

TESIS DEFENDIDA POR
José Gamaliel Rivera Ibarra
Y APROBADA POR EL SIGUIENTE COMITÉ

M. en C. Josefina Rodríguez Jacobo
Director del Comité

Dr. Jesús Favela Vara
Miembro del Comité

Dr. David Hilario Covarrubias Rosales
Miembro del Comité

Dr. Pedro Gilberto López Mariscal
*Coordinador del programa de posgrado en
Ciencias de la Computación*

Dr. David Hilario Covarrubias Rosales
*Encargado del Despacho de la Dirección
de Estudios de Posgrado*

16 de octubre de 2008

**CENTRO DE INVESTIGACIÓN CIENTÍFICA Y DE EDUCACIÓN SUPERIOR
DE ENSENADA**



**PROGRAMA DE POSGRADO EN CIENCIAS
EN CIENCIAS DE LA COMPUTACIÓN**

**MODELO DE DESARROLLO DE COMPETENCIAS PARA INGENIEROS DE
SOFTWARE**

TESIS

que para cubrir parcialmente los requisitos necesarios para obtener el grado de
MAESTRO EN CIENCIAS

Presenta:

JOSÉ GAMALIEL RIVERA IBARRA

Ensenada, Baja California, México, Octubre de 2008.

RESUMEN de la tesis de **JOSÉ GAMALIEL RIVERA IBARRA**, presentada como requisito parcial para la obtención del grado de **MAESTRO EN CIENCIAS** en **CIENCIAS DE LA COMPUTACIÓN**. Ensenada, Baja California, Octubre de 2008.

MODELO DE DESARROLLO DE COMPETENCIAS PARA INGENIEROS DE SOFTWARE

Resumen aprobado por:

Dra. Josefina Rodríguez Jacobo
Director de Tesis

Una de las preocupaciones principales de las organizaciones desarrolladoras de software es desarrollar el talento de su capital humano (los ingenieros de software), ya que la calidad e innovación en sus productos y servicios dependen en gran medida del conocimiento y la capacidad del ingeniero de software para realizar el proceso de desarrollo de software.

Una forma de desarrollar el talento humano es a través del desarrollo de competencias. Las competencias son el conjunto de conocimientos, habilidades y comportamientos que el ingeniero de software utiliza para realizar su trabajo de forma sobresaliente. El objetivo principal en este trabajo de investigación es analizar y diseñar un modelo que guíe el desarrollo de las competencias del ingeniero de software.

El desarrollo de competencias del ingeniero de software es un tema relativamente poco explorado. Para realizar el diseño del modelo fue necesario obtener información de diversas áreas del conocimiento como la administración de recursos humanos, psicología y desarrollo organizacional, para aplicarla al contexto específico de la ingeniería de software.

Además del diseño del modelo de desarrollo de competencias, se realizó el análisis y diseño de un sistema de software que proporcione soporte al desarrollo de competencias.

Palabras clave: competencias, marco de competencias, talento, ingeniero de software, desarrollo profesional, capacitación.

ABSTRACT of the thesis presented by **JOSÉ GAMALIEL RIVERA IBARRA** as a partial requirement to obtain the **MASTER OF SCIENCE** degree in **COMPUTER SCIENCES** Ensenada, Baja California, Mexico, October of 2008.

COMPETENCY DEVELOPMENT MODEL FOR SOFTWARE ENGINEERS

One of the main concerns of the software industry is to develop the talent of its human resources (software engineers), since the quality and innovation of its products and services depend to a great extent on the knowledge, the ability and the talent that software engineers apply in the software development process.

One way to develop the human talent is through the competency development. Competencies are the set of knowledge, skills, and behaviors that software engineer uses to excel in the performance of their job functions. The objective of this thesis is to analyze and design a model to guide the competency development of software engineers.

The competency development of software engineers is a relatively unexplored topic. To carry out the model design, it was necessary to obtain information from several areas such as human resources, psychology, and organizational development to implement it on the specific context of software engineering.

Additionally to the design of the competency development model, we analyzed and designed a software system that provides support to competency development.

Keywords: competencies, competency framework, talent, software engineer, professional development, training.

A las personas que dan sentido a mi vida

Mamá, papá, Sergio y Angélica

Mi novia Kelvia

Primos, tíos y abuelos

“Todo lo que necesitas es amor.”

John Winston Lennon

A mi directora de tesis:

Dra. Josefina Rodríguez Jacobo

A los miembros del comité de tesis:

Dr. Jesús Favela Vara

Dr. David Hilario Covarrubias Rosales

Agradecimiento especial al:

Centro de Investigación Científica y Educación Superior de Ensenada (CICESE)

Consejo Nacional de Ciencia y Tecnología (CONACYT)

“Largo es el camino de la enseñanza por medio de teorías; breve y eficaz por medio de ejemplos.”

Lucio Anneo Séneca

CONTENIDO

	Página
Resumen en español	i
Resumen en inglés	ii
Dedicatoria	iii
Agradecimientos	iv
Lista de figuras	ix
Lista de tablas	xiii
Capítulo I. Introducción	1
<i>I.1 Antecedentes</i>	1
<i>I.2 Planteamiento del problema</i>	6
<i>I.3 Objetivos</i>	8
I.3.1 Objetivo general.....	8
I.3.2 Objetivos específicos.....	8
<i>I.4 Alcances</i>	9
<i>I.5 Relevancia</i>	10
<i>I.6 Contenido</i>	11
Capítulo II. Marco teórico	14
<i>II.1 Desarrollo de software centrado en la persona</i>	14
II.1.1 Introducción.....	14
II.1.2 Modelos del proceso de software.....	15
II.1.3 Actividades y roles del desarrollo de software.....	19
<i>II.2 Competencias</i>	21
II.2.1 Introducción.....	21
II.2.2 Fundamentos.....	22
II.2.3 Marco de competencias.....	27
<i>II.3 Obsolescencia profesional</i>	28
II.3.1 Introducción.....	28
II.3.2 Fundamentos.....	35
II.3.3 Modelos de obsolescencia.....	37
<i>II.4 Desarrollo profesional</i>	34
II.4.1 Introducción.....	34
II.4.2 Fundamentos.....	35
II.4.3 Factores que influyen en el desarrollo profesional.....	37
II.4.4 Métodos de capacitación.....	43

CONTENIDO (continuación)

	Página
Capítulo III. Marco conceptual.....	47
<i>III.1 Introducción.....</i>	<i>47</i>
<i>III.2 Conceptos.....</i>	<i>48</i>
<i>III.3 Estructura.....</i>	<i>49</i>
<i>III.4 Objetivos del modelo conceptual.....</i>	<i>51</i>
<i>III.5 Alcances.....</i>	<i>52</i>
<i>III.6 Trabajo relacionado.....</i>	<i>52</i>
<i>III.7 Descripción de los elementos del modelo.....</i>	<i>56</i>
III.7.1 Condición de interés.....	56
III.7.2 Factores.....	56
III.7.3 Representación gráfica.....	64
<i>III.8 Forma de uso.....</i>	<i>65</i>
III.8.1 Escenario 1: conseguir empleo.....	66
III.8.2 Escenario 2: conservar el empleo.....	68
III.8.3 Escenario 3: progresar en la misma organización.....	70
III.8.4 Escenario 4: cambiar a otra organización.....	71
Capítulo IV. Marco de competencias.....	74
<i>IV.1 Introducción.....</i>	<i>74</i>
<i>IV.2 Conceptos.....</i>	<i>75</i>
<i>IV.3 Descripción.....</i>	<i>76</i>
IV.3.1 Criterios de elección.....	76
IV.3.2 Actividades y roles.....	78
IV.3.3 Competencias.....	80
IV.3.4 Escala.....	86
IV.3.5 Representación.....	86
Capítulo V. Estrategia.....	88
<i>V.1 Introducción.....</i>	<i>88</i>
<i>V.2 Proceso de cambio.....</i>	<i>91</i>
V.2.1 Descripción.....	91
V.2.2 Actividades.....	96
<i>V.3 Proceso de adaptación del marco de competencias.....</i>	<i>107</i>
V.3.1 Descripción.....	107
V.3.2 Actividades.....	110
<i>V.4 Proceso de desarrollo de competencias.....</i>	<i>116</i>
V.4.1 Descripción.....	116
V.4.2 Descripción de las actividades.....	118

CONTENIDO (continuación)

	Página
Capítulo VI. Análisis y diseño de la herramienta de soporte.....	127
<i>VI.1 Introducción.....</i>	<i>127</i>
<i>VI.2 Análisis de herramientas similares.....</i>	<i>129</i>
<i>VI.3 Análisis de requerimientos.....</i>	<i>136</i>
VI.3.1 Introducción.....	136
VI.3.2 Descripción general.....	137
VI.3.3 Requerimientos específicos.....	142
VI.3.4 Validación de requerimientos.....	146
<i>VI.4 Análisis de casos de uso.....</i>	<i>147</i>
VI.4.1 Introducción.....	147
VI.4.2 Representación de los casos de uso.....	147
VI.4.3 Casos de uso.....	150
<i>VI.5 Diseño del sistema.....</i>	<i>155</i>
VI.5.2 Arquitectura del sistema.....	156
VI.5.3 Diagrama de clases.....	159
VI.5.4 Diccionario de datos.....	161
VI.5.5 Diseño de la base de datos.....	164
<i>VI.6 Descripción de la funcionalidad.....</i>	<i>166</i>
<i>VI.7 Casos de prueba.....</i>	<i>184</i>
VI.7.1 Diseño de los casos de prueba.....	184
Capítulo VII. Diseño del experimento.....	190
<i>VII.1 Introducción.....</i>	<i>190</i>
<i>VII.2 Objetivos del experimento.....</i>	<i>191</i>
<i>VII.3 Variables.....</i>	<i>191</i>
VII.3.1 Variables independientes.....	191
VII.3.2 Variables dependientes.....	191
<i>VII.4 Validez interna.....</i>	<i>192</i>
<i>VII.5 Hipótesis.....</i>	<i>194</i>
<i>VII.6 Características del experimento.....</i>	<i>196</i>
VII.6.1 Unidades experimentales.....	196
VII.6.2 Intención de uso.....	197
VII.6.3 Instrumentos de medición.....	200
VII.6.4 Procedimiento experimental.....	203
VII.6.5 Plan de ejecución de la prueba para valorar la intención de uso.....	203
<i>VII.7 Análisis de los datos.....</i>	<i>204</i>

CONTENIDO (continuación)

	Página
Capítulo VIII. Discusión, aportaciones y trabajo futuro.....	205
<i>VIII.1 Discusión.....</i>	205
<i>VIII.2 Aportaciones.....</i>	210
<i>VIII.3 Trabajo futuro.....</i>	211
Referencias.....	212
Anexo A. Formato para el plan de desarrollo de competencias.....	220
Anexo B. Instrumento de valoración de la intención de uso.....	222
Anexo C. Diagrama entidad relación de la base de datos.....	225

LISTA DE FIGURAS

Figura	Página
1. Componentes básicos para el modelado del proceso de software (versión original en Acuña et al. 2001).....	18
2. Modelo del Iceberg de la competencia (adaptado de Spencer et al., 1993).....	26
3. Modelo de los factores directos de la obsolescencia profesional (versión original en Blanton et al., 1998).....	31
4. Modelo teórico de la respuesta del individuo a la amenaza de obsolescencia profesional (versión original en Tsai et al., 2004).....	32
5. Modelo de los factores involucrados en la obsolescencia de habilidades (versión original en Fossum et al., 1986).....	34
6. Fases del modelo sistemático de la capacitación y desarrollo profesional.....	36
7. Representación de los componentes del modelo de desarrollo de competencias para ingenieros de software (ubicación del modelo conceptual).....	51
8. Arquitectura del P-CMM (versión original en Curtis et al., 2001).....	53
9. Representación gráfica del modelo conceptual del desarrollo de competencias.....	65
10. Representación de los componentes del modelo de desarrollo de competencias para ingenieros de software (Marco de competencias).....	75
11. Modelo del desarrollo de software desde la perspectiva técnica.....	79
12. Modelo del desarrollo de software desde la perspectiva social.....	80
13. Representación de los componentes del modelo de desarrollo de competencias para ingenieros de software (Estrategia).....	90
14. Proceso general del cambio basado en la persona que propone [Rodríguez, 2003] (versión original en Rodríguez, 2003).....	91

LISTA DE FIGURAS (continuación)

Figura	Página
15. Representación gráfica de las actividades del proceso de cambio.....	94
16. Estrategia y guía para el desarrollo y uso de competencias que propone el Departamento de Servicios Civiles del Estado de Nueva York (EUA) (versión original en Sinnott et al., 2002).....	108
17. Representación gráfica de las actividades del proceso de adaptación del marco de competencias.....	110
18. Modelo basado en competencias para mejorar el desempeño de David Dubois (versión original en Dubois, 1993).....	117
19. Representación gráfica de las actividades que conforman la estrategia de adquisición de competencias.....	118
20. Representación de los componentes del modelo de desarrollo de competencias para ingenieros de software (Herramienta de soporte).....	128
21. Mapa del sitio web (herramienta de soporte).....	139
22. Diagrama del caso de uso administrar usuarios.....	150
23. Diagrama del caso de uso administrar competencias.....	152
24. Caso de uso administrar cuestionarios.....	153
25. Caso de uso administrar planes.....	154
26. Diagrama de la arquitectura del sistema SAAC.....	157
27. Diagrama de la arquitectura del sistema SAAC (Modelo Vista Controlador).....	158
28. Diagrama UML de las clases del sistema SAAC.....	160
29. Diagrama Entidad/Relación de la base de datos del sistema SAAC.....	165
30. Agregar nueva competencia.....	166

LISTA DE FIGURAS (continuación)

Figura	Página
31. Agregar información sobre nueva competencia (datos generales).....	167
32. Agregar información sobre nueva competencia (criterios).....	168
33. Enlace para ingresar al diccionario de competencias.....	169
34. Subsección diccionario de competencias.....	169
35. Ejemplo de descripción detallada de competencia.....	170
36. Agregar nuevo cuestionario.....	171
37. Agregar información sobre nuevo cuestionario.....	172
38. Subsección de encuestas del sistema SAAC.....	173
39. Responder una encuesta.....	173
40. Subsección autoevaluar competencia.....	174
41. Responder autoevaluación de competencia.....	174
42. Responder segunda etapa de autoevaluación.....	175
43. Despliegue del resultado de autoevaluación de competencia.....	176
44. Subsección perfiles de competencias de puestos.....	177
45. Agregar información sobre el perfil de competencias del puesto (datos generales).....	177
46. Agregar información sobre el perfil de competencias del puesto (competencias).....	178
47. Subsección detección de necesidades de capacitación del sistema SAAC.....	179
48. Subsección de foros del sistema SAAC.....	180
49. Información sobre los temas del foro de cooperación seleccionado.....	180

LISTA DE FIGURAS (continuación)

Figura		Página
50.	Despliegue de los mensajes de un tema de un foro de cooperación.....	181
51.	Responder un mensaje en un tema de un foro de cooperación.....	182
52.	Subsección plan del sistema SAAC.....	183
53.	Agregar información sobre el plan de desarrollo de competencias.....	183
54.	Relación entre casos de uso, casos de prueba y el plan de pruebas (versión original en Leffingwell et al., 2003).....	185

LISTA DE TABLAS

Tabla	Página
I. Investigaciones sobre el desarrollo de software centrado en la persona.....	15
II. Roles que realiza el ingeniero de software en el proceso de desarrollo de software.....	19
III. Comparación entre la clasificación de los factores que afectan la obsolescencia profesional que realizan [Blanton et al., 1998] y [Tsai et al., 2004].....	33
IV. Etapas profesionales y principales características (adaptado de Fernández Losa, 2002).....	35
V. Niveles de madurez y áreas de procesos del P-CMM (versión original en Curtis et al., 2001).....	54
VI. Categorías de las áreas de procesos (versión original en Curtis et al., 2001).....	55
VII. Escalas para medir el grado de desarrollo de una competencia específica...	86
VIII. Marco de competencias para ingenieros de software.....	87
IX. Correspondencias entre fases de la estrategia de [Rodríguez-Jacobo, 2003] y las actividades de la estrategia de cambio que se propone.....	95
X. Resumen de la actividad conocer y concientizar a la organización.....	98
XI. Resumen de la actividad crear clima organizacional de actualización.....	101
XII. Resumen de la actividad planear programa de desarrollo de competencias.....	102
XIII. Resumen de la actividad evaluar al programa de desarrollo de competencias.....	104
XIV. Resumen de la actividad proporcionar seguimiento al programa de desarrollo de competencias.....	106
XV. Resumen de la actividad comunicar resultados.....	107

LISTA DE TABLAS (continuación)

Tabla	Página
XVI. Resumen de la actividad identificar competencias.....	112
XVII. Resumen de la actividad definir diccionario de competencias.....	113
XVIII. Resumen de la actividad desarrollar marco de competencias.....	115
XIX. Resumen de la actividad definir perfil de puestos.....	116
XX. Características generales del sistema Cenzanne Software.....	129
XXI. Características generales del sistema Evacom.....	130
XXII. Características generales del sistema SnowDrop.....	131
XXIII. Características generales del sistema TPMG.....	132
XXIV. Características generales del sistema Winper.....	132
XXV. Características generales del sistema Mid-market.....	133
XXVI. Características generales del sistema RealPath.....	134
XXVII. Comparación de las características de los sistemas y el sistema que se propone.....	135
XXVIII. Ejemplo de formato para la validación de requerimientos.....	147
XXIX. Variables no tomadas en cuenta por el experimento.....	194
XXX. Descripción de las afirmaciones del instrumento agrupadas por constructor.....	201

I.1 Antecedentes

La demanda de sistemas basados en software por parte de las distintas organizaciones, es una de las razones por la cual el desarrollo de software se convierte en una de las actividades económicas más importantes de la actualidad y la de más alto crecimiento a nivel internacional en los últimos años [Mochi Alemán, 2006]. Sin embargo, la industria del software enfrenta problemas como proyectos mal planeados, incumplimiento en la fecha de entrega, errores en el cálculo del presupuesto, errores en la funcionalidad del sistema, documentación inapropiada y otros inconvenientes que provocan la inconformidad del cliente.

Para afrontar este tipo de problemas, una parte de la investigación sobre ingeniería de software se concentra en desarrollar estándares, metodologías y tecnologías que permitan mejorar la calidad del proceso de desarrollo [Green *et al.*, 2005; Umarji, 2005; Khamisani *et al.*, 2006; Niazi, 2006]. El objetivo de estas investigaciones es lograr la mejora del proceso de software (MPS), ya que al utilizar un proceso de alta calidad se debe obtener como resultado un producto de software también de alta calidad [Humphrey, 1989; Glass, 2002]. No obstante, los múltiples avances que se alcanzan gracias a los resultados de estas investigaciones, estos no son suficiente para evitar que un alto porcentaje de los proyectos fracasen [Klappholz *et al.*, 2003].

Para que las organizaciones que desarrollan software puedan alcanzar un alto nivel de productividad y de calidad en el producto de software, deben considerar por lo menos tres aspectos fundamentales: *procesos, tecnologías y personas* [Vakaslahti, 1997; CMMI, 2006]. La calidad del software no depende únicamente de los procesos y de las tecnologías que se utilizan para su elaboración, sino que también influye el conocimiento, la capacidad y el talento que tiene el ingeniero de software para aplicar los conceptos de ingeniería, interactuar con la tecnología y con grupos de trabajo heterogéneos [Perry, 1994; Glass, 2002; Sharp *et al.*, 2005; Khamisani *et al.*, 2006].

A pesar que a través de diversas investigaciones se reconoce la importancia del ingeniero de software, la mayor parte de los investigadores insisten en mejorar las herramientas, técnicas y procesos, antes que a las personas [Perry, 1994; Glass, 2002; Allison 2007]. Si bien, contar con buenos procesos es importante para obtener software de alta calidad, es el ingeniero de software quien proporciona el verdadero valor a los procesos a través del conocimiento y talento que utiliza para ejecutarlos y la destreza que tiene para manejar las herramientas que le suministran el soporte que necesita en la ejecución de los procesos.

El impacto que tiene el ingeniero de software en el desarrollo de software lo resaltan algunos investigadores, por ejemplo Glass [2002] escribe el libro “*Facts and Fallacies of Software Engineering*”, en él describe los principales mitos y realidades sobre la ingeniería de software. En una de las secciones de su libro, la cual dedica a las realidades que están relacionadas con las personas en la ingeniería de software indica lo siguiente:

“El factor más importante en el desarrollo de software es la calidad de los programadores” [Glass, 2002].

De manera similar, Tucker [Tucker, 2004] describe con palabras contundentes la importancia que tiene el factor humano en el desarrollo de software, él dice:

“Incluso con las mejores metodologías para el desarrollo y la sofisticación tecnológica, la dimensión humana en el desarrollo de sistemas y software sigue siendo el elemento clave para el éxito” [Tucker, 2004].

Los párrafos anteriores son sólo dos ejemplos que indican la importancia del ingeniero de software. Sin embargo, ésta la reconoce incluso el Instituto de Ingeniería de Software (Software Engineering Institute, SEI), que años después de haber desarrollado el Modelo de Madurez de Capacidad (Capability Maturity Model, CMM), se percató que éste carece de una sección dedicada a las personas en el desarrollo de software y es entonces que diseña el Modelo de Madurez de Capacidad de las Personas (People-CMM o P-CMM), aunque éste último todavía no tiene la misma popularidad que CMM [Glass, 2002].

El P-CMM [Curtis *et al.*, 2001] es un marco de referencia que sirve para establecer las mejores prácticas en la administración de recursos humanos, administración del conocimiento y desarrollo organizacional, con la finalidad de alcanzar la mejora continua en los procesos de administración y desarrollo de las capacidades del personal de la organización. Las organizaciones pueden utilizar el P-CMM con dos objetivos principales: a) como estándar para evaluar las prácticas relacionadas con el personal o b) como guía para planear o implementar actividades de mejora continua del personal.

Pasi Vakaslathi y otros [Vakaslathi *et al.*, 1997] realizan un estudio sobre el P-CMM en el cual comentan que se requiere mayor investigación sobre cómo interpretar y adoptar las prácticas del modelo P-CMM en un ambiente operativo para lograr un acercamiento simplificado con una apropiada administración de recursos.

El interés del SEI por la mejora continua de las capacidades del ingeniero de software también se hace notar a través del Proceso Personal de Software (Personal Software People, PSP) desarrollado por Watts Humphrey [Humphrey, 1995]. El PSP define un conjunto de buenas prácticas que el ingeniero de software adopta con la finalidad de cumplir ciertos lineamientos que exige el CMM y así alcanzar la MPS, la idea es que al mejorar las prácticas a nivel individual también se favorece la MPS [Disney *et al.*, 1998].

Una característica importante del PSP es que indica al ingeniero de software “cómo” realizar las prácticas que define, ya que es común que los modelos de calidad y de mejora del proceso de software se distingan por servir como un marco de referencia que define “qué” se debe hacer, pero sin indicar “cómo” hacerlo [Humphrey, 1999; Allison, 2007].

Algunas investigaciones [Rainer *et al.*, 2002; Niazi, 2006; Allison, 2007] sugieren que el hecho que los modelos de calidad indiquen únicamente el qué y no el cómo, provoca que el ingeniero de software muestre resistencia al cambio en sus prácticas y que esto conduzca al fracaso de las iniciativas para la MPS, ya que se le dice al ingeniero de software qué hacer, pero no se le proporcionan métodos que digan cómo pueden hacerlo.

Se observa que cuando las organizaciones intentan establecer un programa de MPS, éste suele complicarse porque dependen de la actitud de las personas involucradas. El cambio que producen las iniciativas para la MPS es un elemento importante y por lo general éste no se toma en cuenta como parte de estos modelos de calidad. Se debe considerar que mejorar implica cambiar, sin embargo el cambio lo interpreta de manera diferente cada persona, su interpretación depende de la percepción, sentimientos y pensamientos hacia éste, la idea que cada individuo se forme acerca del cambio determina la manera en cómo reacciona ante éste. Esos sentimientos están condicionados por la historia personal, procesos biológicos, antecedentes y todas las experiencias sociales en la vida del individuo. También el ambiente laboral influye por medio de los patrones y normas de comportamiento [Humphrey, 1999; Davis *et al.*, 2007].

A pesar que para mejorar se requiere cambiar, no siempre cambiar significa mejorar. Para que la mejora ocurra de manera efectiva, el cambio debe surgir de la propia persona. Por lo general, las organizaciones que adoptan iniciativas para la mejora continua, lo hacen principalmente como estrategia de mercado y no como una necesidad o deseo de mejorar. Cuando una organización establece un programa de mejora continua, se exige a las personas que cambien la forma en la cual realizan su trabajo, esta imposición de cambio normalmente provoca una actitud de *resistencia al cambio* en las personas involucradas. Por el contrario, cuando es la persona quien detecta la necesidad de mejorar, el cambio se realiza de manera sencilla (considerando ciertos factores como el apoyo y compromiso de ambas partes, la organización y la persona, entre otros). Una vez que el ingeniero de software logra detectar la necesidad de mejorar, es entonces cuando acude a la organización (la cual ofrece su apoyo) para determinar qué y cómo mejorar [Rodríguez-Jacobo, 2003].

Por otro lado, es importante mencionar que la mayoría de los problemas en ingeniería de software no son tecnológicos sino de naturaleza sociológica [DeMarco *et al.*, 1999; Sharp *et al.*, 2005] y que son relativamente pocas las investigaciones que estudian el desarrollo de habilidades blandas (del término anglosajón *soft skills*) en el ingeniero de software [Khamisani *et al.*, 2006]. Las habilidades blandas son parte de las “*competencias*”, el cual es un término que se utiliza en el área de recursos humanos para referirse al conjunto de conocimientos, habilidades y comportamientos que el profesional utiliza y que le permiten tener un desempeño sobresaliente en su actividad laboral. Es decir, la competencia del ingeniero de software está compuesta por sus conocimientos y habilidades técnicas así como por sus habilidades blandas y comportamientos.

A menudo se piensa que la ingeniería de software es una actividad puramente técnica, pero la actividad técnica y la social están entrelazadas y son mutuamente influyentes: “*los aspectos sociales surgen de la actividad técnica, y la actividad técnica influye en la social*” [Sharp *et al.*, 2005]. El desarrollo de software se caracteriza por ser un trabajo principalmente intelectual [Hanne *et al.*, 2004], en el cual ocurren un conjunto de interacciones del tipo técnico-psicosocial, por lo que el ingeniero de software además de conocimiento y habilidad sobre los aspectos técnicos, debe tener la capacidad para trabajar en equipo, bajo presión, servir al cliente, comunicación, colaboración, resolver conflictos, pensar críticamente y otras competencias que rara vez se intentan desarrollar, pero que son fundamentales para su buen desempeño [Rodríguez-Jacobo, 2003; Tucker, 2004; Dawes *et al.*, 2007; Morneau *et al.*, 2007].

Debido a que se considera al desarrollo de software una rama de la ingeniería, se tiene una imagen del ingeniero de software como un profesional centrado en actividades meramente técnicas, incluso su formación académica y su capacitación en el trabajo se centran en el desarrollo de conocimientos y habilidades con un enfoque principalmente técnico. A pesar de que el dominio sobre aspectos técnicos es importante, en la actualidad el ingeniero de software enfrenta escenarios que le exigen actividades interdisciplinarias e interpersonales, por lo que éste además debe desarrollar habilidades blandas, como el trabajo en equipo y cooperación, que le permitan lograr un efectivo desempeño laboral [Tucker, 2004].

Generalmente, el ingeniero de software tiende a evitar la capacitación que se enfoca en el desarrollo de destrezas interpersonales y de comunicación, se argumenta que tal inversión de tiempo y recursos no es tan benéfica como la inversión en el desarrollo de una nueva destreza técnica. Al mismo tiempo, es difícil encontrar un ingeniero de software quien no haya experimentado conflictos con los miembros del equipo, malentendidos con los jefes o confusión con el cliente o usuario final. Conflictos como los que se mencionan pueden deberse principalmente a aspectos interpersonales como a la comunicación no efectiva, en lugar de deficiencias en aspectos técnicos. El ingeniero de software que logra reconocer la raíz de estas dificultades, está mejor preparado para manejar problemas efectivamente y como consecuencia poder destacar con un mejor trabajo técnico [Tucker, 2004].

Así para que una organización pueda aspirar a producir software de calidad, además de contar con buenos procesos y tecnologías, es importante tener ingenieros de software competentes, que tengan el conocimiento que requieren para asumir diferentes roles de la ingeniería de software, la capacidad y la habilidad para aplicar el conocimiento en las actividades del desarrollo de software y además la habilidad que le permita relacionarse con otros, colaborar y trabajar en equipo, trabajar bajo presión, etc. Sin embargo, generalmente el ingeniero de software al egresar de su formación académica no tiene las competencias que requiere para trabajar en la industria [Khamisani *et al.*, 2006] y si a esto se agrega que la industria del software se caracteriza por ser un área dinámica, en la que constantemente surgen nuevas tecnologías, cambios organizacionales, entre otros, la actualización y desarrollo continuo de competencias durante la vida laboral del ingeniero de software es importante.

