Generación y Actualización automática de un e-learning basado en una definición de proceso SPEM-compatible

Carlos Bertoni, Natalia Andriano, Diego Rubio

Laboratorio de Investigación en Ingeniería y Calidad de Software
http://www.institucional.frc.utn.edu.ar/sistemas/lidicalso/
Departamento de Ing. en Sistemas de Información
Universidad Tecnológica Nacional
Maestro M. López esq. Cruz Roja Argentina
(X50165ZAA) Ciudad Universitaria, Córdoba, Argentina
{51429, nandriano, drubio}@sistemas.frc.utn.edu.ar

Resumen. En la actualidad los equipos de ingeniería de software necesitan tener acceso permanente a una amplia gama de información. En este contexto las organizaciones y Universidades se encuentran con el desafío constante de generar entrenamientos que permitan reflejar las últimas prácticas existentes. Por ello, el presente trabajo forma parte de la propuesta de generar un entorno de aprendizaje activo que, utilizando la información actualizada del entorno, genera *e-learnings* basados en simulaciones permitiendo que la información se encuentre disponible en el momento que el estudiante la necesita, personalizada a su contexto de aplicación (proceso de su compañía) y actualizada tanto con la información disponible en la empresa como en la industria circundante y en los modelos y estándares aplicables. En particular, se presenta una experiencia piloto enmarcada en el contexto de esta investigación con la finalidad de validar la hipótesis de que *´es posible el desarrollo de un generador automático de e-learning que sea capaz de interpretar un proceso previamente definido´*.

Palabras claves: Proceso definido, *SimSe*, simulación, e-learning.

1 Introducción

En la actualidad, la industria de desarrollo de software está enfocada a la mejora de la calidad de sus productos. Para ello, muchas empresas se han comprometido en la implementación de marcos de mejora de procesos (SEI, 2010) (ISO, 2002), buscando alcanzar la mejora de la calidad del software que producen. (Rico, 2004) (McGibbon, Ferens, & Vienneau., 2007). Para lograr un producto de software, no basta tener un proceso definido sino es debidamente ejecutado (Sommerville, I. 2010). Por lo tanto, la calidad del producto dependerá tanto de la calidad del proceso como de la calidad de la ejecución del mismo (Humphrey, 1989). La ejecución del proceso es realizada por las personas que componen la empresa, y el nivel de entendimiento que las personas tengan del proceso junto con sus habilidades personales provocarán variaciones en la calidad de la ejecución (Laycock, 2005) (Rolland, 1995). Es de esperar, que si se mejora el entendimiento y conocimiento del proceso además de las habilidades personales, se obtendrán resultados beneficiosos en términos de calidad del producto final (Miller, 2001). Un medio efectivo para lograr la mejora del conocimiento y entendimiento del proceso de desarrollo junto a las prácticas de

ingeniería, es el entrenamiento del personal. Por décadas las formas tradicionales de entrenamiento (Nien-Lin Hsueh) (O.J. Dahl, 1972) (Ambler, 2009) y las nuevas metodologías de desarrollo de software han dado buenos resultados, sin embargo de la mano de nuevas tecnologías emergentes se hace evidente que nuevas formas alternativas más rápidas y más efectivas para entrenar a las personas sean necesarias. Una forma alternativa es a través de las tecnologías de simulación basadas en e-learnings (Goldschneider:2009).

La palabra E-learning abarca todas las formas electrónicas que dan soporte al aprendizaje y a la enseñanza (Berman P:2006). Por su parte las Simulaciones proveen una representación interactiva de la realidad que permite a los estudiantes probar y descubrir cómo funciona o cómo se comporta un fenómeno, qué lo afecta y qué impacto tiene sobre otros fenómenos. El uso de este tipo de herramienta educativa alienta al estudiante para que manipule un modelo de la realidad y logre la comprensión de los efectos de su manipulación mediante un proceso de ensayo-error (Eduteka, 2010-2011)

En este contexto tanto las organizaciones como las Universidades se encuentran con el permanente desafío de brindar los conocimientos actualizados necesarios para el correcto desempeño de los profesionales de esta industria. Asimismo, la cantidad de conocimiento disponible y necesario para desarrollar y mejorar la calidad del software generado, plantea un desafío extra hacia las metodologías de enseñanzas a utilizar, requiriendo cada vez más de metodologías que no sólo puedan presentar el conocimiento al estudiante en el momento que lo necesita sino también simulando las situaciones reales que el mismo deberá enfrentar permitiéndole construir su conocimiento a partir del entendimiento actualizado pre-existente en las diversas organizaciones(Goldschneider:2009).

