1, t_2, t_3, ...\} \quad (3)$$

donde t_i es una tarea que el usuario domina. El conjunto T representa todo el conocimiento que tiene el usuario experto para ser productivo en su ámbito de trabajo. Para obtener este

modelo mental, se sugiere aplicar técnicas de usabilidad tales como: recorridos cognitivos por expertos, entrevista con usuarios expertos y observaciones de campo.

La metáfora de interacción está representada por la distribución de los elementos en el entorno de trabajo, las formas que tienen estos elementos (imagen y silueta) y las técnicas de interacción para manipular la interfaz.

Las tareas representativas vienen dadas por el análisis de aquellas tareas que ya conocen los usuarios y realizan de forma experta, con esto se conocerán los pasos o maneras de ejecutarlas, así como también se contempla la observación de las siluetas de íconos, y su coherencia con relación a las funciones asociadas. El estudio de las tareas representativas y las metáforas de interacción es lo que determina el modelo conceptual de A.

Si el usuario es experto en el uso del software A, entonces, el modelo mental que tiene el usuario acerca de A es subconjunto del Modelo Conceptual de A (*MC(A)*), tal que,

$$U \text{ es experto en } A \Rightarrow MMU(A) \subset MC(A) \quad (4)$$

De la fórmula (4) se observa que un usuario experto no tiene porque dominar todas las funcionalidades provistas por el sistema.

D. Proyectar el modelo conceptual del software origen hacia el modelo conceptual del software destino

Se debe buscar la relación existente entre los modelos conceptuales del software A y el software B para proyectar el modelo de A hacia el de B, estimando que cada tarea de A tenga su correspondiente imagen en B. Esto es que cada tarea que se ejecuta en A pueda llevarse a cabo también en B.

Tomando entonces el conjunto de tareas representativas que se tienen del software A, se establece una función de correspondencia entre A y B, tal que:

$$h : \mathrm{A} \longrightarrow \mathrm{B} \quad (5)$$

La función homologación *h* define que una tarea de A pueda ser homologada en B.

$$\forall t_{Ai} \in \mathrm{A} \quad \exists t_{Bi} \in \mathrm{B} \mid (t_{Ai}, t_{Bi}) \in h \quad (6)$$

Esto significa que a cada tarea t_{Ai} de A, le corresponde por h una tarea t_{Bi}, y al menos una, de B, que se denomina imagen de t_{Ai} por h y que se denota $h(t_{Ai}) = t_{Bi}$

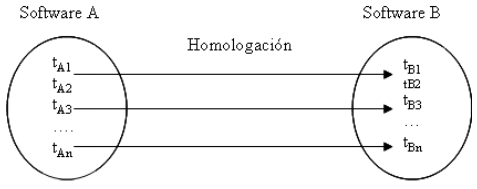

Figura 2. Función homologación

Donde se obtiene el primer movimiento y se expresa como:

$$MM(A) \subseteq h(MC(A)) = PMM(B) \quad (7)$$

En (7), la aplicación de la función *h* sobre el modelo conceptual del software A *(MC(A))*, tiene como resultado la proyección del modelo mental sobre el software B (*PMM(B)).* En principio se expresa que *un usuario experto del producto de Software A puede manipular funcionalmente el producto de Software B*, pero esto no es suficiente, hay que analizar desde la perspectiva cognitiva, si los elementos en B tienen el mismo efectos en el usuario que sus homólogos en A.

En este actividad existen ciertas situaciones donde se requiere reflexionar si trata de una migración o no, por ejemplo, si para una t_{Ai} no existe su homólogo, habría que evaluar cuál es el porcentaje de tareas en esa situación, dependiendo de esto el impacto que tendría el re-aprendizaje con respecto a la cantidad de usuario. Puede tomarse la decisión de no migrar, por ejemplo, si el número de proyecciones encontradas no son significativas bajo cierto criterio fijado previamente.

E. Analizar la compatibilidad entre los productos de software

La base del análisis de compatibilidad es determinar desde el punto de vista ontológico si el software A puede ser utilizado como "ancla conceptual" para aprender efectivamente el software B, si se demuestra que existe la compatibilidad, el aprendizaje del software B se hará de forma significativa.

