LA DEPURACIÓN COMO COMPETENCIA EN LA FORMACIÓN DEL PROFESIONAL INFORMÁTICO

Depurations as competence on informatics professional formation

WALFREDO GONZÁLEZ HERNÁNDEZ*

Recibido: 20 de diciembre de 2016. Aceptado: 27 de diciembre de 2016

DOI: http://dx.doi.org/10.21017/rimci.2017.v4.n7.a22

Resumen

La depuración es una de las actividades más importantes en la actividad ingenieril. En el artículo se toma posiciones de los autores sobre el modelo y sus clasificaciones. Por otro lado, se asumen los elementos esenciales del currículo informático en el mundo y la importancia de la depuración en ellos. En el último acápite se analiza el cómo estructurar la depuración como elemento esencial en su modo de actuación.

Palabras clave: enseñanza de la informática, competencias, depurar.

Abstract

The modelling is one of the more important activities in the engineers' activity. The author assumes theoretic positions as from the critical analysis of the bibliography on the model and its classifications. In addition, in the article the author assumes the information-technology curriculum's essential elements in the world and the importance of the modelling in them. It is examined in the last clause the how structuring the modelling like essential element in your mode of acting.

Keywords: informatics teach, competence, modelling.

I. Introducción

El desarrollo alcanzado por las Tecnologías de la Información y la Comunicación (TIC) y sus aplicaciones en la educación a nivel mundial y en el ámbito nacional, plantean la necesidad de investigar un conjunto de problemas inherentes al proceso de enseñanza-aprendizaje de la Informática. Uno de estos problemas es la formación del profesional de esta área del conocimiento.

La informática ha desarrollado en su devenir histórico varias disciplinas científicas que se estructuran como disciplinas en el currículo de su formación. Sin embargo, existen determinados contenidos que son esenciales para su formación y que no se encuentran enmarcados en una única disciplina. Uno de estos contenidos es precisamente la modelación.

La depuración de los sistemas informáticos es una de las actividades más importantes en la formación del ingeniero informático. Sin embargo, el análisis debe estar en el orden de las estructuras psicológicas de la personalidad que se debe considerar como depuración de los sistemas informáticos en la formación del profesional informático. Un primer punto de partida en este análisis es el papel de la depuración en la actividad informática y su forma fundamental de organización. Un segundo momento de análisis se encuentra en la expresión de la depuración en el diseño curricular de los profesionales informáticos. Por último, entonces determinar qué tipo de estructura de la personalidad ocupa la depuración en la formación de este profesional.

* Doctor en Ciencias Pedagógicas. Profesor Titular de Ingeniería Informática. Universidad de Matanzas, Matanzas, Cuba. Correo electrónico: walfredo.glez@umcc.cu

II. Desarrollo

A. La depuración como parte de la actividad informática y el Proyecto

La detección de errores ha sido una de las formas fundamentales que presentan diversas especies de animales como forma de adaptación al medio en el cual se desenvuelven. Con el advenimiento de la especie humana producto de la evolución este proceso correctivo ha sido llevado a su actividad como uno de sus eslabones fundamentales. La informática no escapa a esta realidad y uno las actividades hoy a las cuales se les dedica mayor tiempo es a la detección de errores [1-3]. Para estos autores la actividad que se realiza para detectar estos errores se denomina pruebas y la actividad de eliminarlos se denomina depuración. Para este artículo se asume entonces que es imposible depurar un sistema informático sin las pruebas. De esta afirmación anterior se deriva entonces que, si bien son actividades diferentes, están íntimamente relacionadas y la depuración depende de las pruebas al software.

Siguiendo las ideas anteriores, se puede inferir que es posible integrar las pruebas y la depuración en una actividad más general. De esta inferencia se desprende que la puesta a punto del sistema es aquella actividad en la cual se realizan un conjunto de acciones para colocar el sistema con la menor cantidad de errores posibles a funcionar en la organización a la cual está destinado. Las acciones fundamentales en esta actividad serían pues probar sistemas y depurar sistemas. En este caso el sistema de acciones es ordenado de manera secuencial y se comenzaría en primer lugar con las pruebas y posteriormente con la depuración. Estas consideraciones llevan a analizar en primer lugar los resultados fundamentales acerca de las pruebas y posteriormente la depuración como actividades informáticas.

Las pruebas se han estructurado en dos grandes grupos, aunque en este sentido en la literatura consultada [4][5]; existen criterios divergentes, casi todos coinciden en varias pruebas: pruebas unitarias, pruebas funcionales, pruebas de integración, pruebas de validación, pruebas de sistema, caja blanca (sistemas), caja negra (sistemas), pruebas de aceptación, pruebas de regresión, pruebas de carga, pruebas de prestaciones, pruebas de recorrido, pruebas de mutación. Cada una de estas pruebas presentan características diferentes que serán analizadas a continuación.

Prueba unitaria es una forma de probar el correcto funcionamiento de un módulo de código. Esta prueba se utiliza para asegurar que cada uno de los módulos funcione correctamente por separado. La idea es escribir casos de prueba para cada función no trivial o método en el módulo de forma que cada caso sea independiente del resto. En esta prueba, como en otras que se utilizan datos de entrada, se particiona el conjunto de datos en clases de equivalencia que le permita al probador verificar con un representante de cada clase. Al probar con un representante de la clase, lo que suceda con él se cumple para el resto de los integrantes de la clase.

La prueba de unidad centra el proceso de verificación en la menor unidad del diseño del software: «el módulo o clase». En el caso de la clase se toman como unidad la clase y sus métodos, en estos últimos se orienta a caja blanca y este paso se puede llevar a cabo en paralelo para múltiples módulos. Usando la descripción del diseño procedimental como guía, se prueban los caminos de control importantes, con el fin de descubrir errores dentro del límite del módulo.