I.2 Planteamiento del problema

En la industria del software, además de que se compite en el mercado de los productos de software, también se compite por el talento de las personas [Curtis *et al.*, 2001]. No obstante, es posible desarrollar el talento humano a través de competencias [Alles, 2005]. Para diseñar un modelo de desarrollo profesional que permita mejorar y desarrollar las

competencias (o el talento) del ingeniero de software es importante considerar los aspectos siguientes:

- *Mejora basada en las necesidades.* Generalmente, los planes de desarrollo profesional del ingeniero de software se basan en la oferta de cursos que existe en el mercado y no en las competencias que necesita para desempeñar de manera sobresaliente su función laboral [Fernández Sánchez, 2001; Fernández Losa, 2002].
- *Desarrollo de habilidades blandas.* La mayoría de los cursos de capacitación que recibe el ingeniero de software se centran en proporcionarle conocimientos y habilidades técnicas, pocos se enfocan en desarrollar habilidades blandas que faciliten y mejoren su desempeño laboral [Turley *et al.*, 1995; McGuire *et al.*, 1998; Kandeel *et al.*, 2001; Mouthaan *et al.*, 2003; Rodríguez-Jacobo, 2004; Khamisani *et al.*, 2006; Morneau *et al.*, 2007].
- *Manejo del cambio.* Se identifica que la resistencia al cambio es uno de los principales factores que afecta la implantación de los programas de mejora continua, a pesar de esto, los modelos de mejora continua no tratan el tema del cambio o no ofrecen un modelo que permita llevar a cabo el manejo del cambio [Dubois, 1993; Moitra, 1998; Stelzer *et al.*, 1998; Rodríguez-Jacobo, 2003; Mathiassen *et al.*, 2005; Allison *et al.*, 2007; Qin, 2007].
- *Modelos prescriptivos.* La mayoría de los modelos se realizan con un enfoque descriptivo, es decir, son marcos de referencia que establecen “qué” debe tener o hacer la organización, no obstante, en su implementación surgen la pregunta “cómo” lo hago. Es importante que los modelos proporcionen o propongan métodos sobre “cómo” implementar lo que proponen [Rainer *et al.*, 2002; Niazi *et al.*, 2006; Allison *et al.*, 2007].

Para lograr el desarrollo profesional del ingeniero de software a través del desarrollo de competencias, es importante considerar los puntos anteriores, pero además surgen las preguntas siguientes:

- ¿Cómo puede saber el ingeniero de software que necesita actualizarse?
- ¿Qué competencias debe desarrollar el ingeniero de software para lograr el desempeño sobresaliente de las actividades de la ingeniería de software?
- ¿Qué actividades se deben realizar para desarrollar las competencias del ingeniero de software?
- ¿Qué características debe tener una herramienta que proporcione soporte al ingeniero de software durante el desarrollo de sus competencias?
- ¿Qué factores (o situaciones) pueden obstruir o dificultar el desarrollo de competencias del ingeniero de software?

I.3 Objetivos

I.3.1 Objetivo general

El objetivo general de la investigación es:

Analizar y diseñar un modelo que apoye al desarrollo de competencias del ingeniero de software.

I.3.2 Objetivos específicos

Los objetivos específicos de la investigación son:

- Identificar los factores que afectan el desarrollo de competencias en el ingeniero de software.
- Analizar y definir un marco de referencia con las competencias que el ingeniero de software debe desarrollar para mantenerse actualizado.
- Analizar y diseñar una estrategia para desarrollar competencias en el ingeniero de software.
- Analizar y diseñar una herramienta que proporcione soporte al ingeniero de software en el desarrollo de competencias.

I.4 Alcances

Marco conceptual

El marco conceptual forma parte del modelo de desarrollo de competencias y su función es responder a la pregunta *¿qué factores (o situaciones) impiden el desarrollo de competencias en el ingeniero de software?* La respuesta que se propone en esta investigación se basa principalmente en la revisión de la literatura y el análisis de la información obtenida a través de estudios empíricos que se realizan en la formación de grupos de desarrollo de software utilizando la metodología de investigación-acción y la técnica de observación participante desde 1994 a la fecha.

Marco de competencias

Para guiar al ingeniero de software durante su actualización y desarrollo profesional es necesario contar con un marco de competencias que sirva de referencia para establecer qué competencias puede desarrollar para alcanzar un nivel competente en el desarrollo de software. El marco que se propone define categorías de competencias, es decir, competencias definidas a un relativamente alto nivel, por lo que es necesario que la organización que desee adoptar este marco tenga que adaptarlo a sus necesidades seleccionando y definiendo las competencias específicas que componen a cada categoría.

Estrategia

La estrategia es la parte del modelo de desarrollo de competencias en la cual se especifica *cómo* desarrollar competencias, es decir, se sugieren un conjunto de actividades que se pueden seguir para lograr el desarrollo de las competencias del ingeniero de software. En la estrategia se explican algunas técnicas que permiten alcanzar el desarrollo de competencias, sin embargo existe una gran variedad de técnicas o formas diferentes para cumplir los objetivos particulares del modelo, algunas de estas técnicas pertenecen a otras áreas del conocimiento como psicología, recursos humanos, desarrollo organizacional, entre otras. Por lo que en esta investigación sólo se proponen algunas de ellas sin profundizar en sus características particulares.

Herramienta de soporte

La herramienta de soporte permite al ingeniero de software administrar información referente a sus competencias, realizar evaluaciones de las mismas y manejar la información sobre su perfil profesional. En la presente investigación se realiza el análisis y diseño de la herramienta de soporte.

I.5 Relevancia

Antes de iniciar el presente trabajo de investigación, al revisar algunos de los temas centrales del mismo, surge la duda sobre si el tema está relacionado con un programa de Ciencias de la Computación o Ingeniería de Software. Es posible que a la gran parte de las personas de estas áreas les pueda parecer extraño e inadecuado tratar temas relacionados con la persona y aspectos sociales en un área de ciencias o ingeniería. Incluso en la literatura se discute y aún no es clara la relación que existe entre las ciencias de la computación, ingeniería de software y los aspectos humanos involucrados en éstas. Sin embargo, conforme se aprende sobre estos temas surge otro tipo de pregunta ¿es la ingeniería de software una rama únicamente de la ingeniería?, ya que al analizar las problemáticas contemporáneas del desarrollo de software se puede notar que la mayor parte de los problemas que se presentan están relacionados con ambos aspectos humanos y aspectos metodológicos o tecnológicos.

Las ingenierías (en general) son profesiones en las que se aplican conocimientos mediante diseños, técnicas y modelos para resolver problemas. La ingeniería de software representa un caso particular, ya que a diferencia de otras áreas de la ingeniería como ingeniería electrónica, civil, mecánica, entre otras, en ella las interacciones que ocurren entre ingenieros de software (y con otras personas, como el cliente y usuario) son fundamentales para lograr el propósito final, un producto de software de alta calidad. Por esto, para desarrollar software (sobre todo de tamaño grande) además del dominio sobre los aspectos ingenieriles (diseños, técnicas y modelos) se requiere también dominio sobre el trabajo en equipo, interacción con el cliente y usuarios del sistema, en un grado superior que la mayor parte de las ingenierías.

Al revisar la literatura se encuentran investigaciones que detectan e indican la necesidad de estudiar el aspecto humano de la ingeniería de software, no obstante estas investigaciones aún son “tímidas” al tratar el tema superficialmente, prácticamente sólo lo mencionan sin profundizar en él. Una posible causa de esto es que la formación académica de las personas del área de ciencias de la computación no permite tratar y ahondar en este tipo de temas. Es importante contar con la colaboración de investigadores de otras áreas del conocimiento que apoyen a encontrar soluciones a este tipo de problemas.

La presente investigación tiene el objetivo diseñar una guía para el desarrollo de competencias, si bien es necesario consultar y aplicar conocimientos de otras áreas como desarrollo organizacional, psicología, recursos humanos, entre otras; para conseguir el objetivo planteado la clave está en conocer profundamente diversos aspectos de la ingeniería de software, como el proceso de desarrollar software, definición de requerimientos, diseño de sistemas basados en software, casos de pruebas, sólo por mencionar algunos.

I.6 Contenido

El contenido del presente documento está organizado en capítulos que llevan al lector de manera gradual a la propuesta de solución que resulta de la investigación que se realizó. En este capítulo (el número uno) se muestran los antecedentes que justifican que se realice la presente investigación. A continuación se describe brevemente el contenido de los capítulos restantes.

El propósito del capítulo dos es mostrar al lector el marco teórico que respalda la investigación, es decir, se presentan las definiciones, conceptos, teorías, técnicas, etc., que proporcionan la base teórica que defiende la propuesta de solución que resulta de esta investigación. El capítulo dos se divide en cuatro temas centrales que son: *desarrollo de software centrado en la persona, las competencias, obsolescencia profesional y el desarrollo profesional.*

En el capítulo tres se presenta el análisis del marco conceptual para el desarrollo de competencias, el cual se diseña para responder a la pregunta de investigación *¿qué factores (o situaciones) pueden obstruir o dificultar el desarrollo de competencias del ingeniero de software?* Primero se presentan los conceptos que permiten entender el marco conceptual, se describe la estructura del modelo de desarrollo de competencias y su relación con el marco conceptual, se enlistan y describen aquellas situaciones que pueden dificultar que el ingeniero de software desarrolle su talento y finalmente se relatan cuatro escenarios en los cuales el ingeniero de software puede utilizar el modelo de desarrollo de competencias.

En el capítulo cuatro se describe el análisis del marco de competencias. El marco de competencias es útil para guiar al ingeniero de software en el desarrollo de su talento. En este capítulo se describen las competencias que el ingeniero de software puede desarrollar para alcanzar el éxito en el desarrollo de software. Para elegir estas competencias se establecen un conjunto de criterios y se analizan las actividades que el ingeniero de software realiza durante el desarrollo de software. Se analiza y propone una clasificación para dichas competencias y una escala que permite medir el nivel de competencia que tiene el ingeniero de software.

En el capítulo cinco se analiza la estrategia para el desarrollo de competencias. La estrategia responde a la pregunta de investigación *¿qué actividades se deben realizar para desarrollar las competencias del ingeniero de software?* Estas actividades están agrupadas en tres grandes procesos, los cuales tienen objetivos particulares. Este capítulo presenta la descripción de estos tres procesos, sus objetivos, tareas, etc.

En el capítulo seis se resumen algunos aspectos importantes sobre el análisis y diseño de la herramienta de soporte. La información completa sobre el análisis y diseño de la herramienta de soporte se encuentra en un reporte técnico, el cual está en un documento diferente al presente trabajo de investigación.

En el capítulo siete se presenta el diseño de un experimento que tiene como objetivo medir el grado de intención de uso del modelo de desarrollo de competencias para ingenieros de

software. Se definen los constructores que se utilizan para elaborar un instrumento de valoración y se describen las características del experimento.

En el capítulo ocho se presenta la discusión del trabajo que se realiza, las principales aportaciones que surgen como producto del presente trabajo y las posibles líneas de investigación que se pueden seguir como trabajo futuro.

II.1 Desarrollo de software centrado en la persona

II.1.1 Introducción

Cuando se menciona “*desarrollo de software centrado en la persona*”, generalmente se piensa en aspectos relacionados con la interacción humano-computadora, usabilidad del sistema y en aquello que concierne al cliente o usuario final del sistema, por ejemplo el diseño de interfaces de usuario. Sin embargo, en este capítulo de la investigación se utiliza para describir principalmente las interacciones humano-humano que ocurren durante el desarrollo de software, es decir, las interacciones del ingeniero de software con los miembros de su equipo de trabajo, con jefes (directivos, mandos medios y otros) y con el cliente o usuario final [Tucker, 2004]. Además de las interacciones sociales, en esta investigación al decir “*desarrollo de software centrado en la persona*” también se hace referencia a los aspectos personales como la motivación, tipo de personalidad, actitud y comportamiento, y a los aspectos técnicos como los conocimientos sobre metodologías y tecnologías para el desarrollo de software.

La mayor parte de los estudios sobre ingeniería de software se realizan desde una perspectiva basada en los procesos. En este apartado, se analizan algunos estudios que examinan los factores que influyen en el desarrollo de software desde una perspectiva de la dinámica humana, esto para definir sus principales actividades tanto técnicas como sociales.

En la literatura se utiliza el término “*peopleware*” para referirse a los aspectos sociales del desarrollo de software [DeMarco *et al.*, 1999]. El enfoque social de la ingeniería de software tiene como propósito estudiar cómo los ingenieros de software trabajan juntos para producir software [Sawyer *et al.*, 2008]. En la Tabla I se indican los principales estudios que influyen en la construcción del presente capítulo:

Tabla I. Investigaciones sobre el desarrollo de software centrado en la persona.

Autores	Año	Título del trabajo
Tom DeMarco y Timothy Lister	1999	Peopleware: productive projects and teams
Josefina Rodríguez-Jacobo	2004	Formación de grupos de desarrollo de software: un enfoque psicosocial
James Tomayko y Orit Hazzan	2004	Human aspect of software engineering
Thomas Hanne y Holger Neu	2004	Simulating human resources in software development processes
Allen Tucker	2004	Computer science: handbook
Silvia Acuña, Natalia Juristo, Ana Moreno y Alicia Mon	2005	A Software Process Model Handbook for Incorporating People's Capabilities
Steve Sawyer, Jay Coopriider, P. Guinan	2008	Social interactions of information systems development teams: a performance perspective

De los estudios que se indican en la Tabla I se abstraen algunos conceptos que se describen en el presente apartado. A continuación se analiza y define qué es un modelo del proceso de software, ya que éste permite establecer qué actividades y roles realiza el ingeniero de software durante el desarrollo de software, estos últimos dos temas también se describen más adelante. La finalidad es mostrar las bases para crear un modelo de las actividades y roles del desarrollo de software que integre las perspectivas técnica y social, esto para identificar algunas de las competencias (técnicas, sociales y personales) que el ingeniero de software requiere para realizar las actividades de la ingeniería de software.

II.1.2 Modelos del proceso de software

Los conceptos ciclo de vida y proceso de desarrollo de software suelen causar cierto grado de confusión incluso entre los propios ingenieros de software, quienes lo utilizan de manera indistinta, a pesar que estos se refieren a cosas ligeramente diferentes [Acuña *et al.*, 2005]. Para fines de la presente investigación el concepto que interesa es el proceso de desarrollo

de software, al cual también se suele referirse a él como proceso de software o proceso de desarrollo. A continuación se definen y se describen las principales características del ciclo de vida y del proceso de desarrollo de software.

Ciclo de vida del software

El *ciclo de vida del software* define los estados por los cuales pasa el producto de software. Es decir, el enfoque en este caso es el producto, por esto define los estados o etapas en las cuales se puede encontrar el producto de software en un momento determinado, estos estados van desde que inicia la necesidad del usuario (primer estado) hasta que el software finalmente se retira (último estado) [Acuña *et al.*, 2005].

De acuerdo con la definición anterior un *modelo del ciclo de vida del software* es una representación abstracta de los estados por los que pasa el producto de software a lo largo de su vida. Pueden existir diferentes modelos del ciclo de vida del software, debido a que las circunstancias en las cuales se desarrolla el software pueden variar de un proyecto a otro [Acuña *et al.*, 2005].

Proceso de desarrollo de software

Por otro lado, el *proceso de desarrollo de software* define las tareas que se deben ejecutar para poder construir el producto de software. El enfoque en este caso son las actividades que permiten manejar, desarrollar y mantener un sistema basado en software. Además de las actividades, el proceso de software debe especificar las técnicas que se pueden utilizar para ejecutar las tareas, los actores que ejecutan las tareas, sus roles y los artefactos (productos de salida en una actividad) que se producen [Acuña *et al.*, 2005].

El proceso de software se forma de dos subprocesos principales: el proceso de producción y la administración del proceso. El proceso de producción se relaciona con la construcción y el mantenimiento del producto de software. Mientras que la administración del proceso se relaciona con la estimación, planeación y control de los recursos necesarios para la producción del software.

Por lo anterior, un *modelo del proceso de software* es una representación abstracta de la secuencia de actividades que se ejecutan para obtener un producto de software. Al igual que con los modelos del ciclo de vida del software, también existen varios modelos del proceso de software, que dependen de las características de cada organización y del cliente, además que cada modelo se define con diferente nivel de detalle, a pesar de esto existen algunas actividades que son comunes para la mayoría de todos los posibles procesos para producir software [Acuña *et al.*, 2005].

Los modelos de procesos se pueden utilizar para control, evaluación y mejora del proceso de desarrollo en una organización, también permiten experimentar con la teoría del proceso de software y automatizar el proceso de desarrollo [Acuña *et al.*, 2005]. Los componentes básicos para el modelado del proceso de software son [Acuña *et al.*, 2001]:

- *Actor (o agente)*: es una entidad que ejecuta una actividad o proceso. El actor puede ser una persona o un sistema. Un actor realiza uno o varios roles.
- *Actividades*: es una meta del proceso que produce un cambio al estado del producto de software. Una actividad se puede componer de una entrada, una salida y algunos resultados intermedios.
- *Roles*: describe un conjunto de actores o grupo de responsabilidades, conocimientos y habilidades que se requieren para desempeñar una actividad específica del proceso de software.
- *Productos (o artefactos)*: es el resultado (o subproductos) del proceso de software. Un artefacto se puede producir en una actividad y ser utilizado por otras actividades. El conjunto de artefactos producidos durante el desarrollo de software es lo que se entrega al cliente bajo el nombre de producto de software. Los artefactos son documentación, datos, software, etc.

En la Figura 1 se muestra la representación gráfica de los componentes básicos para el modelado del proceso de software. El actor (persona o sistema) realiza uno o varios de los roles de la ingeniería de software, a cada uno de los roles le corresponde un conjunto de actividades relacionadas entre sí, la ejecución de esas actividades puede requerir productos

de entrada y genera productos de salida, los cuales se pueden utilizar en otras actividades y por otros roles.

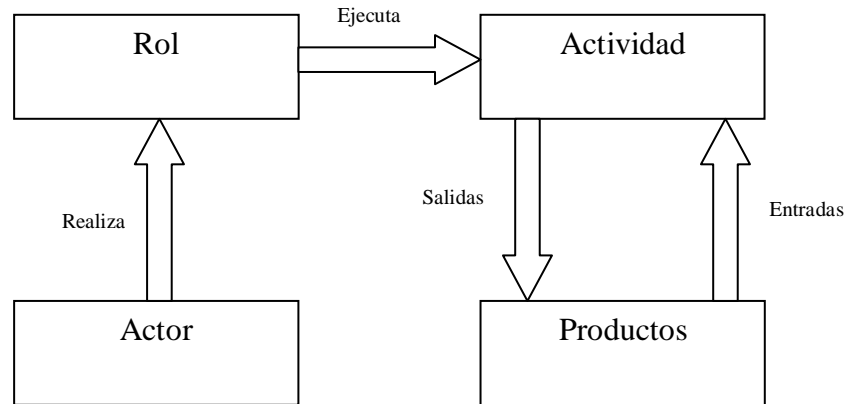


Figura 1.Componentes básicos para el modelado del proceso de software (versión original en Acuña *et al.* 2001).

Los modelos del proceso de software dependiendo de su objetivo se pueden dividir en dos categorías de modelos: *descriptivos* y *prescriptivos*.

- Los *modelos descriptivos* especifican el proceso de software que se utiliza en una organización. Estos modelos responden a la pregunta ¿cómo se está desarrollando el software?
- Los *modelos prescriptivos* especifican cómo se debe presentar el proceso de software. Estos modelos responden a la pregunta ¿cómo se debe desarrollar el software?

Los modelos sobre el desarrollo de software que se encuentran en la literatura describen las actividades del desarrollo de software desde una perspectiva técnica. Para la presente investigación el interés es conocer las actividades y roles del ingeniero de software desde ambas perspectivas (técnica y social), ya que de las actividades técnicas que ejecuta el ingeniero de software surgen las actividades sociales, que a su vez pueden influir en el resultado de las primeras. A continuación se analizan algunas de las características de las actividades y roles del desarrollo de software desde la perspectiva técnica y algunas de las actividades y roles desde la perspectiva social que identifican algunos investigadores.

II.1.3 Actividades y roles del desarrollo de software

Desde la perspectiva técnica, las actividades que componen al desarrollo de software definen los pasos necesarios para lograr las metas y objetivos necesarios para desarrollar software. De acuerdo al SWEBOK (Software Engineering Body of Knowledge, en español Cuerpo de Conocimientos de Ingeniería de Software), las actividades de la ingeniería de software se agrupan en 10 áreas de conocimientos [Abran *et al.*, 2004], las cuales se enlistan a continuación:

- Requerimientos.
- Diseño.
- Construcción.
- Pruebas.
- Mantenimiento.
- Administración de la configuración.
- Administración.
- Procesos.
- Herramientas y métodos.
- Calidad.

Las actividades que forman a estas áreas de conocimientos permiten desarrollar software y es el ingeniero de software la persona que se encarga de realizar dichas actividades. En la literatura, dependiendo del modelo de proceso de software que se sigue, se le asigna al ingeniero de software roles diferente. Para fines de la presente investigación, los roles que puede asumir durante el desarrollo de software, son los que se muestran en la Tabla II:

Tabla II. Roles que realiza el ingeniero de software en el proceso de desarrollo de software.

a) Administrador de proyectos.	b) Analista.
c) Diseñador.	d) Programador.
e) Ingeniero de validación y verificación.	f) Administrador de la configuración.
g) Asegurador de la calidad.	h) Pruebas.
i) Documentador.	j) Ingeniero de mantenimiento.

Desde la perspectiva social, las actividades del desarrollo de software cubren aspectos como la comunicación informal entre los miembros del equipo de desarrollo, compartir

ideas e información, resolver conflictos, coordinación con los miembros del equipo y con otras personas que no son miembros del equipo, manejo del flujo de información, administrar recursos, etc. Sawyer y otros [Sawyer *et al.*, 2008] identifican cinco tipos o grupos de actividades sociales en el desarrollo de software: embajador (ambassador), explorador (scout), guardia (guard), centinela (sentry) y coordinador (coordinator). A continuación se describe a cada uno de ellos:

- a) *Actividades de embajador*: son aquellas que tienen la finalidad de representar a los miembros del equipo ante otras personas. Sirven para establecer jerarquías en la organización, proporcionar retroalimentación sobre el progreso del equipo ante los altos mandos o clientes y para interceder ante oposiciones externas.
- b) *Actividades de explorador*: son aquellas actividades que sobre pasan los límites del equipo y sirven para traer información sobre lo que sucede en otros lugares en la organización y fuera de ella. Las actividades de explorador pueden incluir explorar mercados externos, buscar tecnologías nuevas, identificar actividades relevantes fuera del equipo y el descubrimiento de potenciales competencias.
- c) *Actividades de guardia y centinela*: sirven para cuidar que los miembros del equipo trabajen con las mínimas distracciones posibles. Las actividades de guardia tienen la función de mantener la información y recursos dentro del equipo de trabajo. El guardia se encarga de solicitar información o recursos a las personas externas y ayuda a determinar que el grupo libera en respuesta a esas demandas. Por otro lado, el centinela tiene la función de “policía” que resguarda los límites del equipo para controlar la información y recursos que las personas externas quieren enviar a los miembros del equipo. Los guardias y centinelas esencialmente actúan como filtros que deciden qué información o recursos entran a o salen desde los miembros del equipo.
- d) *Actividades de coordinador*: se enfocan en establecer comunicación. Incluye discusiones de problemas de diseño, obtención de retroalimentación sobre el

progreso del equipo y planeación de eventos tales como negociación de tiempo y recursos adicionales.

De acuerdo con las actividades sociales que indican y agrupan Sawyer y otros [Sawyer *et al.*, 2008], los roles del ingeniero de software, desde una perspectiva social son:

- Embajador.
- Explorador.
- Guardia.
- Centinela.
- Coordinador.

Para poder desempeñar las actividades y roles (técnicos y sociales) que se presentan en este apartado, es necesario que el ingeniero de software tenga ciertos conocimientos, habilidades y comportamientos, a estos en conjunto se les conoce como competencia, este tema se presenta a continuación.

II.2 Competencias

II.2.1 Introducción

Uno de los propósitos de la presente investigación es definir qué y cómo se pueden desarrollar los conocimientos, habilidades, comportamientos, actitudes, valores, etc., que el ingeniero de software requiere para lograr un desempeño sobresaliente en las actividades del desarrollo de software. En el área de recursos humanos se utiliza el concepto “*competencia*” para referirse a estas características del individuo.

Para fines de la presente investigación *competencia* es el parámetro que se utiliza para medir el nivel (o grado) de desarrollo profesional del ingeniero de software. Medir las competencias del ingeniero de software y compararlas con las competencias que exige un determinado puesto de trabajo, permite evaluar si éste es capaz de desempeñar ese puesto de trabajo, de manera similar se puede evaluar si el ingeniero de software es competente para un determinado rol o actividad de la ingeniería de software.

A continuación se analizan los fundamentos que existen detrás del concepto de *competencia* como su definición, clasificación, enfoques y aplicaciones en el área de recursos humanos. También se define qué es un marco de competencias y las características que deben considerar.

II.2.2 Fundamentos

II.2.2.1 Definición

En la literatura no existe una definición única del concepto *competencia*, cada autor lo define de acuerdo a sus necesidades. Para fines de esta investigación se define competencia de la manera siguiente:

“La competencia se compone de conocimientos, habilidades, comportamientos y otras características (actitud, habilidad física, etc.) específicas, definibles y medibles que puede poseer el ingeniero de software para el desempeño eficiente de las actividades contemporáneas de la ingeniería de software, cumpliendo con las normas de calidad regidas por la empresa o la demanda del mercado.”

II.2.2.2 Clasificación

De manera similar que con la definición del concepto de competencia, no existe una forma genérica para clasificar competencias. La clasificación varía de acuerdo al contexto de aplicación y el autor que la realiza. A continuación se enlista y define las clasificaciones más comunes:

- *Competencias técnicas*: hacen referencia al desempeño de una función o al uso de recursos en situaciones específicas [Le Boterf, 1991; Alex, 1991; Bunk, 1994; Echeverría, 2002].
- *Competencias sociales*: hacen referencia a aspectos personales y de relación con el entorno [Le Boterf, 1991; Alex, 1991; Bunk, 1994; Echeverría, 2002].

- *Competencias básicas*: conjunto de requisitos previos para entrar al mundo laboral [ISFOL, 1995; Mertens, 1996].
- *Competencias específicas*: conjunto de conocimientos y técnicas para desempeñar una función específica [ISFOL, 1995; Mertens, 1996].
- *Competencias transversales o genéricas*: conjunto de habilidades para el comportamiento eficaz [ISFOL, 1995; Mertens, 1996].
- *Competencias metodológicas*: capacidad de aplicar procedimientos adecuados a las tareas encomendadas y a las irregularidades [Bunk, 1994; Echeverría, 2002].
- *Competencias participativas*: capacidad de saber participar en la organización del puesto del trabajo [Bunk, 1994; Echeverría, 2002].

A pesar de existir ciertas similitudes entre los autores que se estudian no existe una clasificación general, sino que cada autor las clasifica de acuerdo a sus necesidades. La clasificación de las competencias se hace en base a las características que los autores identifican en éstas, agrupan competencias con características similares.

II.2.2.3 Enfoques

Además de la clasificación (que cada autor propone), las competencias pertenecen a uno de los enfoques siguientes:

- *Enfoque funcional*: son aquellas que describen una actividad laboral.
- *Enfoque conductual*: son aquellas que describen atributos de la persona.
- *Enfoque holístico*: son aquellas que describen las dos anteriores.

Así, las competencias con enfoque funcional se caracterizan por describir una actividad laboral que realiza el profesionista, por ejemplo, para el ingeniero de software una competencia con enfoque funcional es “*obtener los requerimientos del cliente.*” Las

competencias con enfoque conductual se caracterizan por describir atributos de la persona como “*capacidad para trabajar en equipo*”.

McLagan [McLagan, 1997: Klein *et al.*, 2005] identifica que en la literatura las competencias tienen seis enfoques diferentes, estos describen:

- Tareas (actividades) de un trabajo.
- Resultados que se obtienen del esfuerzo en un trabajo.
- Salidas.
- Conocimientos, habilidades y actitudes.
- Cualidades que describen el desempeño superior de un profesional.
- Conjunto de atributos.

En la presente investigación el enfoque que se utiliza es el *conductual*, es decir competencias que describen atributos de las personas, como conocimientos habilidades y actitudes que permiten el desempeño superior del ingeniero de software.

II.2.2.4 Conceptos

Por lo mencionado hasta aquí, el lector puede notar la variedad de interpretaciones que pueden darse del concepto competencia. Cada autor puede incluir diferentes elementos en su definición, realizar diferentes clasificaciones y pertenecer a distintos enfoques. Al final de cuentas la idea del concepto de competencia converge a tres cosas: *saber hacer*, *poder hacer* y *querer hacer*. Es decir, que para ser competente (o tener una determinada competencia) se requiere “*saber hacer*” que se refiere a tener el conocimiento sobre los aspectos teóricos de la competencia, además se debe “*poder hacer*”, en otras palabras tener la habilidad para aplicar los conocimientos teóricos en situaciones reales. Finalmente se debe “*querer hacer*”, que quiere decir tener la actitud y comportamiento adecuado para la aplicación de las competencias.

De acuerdo con lo anterior las competencias se componen de conocimientos, habilidades y comportamientos.