En este análisis de compatibilidad, la metáfora de interacción tiene un papel fundamental, puesto que permite asociar derivar un conjunto de asociaciones o proyecciones entre elementos del producto de software origen y el destino, así como un conjunto de inferencias que resultan posibles.

Las asociaciones entre elementos se denominan correspondencias ontológicas, aquí se introducen los conceptos de *familiaridad* y *transporte* [10], el primero permite traer de la memoria de largo plazo a la de corto plazo, elementos que pueden ser útiles para el aprendizaje, y el segundo concepto permite transportar elementos que deben ser re-aprendidos en el nuevo software.

La compatibilidad va a depender del cumplimiento de los siguientes criterios:

- *Funcionalidad*: Se presentan dos casos posibles, el primero, la funcionalidad de t_{Ai} es la misma en t_{Bi} pero se llevan a cabo con diferentes pasos, esto implica un re-aprendizaje por parte del usuario que genera cambios en el modelo mental, que sirve de *transporte* para iniciar el aprendizaje en B. En el segundo la funcionalidad de t_{Ai} es la misma en t_{Bi} y se ejecutan siguiendo los mismos pasos, lo cual implica la reutilización del modelo mental y por ende la *familiaridad* tiene una alta incidencia.
- *Percepción*: Si la metáfora de interacción para t_{Ai} es similar a la de t_{Bi} se dice que t_{Ai} es familiar a t_{Bi}. Se debe considerar la misma distribución del espacio de trabajo y el uso de las mismas técnicas de interacción para soportar la familiaridad.

- *Nominación*: Que t_{Ai} se identifique con el mismo nombre que t_{Bi}: Si existe una compatibilidad nominal entonces aumenta la compatibilidad, pero si no la tiene entonces los cambios en el modelo mental del usuario pueden afectar el aprendizaje del software B, puesto que implica un doble proceso desaprender-aprender. Esto lleva a definir una estrategia de aprendizaje dentro del proceso de migración.
- *Iconografía*: Que el icono de t_{Ai} posea la misma silueta que el icono de t_{Bi}, lo primero que ve el usuario de un icono es su silueta, todas aquellas diferencias y detalles que puedan estar plasmadas en la imagen resultan irrelevantes para traer su significado desde la memoria de largo plazo hacia la de corto plazo. El análisis de compatibilidad es el mismo que se aplica para el caso de las tareas con el mismo nombre.

Una tarea t_{Ai} es compatible por familiaridad con su homóloga t_{Bi}, si y sólo si cumple con los criterios de funcionalidad familiar, percepción, nominación e iconografía.

En el mismo sentido, tarea t_{Ai} es por transporte con su homóloga t_{Bi}, si y sólo si cumple el criterio de funcionalidad por transporte, pudiendo darse o no la percepción, nominación e iconografía.

La principal condición para que dos tareas no sean compatibles es que, no sean funcionalmente equivalente en ninguno de los dos sentidos (familiar, transporte).

Al determinar la compatibilidad, se espera que los usuarios puedan reutilizar el modelo mental que tienen del software origen para operar el software destino, del análisis se desprende:

a) Compatibilidad por familiaridad implica la reutilización del modelo mental de A en B, caracterizando el proceso de migración de una alta factibilidad.

b) Compatibilidad por transporte implica el re-aprendizaje de nuevas formas para ejecutar las tareas en B, dónde no necesariamente haya que reaprender todo el espacio de trabajo, sino al menos su distribución y las técnicas de interacción, nombres e íconos que formarán parte del nuevo modelo mental. Esto caracteriza al proceso de migración como factible, pero necesariamente harán falta definir estrategias para la formación del modelo mental de B.

c) No compatibilidad entre los modelos, esto supone un proceso de migración lento y difícil para el usuario, puesto que implica el duro proceso de desaprender lo que conoce del software A para aprender una nueva forma de realizar las tareas en B.