Para que una prueba de unidad sea considerada como válida se deben cumplir los siguientes requisitos:

1. Automatizable: no debería requerirse una intervención manual. Esto es especialmente útil para integración continua.

2. Completas: deben cubrir la mayor cantidad de código.

3. Repetibles o Reutilizables: no se deben crear pruebas que sólo puedan ser ejecutadas una sola vez. También es útil para integración continua.

4. Independientes: la ejecución de una prueba no debe afectar a la ejecución de otra.

5. Profesionales: las pruebas deben ser consideradas igual que el código, con la misma profesionalidad, documentación, etc.

Aunque estos requisitos no tienen que ser cumplidos al pie de la letra, se recomienda seguirlos o de lo contrario las pruebas pierden parte de su función. En la actualidad la mayoría de los IDE de los lenguajes de programación ya traen incorporados mecanismos para la depuración de este tipo de pruebas. Ejemplo de ello son PHP, Unit, JUnit entre otros.

Una prueba funcional es una prueba basada en la ejecución, revisión y retroalimentación de las funcionalidades previamente diseñadas para el software. Las pruebas funcionales se hacen mediante el diseño de modelos de prueba que buscan evaluar cada una de las opciones con las que cuenta el paquete informático.

Pruebas integrales o pruebas de integración son aquellas que se realizan en el ámbito del desarrollo de software una vez que se han aprobado las pruebas unitarias. Únicamente se refieren a la prueba o pruebas de todos los elementos unitarios que componen un proceso, hecha en conjunto, de una sola vez. Consiste en realizar pruebas para verificar que un gran conjunto de partes de software funcionan juntos. Las pruebas de integración (algunas veces llamadas integración y testeo I&t) es la fase del testeo de software en la cual módulos individuales de software son combinados y testeados como un grupo. Son las pruebas posteriores a las pruebas unitarias y preceden el testeo de sistema.

En programación, se denomina cajas blancas a un tipo de pruebas de software que se realiza sobre las funciones internas de un módulo. Están dirigidas a las funciones internas. Entre las técnicas usadas se encuentran; la cobertura de caminos (pruebas que hagan que se recorran todos los posibles caminos de ejecución), pruebas sobre las expresiones lógico-aritméticas, pruebas de camino de datos (definición-uso de variables), comprobación de bucles (se verifican los bucles para 0,1 y n iteraciones, y luego para las iteraciones máximas, máximas menos uno y más uno.

Las pruebas de caja blanca se llevan a cabo en primer lugar, sobre un módulo concreto. En los sistemas orientados a objetos, las pruebas de caja blanca pueden aplicarse a los métodos de la clase, pero según varias opiniones, ese esfuerzo debería dedicarse a otro tipo de pruebas más especializadas. Es opinión de este autor, compartida con otros autores [3][5], que esta prueba no debe aplicarse a métodos polimórficos ni otros mecanismos de herencia. Este concepto también es utilizado de manera análoga en la teoría general de sistemas.

Se comprueban los caminos lógicos del software proponiendo casos de prueba que ejerciten conjuntos específicos de condiciones y/o bucles. Se puede comprobar el programa en varios puntos para determinar si el estado real coincide con el esperado. Varias son las pruebas que se pueden realizar como parte de la prueba de caja blanca y la más usada es la del camino básico. Esta técnica permite al probador obtener la medida de la complejidad de una porción de código y usar esa medida como guía para la definición de un conjunto básico (diseño de casos de prueba) de caminos de ejecución. Los casos de prueba derivados del conjunto básico garantizan que durante la prueba se ejecuta por lo menos una vez cada sentencia del programa. Para ello se construye un grafo de flujo y se calcula la complejidad ciclomática en cualquiera de sus tres variantes.

Otra de las pruebas más utilizadas es la de caja negra. En teoría de sistemas y física, se denomina caja negra a aquel elemento que es estudiado por las entradas que recibe y las salidas o respuestas que produce, sin analizar su funcionamiento interno. Se infiere entonces que en la prueba de caja negra lo importante es su forma de interactuar con el medio que le rodea, en el caso de sistemas informáticos pueden ser usuarios u otros sistemas. Por tanto, en esta prueba deben estar bien definidas sus entradas y salidas para cada uno de los casos de pruebas utilizados.

Estas pruebas se centran en los requisitos funcionales del software. Permite al probador obtener conjuntos de casos de prueba para todos los requisitos funcionales de un sistema informático. Estas pruebas se llevan a cabo sobre la interfaz del sistema reduciendo el número de casos de prueba mediante la elección de entradas y salidas válidas y no válidas que ejercitan toda la funcionalidad del sistema. Para estas pruebas también se divide el dominio de la aplicación en clases de equivalencia que permitan definir conjuntos de datos finitos. En el caso de las aplicaciones web, una herramienta muy utilizada para estas pruebas

es el selenium. Este case genera un script propio que puede ser utilizado en interfaces similares de otras aplicaciones web o de la misma aplicación. Este script también puede exportarse a varios lenguajes entre los cuales se encuentran Python, Java y C#.

La prueba de caja negra no es una alternativa a las técnicas de prueba de caja blanca sino complementaria que descubre diferentes tipos de errores que los métodos de caja blanca no pueden detectar. La prueba de caja negra intenta encontrar errores de las siguientes categorías: funciones incorrectas o ausentes, errores de interfaz, errores en estructuras de datos o en accesos a bases de datos externas, errores de rendimiento, errores de inicialización y de terminación.

Por otro lado, las pruebas de regresión se realizan después de haber corregidos los errores en procesos de depuración, carencias de funcionalidad, o divergencias funcionales con respecto al comportamiento esperado del software. Esto implica que el error tratado se reproduce como consecuencia inesperada del cambio realizado en el programa. Este tipo de cambio puede ser debido a prácticas no adecuadas de control de versiones, falta de consideración acerca del ámbito o contexto de producción final y extensibilidad del error que fue corregido (fragilidad de la corrección), o simplemente una consecuencia del rediseño de la aplicación.