- El conocimiento (*o saber hacer*) se refiere al entendimiento del contenido o de la información técnica (herramienta, fenómeno, metodología, etc.) necesaria para desempeñar adecuadamente una función laboral y que se obtiene a través de la educación formal, la formación profesional continua, experiencias laboral y otros medios de adquisición de información [Marrelli *et al.*, 2005; Valdez *et al.*, 2008].
- La habilidad (*o poder hacer*) se refiere a los factores cognitivos que representan la capacidad para aplicar efectivamente los conocimientos sobre una determinada función laboral [Marrelli *et al.*, 2005; Valdez *et al.*, 2008].
- El comportamiento (*o querer hacer*) se refiere a la actitud que se manifiesta como una reacción afectiva positiva o negativa hacia un objeto (abstracto o concreto) y que determina la manera de actuar del individuo.

De acuerdo al modelo del iceberg que Spencer y Spencer [Spencer *et al.*, 1993] aplican a las competencias, los conocimientos y las habilidades son la parte visible y relativamente fácil de desarrollar de una competencia. Mientras que el comportamiento se encuentra en la parte oculta del iceberg, por lo que no es visible fácilmente y es difícil de desarrollar. En la Figura 2 se puede observar el modelo del iceberg, este modelo trata de representar a un iceberg los cuales se caracterizan por tener un parte visible y otra oculta, la parte visible es sólo un pequeño porcentaje del iceberg, mientras que la parte oculta representa casi la totalidad del iceberg, si se realiza una analogía con las competencias, la parte visible son los conocimientos y habilidades que se pueden demostrar y medir relativamente fácil, mientras que la parte oculta de la competencia de las personas se encuentra en sus actitudes, valores, motivación, que son menos fácil de observar y medir.

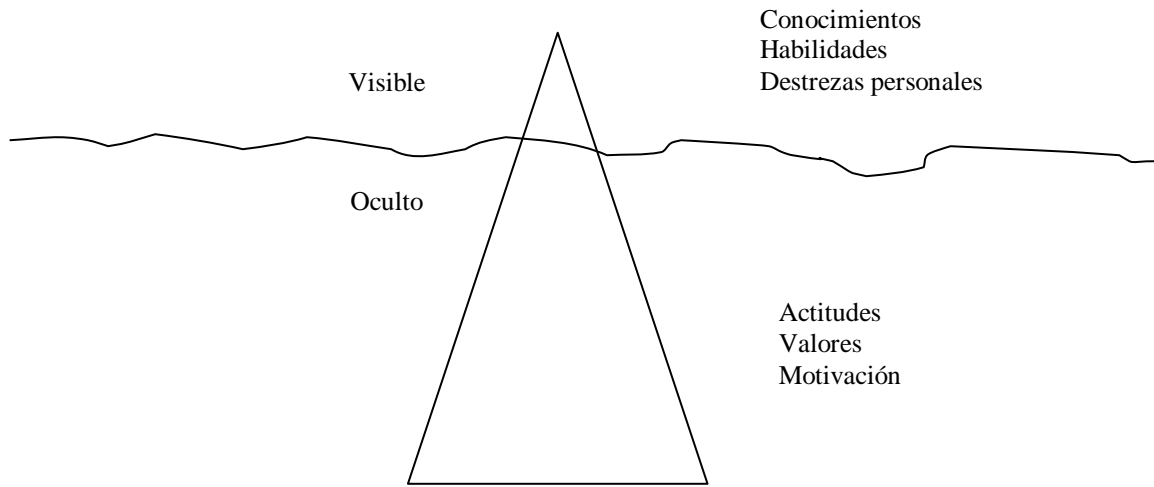


Figura 2. Modelo del Iceberg de la competencia (adaptado de Spencer *et al.*, 1993).

II.2.2.5 Aplicaciones

El describir un puesto de trabajo a través de competencias puede facilitar a la organización actividades de gestión del personal, algunas de las actividades en las que se pueden utilizar las competencias son [Armstrong, 2003; Aragón *et al.*, 2004]:

- Elaboración de planes de carrera.
- Diseño del perfil de un puesto.
- Formación y desarrollo continuo.
- Valoración del potencial.
- Evaluación del desempeño.
- Diseño de programas de compensación.
- Diseño de programas de reclutamiento.
- Planificación de la sucesión.

En la presente investigación la finalidad principal de utilizar las competencias es como parámetro para medir la formación y desarrollo profesional del ingeniero de software. Sin embargo, aunque no se profundiza en estos temas, también se utiliza para evaluar el desempeño, diseñar el perfil de puesto y para elaborar el plan de carrera del ingeniero de software.

II.2.3 Marco de competencias

Un marco de competencias (también referido en la literatura como modelo de competencias) define al conjunto de conocimientos, habilidades y comportamientos que el profesional requiere para alcanzar un alto rendimiento y el desempeño sobresaliente de las actividades correspondientes a una determinada función laboral. De acuerdo con Marrelli [Marrelli *et al.*, 2005] “un modelo de competencias es la clasificación de las competencias identificadas en un marco conceptual que permite a las personas en una organización entender, hablar sobre y aplicar las competencias.”

Berger y otros [Berger *et al.*, 2003] sugieren que los pasos para realizar un modelo de competencias son:

- 1) Definir los criterios de desempeño.
- 2) Analizar la muestra de criterios.
- 3) Colectar datos.
- 4) Desarrollar el modelo de competencias.
- 5) Validar el modelo de competencias.
- 6) Aplicación.
- 7) Evaluación.

Los marcos de competencias implican dos componentes: un trabajo (sitio receptor) y una persona (competencias). Entre mejor cumpla la persona (su competencia) los requerimientos del trabajo, mayor posibilidad existe de que la persona alcance el desempeño sobresaliente en el trabajo [Berger *et al.*, 2003].

La *identificación de competencias* sirve para determinar las competencias que se incluirán en un marco de competencias y que son fundamentales para lograr el desempeño satisfactorio o ejemplar de un trabajo. La identificación de competencias es un paso que va más allá del simple análisis de tareas y trabajos, es un medio para clarificar los requerimientos claves para el trabajo. [Dubois, 1993; Rothwell, 2005].

II.3 Obsolescencia profesional

II.3.1 Introducción

Uno de los temas principales de la presente investigación es la obsolescencia profesional, ya que es el principal indicio de la necesidad de desarrollo de competencias. En el presente apartado se define y explica en qué consiste el proceso de la obsolescencia profesional y se analiza el trabajo de algunos investigadores sobre este tema.

II.3.2 Definición

La permanente innovación tecnológica y la mejora continua del proceso de software son las causas principales de los constantes cambios que ocurren en las organizaciones y en la forma de trabajar en la industria del software. El resultado de esto es que los conocimientos y habilidades del ingeniero de software degradan su valor y se vuelven obsoletos rápidamente. Al estado en el cual los conocimientos y habilidades del ingeniero de software son obsoletos, se le conoce como obsolescencia profesional y es el estado contrario al actualizado o competente. Diversos investigadores estudian el proceso mediante el cual se presenta la obsolescencia profesional en los ingenieros de software [Fossum *et al.*, 1986; Blanton *et al.*, 1998; Damien *et al.*, 2001; van Loo *et al.*, 2001; Tsai *et al.*, 2004].

De acuerdo con Fossum y otros [Fossum *et al.*, 1986] la obsolescencia profesional ocurre cuando los requerimientos de trabajo son incongruentes con el conjunto de conocimientos, habilidades y comportamientos que posee el individuo en un determinado momento, bajo la condición que previamente ese conjunto de conocimientos, habilidades y comportamientos eran congruentes con los requerimientos para ese mismo trabajo. Se refiere a requerimientos del trabajo como el conjunto de conocimientos, habilidades y comportamientos que exigen las tareas, obligaciones y responsabilidades inherentes al trabajo, función laboral o rol en cuestión.

De esta manera,

Obsolescencia profesional es el grado en el cual las competencias que posee el ingeniero de software no son congruentes con los requerimientos de los roles de la ingeniería de software.

A continuación se analizan algunas investigaciones que estudian el tema de la obsolescencia profesional en la ingeniería de software. Algunas de estas investigaciones se realizan para el sector de las Tecnologías de Información y Comunicación (TIC), pero al ser la ingeniería de software una rama de este sector, se considera que lo que los investigadores indican también es válido para el ingeniero de software.

II.3.3 Modelos de obsolescencia

Blanton y otros [Blanton *et al.*, 1998] hacen una evaluación de los factores que afectan la obsolescencia profesional en los profesionales en tecnologías de la información y comunicación (TIC). Los factores que estos investigadores consideran los clasifican en tres grupos:

- Características individuales.
- Naturaleza del trabajo.
- Clima organizacional.

Las características individuales se refieren a factores como la disposición, necesidades y aptitudes que son útiles para predecir el comportamiento y los resultados del profesionista en TIC. La naturaleza del trabajo refleja cómo intelectual y psicológicamente el profesionista en TIC percibe los retos o la rutina del trabajo que se le asigna. Finalmente, el clima organizacional indica la percepción que el individuo tiene sobre el contexto de trabajo, es decir, cómo la influencia social, las prácticas de la administración y las recompensas, estimulan o reprimen al profesionista en TIC para que se mantenga actualizado.

De acuerdo a lo anterior, la obsolescencia profesional es función de variables que se clasifican en tres grupos (enlistados arriba). En primer lugar, las características del

individuo influyen de la siguiente y otras formas: si el profesionalista en TIC no tiene la disposición, necesidad o voluntad para actualizar y desarrollar sus competencias, las actividades de actualización que realice difícilmente tendrán la efectividad que se espera y puede mostrar resistencia al cambio.

Por otro lado, la naturaleza del trabajo se refiere a si el trabajo que realiza la persona le exige utilizar una variedad de competencias o únicamente utiliza algunas pocas de ellas, en este último caso al profesionalista en TIC el trabajo que realiza le puede parecer monótono y poco desafiante, lo que influye de manera negativa en su motivación para mantenerse actualizado.

Por último, el clima organizacional se refiere al entorno en el cual se desenvuelve el profesionalista en TIC para realizar su trabajo. Este grupo abarca factores que se relacionan con los compañeros de trabajo, jefes, el espacio físico, los recursos y todo lo que se encuentra en el entorno de trabajo del profesionalista en TIC y que influye en su voluntad para actualizarse. Por ejemplo, si el profesionalista en TIC percibe que la organización no proporciona oportunidades para la actualización, esto le dificulta que pueda realizar las actividades de actualización.

De acuerdo a los resultados de la investigación Blanton y otros [Blanton *et al.*, 1998], si se aumenta el nivel de desafío (complejidad o responsabilidad) del trabajo del profesionalista en TIC también se debe incrementar su competencia, ya que se le exige utilizar un número y variedad mayor de competencias, lo cual a su vez lo motiva a actualizarse y desarrollarse profesionalmente. Concluyen que el grado de actividades de actualización se relaciona significativamente al nivel de competencia del profesionalista en TIC, es decir, entre mayor cantidad de actividades de actualización realice, existe menor posibilidad de que llegue al estado de obsolescencia profesional.

En la Figura 3 se observa la representación gráfica del modelo de los factores directos de la obsolescencia profesional que proponen Blanton y otros [Blanton *et al.*, 1998]. El modelo indica que los factores que conducen a la obsolescencia profesional (o a la competencia profesional, el estado contrario a la obsolescencia) están divididos en tres grupos

principales: diferencias de la personalidad del individuo, clima organizacional de actualización y la naturaleza del trabajo.

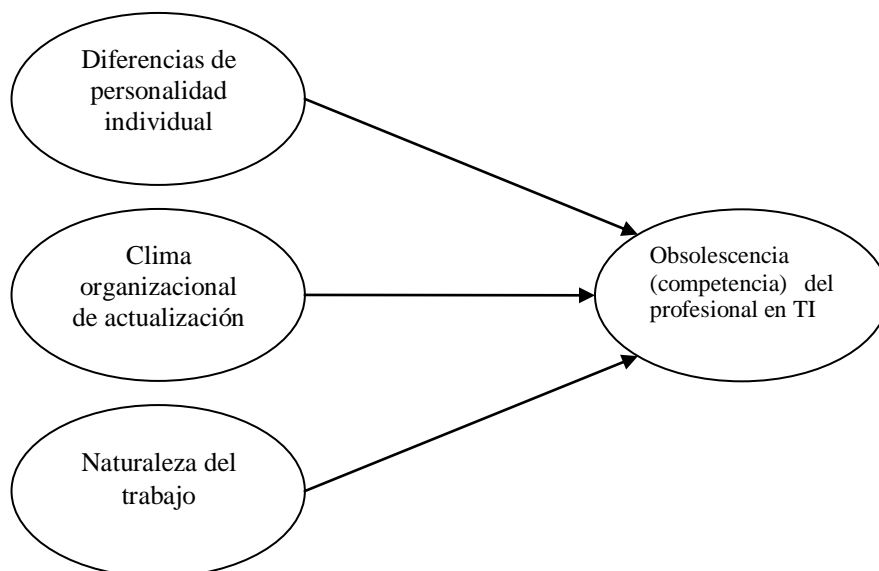


Figura 3. Modelo de los factores directos de la obsolescencia profesional (versión original en Blanton *et al.*, 1998).

Por otro lado, en una investigación también sobre la obsolescencia profesional pero con un enfoque diferente, Damien y Soon [Damien *et al.*, 2001] estudian cómo los profesionistas en TIC se enfrentan a la obsolescencia profesional y el impacto que ésta tiene sobre la movilidad laboral y rotación del personal en las organizaciones. La diferencia de esta investigación con la de Blanton y otros [Blanton *et al.*, 1998] es que en ésta se estudian las consecuencias de la obsolescencia profesional, mientras que en la anterior se estudian los factores que la provocan. Lo importante del resultado de este estudio para la presente investigación está en las conclusiones sobre el trabajo, los investigadores sugieren que los profesionistas en TIC junto con las organizaciones deben desarrollar estrategias que permitan a los profesionistas en TIC permanecer actualizados y de esta manera evitar las consecuencias de la obsolescencia profesional, como la movilidad laboral o rotación del personal, es decir que la actualización continua profesional es la principal forma de evitar la obsolescencia profesional.

Otro aspecto importante de esta investigación, es el énfasis que hacen los investigadores en el *compromiso profesional* de la persona. En la Figura 4 se observa el modelo teórico de la respuesta del individuo a la amenaza de obsolescencia profesional que proponen Damien y Soon [Damien *et al.*, 2001]. En él se puede observar que, según los investigadores, el *compromiso profesional* influye en la decisión que el profesional en TIC toma cuando se enfrenta a la obsolescencia profesional.

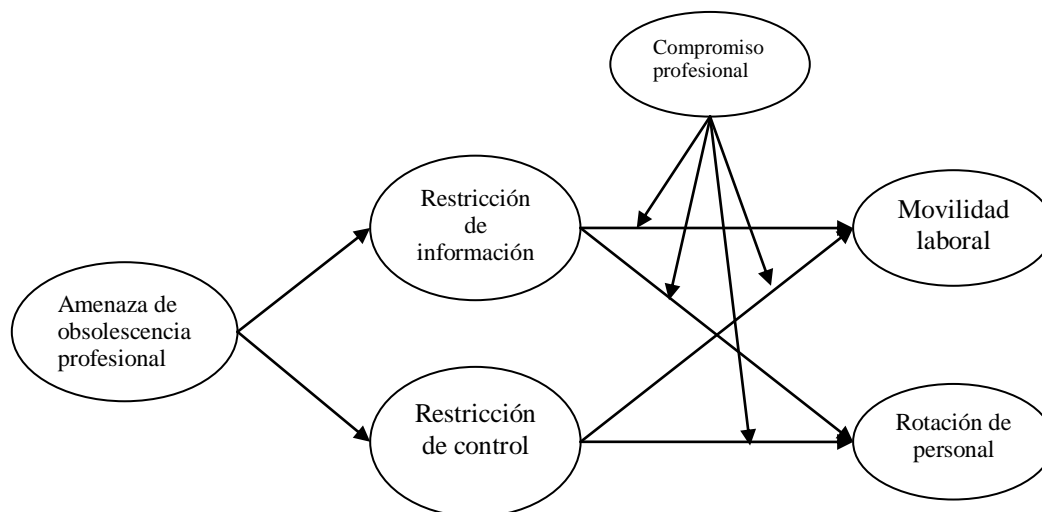


Figura 4. Modelo teórico de la respuesta del individuo a la amenaza de obsolescencia profesional (versión original en Tsai *et al.*, 2004).

Asimismo, Tsai y otros [Tsai *et al.*, 2004] estudian cómo los profesionistas en TIC responden ante la amenaza de la obsolescencia y qué acciones de actualización deciden tomar. Una de las características que interesa de esta investigación es que los investigadores clasifican los factores que provocan la obsolescencia profesional en tres grupos:

- Individuales.
- Relacionados con el trabajo.
- Organizacionales.

Esta clasificación es similar a la que realizan Blanton y otros [Blanton *et al.*, 1998], la relación se muestra en la Tabla III:

Tabla III. Comparación entre la clasificación de los factores que afectan la obsolescencia profesional que realizan [Blanton *et al.*, 1998] y [Tsai *et al.*, 2004].

Clasificación de [Blanton <i>et al.</i>, 1998]	Clasificación de [Tsai <i>et al.</i>, 2004]	Ejemplos de factores de cada grupo
Características individuales	Factores individuales	Disposición, necesidades, edad, actitud, personalidad, motivación, etc.
Naturaleza del trabajo	Factores relacionados con el trabajo	Complejidad, desafío, etc.
Clima organizacional	Factores organizacionales	Variedad de asignación de trabajo, recompensa, apoyo, orientación, etc.

Finalmente, Fossum y otros [Fossum *et al.*, 1986] presentan un modelo de la obsolescencia profesional que, aunque éste no es para alguna profesión en específico, las ideas que proponen se pueden aplicar a la obsolescencia profesional del ingeniero de software. En la Figura 5 se observa la representación gráfica del modelo de los factores involucrados en la obsolescencia de habilidades que proponen Fossum y otros, el modelo indica que en un tiempo determinado, un trabajo en específico exige ciertos requerimientos, los cuales el ingeniero de software cubre con sus conocimientos, destrezas y habilidades (CDH), sin embargo, factores como tecnologías, metas y procedimientos nuevos ocasionan cambios en el trabajo, mientras que factores motivacionales, individuales, organizacionales y externos producen cambios en la persona. Estos cambios (en el trabajo y la persona) ocasionan que los requerimientos para el trabajo ya no concuerden con los CDH de la persona, es en esta situación se dice que está presente la obsolescencia profesional.

Los investigadores de los estudios que se mencionan en este apartado proponen que para evitar la obsolescencia profesional es necesario la capacitación y actualización constante de las competencias del profesionista. Por ejemplo, Jasper van Loo y otros [van Loo *et al.*, 2001] quienes analizan la relación que existe entre los factores de riesgo y los mecanismos para evitar la obsolescencia profesional, sugieren que el entrenamiento y la flexibilidad funcional son los mecanismos principales para evitar la obsolescencia profesional. Con flexibilidad funcional se refiere a la capacidad que tiene el profesionista para realizar diferentes funciones o roles en la organización.

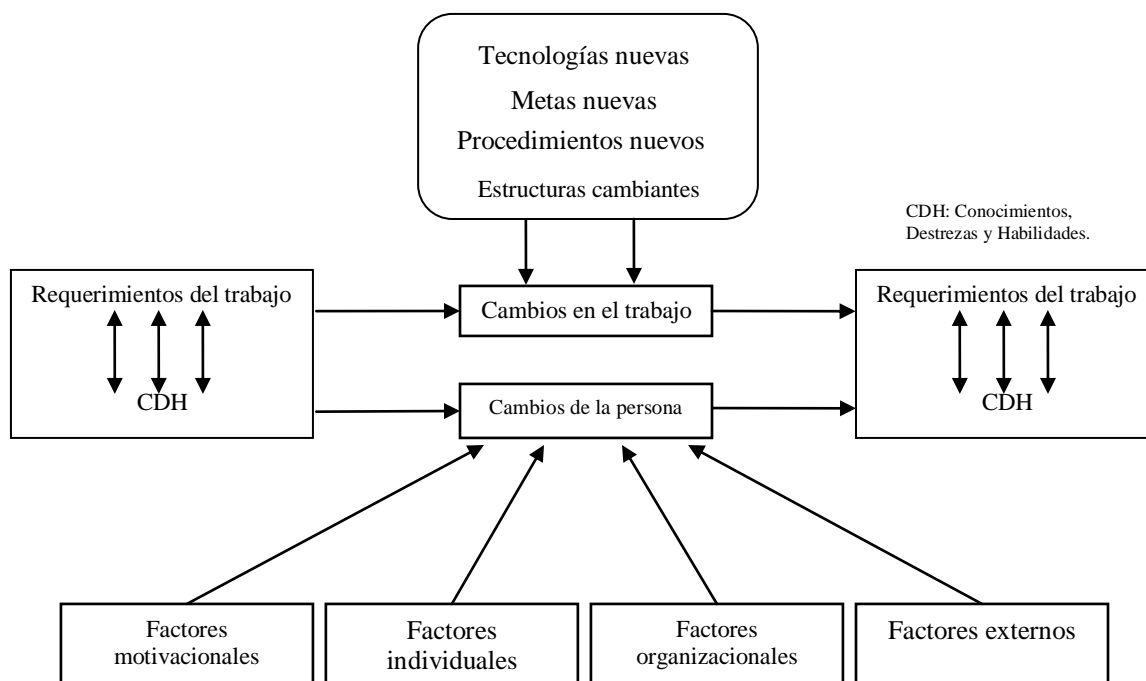


Figura 5. Modelo de los factores involucrados en la obsolescencia de habilidades (versión original en Fossum *et al.*, 1986).

El desarrollo profesional consiste de aquellas actividades que permiten actualizar y desarrollar competencias a través de técnicas como la formación académica, capacitación en el trabajo, entrenamiento, capacitación mixta, entre otros. De esta manera, el desarrollo profesional es la actividad que permite al profesionista evitar la obsolescencia profesional. El tema del desarrollo profesional se trata en el apartado siguiente.

II.4 Desarrollo profesional

II.4.1 Introducción

El desarrollo profesional es un tema bastante amplio y resulta complicado tratar todos los aspectos que éste abarca en un sólo apartado de la investigación. Aquí, únicamente se describen algunos aspectos básicos y útiles para comprender mejor el desarrollo de la presente investigación.

A continuación se describe la definición y algunas de las características de interés para esta investigación sobre el desarrollo profesional. Posteriormente se analizan algunos estudios que identifican algunos factores que facilitan o dificultan que se efectúe el desarrollo profesional y las metodologías que se pueden seguir para lograr el desarrollo profesional del ingeniero de software.

II.4.2 Fundamentos

El desarrollo profesional es el proceso mediante el cual el profesionista progresa por medio de una serie de etapas que se caracterizan por distintas actividades de desarrollo [Fernández Losa, 2002]. Se entiende por actividades de desarrollo profesional como aquellas acciones de aprendizaje que realiza el profesionista para actualizar y adquirir conocimientos, habilidades y comportamientos que son útiles para su desempeño laboral.

Las etapas profesionales por las que puede pasar una persona se indican en la Tabla IV, en ella se observa que en las columnas se enlistan las etapas profesionales y en los renglones la descripción de las tareas de desarrollo, actividades, relación con otros trabajadores, edad y antigüedad que suelen distinguir a cada una de las etapas.

Tabla IV. Etapas profesionales y principales características (adaptado de Fernández Losa, 2002).

	Etapas profesionales			
	Incorporación	Crecimiento	Madurez	Maestría
Tareas de desarrollo	Identificar intereses y habilidades, encajar en el trabajo	Ascenso, crecimiento, seguridad, desarrollo de un estilo de vida	Mantener los logros y actualizar las habilidades	Planificar la jubilación y cambiar el equilibrio entre lo laboral y no laboral
Actividad	Aprendiz	Compañero	Tutor	Consejero
Edad	Menos de 30	Entre 30 y 45	Entre 45 y 60	Más de 60
Antigüedad	Menos de 2 años	Entre 2 y 10 años	Más de 10 años	Más de 10 años

Es posible que el ingeniero de software que se incorpora al mundo laboral tenga una buena cantidad de competencias que son necesarias para comenzar a trabajar. No obstante, en cualquiera de las etapas profesionales es importante la actualización y desarrollo constantes

de competencias para poder tener y mantener un desempeño sobresaliente en las actividades contemporáneas de la ingeniería de software.

La *capacitación* es el término que se refiere a los esfuerzos que realiza la organización para impulsar el aprendizaje del ingeniero de software. Mientras que el término *desarrollo* se le confiere un sentido más amplio (que al término capacitación) y hace referencia a la adquisición de competencias para responsabilidades futuras. Generalmente, se combinan ambos términos "*capacitación y desarrollo*" para referirse a las actividades que realizan la organización y el ingeniero de software para elevar la capacidad del ingeniero de software e instruirlo para el desempeño de sus actividades laborales actuales y futuras [Bohlander *et al.*, 2001].

En la Figura 6 se puede observar el modelo sistemático de capacitación y desarrollo que propone George Bohlander y otros [Bohlander *et al.*, 2001]. El modelo se divide en cuatro grandes fases o actividades: detección de necesidades de actualización, diseño, implementación y evaluación del programa de capacitación.

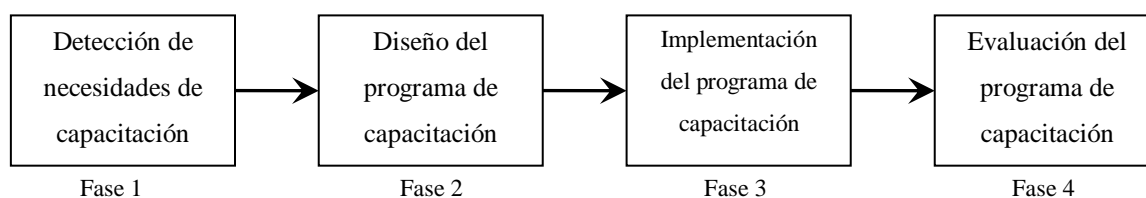


Figura 6. Fases del modelo sistemático de la capacitación y desarrollo profesional.

Fase 1: el ingeniero de software y la organización deben permanecer al pendiente sobre la capacitación que se requiere, cuándo se necesita y qué métodos son los mejores para que el ingeniero de software pueda desarrollar las competencias necesarias.

Fase 2: consiste en diseñar el programa de capacitación y desarrollo. Para que éste pueda tener éxito, además de una efectiva detección de necesidades de capacitación (DNC), se requiere utilizar la información que se obtiene ésta y utilizarla para diseñar un plan de capacitación. El plan de capacitación y desarrollo debe considerar por lo menos: objetivos

de capacitación, deseo y motivación del ingeniero de software, principios de aprendizaje y características de los instructores.

Fase 3: es la implementación del plan de capacitación, para lograr una buena implementación es importante elegir los métodos de instrucción adecuados. Cuando se elige entre varios métodos, una consideración fundamental es determinar cuáles son los más apropiados para adquirir las competencias que se desean. Si la capacitación se concentra en la adquisición de conocimientos técnicos, la conferencia, el aula o la instrucción programada son métodos que pueden funcionar bien. Sin embargo, si la capacitación incluye desarrollar comportamientos del ingeniero de software, otros métodos como la capacitación en el puesto, simulación o instrucción por computadora pueden ser más efectivas.

Fase 4: es la evaluación del programa de capacitación, ésta es necesaria para determinar su eficacia. Existen varios métodos que permiten evaluar si la capacitación mejora el aprendizaje, afecta el comportamiento en el trabajo e influye en el desempeño final de una organización. También, se definen cuatro criterios básicos para evaluar la capacitación: reacción, aprendizaje, comportamiento y resultados.

Para cada una de estas fases existe bastante información que se puede analizar, sin embargo, no es el objetivo de la presente investigación profundizar en cada uno de estos temas.

En la realidad, el desarrollo profesional del ingeniero de software no siempre ocurre de la manera que se plasma en los aspectos teóricos, sino que influyen diferentes factores que pueden facilitar o impedir que las actividades de capacitación y desarrollo profesional se ejecuten y logren su cometido. A continuación se analizan algunos factores que se cree influyen en el desarrollo profesional.

II.4.3 Factores que influyen en el desarrollo profesional

En primer lugar, se debe mencionar que el desarrollo profesional del ingeniero de software es responsabilidad tanto del mismo ingeniero de software como de la organización para la

cual labora. Así, en el desarrollo profesional influyen factores relacionados con la persona, organización y el entorno alrededor de ellos, estos factores pueden propiciar u obstruir el desarrollo profesional del trabajador. En este apartado se explican algunos de esos factores que los investigadores identifican:

Motivación

Uno de los factores más importantes que determinan el éxito en los programas de capacitación y desarrollo profesional es la motivación del trabajador. Pero ¿qué factores influyen en la motivación del trabajador hacia su desarrollo profesional? Muchinsky [Muchinsky, 2002] considera que las acciones y decisiones directivas influyen en la motivación del trabajador con respecto a la capacitación. Explica que el trabajador suele aprender acerca del punto de vista de la organización sobre las actividades de capacitación. Señala la importancia del apoyo de los compañeros y superiores durante las actividades de desarrollo profesional, también que la cantidad de control o participación que tienen el trabajador en el proceso de planeación de la capacitación influye en su motivación para cumplir con la capacitación y su trabajo.

George Bonhander y otros [Bonhander *et al.*, 2001] indican que existen dos condiciones previas para que la capacitación ocurra de manera efectiva: *la buena disposición y motivación del trabajador*. Establece que es posible aumentar la buena disposición de los trabajadores si se les pide que llenen un cuestionario donde indiquen la razón por la cual asisten a la capacitación. Por otro lado, para que el aprendizaje sea óptimo, el trabajador debe ser consciente de la necesidad de las competencias que adquiere durante la capacitación, esto influye en la motivación con la que el trabajador asiste a los cursos de capacitación.