Es por ello que el presente trabajo forma parte de la generación de una alternativa de entorno de generación de entrenamiento que cumpla con las siguientes características:

1. Disponible en el momento que el estudiante lo necesita.
2. Actualizado (con el menor esfuerzo y costo posible) tanto con la información disponible en la empresa como en la industria circundante.
3. Contextualizado a la situación actualmente enfrentada por el estudiante mediante una simulación de la misma.

En particular, se presenta una experiencia piloto enmarcada en el contexto de esta investigación con la finalidad de validar la hipótesis de que *´es posible el desarrollo de un generador automático de e-learning que sea capaz de interpretar un proceso definido en un lenguaje SPEM-Compatible y generar/actualizar un e-learning basado en simulación* (Gonzalez et al.: 2009)

Para ello, este documento se estructura de la siguiente manera: la sección 2 “Estado del arte” describe el contexto en el cual el trabajo ha sido desarrollado; la sección 3 “Exploración y Análisis” define los pasos seguidos en la obtención del conocimiento general de software de simulación y de la definición de las reglas y validaciones definidas; sección 4 “Experimentación” define cómo se configura el generador del modelo (Model Builder); sección 5 “Demonstración e Implementación” presenta los resultados obtenidos; sección 6 “Conclusiones” muestra el significado de los resultados obtenidos y las conclusiones finales del trabajo; sección 7 “Trabajos Futuros y agradecimientos” plantea los pasos a seguir en la investigación y agradecimientos a colegas por su aporte al trabajo presentado.

2. Estado del arte

En el año 2007 y con el objetivo de generar "Un entorno de aprendizaje activo de ingeniería de software basado en la integración Universidad-Industria" se comenzó a trabajar en un programa de investigación y desarrollo que generará los elementos necesarios para la implementación del mismo (Rubio et al:2010).
Debido al tamaño y cantidad de esfuerzo asociado, se dividió el trabajo en 3 etapas que a la vez fueron asociadas a respectivos proyectos de investigación encarados como parte del programa. Cada proyecto se planeo inicialmente con una duración de 2 años con 11, 7 y 8 investigadores participando en ellos respectivamente.
La figura 1 presenta una visión general del entorno planteado y las etapas necesarias para su construcción:

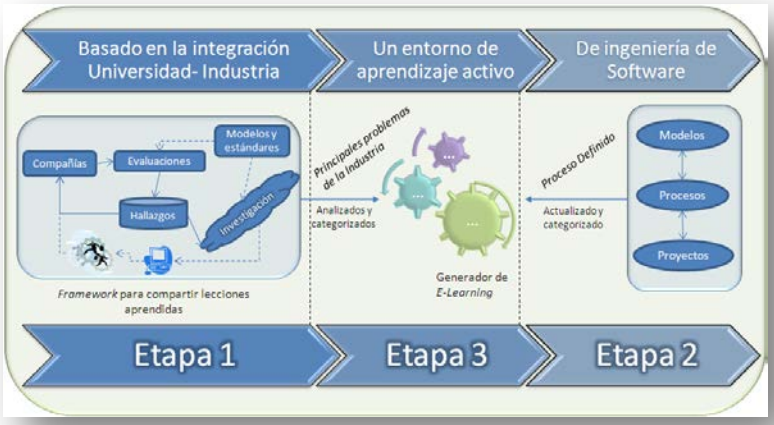

Figura 1: Resumen del entorno propuesto

① **Etapa 1:** En la primera etapa se trabajó en la confección de un marco de trabajo que permitiera obtener la información más relevante de la industria y de la universidad a modo de integrar esta en los entrenamientos propuestos. Para ello se trabajó en el desarrollo de un modelo integrado de colección de información relevante (Rubio et al: 2008:) (Gonzalez et at: 2009).
② **Etapa 2:** Debido a la necesidad de personalizar y contextualizar los contenidos del entrenamiento propuesto, en la segunda etapa se trabajó en el desarrollo de un meta modelo que permitiera, no sólo interpretar el proceso de desarrollo de software de una organización, sino también mantenerlo permanentemente actualizado con las últimas incorporaciones en los modelos de calidad y procesos de referencia, tal como CMMI® (CMMI: 2008) o la norma ISO9001:2008 (ISO9001: 2008) entre otros. A su vez, también se trabajó para obtener un mapeo automático o de mínimo mantenimiento entre la información obtenida en la etapa 1 con la información particular de los procesos sobre los cuales se generaría el entrenamiento (Szyrko et at: 2009).
③ **Etapa 3:** Por último, se planeó una tercera etapa encargada de definir tanto los criterios para la generación de un entorno de aprendizaje activo que utilice las mejoras prácticas de *e-learning* disponibles (Goldschneider: 2009) como las interfaces y desarrollos necesarios para integrar los contenidos obtenidos en las dos