Estas pruebas de regresión se pueden pueden dividir en varios tipos:

a. Clasificación de ámbito: local -los cambios introducen nuevos errores, desenmascarada- los cambios revelan errores previos, remota - los cambios vinculan algunas partes del programa (módulo) e introducen errores en ella.

b. Clasificación temporal: nueva característica - los cambios realizados con respecto a nuevas funcionalidades en la versión introducen errores en otras novedades en la misma versión del software, característica preexistente - los cambios realizados con respecto a nuevas funcionalidades introducen errores en funcionalidad existente de previas versiones.

La prueba de integración es una prueba sistemática para construir la estructura del programa. El objetivo es tomar los códigos probados en una unidad e integrarlos en una estructura de sistema que esté de acuerdo con lo que establece el diseño. Se comprueba la compatibilidad y funcionalidad de las interfaces entre las unidades que componen un sistema, estas 'partes' pueden ser módulos, aplicaciones individuales, aplicaciones cliente/servidor, entre otras.

Las pruebas de recuperación son una prueba del sistema que fuerza el fallo del software de muchas formas y verifica si la recuperación se lleva a cabo apropiadamente. Si la recuperación es automática hay que evaluar la corrección de la inicialización, de los mecanismos de recuperación del estado del sistema, de la recuperación de datos y del proceso de arranque. Si la recuperación requiere la intervención humana, hay que evaluar los tiempos medios de reparación para determinar si están dentro de límites aceptables. Este tipo de pruebas es especialmente relevante en aplicaciones distribuidas.

La prueba de seguridad verifica que los mecanismos de protección incorporados en el sistema lo protegerán de accesos impropios por parte de piratas informáticos. Durante la prueba de seguridad, el responsable (de la prueba) desempeña el papel de un individuo que desea ingresar al sistema, debe intentar conseguir las claves de acceso por cualquier medio, debe producir a propósito errores del sistema. Con tiempo y recursos suficientes, una buena prueba de seguridad terminará por acceder al sistema. El papel del diseñador del sistema es hacer que el coste de la entrada ilegal sea mayor que el valor de la información obtenida. Para esta prueba existen herramientas case que optimizan este proceso como el Vega que implementa hasta inyección SQL. Otras de las herramientas que pueden ser utilizadas son las herramientas de hacker para las funcionalidades críticas.

Asociado a la prueba de seguridad se realizan pruebas de resistencia, en ella se ejecuta un sistema de forma que demande recursos en cantidad, frecuencia o volúmenes anormales. Determinan hasta donde puede soportar el programa determinadas condiciones extremas.

Otra prueba necesaria es la de rendimiento que se realiza en tiempo de ejecución dentro del con-

texto de un sistema integrado. Consisten en determinar que los tiempos de respuesta están dentro de los intervalos establecidos en las especificaciones del sistema.

Para cada una de estas pruebas se genera una documentación asociada con formatos bien concretos que permiten almacenarlas para una posterior revisión de los procesos de desarrollo del sistema en caso de mantenimiento. Cuando se va a realizar el mantenimiento del sistema informático es necesaria toda la documentación de los procesos de desarrollo en el cual las pruebas juegan un papel fundamental.

Generalmente, como se ha abordado hasta el momento, no se concibe en la literatura consultada la detección de errores en los procesos anteriores a la codificación como la detección de los requerimientos y la modelación del sistema. Solamente en la validación o en las pruebas beta es que se detecta si se ha realizado el sistema correcto. Estas afirmaciones llevan a pensar en la necesidad de establecer pruebas a los requerimientos y a la modelación del sistema desde etapas tempranas. Una posible solución para proyectos pequeños en el caso de los requisitos es presentar en un tiempo breve un prototipo que solamente muestre la interfaz del futuro sistema. Además, este prototipo debe presentar todas las funcionalidades para que el cliente especifique si cumple con sus requisitos, pudiera tomarse como una maqueta del sistema.

En el caso de la modelación pudieran establecerse ideas de etapas posteriores como es el caso del pair programming de la metodología XP. En el caso de esta metodología ágil de desarrollo, sería interesante extender esta idea a las historias de usuario, el plan de entregas, las tarjetas CRC y el plan de iteraciones. De tal manera que al culminar cada una de ellas se intercambia con el segundo y se detectan los errores cometidos, así como su depuración de la cual se trataré posteriormente.

Para la modelación que utiliza un lenguaje de modelado como UML es posible situar un probador por cada equipo de modelación. Este probador se encargaría de examinar los modelos contenidos en cada paquete y verificar su correspondencia con los requisitos ya probados. Por tanto, puede ocupar el rol de probador de requerimientos y probador de modelos.

Posteriormente al proceso de detección de las fallas del sistema, como segunda etapa, deben comenzarse el proceso de depuración de estos fallos. Se afirma en la literatura [3][5] que la depuración es la consecuencia de pruebas exitosas. Con esta concepción de depuración se asume que puede encontrarse la causa del error o no encontrarse. En este segundo caso pudiera ser una solución la búsqueda de nuevos casos de pruebas asociado a la sospecha de dónde se encuentra el error.

Existen ciertas características de los errores brindan algunas pistas [3]:

1. El síntoma y la causa pueden ser geográficamente remotos. Es decir, el síntoma puede aparecer en una parte de un programa, mientras que la causa en realidad puede ubicarse en un sitio que esté alejado. Los componentes altamente acoplados exacerban esta situación.

2. El síntoma puede desaparecer (temporalmente) cuando se corrige otro error.

3. El síntoma en realidad puede no ser causado por errores (por ejemplo, imprecisiones de redondeo).

4. El síntoma puede ser causado por un error humano que no se rastrea con facilidad.

5. El síntoma puede ser resultado de problemas de temporización más que de problemas de procesamiento.

6. Puede ser difícil reproducir con precisión las condiciones de entrada (por ejemplo, una aplicación en tiempo real en la que el orden de la entrada esté indeterminado).

7. El síntoma puede ser intermitente, particularmente común en sistemas embebidos que acoplan hardware y software de manera inextricable.