La tabla I está diseñada para registrar las acciones o pasos requeridos para ejecutar cada tarea, tanto en el software A como en el B, y de esa forma identificar el tipo de compatibilidad presente, cuya justificación se explica en las *Observaciones.*

TABLA I. ANÁLISIS DE COMPATIBILIDAD DE LA TAREA T_i

Análisis de compatibilidad de t_i	
Software A	Software B
Nombre de t_{Ai}:	Nombre de t_{Bi}
Forma de realizarla en A:	Forma de realizarla en B:
Compatibilidad: Familiar___ Transportable____ No compatible_____	
Observaciones:	

Después de determinar la compatibilidad entre cada tarea, los resultados pueden ser contabilizados calculando el porcentaje de éstas que son compatibles (ver tabla II), bien sea por familiaridad o por transporte, para definir el nivel de factibilidad del proceso de migración e incluso hacer proyecciones sobre los costos de entrenamiento necesario.

TABLA II. EJEMPLO DE RESULTADOS DE COMPATIBILIDA POR TAREAS

Resultados de Compatibilidad de software por tareas			
Nombre de t_{Ai}	t_{Ai} es familiar a t_{Bi}	t_{Ai} es transportable a t_{Bi}	t_{Ai} no es compatible con t_{Bi}
t_{A1}	**X**		
t_{A2}	**X**		
t_{A3}			**X**
t_{A4}		**X**	
t_{A5}	**X**		
t_{A6}	**X**		
t_{A7}	**X**		
t_{A8}		**X**	
t_{A9}	**X**		
t_{A10}	**X**		
Total	**70%**	**20%**	**10%**

Se asume que si se tienen un gran porcentaje de tareas compatibles por familiaridad entonces el nuevo aprendizaje tendrá lugar a través de la exploración que realice el usuario por si sólo sobre el nuevo software. Si se tiene un alto porcentaje de compatibilidad por transporte, se requerirá de algún entrenamiento previo antes de iniciar el proceso de migración, pero si la no compatibilidad entre tareas es de alto porcentaje, la migración requerirá de cursos de entrenamiento fuera del contexto de trabajo.

Para que los productos de software sean compatibles y por ende la migración factible, el 100% debería estar compartido entre las columnas de familiaridad y transporte. Partiendo de esto se puede obtener que el porcentaje de tareas compatibles sea de un 50% para familiaridad y un 50% para transporte, en cuyo caso los usuarios podrían asimilar la migración de forma fácil en un 50%, mientras que el 50% restante requeriría de un reaprendizaje no tan complejo y asimilable en poco tiempo.

El caso que se presenta en el ejemplo de tabla de compatibilidad se puede predecir un 70% de aceptación inmediata por parte de los usuarios, un 20% necesitará un

reaprendizaje simple y para enfrentar el 10% no compatible se requerirá reentrenamiento externo que implica un gasto no previsto, sin embargo esto no garantiza completamente que el usuario asimile ese 10%. Por ello es recomendable que el porcentaje de la no compatibilidad sea nulo o el mínimo posible, por ejemplo, si se tienen varias alternativas de productos de software para migrar, se sugiere escoger aquella que aporte el menor porcentaje de no compatibilidad, siempre y cuando éste no supere al porcentaje de familiaridad o transporte.

Si el porcentaje de no compatibilidad es considerablemente alto, se puede predecir que la migración no sería viable y no valdría la pena intentarla.

F. Validar con los usuarios

La validación con usuarios puede ser omitida si hasta entonces el método ha sido suficiente para demostrar la inviabilidad de la migración. En caso contrario es recomendable realizar el trabajo de campo con los usuarios para validar los resultados obtenidos en el Análisis de Compatibilidad.

Esta validación pretende comprobar los resultados del método descrito, aplicando pruebas sobre las herramientas a migrar y una muestra de usuarios expertos, cuyos resultados proveerán las mejores prácticas a seguir y serán el soporte para predecir con propiedad la viabilidad de un proceso de migración.

En cuanto a la selección de usuario se hace énfasis que los mismos debe ser expertos, ya que estos representan los de mayor productividad y que tienen estructuras cognitivas muy arraigadas en el producto de software origen. En contraposición, los usuarios inexpertos construirán nuevas estructuras ya que no tienen que pasar por un proceso de des-aprendizaje.

Los criterios contemplados para validación con usuarios son los siguientes

- Número de tareas completadas sin asistencia
- Número de tareas completadas con asistencia
- Número de preguntas realizadas durante la ejecución de las tareas
- Número de tareas no completadas

El número de tareas completadas sin asistencia, en promedio, debe tender al número de tareas compatibles por familiaridad; así como también el número de tareas realizadas con asistencia se asocian con las tareas compatibles por transporte.