etapas iniciales; generando, como consecuencia de ello, "Un entorno de aprendizaje activo de ingeniería de software basado en la integración Universidad-Industria".

Debido a la existencia de investigaciones pre-existentes en la temática de la generación de *e-learning* para ingeniería se Software, se decidió analizar las alternativas existentes a los fines de automatizar la generación y actualización en base a estos. Como resultado se decidió utilizar para la realización de nuestro trabajo un software de simulación basada en aprendizajes *open source* llamado *SimSe*. *SimSe* permite a los estudiantes practicar el proceso de ingeniería de software de manera "virtual", interactiva y divertida en la que la retroalimentación gráfica les permite aprender las relaciones de causa y efecto complejas que existen en los procesos de desarrollo de software (Oh Navarro: 2004). *SimSe* es un juego por simulación educacional de ingeniería de software que utiliza un enfoque de modelado de procesos de software. Este enfoque combina tanto los aspectos predictivos como los prescriptivos para dar soporte a la creación dinámica, interactiva y gráfica de modelos para la educación de procesos de ingeniería del software. A su vez presenta diversos modelos de ciclos de vida como el cascada, incremental, XP y muchos otros (*SimSe*:2009).

3. Exploración y Análisis

Para la realización de la exploración fue necesario el uso de un proceso previamente definido, el cual sirviera de entrada para la generación del *e-learning*. Para ello se utilizó el proceso definido por una empresa del medio para realizar sus actividades de desarrollo de software, sólo abarcando para este piloto las actividades referentes al sub-proceso de "planeación de sprint".
Este sub-proceso cuenta con los siguientes pasos, a saber:

1. Calcular la capacidad del equipo
2. Seleccionar los elementos del producto, llamados elementos de la pila de producto (*product backlog ítems*), de ahora en más PBIs.
3. Establecer la meta u objetivo del sprint
4. Descomponer los distintos en elementos del producto en elementos más pequeños y manejables, llamados elementos de la pila del sprint (*sprint backlog ítems)*, de ahora en más SBIs.
5. Realizar estimaciones para cada uno de los SBIs
6. Revisar el plan de entrega y actualizarlo si fuera necesario.

El proceso tiene definidos 3 (tres) elementos importantes por cada actividad, a saber:

1) Artefactos de entradas (obligatorios y opcionales)
2) Artefactos de salida (obligatorios y opcionales)
3) Roles (obligatorios y opcionales)

Para la realización de este piloto se tomaron en cuenta los siguientes escenarios:

Reglas para artefactos
El usuario debe seleccionar artefactos obligatorios de entradas para la actividad que esté realizando. La simulación sumará puntos en base a los artefactos correctos seleccionados para la actividad en cuestión. Si el usuario seleccionase artefactos que no corresponden a la actividad que se está ejecutando la simulación restará puntos.

Reglas para roles participantes en la actividad
Lo mismo que para los artefactos obligatorios, el usuario sumará puntos a medida que seleccione el/los roles correctos en la participación de una actividad en particular; y se restarán puntos en el caso de una selección incorrecta.

En ambos casos, cuando el usuario selecciona solo una parte de los elementos requeridos para la ejecución de una actividad, ya sean artefactos o roles, la simulación calculará el porcentaje de completitud de la respuesta dada por el usuario, para otorgar el puntaje a la respuesta.

4. Experimentación

SimSe permite la definición personalizada de un proceso de desarrollo de software a través de una herramienta creada para tal fin llamada el *Model Builder*. Dicha herramienta oculta completamente el lenguaje de implementación de la simulación y provee una interfaz gráfica para la especificación de: tipos de objetos, estado de inicio, acciones, reglas y gráficos para el modelo. Una vez que el modelo está especificado, *Model Builder* genera un código *Java* para ejecutar un juego de simulación personalizado basado en el modelo dado. (Oh Navarro Emily, 2005). La figura 2 muestra la interfaz gráfica provista por el *Model Builder*.