8. El síntoma puede deberse a causas que se distribuyen a través de algunas tareas que corren en diferentes procesadores.

Los métodos de depuración más comunes son la fuerza bruta, la vuelta atrás y la eliminación de causas. El primero se basa fundamentalmente en el rastreo del código del programa con las trazas y que en algún momento se detecte una salida no esperada o no deseada. Este método conlleva un gasto de esfuerzo que generalmente no brinda los resultados esperados sobre todo en las actuales aplicaciones que se enlazan unas con otras usando las más variadas formas.

La vuelta atrás su idea fundamental se encuentra en revisar hacia atrás el código a partir del lugar donde se encontró el error. Este método puede funcionar en programas pequeños, pero en grandes sistemas no funciona eficientemente. En códigos orientados a objetos, para este artículo se asume que deben añadirse a la revisión todas las clases involucradas con aquella donde se generó el error. Para ello puede resultar de mucha ayuda contar con las tarjetas CRC o los diagramas de colaboración en dependencia de la metodología de desarrollo utilizada.

La eliminación de las causas se basa en aislar las causas potenciales a partir de una reorganización de los datos. A partir de ahí se desarrolla una lista de las posibles causas y se realizan pruebas para eliminar cada una. Si las pruebas iniciales indican que una hipótesis de causa particular se muestra prometedora, los datos se refinan con la intención de aislar el error.

También es posible utilizar herramientas automatizadas de apoyo para la depuración de los sistemas informáticos. Estas herramientas permiten capturar el error y determinar la causa para ayudar a los depuradores a corregirlos.

Siendo consecuente con el enfoque amplio de pruebas asumido en este artículo, es necesario entonces abordar la depuración. La detección de errores en los requerimientos y en la modelación de los sistemas pueden ser corregidos por el mismo equipo que los elaboró. En este caso, la actividad de depuración puede seguir las mismas técnicas que en los tradicionales, aunque deben realizarse las adecuaciones pertinentes para ello. En el caso de la vuelta atrás deberían revisarse todos los procesos modelados relacionados con el modelo donde se encontró el error para corregir posibles errores si hablamos de pruebas a los modelos. En el caso de los requisitos deberían revisarse las implicaciones que tiene los cambios en los requisitos que están relacionados con éste para corregirlos.

Los probadores de sistemas deben poseer un alto grado de experiencia en los procesos de desarrollo de sistemas e integrarlos, por lo cual hay un conjunto de conocimientos que deben integrar:

- Programación: test de unidad, auditoría de códigos.
- Ingeniería de software: Riesgo, requisitos, análisis y diseño software.
- Dirección de proyectos: La planeación, estimación, monitoreo y control de las actividades de pruebas que generalmente son ejecutadas por el jefe de pruebas.
- Matemática: la selección de datos válidos y no válidos se realiza a través de la partición del conjunto de datos en clases de equivalencia.

B. La depuración como parte de la actividad informática y el Proyecto

Como se apuntaba anteriormente, los procesos de informatización de las organizaciones transcurren desde el análisis de sus necesidades hasta que éstas se satisfacen. Durante todos estos procesos se van construyendo diferentes modelos y éstos se van concretando en la medida del proceso de informatización que se realice. Estos procesos de informatización transcurren en forma de proyectos, lo cual se asume como la forma fundamental de organización de la actividad informática.

Asumir que el proyecto es la forma fundamental de la organización de la actividad informática implica reconocer que es un proceso colaborativo en el cual intervienen un conjunto de personas. También que estas personas tienen determinadas labores que desempeñan en este proceso a las cuales se les denomina roles [3][6]; aunque algunas de ellas pueden ocupar más de uno. También se reconoce que el desarrollarlo como proyecto implica un sistema de acciones con carácter de sistema que lleve a la culminación del proceso. Este sistema de acciones generalmente comienza con las especifi-

caciones del cliente, posteriormente continúa con la modelación de los procesos de informatización, la implementación de los modelos y seguidamente la depuración del proceso que se ha realizado. Se trata de depurar los errores cometidos durante los procesos que antecedieron en la práctica, esta vez no es solamente para comprobar la veracidad del modelo obtenido sino además para completar una fase de concreción del proceso de informatización. Ejes transversales a estos procesos lo constituyen las depuraciones de los errores cometidos y el aseguramiento de la calidad de estos procesos de tal manera que se realice un proceso con calidad.

Siguiendo estas ideas, las acciones a las cuales se hace referencia constituyen parte de los procesos y una de ellas, la depuración es la encargada de encontrar errores en todos los procesos anteriores y corregirlos. De ahí que este proceso sea diferente en cada proyecto que responda a las necesidades de informatización de una organización. En proyectos de producción de sistemas, la depuración transcurre como se ha abordado en los párrafos anteriores, sin embargo, en otros proyectos no necesariamente tiene por qué ser así.

Cuando en los proyectos de informatización no se requiere solamente de la producción de un nuevo sistema, sí se introducen los modelos, pero intervienen otras acciones posteriores a este. Esta afirmación amerita un análisis detallado. Existen proyectos de informatización en los cuales se desarrolla un sistema como parte del proceso, sin embargo, este sistema es solamente una acción dentro de otras acciones. En este caso, al ejecutarse el sistema de acciones modeladas, no se codifica en su totalidad, sino que es una parte de este; cuestión esta que modifica sustancialmente las concepciones de depuración ya analizadas. Otro caso es cuando no se realiza ninguna acción de codificación, pero se llevan a la práctica diferentes modelos obtenidos durante la marcha del proceso. Un ejemplo que ya ha sido tratado en este artículo es el de VoIP. En este caso se modela la arquitectura de la red a partir de las características de la pizarra telefónica y el conjunto de elementos a conectar en dependencia de los requisitos obtenidos.