Participación activa

La participación activa del trabajador en las fases del programa de capacitación y desarrollo (DNC, diseño, implementación y evaluación) influye en la efectividad de la capacitación y está relacionado con la motivación del trabajador. Baldwin, Magjuka y Loher [Baldwin *et*

al., 1991] identifican que el trabajador que recibe capacitación esperando alguna forma de evaluación o seguimiento, presenta mayor intención de transferir lo aprendido a las actividades de trabajo. El hecho que los superiores soliciten a los subordinados que preparen un reporte del curso significa que se les hace responsable de su propio aprendizaje y aparentemente se transmite el mensaje de que la capacitación es importante.

Baldwin, Magjuka y Loher [Baldwin *et al.*, 1991] manifiestan que si se permite al trabajador especificar en qué quiere capacitarse, su motivación para aprender aumenta, siempre y cuando reciban el entrenamiento que eligieron. Sin embargo, el trabajador a quien se le permite escoger un curso, pero luego se asigna a otro diferente, se presenta menos motivado que aquellos que no participan en la elección.

Retroalimentación

El trabajador necesita que se le diga cómo lo está haciendo, en qué cosas específicas está destacando y en cuáles necesita mejorar su rendimiento. No decirle estas cosas lo priva de la oportunidad de hacerse cargo de su propio desarrollo y su carrera. Señala Morgan McCall en “High Flyers: Developing the Next Generation of Leaders” [McCall, 1997], que la falta de retroalimentación es un factor que con frecuencia desempeña un papel en el descarrilamiento de personas muy talentosas. Conocerse a sí mismo es crítico para un continuo crecimiento [Michaels *et al.*, 2004].

La retroalimentación cumple una función “configuradora” que ayuda al trabajador a concentrarse en aquello que hace bien o mal, por ejemplo cuando se aprende a andar en bicicleta, por lo general existe una persona que explica lo que se hace bien o mal, esto seguramente permite hacerlo mejor y aprender más rápido. Además de los aspectos informativos, la retroalimentación cumple con la importante función de motivar [Bohlander *et al.*, 2001].

Ambiente laboral

Uno de los factores de mayor influencia en la capacitación y desarrollo profesional del trabajador es el ambiente laboral (también se hace referencia a él como el clima

organizacional). Mathieu, Tannenbaum y Salas [Mathieu *et al.*, 1990] detectan que el trabajador que informa de limitaciones en su trabajo (falta de tiempo, equipamiento y recursos) asiste a la capacitación con menos motivación para aprender. Estos trabajadores tienen menos incentivos para aprender, ya que las nuevas habilidades no siempre pueden aplicarse en su ambiente laboral por las limitaciones mencionadas.

Tannenbaum y Yukl [Tannenbaum *et al.*, 1992] observan que la eficiencia del programa de capacitación puede estar influenciada por acontecimientos que ocurren después de que el participante vuelve al trabajo. Algunos trabajadores terminan los cursos de capacitación con nuevas habilidades y con intención de aplicarlas a su trabajo, pero las limitaciones en el ambiente laboral interfieren con la transferencia real del aprendizaje.

La transferencia de competencias se define como el grado de aplicación efectiva de los conocimientos, habilidades y comportamientos obtenidos en un contexto de capacitación a las actividades de trabajo por parte de los trabajadores participantes.

George Bohlander y otros [Bohlander *et al.*, 2001] dicen que enfocar la capacitación en el trabajador en lugar del instructor o tema de capacitación, puede crear un ambiente laboral que permita el aprendizaje. Además propone seis estrategias que pueden ayudar a generar un clima organizacional de capacitación propicio:

- Utilizar el esfuerzo positivo.
- Eliminar amenazas y castigos.
- Hacer que el trabajador establezca metas personales.
- Ser flexible.
- Diseñar una instrucción interesante.
- Eliminar obstáculos físicos y psicológicos del aprendizaje.

Baumgartel y Jeanpierre [Baumgartel *et al.*, 1972] sugieren que la capacitación es más efectiva cuando la dirección proporciona un clima organizacional de apoyo que anima al trabajador a explorar nuevas ideas y a utilizar el conocimiento aprendido.

Apoyo de superiores

Una parte importante del clima organizacional es el apoyo de los superiores, que a su vez está relacionado con otros factores como la motivación del trabajador y la transferencia de competencias. El trabajador que recibe apoyo de los superiores en las decisiones sobre la capacitación, afronta la capacitación con mayor convencimiento de que ésta es útil para su desarrollo profesional. Los superiores comprensivos discuten el futuro profesional con sus subordinados, establecen metas de capacitación, proporcionan tiempo para que se preparen y les motiva.

El apoyo de los supervisores es un factor ambiental primordial que puede afectar al proceso de transferencia de competencias. El apoyo del supervisor incluye el reforzamiento, el modelado de conductas aprendidas y las actividades de establecimiento de metas. Así, para que el desarrollo profesional tenga éxito se debe contar con el apoyo de los superiores, quienes deben trabajar en diseñar y poner en marcha un sistema de desarrollo profesional [Bohlander *et al.*, 2001].

Actitud

La actitud del trabajador hacia su profesión se manifiesta normalmente en su voluntad para aprender vía cursos formales o de la experiencia obtenida por la práctica en el trabajo. La actitud de la organización hacia el entrenamiento (clima organizacional de actualización) facilita u obstruyen oportunidades de entrenamiento. Sin embargo, es la actitud individual del trabajador la que permite explotar las oportunidades de aprendizaje o sobre ponerse a los obstáculos que las obstruyen. Al mismo tiempo la actitud de la persona hacia el trabajo influye en el desempeño en el mismo. La actitud de una persona puede ser positiva o negativa, en ella influye la cultura que prevalece en la organización [Downey *et al.*, 2008].

Compromiso con las personas

En las relaciones laborales cooperativas se considera como aspecto prioritario la existencia de un compromiso entre la organización y los trabajadores, de carácter mutuo. Este compromiso se desarrolla en estrecha relación con el grado y nivel que ofrecen las

relaciones laborales cooperativas, con los estilos modernos de gestión cooperativa y participativa. Se vincula, igualmente, al reconocimiento del aporte del recurso humano. Se manifiesta en oportunidades, en disposición al trabajo y en cooperación frente los requerimientos de trabajo, que a veces pueden ir más allá de lo ordinario.

Una de las exigencias principales del compromiso por parte de la organización es la capacitación y el desarrollo del trabajador, que implica fundamentalmente, el estímulo y desarrollo de los conocimientos y habilidades que permitan al trabajador lograr los objetivos de calidad y los resultados en su trabajo. Por parte del trabajador, implica la orientación a los objetivos del negocio, el involucramiento en la línea y la excelencia del proceso. Esta orientación supone y exige una participación en la gestión de la calidad en todos los procesos.

Por lo anterior, el compromiso por la capacitación y el desarrollo profesional tanto por parte de la organización como del trabajador, es uno de los factores claves para que el desarrollo profesional ocurra de manera eficiente.

La comunicación

La comunicación es uno de los factores más importantes, ya que ésta influye en el efecto de otros factores como la resistencia al cambio [Wayne, 2005], el compromiso y la participación activa [Baldwin, 1991]. Hoy en día la Internet y el correo electrónicos son dos de los medios de comunicación más explotados en las organizaciones [Hansen, 2006]. La comunicación es importante para abrir espacios para la discusión y solución de conflictos, buscar soluciones colectivas y realizar cambios [Rebeil Corella *et al.*, 2006].

Características de los instructores

George Bohlander y otros [Bohlander *et al.*, 2001] establecen de manera contundente que “*el éxito en cualquier esfuerzo de capacitación depende en gran medida de las habilidades pedagógicas y las características personales de los responsables de la capacitación.*” Un buen instructor hace un esfuerzo adicional y demuestra mayor preparación e instrucción. También proponen una lista de características deseables en el instructor de la capacitación:

- Dominio del tema.
- Adaptabilidad.
- Sinceridad.
- Sentido del humor.
- Interés.
- Cátedras claras.
- Asistencia individual.
- Entusiasmo.

Apoyo familiar

Uno de los principales factores que influyen en la carrera profesional de las personas es el apoyo familiar. El trabajo puede absorber gran cantidad del tiempo del trabajador, lo que resta tiempo que el trabajador pasa con su familia [Bohlander *et al.*, 2001; Reig *et al.*, 2003].

Estos son algunos investigadores y factores que se obtienen de la literatura, es posible que existan más autores y factores que se hayan identificado.

II.4.4 Métodos de capacitación

El diseño de la capacitación del personal comienza con la detección de necesidades de capacitación y culmina con la evaluación de los resultados del programa de capacitación. En los pasos intermedios se encuentran el diseño e implementación del programa de capacitación y desarrollo profesional, estos últimos consideran utilizar un método de capacitación.

Algunos métodos de capacitación son:

- Capacitación en el puesto.
- Capacitación combinada.
- Instrucción programada.
- Capacitación por computadora.
- Seminarios y conferencias.
- Capacitación de aprendices.
- Instrucción escolarizada.
- Métodos audiovisuales.
- Métodos de simulación.
- Representación de roles.

Cada uno de estos métodos tiene sus características particulares, que resultan útiles para unos u otros casos. La revisión de cada uno de estos métodos queda fuera del alcance de la presente investigación. Sin embargo, aquí se presenta algunos aspectos sobre el método mixto de aprendizaje, que consiste básicamente en utilizar varios de los métodos que se enlistan arriba y otros.

Antes de revisar en qué consiste y las características principales del método mixto de aprendizaje, se mencionan algunos aspectos que se relacionan con la capacitación por computadora, la cual supone dos técnicas: la instrucción asistida y la instrucción dirigida.

Un sistema de instrucción asistida por computadora consiste en colocar el material de capacitación en un formato interactivo. Las computadoras permiten hacer ejercicios y prácticas, solucionar problemas, efectuar simulaciones, usar formatos divertidos de instrucción y ciertas formas muy elaboradas de instrucción individualizada [Bohlander *et al.*, 2001].

Mientras que el sistema de capacitación administrada por computadora consiste en utilizar la computadora para generar y calificar pruebas y determinar el nivel de aprovechamiento de la capacitación. Asimismo, este sistema puede seguir el desempeño de los participantes y dirigirlos al material de estudio que satisfaga sus necesidades [Bohlander *et al.*, 2001].

II.4.4.1 Modelo mixto de aprendizaje (blended learning)

En este apartado se realiza una presentación de algunos puntos relevantes sobre el trabajo de David Aguado y Virginia Arranz [Aguado *et al.*, 2005] en el cual realizan un análisis descriptivo sobre el desarrollo de competencias utilizando modelo mixto de aprendizaje (blended learning).

La mayoría de los métodos de capacitación no se diseñan para desarrollar comportamientos o actitudes positivas en los participantes, estos se centran principalmente en la transmitir conocimientos técnicos. En los últimos años se desarrollan las herramientas de *e-learning* (aprendizaje utilizando medios electrónicos como la computadora). Aguado y Arranz [Aguado *et al.*, 2005] resaltan la dificultad de desarrollar el comportamiento de los

participantes a través de medios electrónicos, es por esta razón que proponen combinar métodos de aprendizaje en línea (*e-learning*) con los presenciales, a esta combinación es la que nombran como modelo mixto de aprendizaje.

Aguado y Arranz [Aguado *et al.*, 2005] realizan especial énfasis en la dificultad que representa desarrollar competencias y por otro lado también la dificultad en el uso de sistemas informáticos para el aprendizaje, es decir, que existe doble problemática. En comparación con los métodos de capacitación tradicionales, los métodos *e-learning* presentan las ineficiencias siguientes:

- La posibilidad del alumno discutir y compartir con otros sus experiencias.
- La relación alumno-realidad-personas que forman el contexto de aplicación.
- El contacto con la figura de un tutor cercano que guíe al alumno.
- Las oportunidades de prácticas reales necesarias para lograr el cambio y la mejora de los comportamientos.
- Atención, comprensión y memoria.

El modelo mixto de aprendizaje (*blended learning*) surge como alternativa para cubrir las ineficiencias de las soluciones *e-learning*. Aguado y Arranz [Aguado *et al.*, 2005] establecen que un modelo mixto de aprendizaje debe considerar tres elementos:

- Un modelo instructivo que tenga como objetivo el desarrollo de competencias.
- Una herramienta *e-learning* capaz de soportar el modelo instructivo anterior.
- El apoyo de profesionales para contemplar y optimizar el modelo de aprendizaje anterior.

El reto es construir un modelo instructivo que tenga como característica principal servir como un medio de aprendizaje eficaz que combine métodos presenciales con los métodos *e-learning*. Además, se debe asegurar que lo que se aprenda se lleve a la práctica, que el proceso de capacitación sea personalizado y considere la práctica guiada por un tutor.

En cuanto a la herramienta *e-learning* que proporciona soporte al modelo instructivo, para que ésta sea eficiente y tenga éxito se debe considerar los aspectos metodológicos siguientes:

- Diseño atractivo.
- Diseño de actividades.
- Retroalimentación inmediata.
- Sencillez de uso.
- Calidad de los contenidos.
- Interactividad con el programa.
- Ejercicios.

Finalmente, el modelo mixto de aprendizaje considera la actuación cercana y apoyo de un profesional que permita potenciar y perfeccionar los conocimientos, habilidades y comportamientos adquiridos a través de los métodos de capacitación, a esta persona se le llama tutor.

Marco conceptual

III.1 Introducción

En este capítulo del documento se analiza y define el *marco conceptual del desarrollo de competencias*. Este modelo se construye con el propósito de dar respuesta a la pregunta de investigación: *¿qué factores inhiben o estimulan las actividades de desarrollo de competencias?* Es decir, cuáles son las circunstancias, situaciones o características tanto de la organización como del ingeniero de software que impiden (o en su caso facilitan) que el ingeniero de software pueda y además quiera realizar actividades que le permiten actualizar y desarrollar sus conocimientos, habilidades y comportamientos con el propósito de alcanzar un alto rendimiento en el desempeño de su función laboral. Adicionalmente se propone una agrupación de estos factores de acuerdo a sus características.

El marco conceptual representa a los factores que afectan el desarrollo de competencias del ingeniero de software es uno de los componentes principales del *modelo de desarrollo de competencias para el ingeniero de software* (tema central de la investigación). La finalidad de identificar y representar estos factores es proporcionar un marco de referencia para que la organización y el ingeniero de software conozcan qué factores deben evaluar para determinar si las condiciones son adecuadas para facilitar el desarrollo de las competencias del ingeniero de software.

Para responder la pregunta de investigación *¿qué factores inhiben o estimulan las actividades de desarrollo de competencias?* es necesario revisar algunos conceptos, los cuales analizan en el siguiente apartado. Después, se plantean los objetivos y alcances del marco conceptual que se propone, también se analizan y describen algunos trabajos similares o relacionados con el tema y finalmente se describe el *marco conceptual del desarrollo de competencias*, sus elementos, sus principales ventajas y sus desventajas con respecto a los trabajos que se revisan en la literatura. Los factores que componen al modelo conceptual que se propone, se identifican a través de la revisión de la literatura.

III.2 Conceptos

Antes de continuar con la descripción del marco conceptual, es necesario definir algunos conceptos que se mencionan a lo largo de este apartado.

Modelo

Un modelo es una representación que sirve de referencia para imitar o reproducir un sistema o una realidad compleja, tiene la finalidad de facilitar la comprensión y el estudio del comportamiento de un sistema o una realidad. El modelo debe abstraer aquellos elementos que se consideren importantes para explicar el fenómeno que se está representando.

Existen diferentes tipos de modelo como los matemáticos, físicos, conceptuales, ambientales, económicos, de negocios, de comportamiento social, entre otros. Dependiendo del tipo del modelo, la representación del sistema o realidad se realiza en distintas formas, por ejemplo a través de fórmulas matemáticas, representaciones gráficas, etc.

Modelo conceptual

Un *modelo conceptual* es una representación que muestra un conjunto de *relaciones* entre ciertos *factores* que se cree impactan o conducen a una *condición de interés*. Los cuatro elementos principales que se utilizan para construir un modelo conceptual, son los que se describen a continuación:

Condición de interés: es la situación o estado que se intenta incidir a través de ciertas actividades. En general, la condición de interés es la variable dependiente en la investigación.

Factores: son eventos, situaciones, condiciones, políticas, actitudes, creencias o comportamientos específicos que se cree afectan a la condición de interés. Los factores corresponden a las variables independientes en la investigación.

Actividades: son las acciones que se llevan a cabo para modificar los factores que influyen sobre el estado de la condición de interés. Una actividad puede ser un evento, tratamiento o exposición que ocasiona un cambio en los factores y por consecuencia sobre la condición de interés.

Relaciones: es una conexión que se establece entre actividades, factores y la condición de interés. Se representan con una flecha que apunta en una sola dirección y pueden ir de las siguientes maneras:

- De una actividad a uno o más factores.
- De un factor a uno o más factores.
- De un factor a la condición de interés.

III.3 Estructura

En el planteamiento del problema (ver capítulo uno) se mencionan las preguntas que guían la presente investigación. Es en base a las respuestas a estas preguntas que se forma la estructura que permite desarrollar competencias en el ingeniero de software, es decir el *modelo de desarrollo de competencias para ingenieros de software* que se propone en esta investigación. Este modelo tiene cuatro componentes principales que son los que se describen a continuación:

- **Modelo conceptual del desarrollo de competencias:** se diseña con la finalidad de responder la pregunta: *¿qué factores (o situaciones) pueden obstruir o dificultar el desarrollo de competencias del ingeniero de software?* En este modelo se

representan las condiciones que permiten (o en su caso obstruyen) que el desarrollo de competencias se realice de manera eficiente.

- **Marco de competencias para ingenieros de software:** se diseña con la finalidad de responder la pregunta *¿qué competencias debe desarrollar el ingeniero de software para lograr el desempeño sobresaliente de las actividades de la ingeniería de software?* El marco indica al ingeniero de software y organización qué competencias requiere tener el ingeniero de software para alcanzar un alto desempeño en su actividad laboral.
- **Estrategia:** en total se compone de tres procesos y la finalidad de ella es responder a la pregunta *¿qué actividades se deben realizar para desarrollar las competencias del ingeniero de software?* La estrategia establece los pasos que se deben seguir (tanto el ingeniero de software como la organización) para realizar el desarrollo de competencias.
- **Herramienta de soporte:** se diseña con la finalidad de responder las preguntas *¿cómo puede saber el ingeniero de software que necesita actualizarse?* y *¿qué características debe tener una herramienta que proporcione soporte al ingeniero de software durante el desarrollo de sus competencias?* La herramienta de soporte guía al ingeniero de software durante el desarrollo de sus competencias.

En este capítulo se describe en específico el modelo conceptual para el desarrollo de competencias. En capítulos posteriores se describe el resto de los componentes del modelo de desarrollo de competencias para el ingeniero de software. En la Figura 7 se puede observar la relación que guardan los componentes del modelo de desarrollo de competencias para ingenieros de software, hay que notar que la estrategia está formada por tres procesos: cambio, adaptación del marco de competencias y desarrollo de competencias. En la Figura 7 resalta (línea más gruesa) los recuadros que representan al modelo conceptual (que se explica en este capítulo), se observa que tiene relación directa con el proceso de cambio que forma parte de la estrategia.

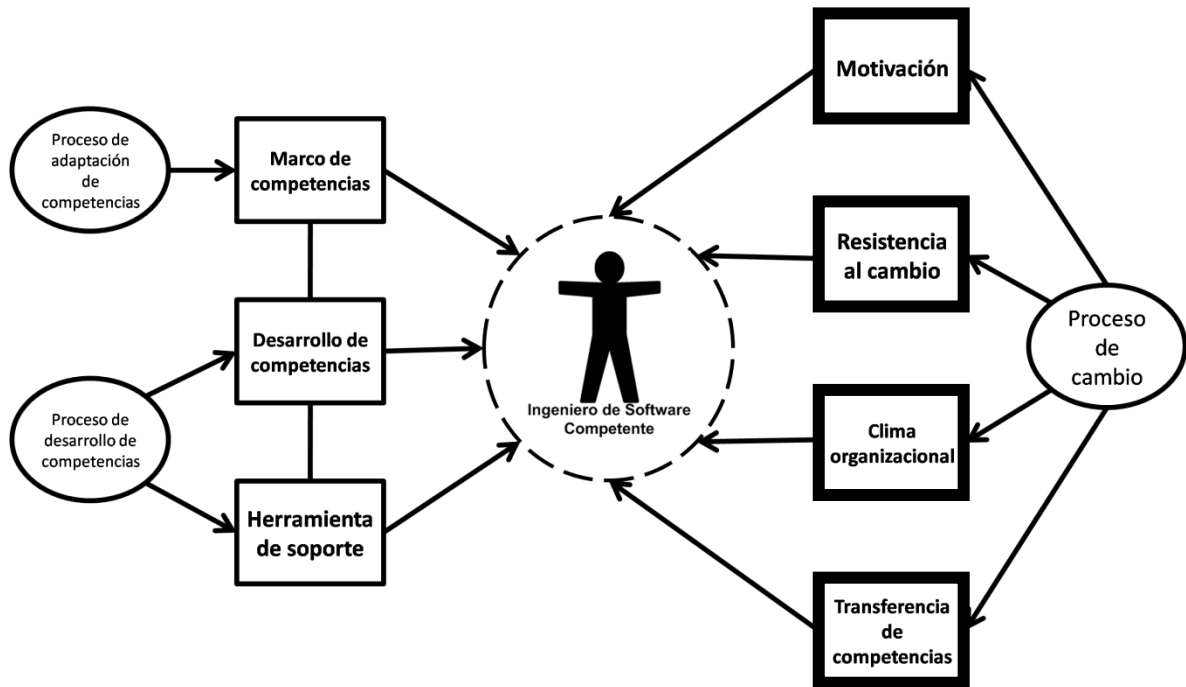


Figura 7. Representación de los componentes del modelo de desarrollo de competencias para ingenieros de software (ubicación del modelo conceptual).

III.4 Objetivos del modelo conceptual

El objetivo principal del modelo conceptual es:

Representar los factores que inhiben o estimulan el desarrollo de las competencias del ingeniero de software.

Los objetivos particulares son:

- Identificar la condición de interés.
- Identificar a través de la revisión literaria los factores que inhiben o estimulan el desarrollo profesional
- Analizar y seleccionar los factores que afectan de manera directa a la condición de interés.
- Representar gráficamente los elementos que componen al modelo así como sus relaciones.

III.5 Alcances

La identificación de los factores que inhiben o estimulan el desarrollo de competencias se realiza a través del análisis de la literatura y de estudios empíricos, queda fuera de los objetivos de esta investigación realizar una evaluación para comprobar el efecto de estos factores sobre la condición de interés.

El objetivo del *modelo de desarrollo de competencias para ingenieros de software* es proporcionar un marco de trabajo que abarque los diferentes aspectos que están involucrados en el desarrollo de competencias del ingeniero de software. Uno de estos aspectos son los factores que afectan el desarrollo de competencias, este apartado se limita a presentar los factores detectados en la literatura.

III.6 Trabajo relacionado

El desarrollo de competencias del ingeniero de software es un tema de investigación relativamente poco estudiado, en la literatura se puede constatar que el estudio de las competencias del ingeniero de software se limita a identificar qué competencias son deseables, sin embargo, prácticamente no existen trabajos que se enfoquen en desarrollarlas, a excepción de P-CMM.

El P-CMM es uno de los esfuerzos más notables que tiene como finalidad el desarrollo de las capacidades de los ingenieros de software. De manera similar que el modelo que se propone en esta investigación, el P-CMM se fundamenta en que el rendimiento organizacional está directamente relacionado con el rendimiento del personal, por esta razón la mejora continua de los ingenieros de software es un aspecto competitivo y fuente de ventaja estratégica para mejorar el desempeño y productividad de las organizaciones desarrolladoras de software.

A pesar de lo anterior el modelo de desarrollo de competencias no se desarrolla para sustituir o competir contra el P-CMM, más bien éste puede complementar o servir de apoyo para implementar algunas prácticas de las áreas de procesos que establece P-CMM. En los

párrafos siguientes se describe con mayor detalle al P-CMM y al mismo tiempo se establece la relación que existe con el modelo que se propone en esta investigación.

El P-CMM, de manera similar a CMMI está organizado en niveles de madurez que a su vez se componen de un conjunto de áreas de proceso (o capacidades), para cada área de proceso define un conjunto de metas, para alcanzar dichas metas es necesario que se implementen ciertas prácticas, así la organización implementa un conjunto de buenas prácticas que le permiten alcanzar las metas correspondientes a un área de proceso, lo que a su vez (al cumplir las áreas de proceso) le lleva a alcanzar cierto nivel de madurez (en el que se encuentran estas áreas de proceso). En la Figura 8 se puede observar la estructura del P-CMM.

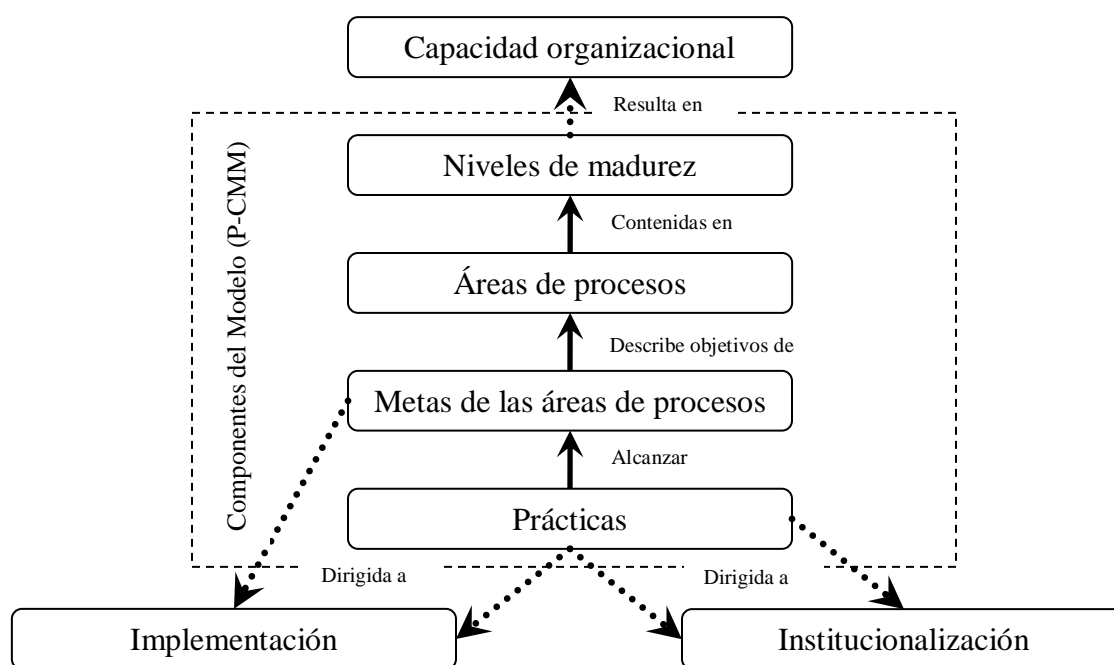


Figura 8. Arquitectura del P-CMM (versión original en Curtis *et al.*, 2001).

En la Tabla V se observan en la columna de la izquierda los cinco niveles de madurez en los que se divide el P-CMM, mientras que en la columna de la derecha se enlistan las diferentes áreas de procesos que abarca cada nivel de madurez. Para que una organización tenga un determinado nivel de madurez, es necesario que cumpla con todas las áreas de

procesos del nivel de madurez correspondiente y las áreas de procesos de los niveles de madurez anteriores.

Tabla V. Niveles de madurez y áreas de procesos del P-CMM (versión original en Curtis *et al.*, 2001).

Nivel de madurez	Áreas de procesos
1. Inicial	
2. Administrado	<ul style="list-style-type: none"> ✓ Compensaciones. ✓ Entrenamiento y desarrollo. ✓ Administración del rendimiento. ✓ Entorno de trabajo. ✓ Comunicación y coordinación. ✓ Personal.
3. Definido	<ul style="list-style-type: none"> ✓ Cultura participativa. ✓ Desarrollo de grupos de trabajo. ✓ Prácticas basadas en competencias. ✓ Plan de carrera. ✓ Desarrollo de competencias. ✓ Planificación del personal.
4. Predecible	<ul style="list-style-type: none"> ✓ Capacitación (tutoría). ✓ Administración de la capacidad organizacional. ✓ Administración cuantitativa del rendimiento. ✓ Evaluaciones basadas en competencias. ✓ Acreditación (creación) de equipos.
5. Optimizado	<ul style="list-style-type: none"> ✓ Innovación continua del personal. ✓ Alineación del rendimiento organizacional. ✓ Mejora continua de las capacidades.

La agrupación de las áreas de procesos en niveles de madurez del P-CMM es para permitir a las organizaciones implementar gradualmente las mejores prácticas de administración de recursos humanos hasta alcanzar la mejora continua de las capacidades (nivel optimizado).

Una de las características del P-CMM es que se delimita a describir el "qué" y no el "cómo" de las prácticas que propone. Estas prácticas indican el "qué" en términos generales con la finalidad de que las organizaciones tengan un margen de maniobra para aplicar con creatividad el "cómo". Por ejemplo, el P-CMM indica que el rendimiento individual debe revisarse en forma periódica, sin embargo no específica con qué frecuencia, qué dimensión se debe revisar o cómo debe realizarse la medición del rendimiento. Las decisiones sobre cómo deben aplicarse las prácticas, se dejan a la organización [Curtis *et al.*, 2001].