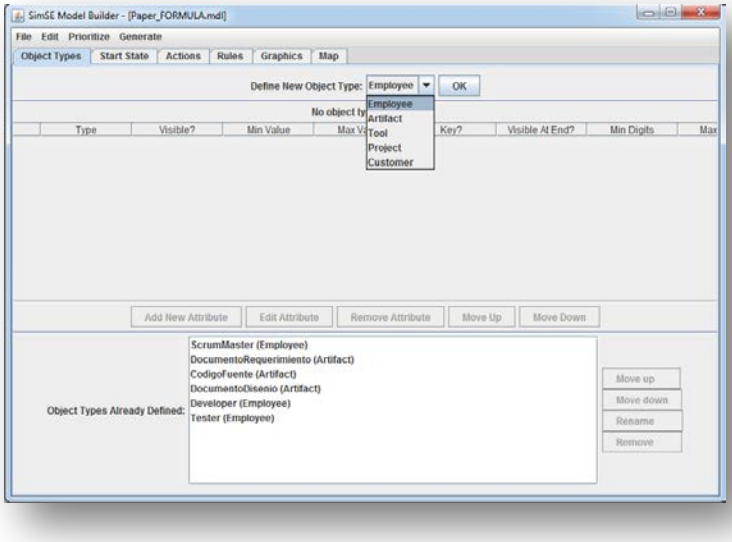

Figura 1: Interfaz gráfica *Model Builder*.

Si cada actividad de un proceso es definida haciendo uso de los siguientes atributos:

- Entradas de la actividad
- Salidas de la actividad
- Roles participantes
- Actividades predecesoras

Y si se conoce el impacto de la presencia/ausencia de elementos en la ejecución de una actividad, la cantidad necesaria/requerida de cada elemento y si el mismo es obligatorio u opcional para la ejecución de la actividad, es posible simular dicho proceso haciendo uso de *Model Builder* como herramienta de generación de un *e-learning* para dicho proceso.
En el caso de los artefactos, que serán entradas o salidas de los procesos, se seleccionaron los siguientes atributos como los mínimos necesarios para poder representar un proceso:

Tabla 1. Atributos definidos para artefactos

Atributo	*Tipo*	*Valor min.*	*Valor max.*	*Visible?*	*Clave?*
Nombre	String	N/A	N/A	Si	Si
Completitud	Double	0	100	Si	No
Calidad	Double	0	100	Si	No
CalidadActividad1	Double	0	100	No	No
CompletitudActividad1	Double	0	100	No	No
(...)	Double	0	100	No	No
CalidadActividadN	Double	0	100	No	No
CompletitudActividadN	Double	0	100	No	No

Los atributos nombre, calidad y completitud son valores visibles para el jugador y servirán como retroalimentación del avance de la ejecución de la actividad de acuerdo con las decisiones tomadas. Durante la ejecución del proceso, las diferentes actividades modificarán los valores de estos para darle retroalimentación al usuario.
La calidad y completitud por actividad permiten que cada actividad marque la correcta/incorrecta ejecución de la misma, eso permitirá que las diferentes actividades del proceso puedan comunicarse y así calcular la calidad y completitud de sus salidas, siendo afectadas las mismas por cómo fue ejecutada la simulación.

Para las actividades presentes en el proceso, se deben definir acciones que serán ejecutadas por el usuario de la simulación. A cada una de estas acciones se le definen tres grupos de participantes, entre los cuales tendremos, las entradas, las salidas y los participantes propiamente dichos (roles).
Por otro lado, las reglas permitirán modelar la forma en que la ejecución de las acciones modificará los atributos de los objetos y así permitirá que las distintas actividades se comuniquen entre sí y se simule el proceso.
Para realizar esto debe crearse una "*Effect Rule*" por cada acción y colocar en ella los efectos de dicha regla en cada uno de los participantes de la acción. El ejecutar una acción causará que las reglas modifiquen los atributos de los artefactos de salida para dicha acción.