A partir de la obtención del modelo se comienza la implantación de la red configurada según los modelos y consecuentemente se hacen pruebas muy parecidas que en el caso. Cuando el sistema no se comporte de la manera esperada, se comienza a realizar las pruebas de cada una de las acciones integrantes del proceso de informatización y se depurarán en dependencia de las características de cada tipo de proyecto. Para proyectos de redes las pruebas deben ser muy parecidas a las pruebas de las aplicaciones web en cuanto a su concepción. Quiere ello decir que debe probarse los sistemas informáticos involucrados, la infraestructura que los sustenta y las políticas implantadas. Una vez detectados los errores la corrección de los mismos puede pasar por la sustitución de un equipo hasta la instalación de nuevos sistemas. Es opinión del autor, que en cada uno de los casos analizados se mantienen las etapas genéricas de desarrollo de un software por lo cual se puede generalizar a los procesos de informatización en organizaciones.

C. La depuración de procesos de informatización en el currículo del profesional informático

Al asumir que el currículo «... es un sistema integral que manifiesta relaciones de subordinación y coordinación entre sus niveles organizativos, es por ello que el perfil de egreso se constituye en el documento rector que expresa la intención formativa de la carrera» [7] es necesario destacar si la depuración se encuentra dentro del perfil de egreso de estos profesionales. Un análisis de este tipo lleva a un análisis comparativo de los currículos en las universidades de países de reconocido prestigio en la formación informática.

En un estudio realizado sobre el currículo informático en diferentes universidades [8] se llega a la conclusión que los contenidos esenciales se encuentran en algoritmos y desarrollo de sistema, sistemas informáticos y su arquitectura, formalismos informáticos y representación y modelación de la información. Como ya se ha evidenciado en el primer acápite, en estos contenidos informáticos las acciones fundamentales tienen como resultado un sistema que debe ser depurado.

Para la Universidad de Bourgogne en Francia se ha mantenido desde el 2007 programación, base de datos, ingeniería de sistemas entre otras asignaturas que conllevan a una modelación de los procesos de informatización como ya se ha analizado anteriormente. Para la Universidad de Dublín, de Nueva Delhi, el MIT entre otras las invariantes de asignaturas se han mantenido en el orden de Inge-

niería del Software, la Inteligencia Artificial, la Arquitectura del Computador, la Informática Educativa, Bases de Datos y Programación. Todas estas asignaturas del diseño curricular en la formación del profesional informático llevan implícitas la depuración como una de sus actividades fundamentales. De esta manera es comprensible que existe una total concordancia entre la informática, su forma de organización fundamental, la depuración de la informatización de procesos como uno de los resultados fundamentales como acciones para la concreción del modelo y su expresión en forma de asignaturas del currículo.

Esta afirmación en el párrafo anterior permite entonces considerar que la depuración es una de las formas de actuación del profesional informático con una gran repercusión en sus actividades profesionales. Se infiere entonces que los profesionales informáticos crean estrategias de pruebas para detectar errores y depurarlos como parte de su actividad para la informatización de los diferentes procesos organizacionales.

Al comprender la depuración como parte del modelo del profesional de la informática y una de las actividades más importantes como futuro informático en organizaciones, es necesario dilucidar si es una habilidad o una competencia. Para algunos autores, la modelación es una habilidad, pero no se refiere a la depuración [6]. Esta conceptualización de la modelación como habilidad adolece de una insuficiencia que, a juicio de este autor, la lleva a restringir su estructura para el contexto de este artículo por lo tanto la depuración como proceso siguiente también sería una habilidad. La insuficiencia de estos autores radica fundamentalmente en la contextualización de la modelación en un currículo de formación básica que posee diferencias sustanciales respecto al currículo de formación profesional en el área de informática.

Para otros autores se estructuran habilidades generalizadas que tienen como característica esencial las acciones y operaciones generalizadas con un conocimiento que también es general [9]. Sin embargo, en esta concepción se aprecia una clara concepción cognitivista al incluir solamente los conocimientos y las habilidades, dejando de lado otros de los procesos subjetivos que intervienen en la actividad profesional como la imaginación, la autorregulación, los sentidos y las vivencias que no se integran en su concepción. Además, se percibe la depuración asociada a procesos de desarrollo de sistemas dejando de lado la depuración de procesos de informatización y aunque como aspecto positivo esta se concibe como parte del proceso de desarrollo de un sistema.

El profesional del área de informática debe incorporar la depuración en su actuación como uno de los requisitos esenciales de concreción de su actividad. En esta proyección y planificación de su actividad como futuro profesional se desprenden un conjunto de cualidades esenciales como la toma de decisiones en la proyección de procesos de informatización, la honestidad y responsabilidad al asumir los errores propios y del colectivo en la depuración de sistemas, entre otras. Ello implica que la depuración rebase a la habilidad como estructura sistémica.

Para varios autores las competencias pueden ser definidas como aprendizajes o logros complejos que integran aspectos cognitivos, procedimentales, actitudinales, habilidades, características de la personalidad y valores, que puestos en práctica en un determinado contexto, tendrán un impacto positivo en los resultados de la actividad desempeñada [10].

Las competencias son estructuras psicológicas integrativas, de nivel intermedio, que complementan o articulan las funciones de las estructuras principales de la personalidad ante situaciones que demandan un desempeño determinado como expresión del comportamiento de la persona en su contexto social y en un ambiente específico de acción [11]. Asumiendo la definición del último autor se toman sus tres dimensiones de análisis de las competencias.

La necesaria diferenciación de los procesos de depuración que cada estudiante realiza y que se estructura de manera individual posibilita la expresión de sus experiencias, conocimientos y habilidades relacionadas con este proceso de llevar a vías de hecho los modelos realizados. Ello les permite integrar sus proyectos con el resto e ir aprendiendo de los demás colegas. Teniendo en cuenta estos elementos es que se aprecia en este artículo que la depuración es una competencia en el ingeniero informático y a continuación se analizará su estructura.

Se intentará demostrar que la depuración es una competencia del ingeniero informático obviando el sistema de conocimientos que ya ha sido abordado en el primer acápite de este artículo, siendo necesario abordar las habilidades dentro de esta competencia. Se pueden enunciar cinco habilidades esenciales de un profesional informático relacionada con la depuración.