Los objetivos del modelo de desarrollo de competencias que se propone en esta investigación están relacionados con algunos de los objetivos de ciertas áreas de procesos en diferentes niveles de madurez. No obstante, el modelo que se propone no establece el “qué” de estas áreas de procesos, sino más bien propone algunas alternativas que indican el “cómo” realizar las prácticas de dichas áreas de procesos.

En la Tabla VI se observan en la columna del lado izquierdo los niveles de madurez del P-CMM, y en las columnas del lado derecho se agrupan en categorías las áreas de procesos que conforman a cada uno de los niveles de madurez. En total son cuatro categorías: desarrollo de la capacidad individual, construir grupos de trabajo y cultura, motivar y administrar el rendimiento y formar al personal.

Tabla VI. Categorías de las áreas de procesos (versión original en Curtis *et al.*, 2001).

Niveles de madurez	Categorías de las áreas de procesos			
	<i>Desarrollo de la capacidad individual</i>	<i>Construir grupos de trabajo y cultura</i>	<i>Motivar y administrar el rendimiento</i>	<i>Formar al personal</i>
5 Optimizado	Mejora continua de la capacidad		Alienación del rendimiento organizacional	Innovación continua del personal
4 Predecible	Evaluación basada en competencias y tutoría	Integración de competencias y creación de grupos de trabajo	Administración cuantitativa del rendimiento	Administración de la capacidad organizacional
3 Definido	Desarrollo de competencias y análisis de competencias	Desarrollo de grupos de trabajo y cultura participativa	Prácticas basadas en competencias y plan de carrera	Planificación del personal
2 Administrado	Entrenamiento y desarrollo	Comunicación y coordinación	Compensación, administración del rendimiento y entorno de trabajo	Personal

Los objetivos y descripción de las áreas de procesos de P-CMM se caracterizan por estar especificados desde la perspectiva de la organización, esto significa que las prácticas del P-CMM están dirigidas o diseñadas para que la organización adopte las mejores prácticas de la administración de recursos humanos y no para indicar las prácticas que puede adoptar el

ingeniero de software para lograr la mejora continua de sus capacidades como indica el P-CMM. A pesar que el P-CMM se diseño para mejorar las capacidades de los ingenieros de software, éste está diseñado desde la perspectiva de la organización y no desde la perspectiva del ingeniero de software.

III.7 Descripción de los elementos del modelo

Como se menciona anteriormente (en el apartado de conceptos de este capítulo) un modelo conceptual está formado por: una condición de interés, factores que afectan esta condición de interés y sus respectivas relaciones. A continuación se describe a cada uno de estos elementos que forman el modelo conceptual del desarrollo de competencias.

III.7.1 Condición de interés

La condición de interés para el presente modelo es el ingeniero de software competente, que es aquel que tiene los conocimientos, habilidades y actitudes, necesarios para desempeñar de manera eficiente las actividades contemporáneas dentro del área de ingeniería de software, cumpliendo con las normas de calidad regidas por la empresa o la demanda del mercado.

Normalmente, se hace referencia al ingeniero de software competente como un ingeniero de software talentoso, es decir aquél que destaca por su dinamismo y gran participación en los proyectos, establece buenas relaciones con sus compañeros, reconoce sus errores, expresa opiniones constructivas, no emite juicios, coopera con el equipo y un sinfín de competencias más que en conjunto con sus conocimientos lo hacen sobresalir entre los demás.

III.7.2 Factores

Algunos de los factores que propician que el ingeniero de software pueda actualizar y desarrollar sus competencias están relacionados con las características de la organización, de la persona y el entorno que envuelve a ambos. Por esto, los factores se pueden agrupar de la manera siguiente:

- Organizacionales.
- Personales.
- Otros.

A continuación se describen los factores que componen a cada uno de estos grupos.

III.7.2.1 Organizacionales

Los factores organizacionales son aquellas situaciones que se generan en la relación que se establece entre la organización y el ingeniero de software. Los factores organizacionales que se representan en el modelo conceptual son:

- Apoyo de la organización.
- Clima organizacional.
- Esquemas de comunicación.
- Estabilidad económica y seguridad de empleo.
- Estilo de liderazgo.
- Políticas internas de la organización.

En los siguientes párrafos se describe a cada uno de estos factores y la forma en que se relacionan con el desarrollo de competencias.

Apoyo de la organización

El apoyo de la organización es la percepción¹ que el ingeniero de software tiene con respecto a la ayuda y el soporte que la organización le proporciona para realizar actividades de capacitación y desarrollo profesional. Si el ingeniero de software percibe que la organización lo apoya en su desarrollo profesional, esto influye sobre otros factores como en la motivación y compromiso del ingeniero de software para efectuar de forma satisfactoria las actividades para la actualización y desarrollo de competencias.

Las actividades de capacitación son más efectivas cuando la organización proporciona las herramientas y el apoyo que motive al ingeniero de software a explorar nuevas ideas y

¹Percepción es una sensación interior que resulta de una impresión material hecha en nuestros sentidos.

utilizar el conocimiento adquirido [Muchinsky, 1994; Biasca, 2005]. Sin el apoyo de la organización, los programas de capacitación y desarrollo no tendrán éxito, por esto es indispensable que la organización tome parte activa en la capacitación y proporcione los recursos necesarios para ello [Wayne, 2005].

Clima organizacional

El clima organizacional es la percepción que el ingeniero de software tiene con respecto a las políticas, prácticas, procedimientos, interacciones y en sí todas las características del entorno de trabajo en el cual se desenvuelve. El clima organizacional influye de manera crítica en el comportamiento del ingeniero de software [Cotton; 2004] y en la efectividad que pueda tener en los programas de capacitación [Tannenbaum *et al.*, 1992].

Esquemas de comunicación

Los esquemas de comunicación se refieren a los medios que utilizan la organización y sus integrantes para transmitir y recibir mensajes. La comunicación es fundamental en cualquier actividad en la cual se requiere transmitir conocimientos, sentimientos, ideas, etc., a otras personas. Los esquemas de comunicación efectivos, continuos y variados son fundamentales para prevenir y disminuir la resistencia al cambio [Wayne, 2005], para establecer compromisos, mejorar el clima organizacional e influir de manera positiva en la motivación y voluntad del ingeniero de software.

Los esquemas de comunicación influyen en la percepción del ingeniero de software sobre el apoyo de la organización [Frías Fernández, 2001], estos también permiten eliminar rumores y comunicación informal de mensajes erróneos que suelen generarse en el interior de las organizaciones. Las revistas institucionales, encuestas, correos electrónicos, portal interno, folletos, diario mural, reuniones, etc., son algunos de los medios que las organizaciones pueden utilizar para establecer un canal de comunicación eficaz con el ingeniero de software.

Estabilidad económica y seguridad del empleo

La estabilidad económica es la situación que se caracteriza por leves variaciones en la condición financiera del ingeniero de software y el mantenimiento pleno del su empleo. La estabilidad económica y seguridad del empleo permiten al ingeniero de software alcanzar cierto grado de equilibrio en su vida personal, de no existir éstas dos condiciones, puede disminuir la motivación del ingeniero de software para capacitarse y su compromiso con la organización.

Estilo de liderazgo

El estilo de liderazgo se refiere a la capacidad que tiene un líder para guiar, controlar e influir de manera interpersonal sobre sus subordinados. Esta influencia se ejerce a través de los procesos de comunicación que el líder utiliza y tiene como objetivo lograr una o varias metas de trabajo.

Las personas que participan en un programa de capacitación y reciben el apoyo de su líder, afrontan la capacitación con mayor convencimiento de que ésta será útil para su desempeño en el trabajo. La motivación y el compromiso del ingeniero de software se incrementa cuando su líder discute con él sobre temas relacionados con su futuro profesional, establecen metas de capacitación y le proporciona el tiempo necesario para prepararse [Muchinsky, 1994].

Políticas internas de la organización

Las políticas internas de la organización son las reglas que se establecen para dirigir funciones y asegurar que el desempeño del ingeniero de software sea de acuerdo con los objetivos establecidos por la organización. Las políticas son las guías de acción de los ingenieros de software y éstas influyen y están relacionadas con el clima y apoyo de la organización. Las políticas que la organización establece con respecto a la capacitación y desarrollo del personal son determinantes para lograr la actualización del ingeniero de software.

III.7.2.2 Personales

Los factores personales son aquellos relacionados con las características propias del ingeniero de software. Algunos factores organizacionales como el apoyo y clima de la organización influyen sobre los factores personales, sin embargo, los factores personales pueden ser diferentes en cada ingeniero de software, aunque estos se desenvuelvan en la misma organización. Los factores personales que considera el modelo conceptual son:

- Compromiso.
- Motivación.
- Personalidad.
- Resistencia al cambio.
- Satisfacción laboral.
- Voluntad.

La descripción y su relación con el desarrollo de competencias de cada uno de estos factores se describen a continuación.

Compromiso

El compromiso se refiere al deber moral o psicológico que el ingeniero de software adquiere hacia una persona, situación o institución. El compromiso entre ingeniero de software y la organización debe ser mutuo, ya que éste es la base fundamental en cualquier relación laboral cooperativa [Frías Fernández, 2001].

El compromiso del ingeniero de software con la organización se manifiesta generalmente en la disposición para trabajar, en la cooperación frente a los retos y en la entrega de resultados que van más allá de lo ordinario. Algunos factores organizacionales como el apoyo, clima, estilo de liderazgo y políticas influyen en el nivel de compromiso del ingeniero de software. La capacitación y el desarrollo profesional exigen un alto nivel de compromiso de parte del ingeniero de software [Frías Fernández, 2001; Wayne, 2005].

Motivación

La motivación es el resultado de un estímulo que provoca una determinada acción o conducta en la persona. La motivación del ingeniero de software es uno de los principales factores que determinan su participación y el éxito de las actividades de actualización y desarrollo de competencias [Mathieu *et al.*, 1990].

Algunos de los factores organizacionales y del programa de capacitación influyen en gran medida en la motivación del ingeniero de software. La participación del ingeniero de software en la elección de los programas de capacitación es una forma de motivarlo para actualizar y desarrollar sus competencias [Quiñones, 1995]. También, el estilo de liderazgo influye en la motivación del ingeniero de software [Muchinsky, 1994], el apoyo de la organización, los esquemas de comunicación, entre otros.

Personalidad

La personalidad se refiere a los patrones del pensamiento y comportamiento que persisten a través del tiempo y las situaciones, y que distinguen a una persona de otra. El tipo de personalidad del ingeniero de software en conjunto con otros factores influye en el éxito de los programas de capacitación [Bohlander *et al.*, 2001].

Resistencia al cambio

La resistencia al cambio es una conducta natural que se produce con frecuencia en las organizaciones, sobre todo cuando no se hace un manejo adecuado del cambio. Es indispensable que se planeen actividades para afrontar la resistencia al cambio cuando se implementan un programa de capacitación, éstas pueden ser difíciles sobre todo cuando se requiere modificar la actitud de las personas [Wayne, 2005].

La comunicación juega un papel importante en la reducción de la resistencia al cambio. La organización debe considerar el uso de esquemas de comunicación eficaces, continuos y variados. Es de vital importancia que la comunicación se dé en ambos sentidos, la

participación activa del personal en la planeación del cambio ayuda a reducir la resistencia a éste [Wayne, 2005].

Satisfacción laboral

La satisfacción laboral es la actitud general del ingeniero de software hacia su trabajo, dicha actitud se basa en las creencias y valores que el ingeniero de software desarrolla durante el desempeño de su trabajo. La satisfacción laboral influye sobre la voluntad del ingeniero de software para realizar las actividades de capacitación, si no existe satisfacción laboral, el ingeniero de software no puede percibir los beneficios de actualizarse en un trabajo que no cumple con sus expectativas [Muchinsky, 2002].

Voluntad

La voluntad es la disposición que el ingeniero de software exhibe para efectuar de manera intencionada actividades de capacitación y autodesarrollo de competencias, por encima de las dificultades, los contratiempos y el estado de ánimo [Bohlander *et al.*, 2001].

III.7.2.3 Otros factores

Algunos otros factores que se considera importante representar en el modelo conceptual son:

- Calidad del curso de capacitación.
- Claridad de los roles.
- Participación activa.
- Reconocimiento.
- Retroalimentación.
- Transferencia de competencias.

Calidad del curso de capacitación

Es probable que la elección y planeación de los programas de capacitación se haya efectuado de manera eficiente, sin embargo, cuando el ingeniero de software toma un curso

de capacitación de poca calidad, los objetivos del programa de capacitación se comprometen y es posible que no se alcancen. De aquí, que además de la adecuación elección y planeación de los programas, se elijan cursos de calidad comprobada que ayuden a alcanzar los objetivos planteados [Bohlander *et al.*, 2001].

Claridad de roles

La claridad en la definición de los roles o puestos de trabajo de la organización es fundamental para lograr determinar de manera más acertada las necesidades de capacitación del ingeniero de software [Muchinsky, 2002].

Participación activa

Este factor se refiere a la participación del ingeniero de software en la planeación de los programas para la capacitación. Cuando se permite al ingeniero de software participar activamente en los programas de capacitación desde su concepción, elección y planeación, su motivación por aprender aumenta, siempre y cuando reciba la capacitación que eligió. Por el contrario, cuando se permite al ingeniero de software participar en la elección y planeación del programa de capacitación, pero después se asigna a cursos que no eligió, el ingeniero de software estará menos motivado y es probable que no se alcancen los objetivos de la capacitación, incluso aprenderá menos que aquellos ingenieros de software que no participaron en la elección de los cursos [Baldwin *et al.*, 1991; Muchinsky, 1994].

Reconocimiento

El reconocimiento a ingeniero de software por el esfuerzo y el éxito alcanzado en la ejecución del programa de capacitación, es un factor que influye en la motivación del ingeniero de software para continuar capacitándose [Bohlander *et al.*, 2001].

Retroalimentación

La retroalimentación es el mecanismo mediante el cual se toman los resultados obtenidos durante la ejecución de un programa de capacitación, para utilizarlos como entradas de

información que permitan mejorar las características de los futuros programas de capacitación [Wayne, 2005].

Transferencia de competencias

La transferencia de competencias se refiere a la habilidad para aplicar los conocimientos, habilidades y comportamientos adquiridos durante la capacitación en situaciones reales de trabajo. Las personas que participan en programas de capacitación se enfrentan a un problema, se les pide que aprendan algo en un ambiente (el entrenamiento) y que lo utilicen en otro (el trabajo) [Goldstein, 1991].

El apoyo de un supervisor es un factor importante para que se pueda efectuar el proceso de transferencia del conocimiento [Baldwin *et al.*, 1991]. Una vez que el ingeniero de software concluye su período de capacitación, el apoyo del supervisor se debe centrar en el reforzamiento de los conocimientos adquiridos, el modelado de conductas aprendidas y en el establecimiento de metas.

III.7.3 Representación gráfica

La representación gráfica del modelo conceptual tiene la función de resumir (gráficamente) los factores que pueden afectar al ingeniero de software competente. En la Figura 9 se observa la representación gráfica del modelo conceptual, en ella se puede observar en el centro de la figura al ingeniero de software competente (condición de interés), a su alrededor se muestran cuatro recuadros que resaltan (línea más gruesa) y que representan a los factores que afectan de manera directa al ingeniero de software competente, mientras que el resto de los recuadros representan los factores indirectos y las flechas una de las múltiples posibles relaciones que pueden existir entre estos factores.

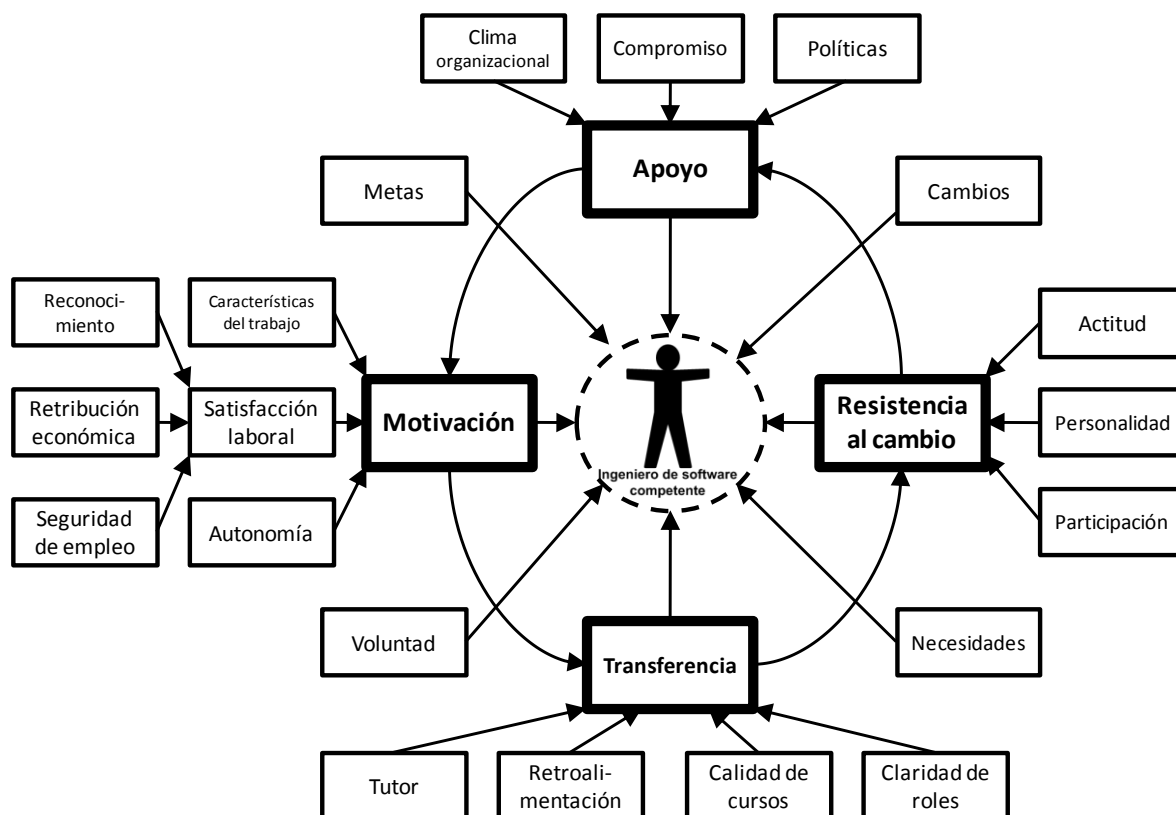


Figura 9. Representación gráfica del modelo conceptual del desarrollo de competencias.

III.8 Forma de uso

Se identifica que el ingeniero de software puede enfrentar diferentes situaciones durante su vida laboral con respecto a su actualización y desarrollo como profesional. Éstas están relacionadas con la *empleabilidad* del ingeniero de software que “*es la habilidad que tiene para conseguir y conservar un empleo, elegir otro cuando lo desee o pierda el que tiene e integrarse de manera más fácil al mercado laboral en diferentes períodos de su vida*”². En todos estos períodos, un nivel alto de desarrollo de competencias es esencial para alcanzar el objetivo deseado (conseguir, conservar, progresar o cambiar de empleo).

²“Resolución sobre el Desarrollo de Recursos Humanos”. Conferencia General de la Organización Internacional del Trabajo, 88ª reunión. Ginebra, junio de 2000. Párrafo 9.

En este apartado se describen las características principales de cuatro escenarios de los posibles en los que el ingeniero de software se puede encontrar en algún momento de su vida laboral.

III.8.1 Escenario 1: conseguir empleo

Este escenario corresponde al período en el que el ingeniero de software intenta conseguir empleo por primera vez (por lo menos en el área de ingeniería de software). Para lograrlo requiere entre otras cosas de esfuerzo, compromiso y disponibilidad para trabajar. La habilidad que el ingeniero de software tenga para conocerse, formarse y “venderse” es fundamental para que incremente la posibilidad de adquirir un empleo en ingeniería de software.

Generalmente, el ingeniero de software en esta situación es joven recién egresado o incluso aún estudiando, suele caracterizarse por el entusiasmo y compromiso por trabajar. Normalmente, el ingeniero de software que busca empleo por primera vez no tiene la certeza sobre si cumple los requerimientos que las empresas exigen o si es capaz de responder a las exigencias del trabajo, por esto surgen preguntas como *¿qué características buscan en mí las empresas?, ¿qué puedo ofrecer para que me empleen?, ¿qué debo saber para hacer bien mi trabajo?*

Ya sea para ingresar a una empresa o iniciar una, es importante que el ingeniero de software conozca sus competencias y las competencias que las empresas de desarrollo de software demandan. También, debe conocer los procesos para actualizar y desarrollar competencias y poder cumplir los requerimientos del puesto al que aspira, ya que estos últimos cambian constantemente, debido por un lado a que cambian las características de los ingenieros de software que egresan y buscan empleo, y por otro lado cambian las necesidades económicas y sociales por lo que las empresas de desarrollo de software ofrecen nuevos productos y servicios [Cruz *et al.*, 1999].

Para que el ingeniero de software consiga un trabajo se deben cumplir por lo menos dos condiciones: 1) debe existir una oportunidad de empleo y 2) el ingeniero de software debe cumplir los requisitos para ese empleo. La primera condición difícilmente la puede

controlar el ingeniero de software, sin embargo la segunda condición es en la que puede hacer algo para conseguir emplearse. Aquí es donde el desarrollo de competencias juega un papel importante para la empleabilidad del ingeniero de software, ya que al incrementar el nivel de su competencia también aumenta su posibilidad de ingresar al mundo laboral.

Una característica de la empleabilidad del ingeniero de software es que puede aspirar a emplearse en diversas organizaciones del sector público o privado de las diferentes ramas de la producción, además puede aspirar a trabajar en la industria y empresas pequeñas, medianas o grandes. Por otro lado, también tiene la posibilidad de crear su propia empresa, dedicarse a la instrucción o ser consultor de productos y servicios relacionados con el software. Existe un amplio número de posibles competencias que el ingeniero de software puede desarrollar, dependiendo del tipo de actividad que vaya a realizar y de las características de la empresa en la que desea trabajar, de aquí la importancia que el ingeniero de software defina sus objetivos personales y busque emplearse en una empresa que tenga objetivos similares y estos se puedan alinear.

Forma de uso del modelo de desarrollo de competencias

Bajo las condiciones del escenario uno (conseguir empleo), el modelo de desarrollo de competencias para ingenieros de software lo puede utilizar por un lado la organización que ofrece el empleo y por otro el ingeniero de software que aspira a ese empleo.

El uso que puede hacer la organización del modelo de desarrollo de competencias para ingenieros de software es tomar como base el marco de competencias que propone el modelo y definir qué competencias se requieren para el puesto en el cual ofrece el empleo, es decir, realizar un perfil de competencias para ese puesto. Después, la organización debe medir el nivel de competencia de los aspirantes para cada una de las competencias que se requieren para ese puesto (se especifican en el perfil de competencias del puesto). El resultado de la medición del nivel de competencia de cada uno de los aspirantes, sirve para evaluar quién de los aspirantes tiene mayor probabilidad de éxito en ese puesto.

Por otro lado, el ingeniero de software que aspira a un determinado puesto que ofrece alguna organización, puede realizar con ayuda del modelo de desarrollo de competencias para ingenieros de software una autoevaluación que le permita identificar las competencias que tiene en ese momento y si la organización tiene un perfil de competencias para el puesto que ofrece, el ingeniero de software puede evaluar si es un buen candidato o no para ese puesto. Además, es posible realizar un plan de desarrollo de competencias y consultar algunas estrategias para desarrollar las competencias necesarias para cubrir los requisitos del puesto al que aspira.

III.8.2 Escenario 2: conservar el empleo

El escenario dos corresponde al período en el cual el ingeniero de software tiene la necesidad de adquirir competencias que le permitan cumplir con los requisitos que exige el puesto de trabajo en el que actualmente se encuentra y así poder conservarlo. Hay que recordar que de acuerdo al modelo de los factores involucrados en la obsolescencia de habilidades de Fossum y otros [Fossum *et al.*, 1986] los requisitos del trabajo y las competencias del profesional cambian con el tiempo debido a factores como el desarrollo tecnológico.

Algunas organizaciones acostumbran a dar contratos temporales (comúnmente de tres meses) a los empleados nuevos, como un período de “*prueba*”. Una vez que el ingeniero logra consolidarse, la empresa proporciona un contrato más extenso en tiempo. De no cumplir con los requerimientos del puesto, es probable que la organización decida proporcionar otro contrato temporal, cambiarlo de puesto o inclusive despedirlo. Sin embargo, a partir del momento que el ingeniero de software comienza a trabajar inicia el período en el que requiere desarrollar las competencias necesarias para cumplir eficazmente con los requisitos del puesto que desempeña y poder así conservar el empleo.

En esta situación pueden presentarse con una gran cantidad de variantes, que dependen de las condiciones a las que se enfrenten el ingeniero de software y la organización. En algunas de estas situaciones el ingeniero de software puede sentir que tiene su puesto de trabajo asegurado por largo tiempo de tal manera que no ve la necesidad de actualizarse

para permanecer con ese puesto. En un caso contrario, puede existir una situación en la cual el ingeniero de software sienta que corre el riesgo de perder el puesto de trabajo, por lo que puede considerar necesario actualizar sus competencias, en estas situaciones dependen de la actitud que el ingeniero de software asuma.

Por lo general, existe diferencia entre las competencias que tiene el ingeniero de software y las que requiere para desempeñar eficazmente el puesto de trabajo, por lo tanto es necesario que realice constantemente actividades de actualización y desarrollo de competencias. No obstante, en ocasiones es difícil para el ingeniero de software percibir esta necesidad, o simplemente no sabe cómo cubrirla.

En este escenario la organización juega un rol muy importante, aunque en principio pareciera que el principal interesado y responsable en el desarrollo del ingeniero de software es el mismo ingeniero de software, la organización también debe estar interesada y compartir esta responsabilidad. Debe estar interesada porque su competitividad depende en gran medida del nivel de competencia del ingeniero de software y debe compartir la responsabilidad para facilitar al ingeniero de software las condiciones que permitan alcanzar el estado competente deseado.

Forma de uso del modelo de desarrollo de competencias

Bajo las condiciones de este escenario, es importante la colaboración y apoyo entre la organización y el ingeniero de software. La participación de la organización es en primer lugar a través de la constante actualización de los perfiles de competencias. Mantener actualizados los perfiles de competencias permite al ingeniero de software conocer de manera acertada las competencias que requiere desarrollar. Es importante también que la organización proporcione el apoyo necesario al ingeniero de software durante el proceso de desarrollo de las competencias que necesita para desempeñar su puesto actual.

El ingeniero de software debe tener la voluntad necesaria para periódicamente realizar autoevaluaciones de sus competencias y compararlas con las que requiere para el puesto en el que trabaja, el resultado de este análisis debe utilizarse para definir un plan de desarrollo

de competencias. El modelo de desarrollo de competencias para ingenieros de software específica los factores que frenan o inhiben el desarrollo de competencias, es importante que tanto la organización y el ingeniero de software participen de manera activa en detectar y disminuir el efecto negativo que causan los factores que se proponen en el modelo u otros que hayan identificado.

III.8.3 Escenario 3: progresar en la misma organización

Este escenario corresponde al período en el que el ingeniero de software logra desarrollar competencias que le permiten cumplir eficazmente las actividades que demanda el puesto de trabajo en el que se encuentra actualmente, de tal manera que, ya sea por motivación o satisfacción personal, profesional o cualquier otra razón, requiere cambiar a un puesto que le exija un reto y responsabilidad mayor, además que le demande utilizar nuevas o diferentes competencias.

Existen diferentes motivos por los cuales el ingeniero de software tiene la necesidad y el deseo de conseguir un mejor puesto de trabajo que el actual, algunos de estos motivos son superación o satisfacción personal, profesional, económica y social. Por lo general, el ingeniero de software busca que el nuevo puesto represente un reto mayor o le proporcione una mejor retribución económica. En otros casos, debido a condiciones de la organización es necesario que el ingeniero de software tenga que cubrir diferentes roles en el proceso de desarrollo de software. Aunque existen competencias que son iguales para los diferentes roles, es necesario desarrollar algunas otras que son específicas para cada rol.

Por lo general, en este período de la vida laboral del ingeniero de software, a éste ya se le considera un ingeniero de software con cierto grado de madurez y experiencia lo cual debería ayudarle a detectar con menor dificultad las necesidades de actualización. Aunque, dependiendo de diversos factores como la edad, puede llegar el momento en el que al ingeniero de software no sienta necesidad, interés o motivación por realizar más actividades de actualización y desarrollo de competencias.

Forma de uso del modelo de desarrollo de competencias

Bajo las condiciones de este escenario, la forma de uso del modelo de desarrollo de competencias es similar al escenario anterior. Se requiere gran cooperación y apoyo entre la organización y el ingeniero de software. La organización debe actualizar constantemente los perfiles de competencias, orientar al ingeniero de software a detectar su déficit de competencias para el nuevo puesto al que aspira y facilitarle opciones de cursos para el desarrollo de esas competencias.

El ingeniero de software tiene que realizar una autoevaluación para determinar qué y en qué nivel de competencia se encuentra, comparar las competencias que tiene en ese momento contra las competencias que el perfil de competencias actualizado del puesto al que aspira, de acuerdo a los resultados realizar un plan de desarrollo de competencias y finalmente seguir el plan de desarrollo de competencias.