Para modelar el proceso se utilizarán formulas matemáticas que relacionen los atributos de los objetos. Estas fórmulas fueron definidas en base a (Rubio, 2012)

- **Fórmula de selección base:** Esta fórmula se utiliza para conocer la pertinencia de la selección realizada por el usuario al momento de ejecutar una actividad del proceso dentro del juego y seleccionar los participantes para la misma. Esta fórmula no afecta valores de los artefactos y solo es creada para poder ser referenciada por las formulas de calidad y completitud.

$$SeleccionBase = \sum_{j=tipoParticipanteJ} PesoParticipante_j * \left(\sum_{i=participanteItipoJ} \frac{Obligatoriedad_i * CantSeleccionada_i}{CantOpcionesObligatorias_i}\right) \quad (1)$$

- **Fórmula de calidad:** Esta fórmula se utiliza para simular la calidad que tendrán las salidas de una actividad, basándose en la calidad de las entradas seleccionadas (salidas de sus actividades predecesoras) y en la pertinencia de la selección de participantes (entradas, salidas y roles) para la actividad actual.

$$Calidad = PesoPredecesor * \left(\frac{\sum_{i=entradaObligatoriaI} CalidadEntradaObligatoria_i}{CantidadEntradasObligatorias}\right) + PesoSeleccion * SeleccionBase * 100 \quad (2)$$

- **Fórmula de completitud:** Esta fórmula se utiliza para simular la velocidad con la que se completarán los artefactos de salida de una actividad, basándose en la completitud de las entradas (salidas de sus actividades predecesoras) y en la pertinencia de la selección de participantes (entradas, salidas y roles) para la actividad actual.

$$Completitud = Completitud + \left[PesoPredecesor * \left(\frac{\sum_{i=entradaObligatoriaI} CompletitudEntradaObligatoria_i - CantidadEntradasObligatorias * 100}{100} + \frac{\sum_{i=entradaObligatoriaI} CompletitudEntradaObligatoria_i}{CantidadEntradasObligatorias * 100}\right) + PesoSeleccion * SeleccionBase\right] \quad (3)$$

Cabe destacar que las formulas contemplan además la correcta ejecución de las actividades en cuanto a precedencia se refiere, dado que el no ejecutar una actividad o ejecutarla en el orden incorrecto, causará que las entradas no posean buena completitud/calidad y esto afectará la completitud/calidad de las salidas de la actividad en cuestión.

5. Demostración e implementación

Como es de esperarse, la simulación debe terminar y entregar al usuario un resultado que evalúe la ejecución del proceso. El resultado de esta evaluación se basará en la calidad de las salidas de las diferentes etapas del proceso, y para ellos podemos utilizar uno de dos modos

- **Evaluación por artefacto final:** En caso de tener un artefacto único que sea la salida de la última actividad del proceso, podremos utilizar la calidad de este artefacto como un indicador de la correcta ejecución del proceso, debido a que este atributo fue afectado por la calidad de todos sus predecesores.
  El juego finaliza cuando la última actividad es completada y la puntuación obtenida de la simulación es un número entre 0 (cero) y 100 (cien) dado por la calidad de la salida de esta actividad. Por lo que seleccionaremos dichos elementos en la condición de la salida y cálculo del puntaje final.

- **Evaluación por promedio:** En caso de tener un proceso más complejo en el cual exista una ramificación de dependencias entre actividades e incluso varios puntos y artefactos de finalización, podremos utilizar un nuevo artefacto que no sea utilizable por el usuario durante el juego y que permita el cálculo del promedio de la calidad de los productos.

Para realizar este método de evaluación deberemos comenzar creando un nuevo artefacto Proyecto con los siguientes atributos:

Tabla 2. Atributos definidos para el artefacto Proyecto.

Atributo	*Tipo*	*Valor min.*	*Valor max.*	*Visible?*	*Clave?*
Nombre	String	N/A	N/A	Si	Si
PuntuacionActividad1	Double	0	100	Si	No
(…)	Double	0	100	Si	No
PuntuaciónActividadN	Double	0	100	No	No
PuntuaciónTotal	Double	0	100	No	No

Cada una de las actividades actualizará la puntuación del proyecto que le corresponda haciendo uso de la calidad de sus salidas y recalculará el total utilizando la siguiente fórmula:

$$PuntuacionTotal = \frac{\sum_{ActividadI} PuntuacionActividadI}{CantidadDeActividades} \quad (4)$$

Por último se deberá marcar como actividad de finalización a todas las actividades que potencialmente finalizan el proceso y seleccionar como puntaje de evaluación la *PuntuacionTotal* del proyecto.

6. Conclusiones

Tal como se desprende de las secciones anteriores, el modelo propuesto nos permitió sistematizar la generación de e-learning (y sus actualizaciones), generando pasos automatizables.