Una primera se denomina en este artículo elaborar plan de pruebas que se representan el sistema de pruebas que se van a realizar utilizando el sistema de artefactos adecuados que le permite estructurar las pruebas que se han analizado hasta el momento.

La segunda se denomina elaborar la estrategia de pruebas en los procesos de informatización y llevarlos a cabo según la concepción de los encargados de estos procesos: gestor de pruebas y probador. Para ello es importante retomar el proyecto, ahora como eje articulador de los procesos formativos en los ingenieros informáticos. La tercera habilidad está relacionada con la ejecución de las pruebas para la concreción del plan y la estrategia de pruebas en la solución de la problemática planteada al proyecto.

Dentro de esta tercera habilidad, la selección de las herramientas automáticas, en caso que el proyecto los necesite, es importante para la depuración de los sistemas por variadas causas establecidas en la literatura [3][5][12][13].

La cuarta habilidad está relacionada con la selección del método de depuración de estos procesos en la práctica. La última, y no menos importante, es la ejecución de la depuración de los errores detectados en el proceso de informatización de las organizaciones. La integración de estas cinco habilidades en un sistema armónico conjuntamente con los conocimientos acerca de la depuración permitirá a los estudiantes conformar con éxito un proyecto.

La formación basada en proyectos permite al estudiante aplicar los contenidos apropiados y analizar rápidamente la pertinencia de éstos para su formación profesional. En un entorno de proyecto el estudiante desempeña los roles de su futuro profesional y va articulando los modos de actuación que desempeñará en el futuro, de tal manera que se va articulando dentro de su proyecto de vida la futura profesión que va a desempeñar. Este proceso tiene una especial relevancia en la juventud por ser una de las características fundamentales de la situación social de desarrollo en la cual se encuentran.

Sin embargo, la educación de los valores en el proyecto es también importante a tener en cuenta en este artículo. En este mismo orden de ideas, se van conformando aquellos valores propios de la profesión como la responsabilidad, comprometido, honesto y humilde [14]. La responsabilidad es uno de los valores más importantes en el profesional informático por las características propias de la actividad, así como por el impacto social que tienen las tecnologías para la sociedad. En cualquiera de los roles que deba desempeñar debe ser responsable de sus actos y constituir un verdadero eje regulador de sus acciones puesto que de ellas depende las del resto del proyecto. La depuración es una de las actividades con potencial para el desarrollo de la responsabilidad por el papel que desempeña en la concreción del proceso de informatización a desarrollar y por su marcado carácter práctico. Además, es esencial el proceso de depuración para que el proceso de informatización carezca de errores y sea eficiente.

Durante el desarrollo del proyecto la honestidad con los colegas en el proceso de depuración es primordial para entender las relaciones sociales que se establecen. Estas relaciones en las cuales el proyecto, el posicionamiento de la empresa y la confianza entre los integrantes juegan un papel fundamental.

La educación de estos valores debe basarse en el conjunto de vivencias que hacen comprender al sujeto su responsabilidad ante los demás integrantes del proyecto y la sociedad por el resultado del proyecto que van a obtener. Este proceso debe estar centrado en la importancia del rol que desempeña y el resultado de la actividad para la organización. En un ambiente de proyecto todos los roles son importantes. Juega un papel esencial la explicación detallada de los errores, así como la oportunidad de expresarlos y corregirlos sin constituir una oportunidad de castigo, a través del diálogo, la confrontación y la polémica constante y constructiva. También es importante para la depuración el ir asignando tareas de mayor compleji-

dad y que involucren con el resto de los integrantes del proyecto para explicar las especificidades de su proyecto en la organización.

La integración de estos valores, en las configuraciones que se estructuran en la actividad informática, con el proyecto de vida hace que se incorporen de manera real al potencial regulador de la personalidad del profesional. De esta manera estos valores se constituirán en parte de la subjetividad del sujeto y no generarán formalismos.

De la misma manera que los valores, se tienen en cuenta los restantes componentes de la competencia depuración de modelos. La integración de estudiante – realidad – enseñanza propicia que el trabajo de los estudiantes adquiera un carácter social tanto por la implicación de los resultados del proyecto para las organizaciones, así como el sistema de relaciones a desarrollar con el resto del colectivo en la solución de los problemas. Lo anteriormente planteado conlleva al análisis de la situación y una postura reflexiva ante las críticas y los cuestionamientos.

La transformación de la realidad por parte del estudiante a partir del proceso de informatización y la selección de las herramientas necesarias para lograrlo evidencian el carácter activo de la función reguladora de su personalidad. La amplia variedad de herramientas para una misma actividad, así como la selección de las metodologías implica que se han tomado decisiones con respecto al proyecto y los modelos que en él intervienen. Este proceso de transformación debe ocurrir primeramente en el plano mental jugando un papel importante de la imaginación, con lo que se contribuye a su desarrollo. Este es uno de los aspectos esenciales que diferencian la depuración de la modelación. En este caso, por la posición que ocupa este proceso en el desarrollo del sistema, los estudiantes aprecian con mayor claridad cómo se da el tránsito de lo abstracto a lo concreto durante la actividad informática. De esta manera los estudiantes van integrando estas vivencias de participación en actividades de generación casos de pruebas, estrategias de pruebas entre otras actividades al mismo tiempo que van regulando el aprendizaje de la depuración en la actividad informática.

A partir de la búsqueda de problemas en la realidad se comienza el ciclo de vida de un software hasta que concluye con la puesta punto y mantenimiento. Los problemas derivados del proyecto individual motivan a los estudiantes hacia su solución y en ellos se encuentran los conocimientos del curso que a su vez generan las situaciones problémicas para los demás estudiantes. Las acciones y operaciones asociadas a la depuración, a partir de la utilización del conjunto de símbolos y signos asociados a ello, y el trabajo conjunto con el resto de colegas de mayor experiencia van estructurando diferentes escenarios de su futuro perfil profesional.