Es de gran importancia que las condiciones para el desarrollo de competencias sean las adecuadas, el modelo de desarrollo de competencias indica qué factores pueden afectar, la organización en cooperación con el ingeniero de software realizan evaluaciones que permitan medir la presencia de estos factores y realizar acciones que permitan disminuir el efecto negativo de los factores que se detectaron.

III.8.4 Escenario 4: cambiar a otra organización

Este escenario tiene un planteamiento similar al anterior, sólo que en este caso el ingeniero de software no tiene la posibilidad de aspirar a un mejor puesto de trabajo en la organización en la que labora actualmente, por las mismas limitaciones de la organización, esto hace necesario que tenga que buscar mejores oportunidades en otras organizaciones.

Pueden existir distintos factores que originen la salida del ingeniero de software de una organización. Puede ser que la organización ya no cumple con las expectativas o los objetivos del ingeniero de software, por lo que el ingeniero de software considere conveniente cambiar de organización para obtener mejores posibilidades de desarrollo personal y profesional. Otro caso es que, por cualquier motivo, el ingeniero de software no

haya cumplido los requisitos del puesto y éste haya sido despedido. Sin embargo, sin importar cuál sea la razón, al cambiar de organización y de puesto, si es el caso, también cambian las competencias que requiere para desempeñar su trabajo. Por esto, también en esta situación es importante la actualización y desarrollo de nuevas competencias.

La principal diferencia con el primer escenario (encontrar trabajo) es que el ingeniero de software ha experimentado y conoce mejor el mundo laboral, lo que le permite ser más consciente de sus capacidades; además de haber desarrollado ciertas competencias que le facilitan su reinserción a otro empleo.

Forma de uso del modelo de desarrollo de competencias

Las condiciones de este escenario combinan algunas condiciones de los anteriores. El ingeniero de software realiza una autoevaluación para conocer que competencias tiene en ese momento y compara contra los perfiles de competencias de los puestos que aspira en otras organizaciones.

En cada una de las cuatro situaciones planteadas anteriormente, se identifica que a lo largo de la vida laboral del ingeniero de software puede presentarse en distintas maneras la necesidad de actualizar y desarrollar competencias.

En los ingenieros de software las barreras más fuertes para su desarrollo como profesional son las actitudes o comportamientos de ellos mismos y de la organización, más que el mismo avance tecnológico. Aunque, factores como el avance tecnológico, la globalización y los programas de mejora continua conducen a la obsolescencia profesional del ingeniero de software, estos no son necesariamente los mismos factores que afectan directamente al desarrollo del ingeniero de software. La disposición o voluntad del ingeniero de software para adoptar y adquirir nuevas competencias es un factor más definitivo para su desarrollo como profesional [Hansen, 2006].

Si el ingeniero de software no tiene la voluntad para realizar las actividades para la actualización y desarrollo de sus competencias, con la presencia de alguna dificultad o contratiempo, puede disminuir sus esfuerzos en el ejercicio de dichas actividades, lo que

puede llevar a no alcanzar los resultados esperados. Generalmente, en esta situación se argumentan motivos como “*no tengo tiempo*”, “*estoy cansado*”, “*no me beneficia*”, “*no es lo que necesito*” o cualquier otra explicación que lo exonere de realizar cualquier actividad de capacitación.

Marco de competencias

IV.1 Introducción

En este capítulo se analiza y define el *marco de competencias para el ingeniero de software*. El marco de competencias tiene el propósito de responder a la pregunta: *¿en qué se debe desarrollar o capacitar el ingeniero de software para tener éxito profesional en la ingeniería de software?* En otras palabras, cuáles son las competencias que facilitan al ingeniero de software realizar de manera eficiente y sobresaliente las actividades del desarrollo de software.

El marco de competencias para el ingeniero de software es uno de los componentes del modelo de desarrollo de competencias para el ingeniero de software. La finalidad es indicar a la organización y principalmente al ingeniero de software qué competencias puede adquirir durante su desarrollo profesional, estas competencias deben facilitar su desempeño en la industria del software.

En la Figura 10 se observa la relación que existe entre los componentes del modelo de desarrollo de competencias, en ella se observa que resalta el recuadro (línea más gruesa) que representa al marco de competencias, se observa que tiene relación directa con el proceso de adaptación del marco de competencias que forma parte de la estrategia.

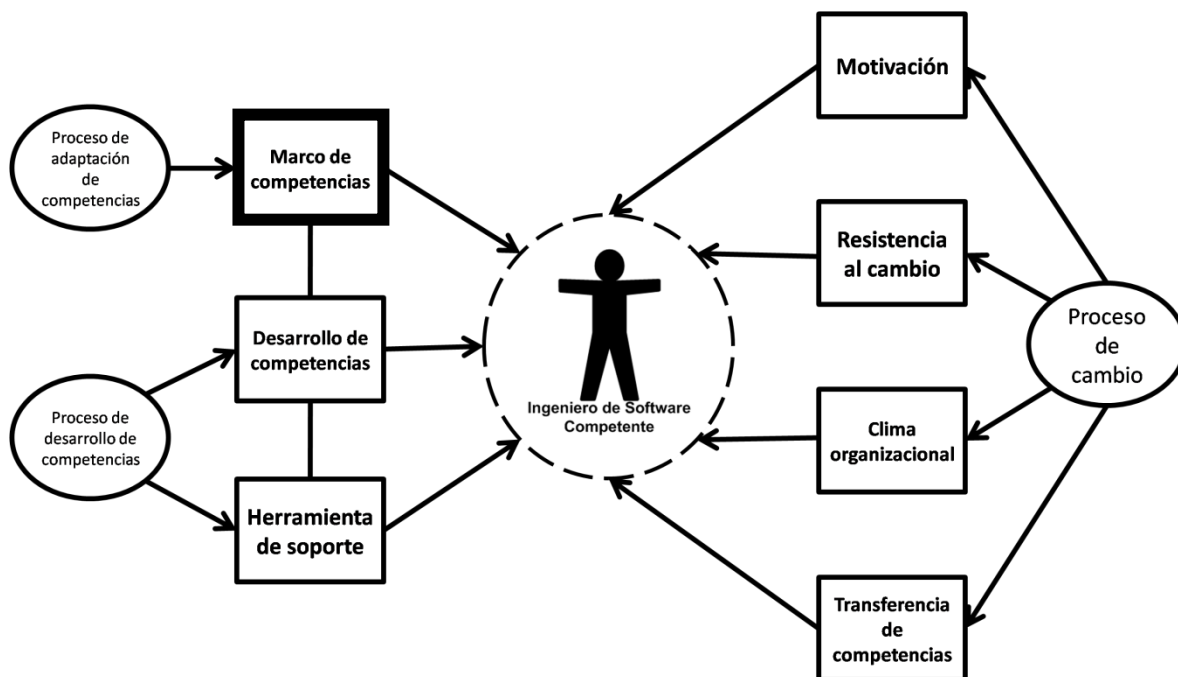


Figura 10. Representación de los componentes del modelo de desarrollo de competencias para ingenieros de software (Marco de competencias).

En los siguientes temas se describen las características principales del marco de competencias, se indican qué competencias forman el marco y por qué. Además se propone una escala para medir el nivel de competencia que tiene el ingeniero de software.

IV.2 Conceptos

Un marco de competencias (también referido en la literatura como modelo de competencias) para ingenieros de software define al conjunto de conocimientos, habilidades y comportamientos que éste requiere para alcanzar un alto rendimiento y el desempeño sobresaliente de las actividades contemporáneas de la ingeniería de software. Para elaborar el marco de competencias para el ingeniero de software es necesario definir qué actividades, roles e interacciones ocurren en el proceso de desarrollo de software.

En la literatura existen diferentes formas de clasificar a las competencias, por convenir a esta investigación se les clasifica en tres categorías generales:

- a) *Técnicas*: son aquellas que están relacionadas con la aplicación de los procedimientos correspondientes a la ingeniería de software. Por ejemplo, obtener los requerimientos del sistema.
- b) *Sociales*: son aquellas que describen una cualidad del ingeniero de software para interactuar con su entorno. Por ejemplo, la habilidad para trabajar en equipo.
- c) *Personales*: son aquellas que describen características particulares del ingeniero de software. Por ejemplo, la capacidad de aprender él solo.

Estas categorías, cada una con un enfoque específico, están interrelacionadas, y colectivamente, forman la base para identificar y mapear las competencias claves para que el ingeniero de software se mantenga competente. La necesidad de que los ingenieros de software piensen y actúen reflexivamente es fundamental en este marco de competencias. La reflexión involucra no sólo la habilidad de aplicar de forma rutinaria un proceso o método para realizar una tarea, sino también la capacidad de adaptarse al cambio, aprender de las experiencias, pensar y actuar con actitud crítica [Rychen, 2002].

IV.3 Descripción

A continuación se describen los criterios que se siguen para elegir las competencias que forman parte del marco de competencias para el ingeniero de software. También se muestra el modelo del desarrollo de software (perspectiva técnica y social) que se toma de referencia para identificar las competencias claves. Después se describen las competencias que se eligieron y finalmente se explica la valoración (escala) que se utiliza para dichas competencias.

IV.3.1 Criterios de elección

Para la elección de competencias claves se utiliza un enfoque que se basa en las necesidades del mercado, en los ingenieros de software y en las organizaciones que los contratan. Con la idea de identificar los criterios para la elección de las competencias claves se formulan las preguntas siguientes: *¿qué deben poseer los ingenieros de software para*

interactuar en un equipo de trabajo heterogéneo?, ¿qué competencias deben poseer para encontrar y retener un trabajo?, ¿qué tipo de cualidades de adaptación requieren para mantenerse al tanto de la tecnología cambiante?

Antes de continuar, es importante mencionar que el marco de competencias aquí descrito se relaciona con competencias individuales y no con capacidades colectivas de organizaciones o grupos.

Para que el ingeniero de software se sienta exitoso requiere tener: seguridad en el empleo, sueldo aceptable, salud personal y un entorno social amigable. Por otro lado, la empresa necesita satisfacer las demandas del mercado y mantener una productividad económica rentable, para lograr estos objetivos se tiene que considerar que la suma de las competencias individuales afecta la habilidad para alcanzar las metas compartidas entre individuo y organización.

El ingeniero de software necesita desarrollar competencias que le permitan adaptarse a un contexto que se caracteriza por el cambio, la complejidad y la interdependencia. Estas competencias deben ser adecuadas para un entorno en el cual:

- La tecnología cambia rápida y continuamente, y aprender a trabajar con ella, no requiere dominio único de los procesos, sino también capacidad de adaptación.
- La transformación de las sociedades es diversa y fragmentada, y las relaciones interpersonales requieren de mayor contacto con personas diferentes a uno.

Así, las competencias claves para el ingeniero de software deben permitirle adaptarse al entorno cambiante y heterogéneo. Estas características del entorno exigen al ingeniero de software diferentes cualidades. Sin embargo, las competencias claves son aquellas de valor particular, que tienen utilidad en diferentes áreas y son necesarias para todos los ingenieros de software.

- La primera condición, sobre las que deben valorarse las competencias, se relaciona con beneficios medibles para fines tanto económicos como sociales y personales.

- La segunda condición es que las competencias que se selecciona deben ser útiles en un amplio espectro de contextos, es decir, deben ser aplicables a múltiples áreas de la vida.
- La tercera condición es elegir aquellas competencias que son de uso específico para un rol, ocupación o tarea en particular que desempeñe el ingeniero de software en el desarrollo de software.

IV.3.2 Actividades y roles

Para seleccionar las competencias claves que necesita tener el ingeniero de software, es necesario conocer qué actividades e interacciones realiza con otras personas durante el desarrollo de software. Es en base a éstas que se determina el grupo de conocimientos, habilidades y comportamientos que garantizan el desempeño eficiente del ingeniero de software.

En el capítulo dos (en el apartado: *desarrollo de software basado en la persona*) se indican las actividades y roles que el ingeniero de software realiza durante el desarrollo de software, tanto desde la perspectiva técnica como desde la perspectiva social. Esas actividades y roles son las que se utilizan para formar el modelo del desarrollo de software que funciona como referencia para decidir qué competencias seleccionar (apartado siguiente en este capítulo).

En la Figura 11 se puede observar las actividades y roles del ingeniero de software desde la perspectiva técnica. Existen dos actores: el cliente (o usuario final) y el ingeniero de software. Las actividades se agrupan en fases de acuerdo al Proceso Unificado de Rational (Rational Unified Process, RUP), estas fases son: *inicio*, *elaboración*, *construcción* y *transición*, que forman las columnas en la Figura 11. Además, algunas actividades se agrupan dentro de las actividades de soporte que también indica el RUP, se observa que las actividades de soporte abarcan las diferentes fases del desarrollo de software.

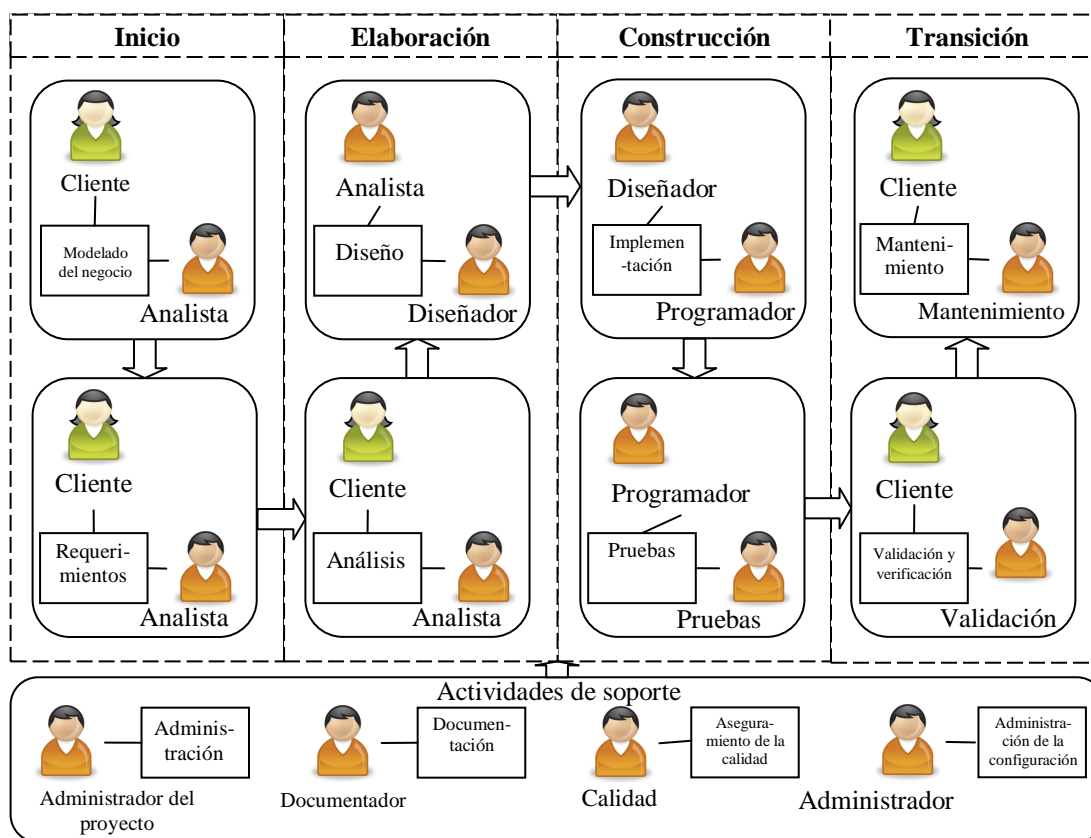


Figura 11. Modelo del desarrollo de software desde la perspectiva técnica.

En la Figura 12 se representan las actividades y roles del ingeniero de software desde una perspectiva social [Sawyer, 2008]. En este caso existen tres actores principales: el cliente (o usuario final), el alto mando (directivo, jefe, etc.) y el ingeniero de software. Las actividades sociales no se agrupan en fases, ya que estas pueden ocurrir en diferentes momentos durante el desarrollo del software.

En estos modelos del desarrollo de software se representa las actividades y roles del ingeniero de software, los productos generados por estas actividades no se representan, por no ser de interés para la finalidad que se busca, identificar competencias.

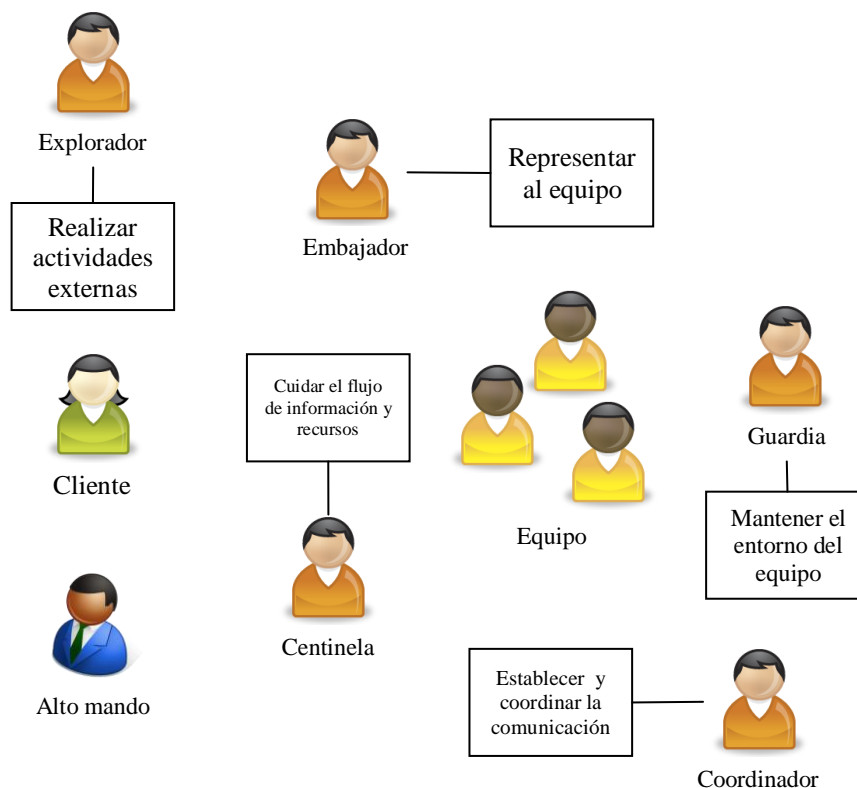


Figura 12. Modelo del desarrollo de software desde la perspectiva social.

IV.3.3 Competencias

En este apartado se describe a cada una de las competencias claves para el ingeniero de software. Estas competencias están agrupadas en técnicas, sociales y personales. A continuación se explican algunos aspectos relevantes sobre estas competencias.

IV.3.3.1 Competencias técnicas

El ingeniero de software necesita crear y adaptar el conocimiento y las habilidades a su contexto de operación. Esto demanda cierta familiaridad con el conocimiento y las herramientas, así como un entendimiento sobre cómo cambian, en qué se pueden aplicar a su ambiente de trabajo y cómo se pueden utilizar para alcanzar metas más amplias. En este sentido, el conocimiento técnico no es solamente un mediador pasivo, es un instrumento que el ingeniero de software utiliza para interactuar con su ambiente. De aquí, la necesidad de mantenerse al día con los avances tecnológicos, de adaptar herramientas para sus propios

propósitos y mantenerse competitivo en el mercado. De aquí que, los conocimientos establecen cómo el ingeniero de software entiende y se hace competente en el mundo, cómo enfrenta la transformación y el cambio y cómo responde a los desafíos de largo plazo. Al usar el conocimiento de manera interactiva se abren nuevas posibilidades en la manera cómo los ingenieros de software perciben y se relacionan con su ambiente laboral.

A partir de la competencia técnica se hace una segunda clasificación como se indica a continuación:

Conocimiento e información técnica

La competencia de conocimiento e información técnica es necesaria como base para comprender opciones, formar opiniones, tomar decisiones y llevar a cabo acciones informadas y responsables. Usar el conocimiento y la información de manera interactiva requiere que el ingeniero de software:

- Reconozca y determine lo que no sabe.
- Identifique, ubique y acceda a fuentes apropiadas de información.
- Evalúe la calidad, propiedad y el valor de dicha información, así como sus fuentes.
- Organice el conocimiento y la información.

Uso de la tecnología

El uso de la tecnología requiere de un conocimiento de nuevas formas de interacción de los ingenieros de software con su labor diaria. Identificar las herramientas de soporte a cada rol o puesto, puede ayudar a mejorar la productividad de todas las funciones de la empresa, además de mejorar el flujo de información dentro y entre las unidades del negocio. Una organización que pretenda ser efectiva debe de explotar y administrar todas estas tecnologías para dar un valor agregado a toda la organización. Implementarla en todas las etapas de la empresa (entrada, desarrollo y salida), ya que esto les permitirá evolucionar hacia el siguiente nivel: las empresas en la era de la información.

IV.3.3.2 Competencias sociales

En ambientes como el del ingeniero de software es importante manejar bien las relaciones interpersonales para beneficio mutuo y para construir nuevas formas de cooperación. Una de las fuentes potenciales de competitividad es la diferencia en las competencias de distintos grupos para construir y beneficiarse del capital social. Las competencias en esta categoría son necesarias para que los ingenieros de software aprendan, vivan y trabajen con otros.

A partir de la competencia social se hace una segunda clasificación tomando como base la clasificación realizada por la *Organisation for Economic Co-operation and Development* (OECD) en su artículo competencia clave para el conocimiento de la sociedad [Rychen, 2002] como se indica a continuación:

Relación interpersonal

Esta competencia permite a los ingenieros de software iniciar, mantener y manejar relaciones personales con, por ejemplo, amigos, colegas y clientes. Relacionarse bien no es sólo un requerimiento para cohesión social sino, cada vez más, para el éxito económico conforme las compañías y economías cambiantes enfatizan en la inteligencia emocional. Esta competencia supone que los individuos pueden respetar y apreciar los valores, las creencias, culturas e historias de otros para crear un ambiente en el que se sientan bienvenidos, sean incluidos y puedan crecer. Para relacionarse bien con otros se requiere:

- Empatía, adoptar el rol de la otra persona e imaginar la situación desde su perspectiva. Esto lleva a la autoreflexión cuando, al considerar una amplia gama de opiniones y creencias, los ingenieros de software reconocen que lo que se da por sentado en una situación no es necesariamente compartido por los demás.
- El manejo efectivo de las emociones, conocerse a sí mismo e interpretar de forma efectiva sus propios estados emocionales y motivacionales no evidentes y los de los demás [Rodríguez-Jacobo, 2004].

Cooperar y trabajar en equipo

Existen ciertas metas que no las puede alcanzar un solo ingeniero de software, se requiere de grupos como equipos de trabajo. La cooperación exige que cada ingeniero de software tenga ciertas cualidades. Cada uno debe poder equilibrar su compromiso con el grupo y sus metas con sus propias prioridades y debe poder compartir el liderazgo y apoyar a otros. Los componentes específicos de esta competencia incluyen:

- La habilidad de presentar ideas y escuchar las ideas de otros.
- Un entendimiento de las dinámicas del debate y el seguimiento de una agenda.
- La habilidad de construir alianzas tácticas y sostenibles.
- La habilidad de negociar.
- La capacidad de tomar decisiones que permitan diferentes opiniones.

Manejar y resolver conflictos

En todos los aspectos de la vida ocurren conflictos, ya sea en el hogar, el lugar de trabajo o la comunidad y sociedad. El conflicto es parte de la realidad social, una parte inherente de las relaciones humanas. Surge cuando dos o más individuos o grupos se oponen uno al otro por necesidades, intereses, metas o valores divergentes. La clave para manejar efectivamente un conflicto es enfrentarlo y resolverlo, y no negarlo. Para ello es necesario considerar los intereses y las necesidades de otros y las soluciones en las que ambas partes ganen. Para que los ingenieros de software participen activamente en el manejo y la resolución de conflictos, necesitarán poder [Rodríguez-Jacobo, 2003]:

- Analizar los elementos y los intereses en juego (ej. poder, reconocimiento de méritos, división del trabajo, equidad), los orígenes del conflicto y el razonamiento de todas las partes, reconociendo que hay diferentes posiciones posibles.
- Identificar áreas de acuerdo y áreas de desacuerdo.
- Volver a plantear el problema.
- Priorizar las necesidades y metas, decidiendo lo que están dispuestos a dejar de lado y bajo qué circunstancias.

IV.3.3.3 Competencias personales

Esta competencia permite que los ingenieros de software actúen de forma autónoma para participar efectivamente en el desarrollo de la sociedad y para funcionar bien en diferentes esferas de la vida incluyendo el lugar de trabajo, la vida familiar y la vida social. En general, la autonomía requiere de una orientación hacia un futuro y un conocimiento del ambiente que lo rodea, de las dinámicas sociales y los roles que él juega y desea jugar.

A partir de la competencia personal se hace una segunda clasificación tomando como base la clasificación realizada por la OECD en su artículo competencia clave para el conocimiento de la sociedad [Rychen, 2002] como se indica a continuación:

Desarrollarse en el ambiente laboral

Esta competencia requiere que los ingenieros de software entiendan y consideren el contexto más amplio de sus acciones y decisiones, por ejemplo que comprendan patrones de comportamiento, sus estructuras, cultura, prácticas y reglas formales e informales y expectativas y roles que juegan dentro de la misma, incluyendo una mayor comprensión de las leyes y regulaciones, y también de las normas sociales no escritas, los códigos morales, los modales y el protocolo. Identificar las consecuencias directas e indirectas de sus acciones. Elegir entre diferentes cursos de acción reflexionando en sus consecuencias potenciales en relación con las normas y metas individuales y compartidas. Esta competencia requiere por ejemplo que los ingenieros de software [Rodríguez-Jacobo, 2003]:

- Comprendan patrones de comportamiento, sus estructuras, cultura, prácticas y reglas formales e informales y expectativas y roles que juegan dentro de la misma, incluyendo una mayor comprensión de las leyes y regulaciones, y también de las normas sociales no escritas, los códigos morales, los modales y el protocolo. Complementa un entendimiento de los derechos con conocimiento de los límites de las acciones.
- Identificar las consecuencias directas e indirectas de sus acciones.

- Elegir entre diferentes cursos de acción reflexionando en sus consecuencias potenciales en relación con las normas y metas individuales y compartidas.

Desarrollo personal

Esta competencia aplica el concepto de manejo de proyectos en los individuos. Requiere que los individuos interpreten la vida como una narrativa organizada y le den significado y propósito en un ambiente cambiante en el que la vida, con frecuencia, se ve fragmentada. Esta competencia supone una orientación hacia el futuro, implicando tanto optimismo como potencial, pero también raíces fuertes dentro de lo posible. Por ejemplo, los ingenieros de software deben poder:

- Definir un proyecto y fijar una meta personal.
- Identificar y evaluar tanto los recursos a los que se tiene acceso, como los recursos necesarios (ej. tiempo y dinero).
- Priorizar y refinar las metas.
- Balancear los recursos necesarios para satisfacer metas múltiples.
- Aprender de acciones pasadas, proyectando resultados futuros.
- Monitorear el progreso, haciendo los ajustes necesarios conforme se desarrolla el proyecto.

Derechos y límites

Esta competencia es importante para la seguridad de los propios intereses de los ingenieros de software. A pesar de que muchos derechos y necesidades se establecen y se protegen en las leyes o en contratos, son los individuos quienes identifican y evalúan sus derechos, necesidades e intereses. Por un lado, esta competencia se relaciona con derechos y necesidades autodirigidos; también se relaciona con los derechos y las necesidades del individuo como miembro de una colectividad (ej. participando activamente en instituciones democráticas y en procesos políticos locales y nacionales). Esta competencia implica la habilidad, por ejemplo, de:

- Comprender los propios intereses (ej. en una elección).
- Conocer las reglas y principios escritos para basar un caso.
- Construir argumentos para que nuestros derechos y necesidades sean reconocidos
- Sugerir arreglos o soluciones alternativas.

El marco de competencias que se propone define un conjunto de conocimientos, habilidades y comportamientos clave, con especial énfasis en las destrezas blandas (*soft skills*). Es necesario que la empresa que lo adopte defina e integre las competencias específicas que requiere el ingeniero de software para permitir a ésta conseguir sus objetivos y metas.

IV.3.4 Escala

La escala de competencias se utiliza para medir e indicar el grado de dominio que tiene una persona sobre una competencia específica. Para algunas competencias no es necesario contar con un dominio total de ella, por lo que para indicar el grado de competencia que se requiere, se definen las escalas o niveles que se muestran en la Tabla VII.

Tabla VII. Escalas para medir el grado de desarrollo de una competencia específica.

Nivel	Porcentaje de requerimientos que un ingeniero de software satisface con respecto a los requerimientos que la compañía establece para una competencia específica.
0	Menos del 25 %.
1	Más del 25 % y menos del 50 %.
2	Más del 50 % y menos del 80 %.
3	Más del 80 %.

IV.3.5 Representación

La representación del marco de competencias se muestra en la Tabla VIII, en ella se puede observar a las competencias agrupadas por tipos y categorías. Además en las columnas de la derecha se indica el nivel (o grado) de desarrollo que se propone para cada uno de los roles de la ingeniería de software.

V.1 Introducción

Según la definición de la Real Academia Española, en su tercera acepción, una estrategia es “*un proceso regulable, un conjunto de reglas que aseguran una decisión óptima en cada momento.*” Mientras que WordNet 2.0 de Princeton University define como “*un plan sistemático de acción*”. De las diferentes definiciones de la palabra *estrategia* se abstraen dos partes fundamentales: los objetivos y las actividades (o acciones) que permiten alcanzar los objetivos.