En trabajos anteriores (Gonzalez et al.: 2009) (Rubio et al: 2008) (Rubio et al: 2010) se presentaron desarrollos que permiten tomar un proceso definido en lenguaje SPEM-compatible y generar automáticamente un metamodelo de mapeo de componentes. En este trabajo se demostró la factibilidad de sistematizar el procesamiento de dicho metamodelo para la generación y actualización de e-learning. Como modelo de ejemplo, se utilizó *SimSe* como motor de generación del e-learning. El mismo fue seleccionado debido no sólo a sus reconocimientos en la industria como modelo de simulación para cursos de Ingeniería de Software sino también por su alta puntuación respecto al modelo de evaluación de E-learning generado en trabajos anteriores (*SimSe*:2009) y a su compatibilidad con el "modelo para generación de e-learnings" desarrollado en (Rubio: 2010). Se presentó y desarrollo un ejemplo completo de la automatización de un sub-proceso, tomando el mismo de una empresa del medio sin ninguna adaptación para el ejercicio con la finalidad de comprobar su aplicabilidad a modelos generados por la Industria local.

Como conclusión, este trabajo nos permite validar la hipótesis originalmente planteada y afirmar que *´es posible el desarrollo de un generador automático de e-learning que sea capaz de interpretar un proceso definido en un lenguaje SPEM-Compatible y generar/actualizar un e-learning basado en simulación´*.

7. Trabajos futuros y agradecimientos.

En este momento nos encontramos en el proceso de la mejora y codificación del software de generación de e-learning basado en las premisas descriptas en este trabajo. Como trabajo inmediato futuro se plantea la generación y despliegue de una simulación completa del proceso de desarrollo de la empresa seleccionada y su implementación piloto para poder evaluar la adecuación y recepción de *SimSe* como modelo de presentación del mismo.

Asimismo, se plantean trabajos paralelos focalizados en optimizar el modelo de cálculo del peso relativo de los valores expresados en este modelo para poder actualizar automáticamente los mismos a través de nuestro marco de trabajo para compartir lecciones aprendidas descripto en la sección 1 de este documento.

Por último, nos gustaría destacar que el trabajo presentado en esta publicación fue desarrollado como parte de la investigación en curso en el laboratorio de investigación y desarrollo en ingeniería y calidad de software perteneciente a la "Universidad Tecnológica Nacional Córdoba". Muchas personas además de los autores participaron en las investigaciones asociadas. En particular, nos gustaría agradecer a Paula Izaurralde y Claudio Gonzalez por sus contribuciones al presente trabajo.

8. Bibliografía

1. Ambler, S. W. (2009). Agile Modeling (AM) Home Page Effective Practices for Modeling and Documentation. Retrieved from Ambysoft Copyright 2001-2009 : http://www.agilemodeling.com/
2. Berman, P. (2006.). "E-learning concepts and techniques - instructional strategies for e-learnings". Retrieved from Institute for Interactive Technologies, Bloomsburg University of Pennsylvania, USA: http://iit.bloomu.edu/Spring2006_eBook_files/chapter5.htm
3. (CMMI: 2008) Software Engineering Institute (SEI). Capability Maturity Model Integration Version 1.1 (CMMI-SE/SW/IPPD/SS, V1.1). CMU/SEI-2002-TR-011 - ESC-TR-2002-011. Pittsburgh, Pennsylvania, USA . CMMI Product Team. Marzo 2002.
4. Eduteka. (2010-2011). Simulaciones. Retrieved from http://www.eduteka.org/instalables.php3
5. Goldschneider (Goldschneider:2009) , B. (n.d.). e-learning Best Practices. Retrieved from http://www.syberworks.com/articles/bestpractices.htm
6. Gonzalez, Claudio (Gonzalez et al.: 2009); Izaurralde, Paula M.; Marzo, Luciano; Rubio, Diego M. Experiencia de la Aplicación de Aprendizaje Activo en un Marco Universidad-Empresa. IV Congreso de Tecnología en Educación y Educación en Tecnología (TE&ET'09). La Plata – Argentina. Julio 2009. (En Línea). http://teyet.info.unlp.edu.ar/archivos/Articulos_Aceptados.pdf
7. ISO, I. O. (2002). "ISO9001:2000 Quality management systems -- . Requirements. s.l.: ISO copyright office, 2002. ICS 01.04|0.03.
8. (ISO9001: 2008) International Organization for Standarization. ISO9001:2008 Sistemas de gestión de la calidad - Requisitos. s.l. ICS 01.040.03. ISO copyright office. En línea. 2008.
9. Laycock, M. (2005). Collaborating to compete: achieving effective knowledge sharing in organizations. . The Learning Organization: Emerald Group Publishing Limited. DOI:10.1108/09696470510626739.