En cada encuentro, a partir de la interacción previa entre estudiante – profesor – grupo, se determinan las situaciones problémicas para los restantes estudiantes. Es el profesor quien decide la situación problémica a presentarse en el encuentro basándose previamente en la interacción grupal y el desarrollo de los proyectos de los estudiantes. La toma en cuenta de las situaciones problémicas asociadas a la depuración, y su solución, desde un proyecto les posibilita que se tengan en cuenta sus metas, proyectos y expectativas. En la medida que estas situaciones problémicas se estructuren y se concatenen con su práctica los lleva a reflexionar sobre esta práctica y mejorarla. En este mejoramiento continuo se va creciendo como profesional, propicia que se continúe el aprendizaje de sí mismo y de su aprendizaje de manera que propicie su regulación.

Para muchos autores, la formación de habilidades en la informática se da en el momento de enseñar un sistema en particular [8] [9][14][15]. En este artículo se asume una concepción sistémica propuesta por varios autores [16-20]. En esta concepción de la enseñanza de la informática se aborda el proceso como un todo integrado, en el cual se le presta especial atención a la concatenación de los conceptos y procedimientos informáticos que no se pueden formar en una única clase como en el caso que ocupa este artículo: la competencia depurar procesos de informatización a organizaciones. Por el carácter interdisciplinar y la complejidad de su estructura, su formación en el profesional informático debe abordarse desde una postura sistémica.

Fortalecer el enfoque de sistema en la enseñanza de la informática significa establecer agrupamientos de los contenidos de la Informática a partir de los cuales se exprese la concatenación de estos

que puedan establecer lineamientos generales para organizar su enseñanza. El trabajo integrador y sistemático que se propone garantiza una sólida formación informática en el estudiante como un elemento importante para el desarrollo de la competencia implementar procesos de informatización.

Aplicar el enfoque de sistema conlleva al análisis de los sistemas de aplicación porque anteceden a la enseñanza de la programación en la preparación informática de los estudiantes. Una de las cuestiones en la cual realiza énfasis la enseñanza de la Informática, por la importancia que reviste para esta ciencia, es el procesamiento de la información. La depuración de los procesos de informatización y sus flujos puede estructurarse desde los inicios de la formación informática y en las diferentes disciplinas informáticas que componen la formación del profesional de esta ciencia.

Para comprender la posición en este artículo sobre el desarrollo de la competencia implementar procesos de informatización es importante asumir diferentes posiciones. Existen hoy dos criterios divergentes en cuanto la formación informática y las disciplinas de programación e ingeniería de software. Algunos plantean que es preferible comenzar por la enseñanza de la programación y otros por la ingeniería del software. Cuando la enseñanza de la informática se estructura sobre la base del proyecto como eje formativo la ingeniería de software alcanza prevalencia pues es la proyección de lo que se quiere alcanzar. Es la disciplina que provee de los símbolos y sus relaciones que permiten lograr las primeras 4 habilidades necesarias para la comprensión de la representación del proceso a informatizar. Se forma entonces las habilidades representar procesos y estructuras y comprender representaciones que fueron definidas anteriormente. De esta manera se prepara al estudiante desde el aprendizaje para la futura actividad profesional que va a realizar. Estos procesos son necesarios que el estudiante los conozca antes de realizar un proceso de depuración en la concepción amplia sustentada en este artículo.

Asumiendo que la Ingeniería de Software deba anteceder a la programación es importante destacar que las Bases de Datos deben comenzar a enseñarse desde la introducción de la Ingeniería de Software para determinar el modelo entidad relación desde el dominio de la aplicación. De esta manera el estudiante se apropia de las formas de trabajo propias de cada asignatura, pero tributando a la modelación y a la depuración y a la estrecha relación que existe entre estas disciplinas informáticas. La integración de estas dos asignaturas en el proyecto le provee al estudiante de los elementos necesarios para realizar una primera modelación del análisis de la aplicación que le resultará necesario para comprender este proceso e implementarlo.

Posteriormente, al refinar este primer modelo de análisis se puede comenzar a enseñar programación orientada a objetos desde los inicios. El paradigma imperante es la programación orientada a objetos y se puede comenzar a analizar los conceptos de clase, objeto, herencia y polimorfismo. Posteriormente se abordan los conceptos de algoritmo, variable y código a partir de estas clases y los métodos que en ellas se incluyen. Este proceder metodológico propicia la integración de conocimientos tal y como transcurre en un proceso de desarrollo de software. Siguiendo este orden de ideas, la evaluación final de las asignaturas debe ser integrada para evaluar precisamente los objetivos de cada una de las asignaturas y la integración de ellas para resolver un proyecto real.

Cuando se enseña programación entonces se comienza a trabajar desde la modelación del sistema informático usando cualquiera de los elementos estructurales de los dos grandes grupos de metodologías: pesadas y ágiles para posteriormente lograr estructurar coherentemente la depuración tal y como se concibe en este artículo.

En este proceso se va desarrollando la habilidad comprender representaciones. El estudiante va analizando las representaciones realizadas por otros, relaciona los símbolos expresados que posee de su modelo y los va concatenando para la representación en el plano mental del proceso o estructura a informatizar para posteriormente implementarlo.

Ya en este momento, el estudiante puede estructurar mejor su accionar en función de ejecutar las acciones contenidas en el modelo, como parte del proceso de depuración dentro de su esfera de actuación como profesional de la informática. Este análisis vuelve a situar al estudiante ocupando los roles de su futuro como profesional. Sin embargo,

el tratamiento de la depuración no acaba con la enseñanza de la programación.