En esta investigación, la *estrategia* especifica a un conjunto de actividades definidas y planificadas, que tienen como objetivo asegurar el desarrollo de competencias del ingeniero de software. Una actividad abarca un conjunto de tareas que tienen como propósito generar un cierto producto o resultado. Estas tareas las ejecuta la organización, el ingeniero de software u otra persona o sistema.

Uno de los objetivos particulares de la investigación es que el modelo que se propone, pueda responder a las preguntas *qué* competencias y *cómo* desarrollar esas competencias. El propósito general de la estrategia es responder a la pregunta *¿cómo desarrollar las competencias del ingeniero de software?* La estrategia que aquí se propone se compone de tres procesos que tienen los objetivos específicos siguientes:

- 1) *Proceso de cambio*: se diseña para cumplir tres objetivos principales. El primero es indicar cómo realizar un manejo adecuado del cambio basado en el ingeniero de software. El segundo es indicar cómo identificar los factores que potencialmente pueden o están afectando al desarrollo profesional del ingeniero de software. Finalmente, indicar cómo crear un clima organizacional de actualización que propicie al desarrollo profesional del ingeniero de software.
- 2) *Proceso de adaptación del marco de competencias*: indica cómo la organización puede construir un marco de las competencias para que el ingeniero de software lo adopte como guía para saber qué competencias. En esta investigación se propone un marco de competencias con algunas competencias claves para el éxito laboral en la ingeniería de software, a este marco de competencias las organizaciones desarrolladoras de software lo puede adaptar a sus necesidades.
- 3) *Proceso de desarrollo de competencias*: se diseña para indicar cómo realizar el desarrollo de competencias. Este proceso se compone de una serie de actividades de desarrollo profesional.

Los procesos que se indican arriba tienen la función de guiar y proponer alternativas a la organización y al ingeniero de software sobre cómo realizar el desarrollo de competencias. Sin embargo, la estrategia general (junto con sus procesos) se puede modificar de acuerdo a las necesidades de cada organización y del ingeniero de software, ya sea agregando, modificando o quitando actividades que propone. Cada organización puede necesitar una estrategia diferente y también una misma organización puede requerir hacer cambios en las actividades de los procesos cada vez que las ejecuten, por lo que la estrategia que aquí se presenta es un marco general que la organización puede o no requerir modificar.

En la Figura 13 se puede observar la relación que tiene la estrategia con el resto de los componentes del modelo de desarrollo de competencias. En la Figura 13 se observa los tres procesos que forman la estrategia, el proceso de cambio indica el “cómo” del modelo conceptual, el proceso de adaptación del marco de competencias indica el “cómo” del marco de competencias, y finalmente el proceso de desarrollo de competencias indica el

“cómo” del desarrollo de competencias, que se realiza a través de la herramienta de soporte. Estos tres procesos se resaltan en la Figura 13 con recuadros con línea más gruesa.

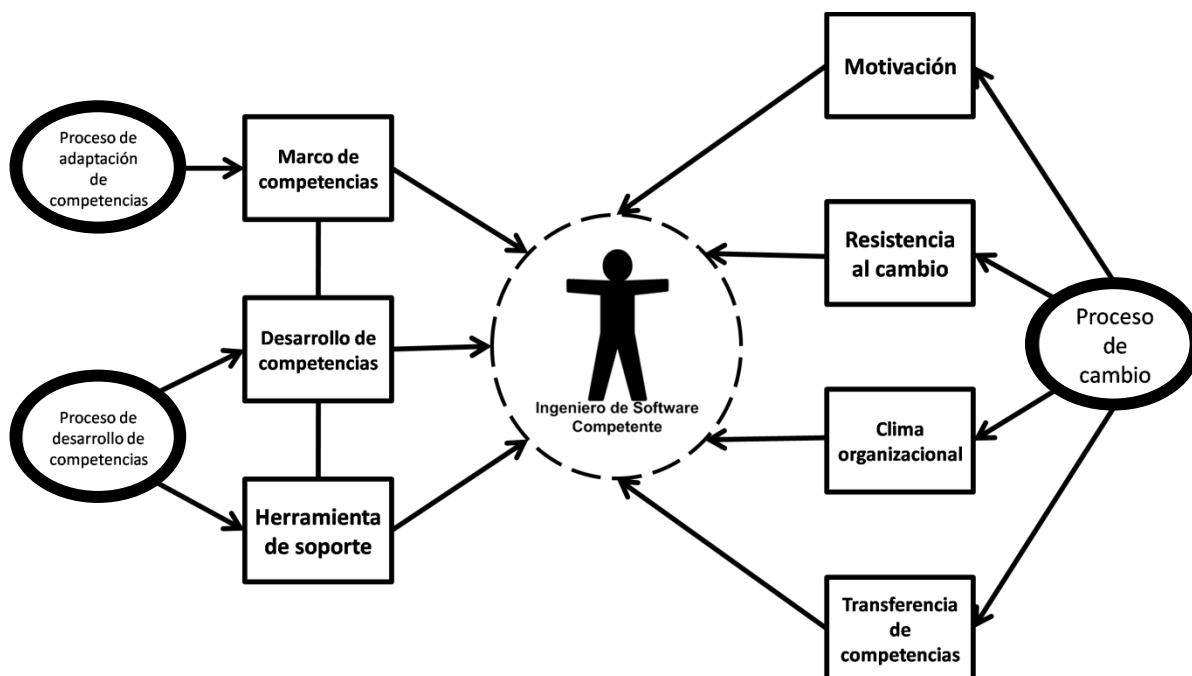


Figura 13. Representación de los componentes del modelo de desarrollo de competencias para ingenieros de software (Estrategia).

Además, cada proceso se compone de una serie de actividades, las cuales a su vez están formadas por tareas.

El modelo de desarrollo de competencias se diseña para que lo utilicen tanto el ingeniero de software como la organización. La estrategia que se propone en esta investigación incluye actividades que realizan ambas partes (ingeniero de software y organización). Algunas de estas actividades requieren de la participación de ambos, mientras que otras las puede realizar un solo elemento (el ingeniero de software o la organización). Los procesos de cambio y de adaptación del marco de competencias contienen actividades que realiza, en su gran mayoría, la organización. Mientras que el proceso de desarrollo de competencias está formado por actividades que realiza principalmente el ingeniero de software.

En adelante, se hace referencia a la ejecución de las actividades que forman a los procesos como *programa para el desarrollo de competencias del ingeniero de software*.

En los apartados siguientes se analizan y describen las actividades que forman a cada uno de los procesos. Al final de la descripción de cada actividad se presenta una tabla que resume las características principales de la misma, la información que contienen estas tablas son: el objetivo de la actividad, persona responsable, colaboradores, documentos, tareas, cuándo inicia y cuándo termina dicha actividad. El propósito de mostrar este resumen es facilitar al lector, ingeniero de software y organización de manera compacta las características de cada actividad.

V.2 Proceso de cambio

V.2.1 Descripción

El proceso de cambio que se propone se basa en la “*estrategia para la generación del cambio basada en la persona*” que propone Josefina Rodríguez [Rodríguez-Jacobo, 2003], esta estrategia se compone de siete fases que se indican en la Figura 14, en ella se puede observar que el proceso general del cambio es un ciclo que contempla las fases siguientes: contacto, generación de clima, planificación, acción y desarrollo, observación y seguimiento, reflexión y análisis, y finalmente la comunicación que se ejecuta durante todo el ciclo.

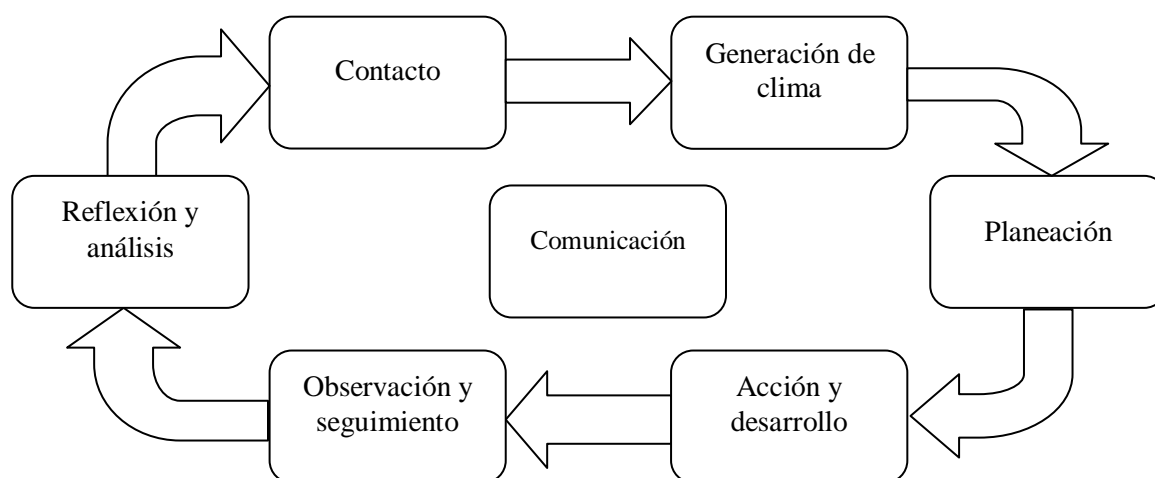


Figura 14. Proceso general del cambio basado en la persona que propone [Rodríguez Jacobo, 2003] (versión original en Rodríguez Jacobo, 2003).

El proceso de cambio que se propone en esta investigación busca cumplir tres objetivos primordiales:

- 1) Realizar un manejo del cambio basado en el ingeniero de software.
- 2) Identificar los factores que afectan el desarrollo de competencias.
- 3) Generar un clima organizacional de actualización propicio para el desarrollo de competencias.

Objetivo 1 del proceso de cambio: manejo del cambio

Realizar un manejo adecuado del cambio permite disminuir la posibilidad de que las personas que están involucradas en el programa de desarrollo de competencias (causa del cambio) presenten resistencia al cambio, que de acuerdo al modelo conceptual de desarrollo de competencias (que se propone en esta investigación) es una de las principales causas de fracaso en los programas de mejora continua [Moitra, 1998; Stelzer *et al.*, 1998; Mathiassen *et al.*, 2005]. Para conseguir el objetivo uno del proceso de cambio, las actividades y tareas deben cumplir lo siguiente:

- Concientizar a las personas involucradas sobre los efectos (positivos y negativos) del cambio.
- Contar con una persona con conocimientos sobre cambio organizacional (*agente de cambio*).
- Hacer participar democráticamente en la planificación del programa de desarrollo de competencias a todas las personas que resultan afectadas por el cambio.
- Motivar a las personas involucradas a que se preparen para realizar el cambio.

La buena comunicación juega un papel importante para realizar un manejo adecuado del cambio.

Objetivo 2 del proceso de cambio: identificar factores

Detectar y conocer los factores (o situaciones) potenciales que pueden afectar el desarrollo de competencias permite tomar decisiones bien informadas que conduzcan a generar un clima organizacional de actualización propicio. Así, la identificación de factores que afectan al desarrollo de competencias es un prerrequisito para seleccionar las acciones que permiten generar en la organización un ambiente propicio para el desarrollo de competencias.

Para detectar la presencia de estos factores es necesario aplicar una serie de instrumentos que permiten medir el clima organizacional de actualización. La medición del clima organizacional de actualización refleja la opinión que tienen los ingenieros de software sobre las condiciones que caracterizan a la organización, sin embargo no se trata de identificar las condiciones laborales existentes en la organización, sino la opinión que las personas tienen sobre esas condiciones [Castillo Aponte, 2006]. Un instrumento para medir el clima organizacional de actualización consiste de un cuestionario que tiene la finalidad de identificar la percepción de los ingenieros de software sobre las condiciones laborales en la organización [Méndez Álvarez, 2006].

Objetivo 3 del proceso de cambio: generar clima organizacional de actualización

No basta con conocer la percepción del ingeniero de software sobre las condiciones del clima organizacional de actualización. Además, es necesario realizar las acciones que permiten cambiar la percepción que tiene el ingeniero de software sobre esas condiciones.

El clima organizacional propicio para el desarrollo de competencias debe tener como características principales el apoyo, cooperación y compromiso entre los ingenieros de software y la organización, bajo un ambiente de respeto, en el cual ingenieros de software y jefes pueden hablar sobre el desarrollo profesional sin prejuicios. El ingeniero de software percibe que tiene la facilidad y el apoyo de la organización para desarrollarse profesionalmente y que la capacitación no está dada solo por los cursos de capacitación sino que también en el autodesarrollo que cada ingeniero de software realiza para

complementar aquellos aspectos que le permiten obtener resultados satisfactorios en su vida y trabajo [Pérez Roquela, 2002].

En base a la estrategia de cambio que propone Josefina Rodríguez [Rodríguez-Jacobo, 2003] y a los objetivos que se plantean, el proceso de cambio que aquí se propone se compone de las actividades siguientes:

- Conocer y concientizar a la organización.
- Crear clima organizacional de actualización.
- Planear el programa de desarrollo de competencias.
- Evaluar al programa de desarrollo de competencias.
- Proporcionar seguimiento al programa de desarrollo de competencias.
- Comunicar resultados.

En la Figura 15 se puede observar la relación que existe entre las actividades de la estrategia de cambio. La Figura 15 indica que las actividades forman un ciclo continuo, es decir, las actividades deben ejecutarse continuamente, además se observa que existen dos actividades (colocadas al centro de la Figura 15) que se ejecutan durante todo el ciclo.

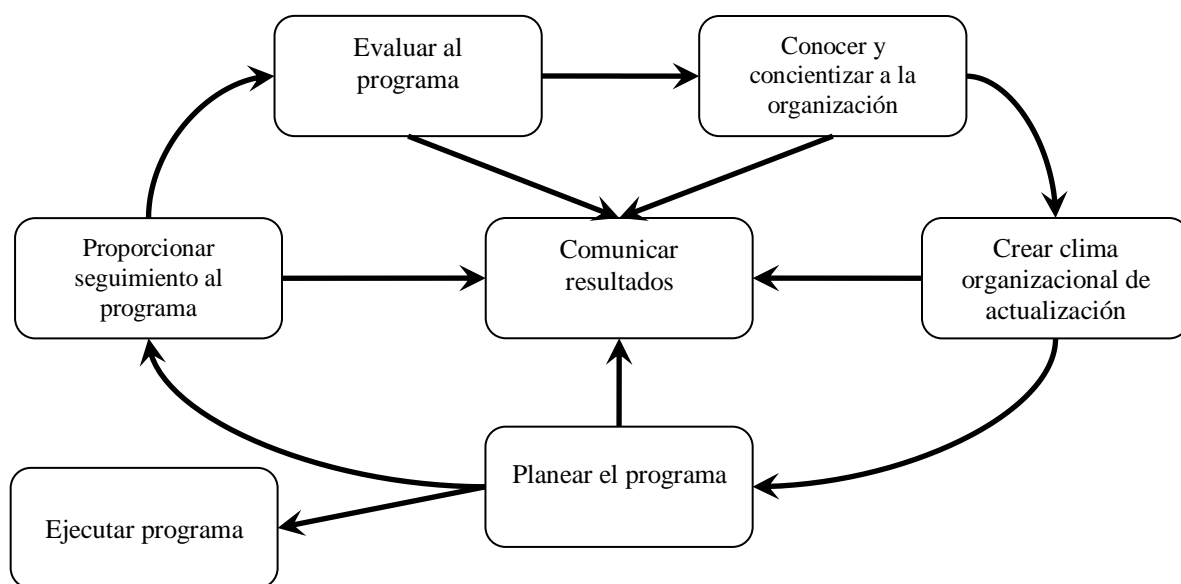


Figura 15. Representación gráfica de las actividades del proceso de cambio.

Si se realiza una analogía de las actividades del proceso de cambio que se propone en esta investigación (Figura 15) con la estrategia de cambio de Josefina Rodríguez [Rodríguez-Jacobo, 2003] (Figura 14), la correspondencia entre las actividades del proceso y las fases de la estrategia por similitud de objetivos queda de la manera como se muestra en la Tabla IX:

Tabla IX. Correspondencias entre fases de la estrategia de [Rodríguez-Jacobo, 2003] y las actividades de la estrategia de cambio que se propone.

Fase	Actividad
Contacto	Conocer y concientizar a la organización
Generación de clima	Crear clima organizacional de actualización
Planeación	Planear el programa
Observación y seguimiento	Proporcionar seguimiento al programa
Reflexión y análisis	Evaluar al programa
Comunicación	Comunicar resultados

En el proceso de cambio que se propone no existe una actividad que corresponda a la fase de acción y desarrollo de la estrategia de cambio de [Rodríguez-Jacobo, 2003], la razón de esto es que la fase de acción y desarrollo corresponde a las actividades que forman a los procesos de adaptación del marco de competencias y de desarrollo de competencias.

Agente de cambio

La mayor parte de las actividades que componen al proceso de cambio, se realizan a nivel organizacional, por lo que debe existir una persona responsable de ellas, a esta persona se le conoce como *agente de cambio* y es designado por la organización.

El *agente de cambio* es la persona responsable del programa para el desarrollo de competencias del ingeniero de software, puede ser una persona interna o externa a la organización. La ventaja para una persona interna es que puede conocer de antemano ciertos aspectos de la organización que son relevantes para la implantación del programa de desarrollo de competencias, aunque por otro lado es probable que no domine áreas como comportamiento organización, gestión del cambio y otras importantes para realizar manejo adecuado del cambio. Un poco contrario a lo que puede suceder si el agente de cambio es

una persona externa a la organización (siempre y cuando sea experto en cambio organizacional), si esta persona tiene conocimientos sobre procesos de cambio, su trabajo más fuerte al iniciar el cambio consiste en estudiar las características de la organización.

V.2.2 Actividades

A continuación se describen las características y tareas que forman a cada una de las actividades del proceso de cambio, en total son siete actividades de las cuales dos de ellas son actividades de soporte (o control).

Actividad 1: Conocer y concientizar a la organización

La actividad *conocer y concientizar a la organización* se refiere al conjunto de tareas que permiten al agente de cambio transmitir a los integrantes de la organización (incluyendo a los ingenieros de software) la información siguiente: en qué consiste el programa desarrollo de competencias, por qué es importante, cómo puede beneficiar a la organización y al ingeniero de software, para qué se utiliza, quién lo requiere, cómo se realiza y otras preguntas o dudas que tengan las personas involucradas.

Los objetivos de esta actividad son:

- Conocer a las personas involucradas (esto incluye conocer sus costumbres, relaciones, cultura, formación, rutinas, sistemas de control, valores, etc.).
- Concientizar a las personas sobre la importancia del desarrollo de competencias.

El agente de cambio es la persona responsable de ejecutar esta actividad y requiere la colaboración de todos los integrantes de la organización (incluyendo directivos, jefes, mandos medios, ingenieros de software, etc.). Esta actividad parte del hecho que el agente de cambio conoce el modelo de desarrollo de competencias que se propone en esta investigación y que domina temas relacionado con el desarrollo profesional, competencias y cambio organizacional. También se supone que la organización acepta la adopción e implantación del modelo de desarrollo de competencias para el ingeniero de software,

además facilita los recursos necesarios al agente de cambio para realizar las distintas actividades que conforman al modelo.

Las tareas necesarias para realizar esta actividad se describen a continuación.

Tarea 1: Solicitar reunión para la presentación del programa de desarrollo de competencias.

El agente de cambio acuerda una reunión con las personas involucradas (directivos, jefes, mandos medios, ingenieros de software, etc.), para presentar el programa de desarrollo de competencias, es importante la presencia y participación de todos ellos. Si los directivos, jefes y/o mandos medios no están presentes en la reunión, el ingeniero de software puede percibir que no es importante lo que ahí se va decir.

Tarea 2: Preparar presentación del programa de desarrollo de competencias.

El agente de cambio, de acuerdo a los recursos disponibles (diapositivas, folletos, proyector, etc.), prepara una presentación que permita transmitir los aspectos más importantes del desarrollo de competencias. Debe considerar las características del público al cual está dirigida la presentación para decidir qué información va transmitir.

Tarea 3: Presentar el programa de desarrollo de competencias.

El agente de cambio durante la reunión con los directivos, jefes, mandos medios e ingenieros de software, presenta las características del programa de desarrollo de competencias.

Tarea 4: Conocer a las personas involucradas.

El agente de cambio acude al área de trabajo y conversa con cada una de las personas que están involucradas en el programa (directores, jefes, mandos medios, ingenieros de software). Es importante que logre conocer los temores, expectativas y dudas sobre el desarrollo de competencias de cada una de las personas. El agente debe acudir con cada persona las veces que sean necesarias y en horarios en los cuales la persona esté disponible.

Tarea 5: Aclarar dudas sobre el programa de desarrollo de competencias.

El agente de cambio conoce cuáles son las principales dudas sobre el programa de desarrollo de competencias y las aclara por medio de reuniones, ejemplos, simulaciones, etc.

En la Tabla X se muestra a manera de resumen la información sobre la actividad conocer y concientizar a la organización.

Tabla X. Resumen de la actividad conocer y concientizar a la organización.

Conocer y concientizar a la organización	
Objetivo:	<ul style="list-style-type: none"> • Conocer las personas involucradas. • Concientizar a las personas de la importancia del desarrollo de competencias.
Responsable:	Agente de cambio (interno y/o externo).
Colaboradores:	Todas las personas que conforman la organización.
Tareas	<ul style="list-style-type: none"> • Solicitar reunión para la presentación del programa de desarrollo de competencias. • Preparar presentación del programa de desarrollo de competencias. • Presentar programa de desarrollo de competencias. • Conocer a las personas involucradas. • Aclarar dudas sobre el programa de desarrollo de competencias.
Documentos:	<ul style="list-style-type: none"> • Documentación sobre el modelo de desarrollo de competencias para ingenieros de software. • Documento de presentación del programa para el desarrollo de competencias. • Reporte sobre los principales temores, dudas y expectativas de las personas involucradas.
Inicia:	Es la primera actividad de la estrategia.
Termina:	Cuando se consigue concientizar a las personas involucradas.

Actividad 2: Crear clima organizacional de actualización

El clima organizacional se refiere a las interacciones, estructura, políticas, actividades, actitudes y otras características de la organización y sus integrantes. Estas condiciones influyen en la percepción que el ingeniero de software tiene sobre su entorno laboral.

El clima organizacional de actualización es un tipo particular de clima organizacional que refleja la percepción de los ingenieros de software sobre la importancia de mantenerse actualizado profesionalmente. Crear un clima organizacional de actualización consiste en

establecer en la organización y sus integrantes un medio laboral en cual el desarrollo de competencias represente un reto, interés y crecimiento individual, grupal y organizacional, con finalidad convertir al desarrollo de competencias en una filosofía y estilo de trabajo para el ingeniero de software.

Los objetivos de realizar esta actividad son:

- Motivar al ingeniero de software a realizar acciones de actualización de competencias.
- Establecer un entorno laboral que facilite el desarrollo de competencias del ingeniero de software.

El encargado de guiar y vigilar que se cumplan las tareas que permiten crear el clima organizacional de actualización es el agente de cambio. Sin embargo, crear un clima propicio requiere de la cooperación y responsabilidad de todas las personas involucradas.

Tarea 1: Aplicar instrumentos para conocer el clima organizacional de actualización.

El agente de cambio adapta y aplica un conjunto de instrumentos que permitan conocer la percepción del ingeniero de software sobre las condiciones del ambiente laboral de la organización.

Tarea 2: Realizar reporte de resultados de la aplicación de instrumentos para conocer el clima organizacional de actualización.

El agente de cambio analiza las respuestas que proporcionan las personas que responden los instrumentos. En base a estas respuestas genera un reporte que especifique cuál es la percepción de los ingenieros de software sobre las condiciones del entorno laboral de la organización.

Tarea 3: Planear la creación del clima organizacional de actualización propicio.

El agente de cambio de acuerdo a los resultados que especifica el reporte, realiza un plan de acciones que permitan cambiar las condiciones que potencialmente puedan obstruir el desarrollo de competencias.

Tarea 4: Presentar resultados y plan para crear clima organizacional de actualización propicio.

El agente de cambio presenta ante la organización el reporte con los resultados sobre el clima organizacional de actualización y el plan que se sigue para crear el clima organizacional propicio para la actualización.

Tarea 5: Ejecutar plan para crear clima organizacional de actualización.

El agente de cambio en colaboración con la organización realiza las acciones necesarias para cambiar la percepción de los ingenieros de software sobre las condiciones del entorno laboral y crear así el clima organizacional propicio para el desarrollo de competencias. En la Tabla XI se muestra el resumen de la actividad crear clima organizacional de actualización.

Actividad 3: Planear programa de desarrollo de competencias

El plan del programa de desarrollo de competencias es el conjunto de acciones sistematizadas que tiene como propósito establecer de manera efectiva el proceso de desarrollo de competencias. El plan se compone de varios planes con objetivos específicos, algunos de ellos son:

- Plan de reducción de la resistencia al cambio.
- Plan de comunicación interna.
- Plan de motivación del ingeniero de software.
- Plan de reducción de riesgos.

Tabla XI. Resumen de la actividad crear clima organizacional de actualización.

Crear clima organizacional de actualización	
Objetivos:	<ul style="list-style-type: none"> • Motivar al ingeniero de software a realizar acciones de actualización de competencias. • Establecer un entorno laboral que le facilite el desarrollo de competencias del ingeniero de software.
Responsable:	Agente de cambio.
Colaboradores:	Organización, ingenieros de software.
Tareas:	<ul style="list-style-type: none"> • Aplicar instrumentos para conocer el clima organizacional de actualización. • Realizar reporte de resultados de la aplicación de instrumentos para conocer el clima organizacional de actualización. • Planear la generación del clima organizacional de actualización propicio. • Presentar resultados y plan para generar clima organizacional de actualización propicio. • Ejecutar plan para generar clima organizacional de actualización.
Documentos:	<ul style="list-style-type: none"> • Instrumentos para valorar clima organizacional de actualización. • Reporte sobre clima organizacional de actualización. • Plan para generar clima organizacional de actualización propicio.
Inicia:	Después de realizar la actividad "conocer y concientizar a la organización".
Termina:	Cuando se genera el clima organizacional de actualización propicio.

Los planes que se analizan y diseñan en esta actividad, se utilizan durante los diferentes procesos que forman parte del modelo de desarrollo de competencias para ingenieros de software.

El objetivo de esta actividad es:

- Diseñar un conjunto de planes que sirvan de guía para realizar el desarrollo de competencias.

El agente de cambio en colaboración con los ingenieros de software son quienes se encargan de elaborar el plan del programa de desarrollo de competencias y los planes específicos necesarios.

Tarea 1: Analizar y diseñar el plan general del programa de desarrollo de competencias.

El agente de cambio en colaboración con los ingenieros de software realiza el plan general para el programa de desarrollo de competencias.

Tarea 2: Analizar y diseñar los planes específicos para el programa de desarrollo de competencias.

El agente de cambio en colaboración con los ingenieros de software realiza los planes específicos para el programa de desarrollo de competencias. En la Tabla XII se observa el resumen de la actividad planear programa de desarrollo de competencias.

Tabla XII. Resumen de la actividad planear programa de desarrollo de competencias.

Planear programa de desarrollo de competencias	
Objetivo:	Diseñar un conjunto de planes que sirvan de guía para realizar el desarrollo de competencias.
Responsable:	Agente de cambio.
Colaboradores:	Ingenieros de software.
Tareas:	<ul style="list-style-type: none"> • Analizar y diseñar el plan general del programa de desarrollo de competencias. • Analizar y diseñar los planes específicos del programa de desarrollo de competencias.
Documentos:	<ul style="list-style-type: none"> • Plan general para el programa de desarrollo de competencias. • Planes específicos para el programa de desarrollo de competencias.
Inicia:	Después generar el clima organizacional de actualización propicio.
Termina:	Cuando diseñan el plan general y los planes específicos para el programa de desarrollo de competencias.

Actividad 4: Evaluar al programa

La evaluación del programa de desarrollo de competencias consiste en realizar un análisis y reflexión crítica sobre todos los componentes y factores que intervienen en la ejecución de los procesos que forman la estrategia. Los resultados de este análisis deben ser útiles para mejorar el programa general.

Los objetivos de la evaluación del programa son:

- Determinar en qué grado se alcanzaron los objetivos planteados.
- Identificar qué modificaciones al programa son necesarias para su mejoramiento.

La evaluación puede dirigirse a una de las siguientes tres dimensiones:

- Evaluación de los procesos.
- Evaluación del impacto.
- Evaluación de los resultados.

El agente de cambio es responsable de hacer un documento en el cual se describa los puntos que se van a evaluar antes, durante y después de realizar un ciclo del desarrollo de competencias (ver proceso de desarrollo de competencias). De acuerdo a los intereses de la organización, algunos de los aspectos que se pueden evaluar del programa de desarrollo de competencias son:

- La reacción de los ingenieros de software a la implantación del programa.
- La reacción de los ingenieros de software ante los resultados alcanzados con el programa.
- Los cambios en el comportamiento de los participantes en el programa.
- Los resultados que se alcanzaron.
- La calidad en el diseño de las actividades para la ejecución del programa.
- Las mejoras alcanzadas como consecuencia de las actividades del programa.

Algunos de los medios que el agente de cambio puede utilizar para realizar la evaluación del programa general son cuestionario, sondeo, entrevistas, etc.