El número de preguntas realizadas es indicativo de que tan difícil le resulta establecer un puente cognitivo entre su modelo mental y el modelo conceptual del software. Incluso, el tipo de preguntas orienta de alguna manera al tipo de asistencia requerida.

El número de tareas no completadas, refiere a las tareas que no son compatibles. Otros criterios, como por el ejemplo el tiempo de ejecución de las tareas, pueden ser introducidos como indicadores en la actividad de validación.

IV. NUEVAS APLICACIONES

Al inicio de la investigación el interés se centró predecir información sobre la factibilidad de migración desde la perspectiva del usuario, posteriormente se identificaron otras aplicaciones del método

A continuación, se describe la base en estas aplicaciones del método.

A. Midiendo la facilidad de aprendizaje

El usuario tiene un conocimiento previo que debe servir de ancla conceptual para el nuevo software. El trabajo consiste en determinar qué estructuras cognitivas presentes en el conocimiento previo del usuario y que, a su vez pueden ser reutilizadas en el software. Para este fin, se realiza las actividades de seleccionar el conjunto de tareas representativas y la validación con los usuarios.

En principio, esta del método está inspirando en las pruebas contextuales de observación de directa a los usuarios, el aporte se encuentra en cómo contabilizar los resultados con el rango difuso: familiar, transporte y no compatibilidad en función de los porcentajes. Los porcentajes indicarán las porciones del software asociadas a los niveles de compatibilidad entre el modelo mental del usuario y el modelo conceptual del sistema, por ejemplo:

- En un software muy fácil de aprender, el porcentaje de tareas compatibles debe ser mayor que la suma de tareas familiares y no compatibles, siendo esta suma muy pequeña.
- En contraposición, un software que no sea fácil de aprender ocurre, que el porcentaje de tareas no compatible debe ser significativamente mayor que la suma de las tareas familiares y transportable.

En la Figura 3 se expresa en términos de conjunto, las situaciones extremas y media que puede arrojar la aplicación del método. La compatibilidad total muestra la contención del modelo mental del usuario en el modelo mental del usuario en el modelo conceptual, en el otro extremo se muestra la incompatibilidad de modelos a través de la disyunción de conjuntos.

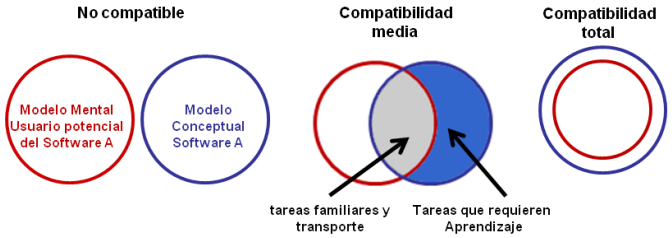

Figura 3. Compatibilidad entre Modelo mental y Conceptual

B. Comparación de software

Medir la facilidad de aprendizaje abre la posibilidad de establecer la comparación entre software, es decir, determinar

para un usuario cuál de dos productos de software es más fácil de aprender.

Las condiciones para esta aplicación, es que se trate de software homólogos, donde los porcentajes arrojados por la validación de usuario son la base para establecer la comparación.

Un producto de software A es más fácil de aprender que un software B, si y sólo si, el porcentaje de tareas compatibles de A con el usuario es mayor que el porcentaje de tareas compatibles de B. La figura 4, muestra la estrategia de comparación en términos de la proyección del usuario sobre los diferentes productos de software.

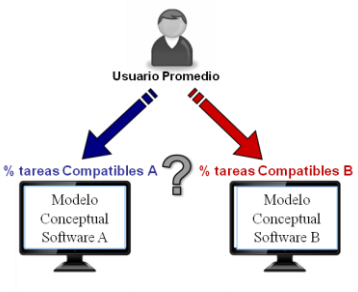

Figura 4. Estrategia de comparación

Es recomendable, realizar la validación utilizando diferentes usuarios para cada software. La validación puede ser combinada recurriendo a usuarios con diferentes niveles de experticia (novato, medio, experto).