10. McGibbon, T., Ferens, D., & Vienneau., R. L. (2007). A Business Case for Software Process Improvement (2007 Update). s.l. : Measuring Return on Investment from Software Engineering and Management. DACS Report Number 347616.
11. Miller, J. (2001). The Internet's Impact On Business Relationships. . Retrieved from Information Week. [Online] Sears, Roebuck and Co,: http://www.informationweek.com/news/management/showArticle.jhtml?articleID=650662
12. Nien-Lin Hsueh, W.-H. S.-W.-L. Applying UML and software simulation for process definition, verification, and validation. . 897-911.
13. O.J. Dahl, E. W. (1972). Structured Programming. London: Academic Press.
14. Oh Navarro Emily, v. d. (2004). Software Process Modeling for an Educational Software Engineering Simulation Game. Retrieved from http://www.ics.uci.edu/~emilyo/papers/SPIP2004.pdf
15. Oh Navarro, Emily Oh Navarro, E. (2005); van der Hoek, André. "Design and Evaluation of an Education Software Process Simulation Environment and Associated Model," In Proceedings of the Eighteenth Conference on Software Engineering Education and Training. Ottawa, Canada: IEEE, 2005. Web Page: http://www.ics.uci.edu/~emilyo/papers/CSEET2005-2.pdf
16. Oh Navarro, E. (2009). “An educational, Game Based Software Engineering Simulation Enviroment”. Retrieved from University of California, Irvine. : http://www.ics.uci.edu/~emilyo/SimSE/
17. Rico, D. F. (2004). ROI of Software Process Improvement (Foreword by Roger S. Pressman). s.l. : J. Ross Publishing, Inc., ISBN: 1-932159-24-X.
18. Rolland, C. S. (1995). An Approach for Defining Ways-of-Working, Information Systems. Oxford : Elsevier Science Ltd.
19. Rubio, Diego M (Rubio et al: 2008); Andriano, Natalia V.; Ruiz de Mendarozqueta, Álvaro; Bartó, Carlos. An integrated improvement framework for sharing assessment lessons learned. Congreso Argentino en Ciencias de la Computación (CACIC) Universidad Nacional de Chilecito. La Rioja – Argentina. (En Línea) http://cacic2008.undec.edu.ar/
20. Rubio et al (2010) “Libro “La tecnología educativa al servicio de la educación tecnológica: Experiencias e investigaciones en la UTN” - Capítulo “Un entorno de aprendizaje activo de ingeniería de software basado en la integración Universidad-Industria”. http://issuu.com/edutecne
21. Rubio, D. (2012). Implementación de un sistema generador de e-learning para el desarrollo de software (UTN1168). Retrieved from http://www.institucional.frc.utn.edu.ar/sistemas/lidicalso/pub/file/Investigaciones/Investigacion_PID_Elearning.pdf
22. SCRUM Alliance. (n.d.). Retrieved from http://www.scrumalliance.org/
23. SEI, S. E. (2010). CMMI Product Team. “CMMI for Development, version 1.3”. Pittsburgh, Pennsylvania, USA: CMU/SEI-2010-TR-033.
24. SIMSE. P. C. (2009). Engineering Pathway. . Retrieved from http://www.k-grayengineeringeducation.com/blog/index.php/2009/10/22/classroom-presenter-is-the-2009-premier-courseware-award-winner/
25. Sommerville, I. (2010). Software Engineering (9th Edition). Madrid: Addison Wesley.
26. Szyrko, Pablo (2009); Silclir, Mauricio; García Favre, Gonzalo; Rubio, Diego; Cohen, Diego; Angeloni, Romina. Un modelo de validación automático para la definición y mantenimiento de procesos de desarrollo de software XI Workshop de Investigadores en Ciencias de la Computación (WICC). San Juan – Argentina. Mayo 2009. (En Línea). http://www.wicc2009.com.ar/
27. Watts S. Humphrey, M. I. (1989). Software Process Modeling: Principles of Entity Process Models. Pennsylvania: ACM.