Para este artículo la competencia implementar procesos de informatización es una de las más sistémicas e integradoras de la actividad profesional informática. Constituye un eje central en dos roles de la actividad informática: gestor de pruebas, probador y gestor de proyecto. Además, juega un papel esencial como concreción de los modelos obtenidos para las acciones de informatización de procesos. Por ende, la formación de competencia depurar procesos de informatización comienza en el segundo año de la carrera y culmina cuando el estudiante expresa su proceso de investigación en forma de memoria escrita en el cual integra todos los modelos estudiados en la carrera para describir el proyecto y los implementa. Ya el proceso de desarrollo de esta competencia corresponde a su ámbito laboral a partir de las diversas problemáticas que este profesional debe resolver en el ámbito organizacional.

III. Conclusiones

La competencia depurar procesos de informatización constituyen el resultado de la actividad modelar que se desarrolla en el proyecto informático y establece las pautas a seguir en el proceso de informatización que se quiere lograr y su concreción en la práctica del ingeniero informático. Estos procesos depuración en la informatización de organizaciones constituyen un acercamiento en la actividad del equipo de desarrollo y el cliente para satisfacer las necesidades de este último.

En el currículo del profesional informático la depuración juega un papel esencial e integra varias de las disciplinas que se conciben en el proceso de formación de este profesional. Asumir la formación de esta competencia implica cambios en la concepción curricular para potenciar en el estudiante los modos de actuación del profesional informático.

La estructura de la competencia implementar procesos de informatización implica asumir el enfoque de sistema en la enseñanza de la informática para su formación durante la carrera. Conjuntamente con este enfoque se asume el aprendizaje basado en proyectos como la vía esencial para formación de la competencia pues en él se integran los modos de actuación del profesional.

Referencias

[1] I. M. S. Jimenez, L. Barroca, E. F. Barbosa & E. A. Júnior. Learning Design for Software Engineering Courses. Paper presented at the International Conference on Computer Supported Education, Barcelona, España, 2014.

[2] B. A. Naranjo. Calidad del software educativo: Metodología de Evaluación de software educativo para determinar el que cumple con las especificaciones basadas en estándares internacionales. (Licenciatura en Educación). Retrieved from http://reposital.cuaed. unam.mx., 2015.

[3] R. S. Pressman. Software engineering: A practitioner´s approach. (Seventh Edition ed.). New York: McGraw-Hill, 2010.

[4] R. A. Vera & J. P. Pech. Developing Virtual Learning Environments for Software Enginnering Education: a ludic proposal. Paper presented at the Proceedings of EDULEARN15 Conference, Barcelona, Spain, 2015.

[5] R. Black & J. L. Mitchell. Advanced Software Testing. Guide to the ISTQB Advanced Certification as an Advanced Technical Test Analyst. USA: O'Reilly Media. 2011.

[6] J. Segura & W. González. La habilidad modelar multimedia en los procesos formativos de los Joven Club. Didasc@alia: Didactica y Educación, 6(2), 2015.

[7] J. C. Arteaga Vera. Algunas reflexiones en torno al perfeccionamiento del diseño curricular de la Carrera de Ingeniería en Sistemas de la Universidad Laica Eloy Alfaro de Manabí. Revista Electrónica Formación y Calidad Educativa, 3(1), 2015.

[8] F. Mulder, K. Lemmen & M. Veen. Variety in views of university curriculum schemes for informatics / computing / ICT A comparative assessment of ICF-2000 / CC2001/ Career Space. Paper presented at the ICTEM 2002 Florianopolis.

[9] R. B. Jiménez. A. D. Barrera & L. E. Hernández. Algunas consideraciones en torno al desarrollo de habilidades profesionales del ingeniero informático y el rol de la comprensión de texto en la modelación de algoritmos computacionales. Revista Científico Metodológica Mendive, 50. 2015.

[10] M. C. Núñez. El docente en el enfoque por competencias. Pensamiento, Papeles de Filosofía, 1(1), 2013.

[11] D. Hernández. Formación para el desarrollo de proyectos de vida reflexivos y creativos en los campos social y profesional. Revista Crecemos Internacional, 5(2), 2015.

[12] R. Lacerda. Proposta de um modelo para análise de requisitos de software educativo. (Programa de Pós-Graduação em Educação - PPGE), Universidade de Brasília - UnB, Brasilia, 2007.

[13] M. Marques de Lima, A. Ribeiro de Lima, A. C. Coêlho, E. H. Cavalcante & L. Leal. Uma Revisão Sistemática da Literatura dos Processos de Desenvolvimento de Software Educativo. Paper presented at the Simpósio Brasileiro de Informática na Educação Rio de Janeiro, 2012.

[14] E. B. de Castro & M. A. D. de Sá. Habilidades, Competências, Valores e Atitudes-Um Perfil Para o Profissional de Computação e Informática. Paper presented at the Anais do Congresso da Sociedade Brasileira de Computação, Florianópolis, Brasil, 2002.

[15] M. Gutiérrez. Una metodología para contribuir al desarrollo de la habilidad resolver problemas en la disciplina Lenguajes y Técnicas de Programación, en estudiantes de la carrera de Licenciatura en Educación, especialidad de Informática. (Doctor en Ciencias Pedagógicas), Universidad Pedaógica Enrique José Varona, Academia de Ciencias de Cuba, 2012.

[16] E. López. El aprendizaje intencional y los entornos informatizados, medios para el desarrollo de las habilidades metalingüísticas: un paso hacia adelante en el área de didáctica de la lengua y la literatura. . In U. d. Murcia (Ed.), Enseñanza e Informática (Vol. 4). España: Editorial de la Universidad de Murcia, 2010.

[17] W. González. Creativity Development in Informatics Teaching Using the Project Focus. International Journal of Engineering Pedagogy (iJEP), 3(1), pp. 63-70. 2013.

[18] W. González. Intuition as Part of Informatics Creativity. iJEP, 3(3), 7. 2013.

[19] W. González, V. Estrada & M. Martínez. El enfoque de sistema en la enseñanza de la Informática para el desarrollo de la creatividad Revista Enseñanza Universitaria, 32, 45-56, 2006.

[20] W. González. La intuición informática: un acercamiento a su studio. Revista Ingeniería, Matemáticas y Ciencias de la Información. 3, 99-109, 2016.