C. Capturando requerimientos de aprendizaje

Al momento de desarrollar un software, una de las cualidades importante es la facilidad de aprendizaje, desde el punto de vista de competitividad del software a desarrollar, se hace necesario revisar las características de otro software que pueden ser su competencia. La idea central es que los usuarios de la competencia migren con facilidad al software que se va a desarrollar.

Se parte de la selección de uno o más producto de software de la competencia, previamente se requiere determinar el conjunto de tareas representativas que se desean tener en el software a desarrollar (Figura 5). Todo esto supone que se conoce las funcionales que deben ser soportadas, pero no se tiene la decisión de cómo estás deben ser percibidas por el usuario en un ambiente de trabajo.

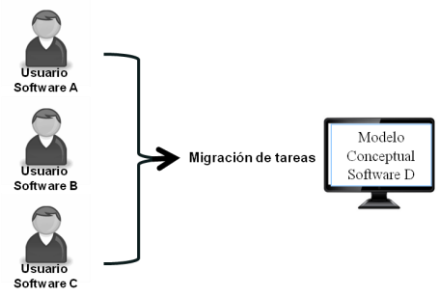

Figura 5. Esquema para la migración en la construcción de Software

Si se dispone de un sólo software, entonces el reto es proyectar el modelo mental de un usuario del software de la competencia sobre el que se va a desarrollar, por ejemplo, si se quiere desarrollar una red social pero con fines académicos, es deseable garantizar que cualquier usuario de la mayor red social (Facebook) migre con facilidad a este nuevo software.

Si se tiene más de un software de la competencia, es requerido realizar la comparación para extraer las tareas más compatibles y proyectarlas en el nuevo desarrollo. Esta proyección implica el rediseño de las tareas para lograr la coherencia del conjunto seleccionado, sin pérdida de compatibilidad.

V. CONCLUSIÓN

El método de estimación permite cuantificar en términos de la compatibilidad entre tareas, el nivel de aprendizaje que puede tener un usuario experto, cuando se enfrenta a un nuevo software dentro del mismo dominio.

Los resultados de la aplicación del método, en el caso de la migración de software, aportan información útil para la toma de decisión, esto conduce a valorar la factibilidad de la migración desde la perspectiva del usuario.

El movimiento de modelos mentales a través de los modelos conceptuales, representa el reuso cognitivo que puede ser aprovechado a fin de construir aplicaciones fáciles de aprender.

Nuevas variantes en la aplicación, muestran la extensibilidad del método en nuevos contexto.

REFERENCES

[1] Decreto N° 3.390 Publicado en la Gaceta oficial Nº 38.095 de fecha 28/ 12/ 2004

[2] República Bolivariana de Venezuela Ministerio de Ciencia y Tecnología: Plan nacional de migración a software libre de la administración pública nacional www.mct.gov.ve.

[3] N. Montano, A. Acosta, N. Merchán, M. Avila, Evaluaciones para garantizar el aprendizaje de aplicaciones en usuarios novatos XXVI Conferencia Latinoamericana de Informática (CLEI_2000). México D.F, México, 2000.

[4] J. Ausubel, J.D Novak, Hanesian H. Psicología Educativa: Un punto de vista cognoscitivo 2da. Edición TRILLAS México (1983).

[5] N. Montaño, Prolibre: migración de software **Pro**pietario a software **Libre** y su incidencia en el RE aprendizaje del usuario, proyecto de investigación aprobado por el Consejo de Desarrollo Científico y Humanístico de la Universidad Central de Venezuela, 2006

[6] F. Palomino, L. Malavé, Una Propuesta para la Migración de Software basada en un Análisis de Compatibilidad, trabajo Especial de grado Escuela de Computación de la UCV, tutores N. Montaño, A. Sanoja, 2007.

[7] D. Norman, Some observations on mental models en Gentner, D. y Stevens, AL eds. Modelos mentales. Lawrence Erlbaum Associates Inc. 1983.

[8] D. Norman, The design of everyday things. Doubleday, Nueva York. 1990.

[9] A.E Acosta, N. Zambrano, Interacción Humano-Computador: Fundamentos de Diseño. Lecturas en Ciencias de la Computación ISSN 1316-6239, ND 99-02 (53 págs.), Escuela de Computación, UCV, Caracas. 1999

[10] G. Lakoff, Johnson M. Metáforas de la vida cotidiana. Madrid: Cátedra. 1995