Las tareas que componen a la actividad de evaluar al programa se describen a continuación.

Tarea 1: Analizar y diseñar evaluación del programa de desarrollo de competencias.

El agente de cambio realizar o adapta uno o varios instrumentos para evaluar el programa de desarrollo de competencias. Estos instrumentos permiten evaluar al programa de desarrollo de competencias en sus tres dimensiones (procesos, impacto y resultados).

Tarea 2: Aplicar evaluación al programa de desarrollo de competencias.

El agente de cambio aplica los instrumentos que contestan los participantes en el programa de desarrollo de competencias. La aplicación de la evaluación se realiza en tres tiempos (al inicio, durante y después) de la ejecución del programa.

Tarea 3: Analizar resultado de evaluación y proponer mejoras.

El agente de cambio realiza el análisis de los resultados de las evaluaciones. Este análisis debe permitir sugerir mejoras al programa de desarrollo de competencias. En la Tabla XIII se observa el resumen de la actividad evaluar al programa de desarrollo de competencias.

Tabla XIII. Resumen de la actividad evaluar al programa de desarrollo de competencias.

<i>Evaluar al programa de desarrollo de competencias</i>	
Objetivos:	<ul style="list-style-type: none"> • Determinar en qué grado se alcanzaron los objetivos planteados. • Identificar qué modificaciones al programa son necesarias para su mejoramiento.
Responsable:	Agente de cambio.
Colaboradores:	Ingenieros de software.
Tareas:	<ul style="list-style-type: none"> • Analizar y diseñar evaluación del programa de desarrollo de competencias. • Aplicar evaluación al programa de desarrollo de competencias. • Analizar resultado de evaluación y proponer mejoras
Documentos:	<ul style="list-style-type: none"> • Instrumentos de evaluación al programa de desarrollo de competencias. • Reporte de análisis de resultados y propuestas de mejora al programa de desarrollo de competencias.
Inicia:	Antes de comenzar a realizar las actividades de la estrategia de adquisición de competencias.
Termina:	Después de realizar un ciclo completo del proceso de desarrollo de competencias.

Actividad 5: Proporcionar seguimiento al programa

El seguimiento consiste de aquellas acciones que proporcionar soporte a las actividades del programa de desarrollo de competencias, aseguran que se estén ejecutando de manera correcta y que no exista ninguna otra situación que impida que éstas se realicen.

El seguimiento del programa general cumple con diversos objetivos:

- Identificar las fuentes de resistencia al cambio.
- Proporcionar retroalimentación.
- Resolver dudas acerca del programa general.
- Detectar errores en la ejecución de las actividades.
- Cambiar el diseño de las actividades que no resulten eficientes.

El seguimiento de las actividades del programa general está a cargo del agente de cambio, es él quien proporciona soporte acerca de la manera cómo se deben ejecutar las actividades del desarrollo de competencias. El agente de cambio tiene la responsabilidad y obligación de ayudar a los participantes cuando tengan dudas o problemas en la ejecución de las actividades, es su obligación también, preguntar a los ingenieros de software si necesitan ayuda con alguna de ellas.

El seguimiento se realiza a lo largo de la duración del programa de desarrollo de competencias. Se pueden utilizar diferentes medios para dar seguimiento a las actividades del programa, tales como visitas al lugar de trabajo, encuestas, entrevistas, buzón de sugerencias, correos electrónicos, etc. El seguimiento es útil para resolver dudas, mitigar la resistencia al cambio, identificar fuentes de resistencia al cambio, tomar acciones correctivas, retroalimentación para futuras ejecuciones, etc.

Tarea 1: Realizar plan de seguimiento al programa.

El agente de cambio elabora un plan en el cual especifica que acciones a seguir para proporcionar seguimiento al programa de desarrollo de competencias.

Tarea 2: Realizar el seguimiento al programa.

El agente de cambio ejecuta el plan de seguimiento al programa de desarrollo de competencias. En la Tabla XIV se observa el resumen de la actividad proporcionar seguimiento al programa de desarrollo de competencias.

Actividad 6: Comunicar resultados

Una actividad clave para la ejecución del programa general, es la comunicación. Para lograr establecer una comunicación efectiva es necesario utilizar medios de comunicación eficaces y variados. Una mala comunicación puede traer como consecuencia que disminuya la eficiencia del programa de desarrollo de competencias.

Tabla XIV. Resumen de la actividad proporcionar seguimiento al programa de desarrollo de competencias.

Proporcionar seguimiento al programa de desarrollo de competencias	
Objetivos:	<ul style="list-style-type: none"> • Identificar las fuentes de resistencia al cambio. • Proporcionar retroalimentación. • Resolver dudas acerca del programa general. • Detectar errores en la ejecución de las actividades. • Cambiar el diseño de las actividades que no resulten eficientes.
Responsable:	Agente de cambio.
Colaboradores:	Ingenieros de software.
Tareas:	<ul style="list-style-type: none"> • Realizar plan de seguimiento al programa. • Realizar el seguimiento al programa.
Documentos:	<ul style="list-style-type: none"> • Plan de seguimiento al programa. • Reporte de errores, dudas o ineficiencias sobre el programa.
Inicia:	Cuando inicia el programa de desarrollo de competencias.
Termina:	Cuando termina el programa de desarrollo de competencias.

El objetivo principal de esta actividad es:

- Mantener informados a los participantes.

Todos los participantes en el programa de desarrollo de competencias son de alguna forma responsables del establecimiento de la comunicación. El agente de cambio debe hacerse de diferentes medios para comunicar la información necesaria para la ejecución del programa, algunos de estos medios pueden ser:

- Correo electrónico.
- Buzón de sugerencias.
- Carteles.
- Pizarras.
- Folletos.
- Reuniones.
- Pláticas.
- Conferencias.

Esta actividad debe ejecutarse durante toda la duración del programa de desarrollo de competencias.

Tarea 1: Realizar plan de comunicación de resultados.

El agente de cambio elabora un plan en el cual especifica las acciones que debe seguir para mantener informados a los participantes sobre los diferentes aspectos relacionados con el programa de desarrollo de competencias.

Tarea 2: Ejecutar plan de comunicación de resultados.

El agente de cambio en colaboración con los participantes ejecuta el plan de comunicación de resultados. En la Tabla XV se muestra el resumen de la actividad comunicar resultados.

Tabla XV. Resumen de la actividad comunicar resultados.

Comunicar resultados	
Objetivo:	<ul style="list-style-type: none"> Mantener informados a los participantes.
Responsable:	Agente de cambio.
Colaboradores:	Ingenieros de software.
Tareas:	<ul style="list-style-type: none"> Realizar plan de comunicación de resultados. Ejecutar el programa de comunicación de resultados.
Documentos:	<ul style="list-style-type: none"> Plan de comunicación de resultados.
Inicia:	Cuando inicia el programa de desarrollo de competencias.
Termina:	Cuando termina el programa de desarrollo de competencias.

V.3 Proceso de adaptación del marco de competencias

V.3.1 Descripción

El proceso de adaptación del marco de competencias consiste de aquellas actividades que permiten definir un conjunto de competencias que el ingeniero de software puede desarrollar para alcanzar el desempeño eficiente en un rol o puesto determinado en una organización desarrolladora de software. El proceso de adaptación del marco de competencia se basa en el reporte que realiza el Departamento de Servicios Civiles del Estado de Nueva York (Department of Civil Service of New York State) sobre cómo desarrollar competencias [Sinnott *et al.*, 2002]. En él, proponen una estrategia y guías para

desarrollar y utilizar competencias. La estrategia que se propone en este reporte se compone de cinco fases: identificar las posiciones para las cuales se establecen las competencias, desarrollar el marco de competencias, evaluar las competencias individuales e identificar la brecha, desarrollar una estrategia para dirigir la brecha, reevaluar competencias y evaluar el retorno de la inversión. De estas fases, únicamente las primeras dos se utilizan en el proceso de adaptación del marco de competencias, el resto de las fases se utilizan en el proceso de desarrollo de competencias que se describe en el apartado siguiente.

En la Figura 16 se observan las fases de la estrategia que se propone en [Sinnott *et al.*, 2002].

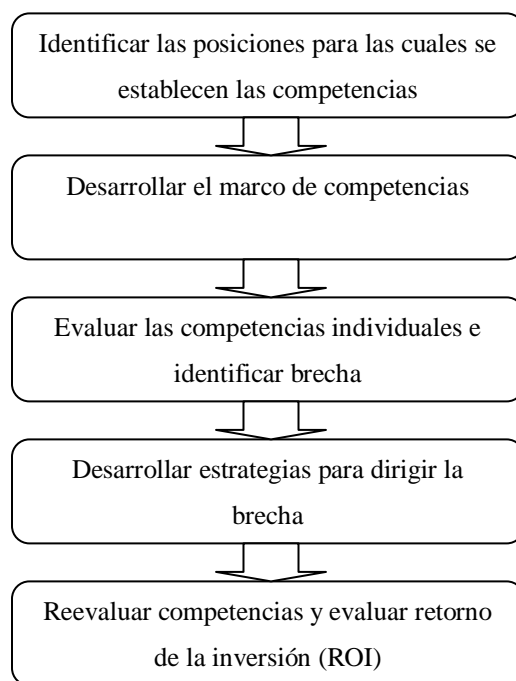


Figura 16. Estrategia y guía para el desarrollo y uso de competencias que propone el Departamento de Servicios Civiles del Estado de Nueva York (EUA) (versión original en Sinnott *et al.*, 2002).

Los objetivos que se deben alcanzar con el proceso de adaptación del marco de competencias son:

- Crear marco de competencias claves.
- Definir los perfiles de puestos de la organización.

Objetivo 1 del proceso adaptación de marco de competencias: crear marco de competencias claves

Crear un marco de referencia con las competencias claves permite encausar el esfuerzo del desarrollo profesional hacia el desarrollo de competencias específicas que garantizan el éxito laboral. De esta manera las competencias claves deben ser aquellas que contribuyen a alcanzar los objetivos personales (del ingeniero de software) y de la organización.

Objetivo 2 del proceso de adaptación de marco de competencias: definir los perfiles de puestos de la organización

Después de crear el marco de competencias, es necesario definir qué competencias se requieren en cada uno de los puestos de la organización. La definición del perfil de puesto que la organización realiza sirve para indicarle al ingeniero de software qué competencias cree la organización conveniente para la ejecución de ese rol o puesto de trabajo.

De acuerdo a la estrategia que se propone en [Sinnott *et al.*, 2002] y los objetivos que se describen arriba, las actividades que forma el proceso de adaptación del marco de competencias son:

- Identificar competencias.
- Definir diccionario de competencias.
- Definir marco de competencias.
- Definir perfil de puesto.

En la Figura 17 se puede observar la relación que existe entre estas actividades, las cuales forman un ciclo que se repite continuamente, esto se debe a que es necesario que la organización actualice periódicamente las competencias del marco y la definición de los perfiles de puestos, para así garantizar la validez del marco y perfil de competencias.

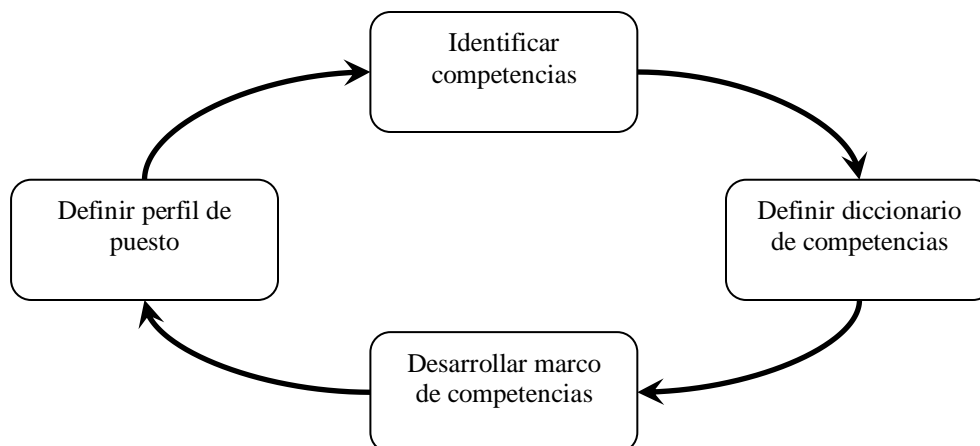


Figura 17. Representación gráfica de las actividades del proceso de adaptación del marco de competencias.

El encargado de ejecutar las actividades de la estrategia de adaptación del marco de competencias es el agente de cambio. Sin embargo, para establecer esta estrategia requiere de la cooperación de un grupo que está conformado por los ingenieros de software más sobresalientes de la organización. A los ingenieros de software que forman este grupo se les conoce como “tutores”. Sus principales funciones son: ayudar al agente de cambio a adaptar el marco de competencias y proporcionar soporte a los ingenieros de software durante el desarrollo de competencias.

V.3.2 Actividades

Actividad 1: Identificar competencias

La actividad identificar competencias consiste en realizar un análisis de las actividades laborales con el propósito de identificar y definir las competencias que el ingeniero de software utiliza para desempeñar de manera eficiente su trabajo.

La identificación de competencias permite crear el marco de competencias que sirve de guía para indicar al ingeniero de software qué competencias puede desarrollar. Es importante la participación activa de los ingenieros de software en la actividad de identificación de competencias, ya que son ellos quienes desempeñan las funciones laborales y por lo tanto pueden proporcionar información más exacta sobre las

competencias que necesitan, esta participación puede ocurrir a través de encuestas, foros de discusión, buzón de sugerencias, entrevistas, etc.

El objetivo principal en esta actividad es:

- Analizar las competencias que permiten el desempeño eficiente de los ingenieros de software en los roles o puestos de la organización.

El agente de cambio en colaboración con los ingenieros de software, se encargan de realizar las tareas que forman la actividad identificar competencias. Existen diferentes técnicas para realizar la identificación de competencias para un contexto laboral, algunas de ellas son el análisis funcional, panel de expertos, incidentes críticos, etc., sin embargo, este tema queda fuera de los alcances de la presente investigación.

Las tareas necesarias para realizar la actividad de identificación de competencias son:

Tarea 1: Identificar y definir objetivos personales y de la organización.

El agente de cambio guía al ingeniero de software y a la organización para identificar y definir sus objetivos profesionales y de negocio, respectivamente. Por ejemplo, el ingeniero de software puede definir qué puestos de trabajo aspira en el futuro

Tarea 2: Diseñar un plan de trabajo para identificar las competencias.

El agente de cambio realiza un plan de trabajo para realizar la identificación de competencias. Existen diversos métodos para realizar la identificación de competencias, algunos de ellos se analizan en [Dubois, 1993], en esta investigación no se profundiza en este tema.

Tarea 3: Realizar la identificación de competencias.

El agente de cambio en colaboración con los ingenieros de software, utilizan algún método de identificación de competencias, para detectar las competencias de los puestos de la organización.

Tarea 4: Verificar y validar competencias.

El agente de cambio y el grupo de tutores se aseguran que las competencias identificadas sean las que realmente requiere el ingeniero de software para el desempeño de un determinado puesto. En la Tabla XVI se muestra el resumen de la actividad identificar competencias.

Tabla XVI. Resumen de la actividad identificar competencias.

Identificar competencias	
Objetivo:	<ul style="list-style-type: none"> Analizar las competencias que permiten el desempeño eficiente de los ingenieros de software en los roles o puestos de la organización.
Responsable:	Agente de cambio.
Colaboradores:	Ingenieros de software y organización.
Tareas:	<ul style="list-style-type: none"> Identificar y definir objetivos personales y de la organización. Diseñar un plan de trabajo para identificar las competencias. Realizar la identificación de competencias. Verificar y validar competencias.
Documentos:	<ul style="list-style-type: none"> Documento con definición de objetivos personales. Documento con definición de objetivos de negocio. Lista de competencias.
Inicia:	Una vez que se establece el clima organizacional propicio.
Termina:	Se ejecuta periódicamente.

Actividad 2: Definir diccionario de competencias

Esta actividad consiste en escribir un diccionario con las definiciones de las competencias de la organización. Una vez que se identifican las competencias para determinado puesto o rol, éstas se definen y agregan al diccionario de competencias. El diccionario de competencia contiene la definición y características principales de cada competencia, esto permite al ingeniero de software establecer qué competencia es la que necesita desarrollar. Además, el diccionario debe indicar cómo se pueden desarrollar dichas competencias, es decir proponer técnicas para el desarrollo de la competencia en particular.

El objetivo de esta actividad es:

- Desarrollar el diccionario de competencias de la organización.

El agente de cambio es el responsable de escribir el diccionario de competencias. Este proceso se ejecuta cada vez que se identifica una competencia nueva. En esta actividad existe una única tarea que es:

Tarea 1: Definir las competencias.

El agente de cambio puede solicitar la ayuda de la organización y los ingenieros de software para definir las competencias. En la Tabla XVII se resume la actividad definir diccionario de competencias.

Tabla XVII. Resumen de la actividad definir diccionario de competencias.

Definir diccionario de competencias	
Objetivo:	• Desarrollar el diccionario de competencias de la organización.
Responsable:	Agente de cambio.
Colaboradores:	Ingenieros de software y organización.
Tareas:	• Definir las competencias.
Documentos:	• Diccionario de competencias.
Inicia:	Cuando termina la actividad identificar competencias.
Termina:	Cuando se definen todas las competencias identificadas.

Actividad 3: Desarrollar marco de competencias

Esta actividad consiste en seleccionar (del diccionario de competencias) las competencias y el nivel de cada una de ellas que permite el desempeño sobresaliente para un determinado puesto. Alternativamente, se puede utilizar un diccionario o marco de competencias existente y adaptarlo a las necesidades particulares de la organización, por ejemplo en este trabajo de investigación se propone un marco de competencias para ingenieros de software, el cual la organización puede modificar para adecuarlo a las características que necesita para poder alcanzar los objetivos planteados en actividad 1 (identificar competencias). Así, el marco de competencias específica para cada puesto de la organización dos cosas: las competencias y el nivel de las mismas.

El objetivo de esta actividad es:

- Desarrollar el marco de competencias de la organización.

El agente de cambio con la colaboración y apoyo de los ingenieros de software eligen las competencias y el nivel de cada una de ellas. Las tareas necesarias para desarrollar el marco de competencias se describen a continuación.

Tarea 1: Decidir si se adopta o modifica un marco de competencias existente o se desarrolla uno propio.

El agente de cambio en colaboración con la organización y los ingenieros de software evalúan si se utiliza o adapta un marco de competencias existente o si se desarrolla uno con las propias competencias.

Tarea 2: Obtener ejemplos de marcos de competencias.

El agente de cambio, la organización y los ingenieros de software buscan ejemplos de marcos de competencias y competencias que pueden incluirse en el marco de competencias de la organización.

Tarea 3: Identificar y definir los resultados que se esperan obtener con las competencias.

El agente de cambio en colaboración con la organización y los ingenieros de software determinan qué resultados desean obtener tanto a nivel personal, del puesto y organizacional.

Tarea 4: Elegir las competencias del marco.

El agente de cambio elige un método para elegir competencias. Algunos métodos son: entrevistas, encuestas, foros de discusión, lluvia de ideas, etc. Con el apoyo de la organización y de los ingenieros de software deciden qué competencias forman el marco de competencias de la organización.

Tarea 5: Determinar la escala de las competencias.

El agente de cambio, la organización y los ingenieros de software analizan y determinan la importancia y el nivel de cada una de las competencias que forman el marco. En la Tabla XVIII se resume la actividad desarrollar marco de competencias.

Tabla XVIII. Resumen de la actividad desarrollar marco de competencias.

Desarrollar marco de competencias	
Objetivo:	<ul style="list-style-type: none"> • Desarrollar el marco de competencias de la organización.
Responsable:	Agente de cambio.
Colaboradores:	Ingenieros de software y organización.
Tareas:	<ul style="list-style-type: none"> • Decidir si se adopta o modifica un marco de competencias existente o se desarrolla uno propio. • Obtener ejemplos de marcos de competencias. • Identificar y definir los resultados que se esperan obtener con las competencias. • Elegir las competencias del marco. • Determinar la escala de las competencias.
Documentos:	<ul style="list-style-type: none"> • Marco de competencias.
Inicia:	Cuando termina la actividad identificar competencias o definición del diccionario de competencias.
Termina:	Cuando se desarrolla el marco de competencia.

Actividad 4: Definir perfil de puestos

Esta actividad consiste en generar un documento en el cual se describan las principales funciones de cada puesto de trabajo de la organización, además de las competencias claves que requiere la persona que realiza dicho puesto.

El objetivo principal de esta actividad es:

- Desarrollar el perfil de puestos de la organización.

El agente de cambio en colaboración con los tutores son los responsable de realizar la descripción del perfil de puestos, pueden utilizar un formato para la descripción de perfil de puesto. Esta actividad se compone por una única tarea que es:

Tarea 1: Definir perfil de puestos.

El agente de cambio en cooperación con la organización y los ingenieros de software definen el perfil de los puestos de trabajo de la organización. En la Tabla XIX se resume la actividad definir perfil de puestos.

Tabla XIX. Resumen de la actividad definir perfil de puestos.

Definir perfil de puestos	
Objetivo:	<ul style="list-style-type: none"> • Desarrollar el perfil de puestos de la organización.
Responsable:	Agente de cambio.
Colaboradores:	Ingenieros de software y organización.
Tareas:	<ul style="list-style-type: none"> • Desarrollar el perfil de puestos de la organización.
Documentos:	<ul style="list-style-type: none"> • Documento con la definición de los perfiles de los puestos de la organización.
Inicia:	Cuando termina la actividad desarrollar marco competencias.
Termina:	Cuando se definen todas los perfiles de puestos de la organización.

V.4 Proceso de desarrollo de competencias

V.4.1 Descripción

El proceso de desarrollo de competencias consiste de aquellas actividades que permiten al ingeniero de software identificar qué competencias tiene y qué competencias necesita desarrollar, además permiten definir un conjunto de acciones para desarrollarlas. Este proceso se basa en los trabajo del Departamento de Servicios Civiles del Estado de Nueva York [Sinnott *et al.*, 2002] y de David Dubois [Dubois, 1993]. En el apartado anterior se presenta la estrategia que propone en [Sinnott *et al.*, 2002] en la Figura 16. David Dubois propone un modelo basado en competencias para mejorar el desempeño, el cual se puede observar en la Figura 18.

El objetivo principal de esta estrategia es:

- Desarrollar competencias del ingeniero de software.

Desarrollar las competencias del ingeniero implica una serie de pasos que se centran en determinar qué, cómo, cuándo, dónde y para qué desarrollar competencias.

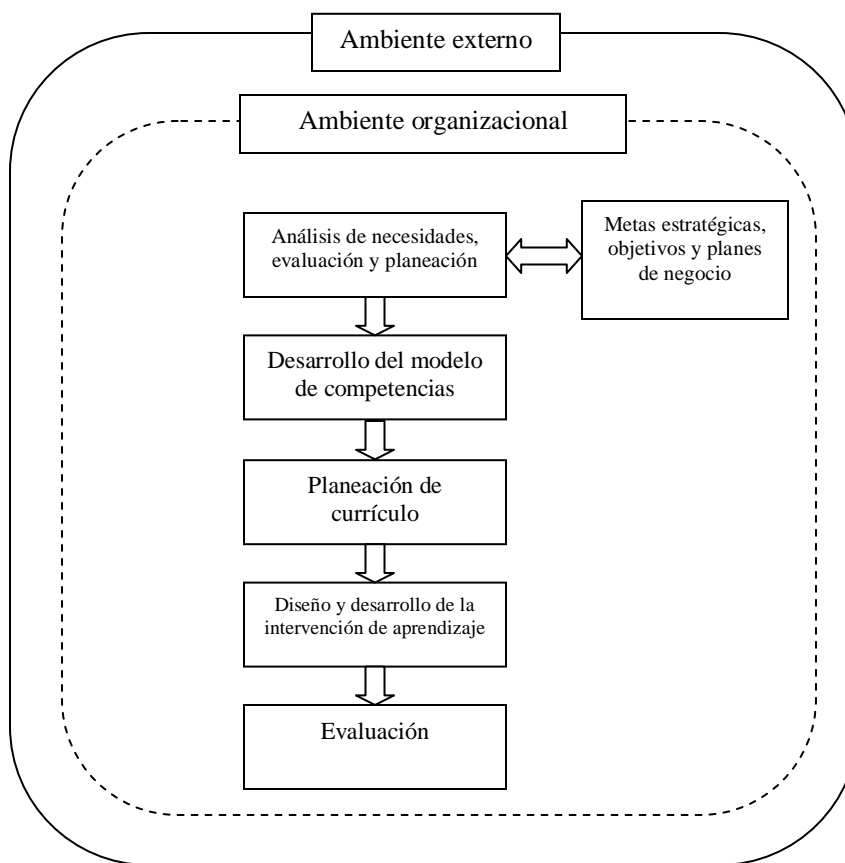


Figura 18. Modelo basado en competencias para mejorar el desempeño de David Dubois (versión original en Dubois, 1993).

Para lograr este objetivo, el proceso de desarrollo de competencias se compone de las actividades siguientes:

- Autoevaluar.
- Detectar necesidades de actualización.
- Realizar plan de desarrollo de competencias.
- Desarrollar competencias.
- Transferir competencias.
- Proporcionar seguimiento al desarrollo de competencias.
- Retroalimentar.

En la Figura 19 se observa la representación gráfica de las actividades que conforman la estrategia de adquisición de competencias. En ella se puede observar que las actividades forman un ciclo continuo y que dos actividades (proporcionar seguimiento y retroalimentación) son actividades que se realizan durante todo el ciclo.

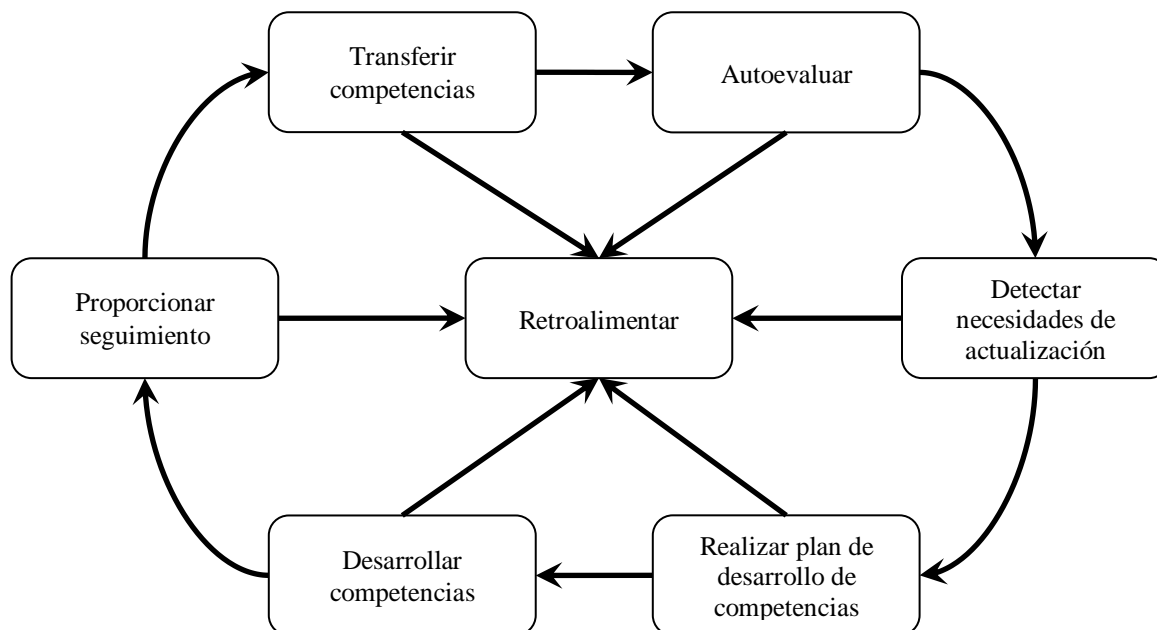


Figura 19. Representación gráfica de las actividades que conforman la estrategia de adquisición de competencias.

V.4.2 Descripción de las actividades

Actividad 1: Autoevaluar

Es la actividad a través de la cual el ingeniero de software identifica y define sus fortalezas. La autoevaluación representa la primera aproximación hacia la detección de necesidades de actualización, el propósito es descubrir y hacer consciente al ingeniero de software de cuáles son las competencias que le dan valor como persona y profesional para reforzarlas. Por otro lado, busca precisar cuáles son las competencias que requieren especial atención para desarrollarlas a un nivel aceptable que le permita alcanzar un mejor desempeño en el puesto de trabajo.

El ingeniero de software puede opcionalmente recibir ayuda y orientación de un tutor al momento de realizar la autoevaluación.

Para realizar la autoevaluación es necesario responder dos cuestionarios:

- 1) En el primero de ellos el ingeniero de software escribe aquellas competencias que considera son las principales que lo hacen sobresalir en el trabajo que desempeña, usan palabras propias redacta una breve definición de la competencia a la que hace referencia y establece el nivel de desarrollo que cree tener para esa competencia.
- 2) El segundo cuestionario consiste en seleccionar de diccionario de competencias aquellas que considere que tiene e indicar el nivel de desarrollo, el ingeniero de software puede consultar la definición y ejemplos que representan a dicha competencia para establecer en qué nivel de desarrollo considera tener esa competencia.

Para realizar una autoevaluación realmente útil, se requiere de la completa honestidad y objetividad del ingeniero de software al momento de precisar las competencias que tiene y el nivel de desarrollo. Esta parte es esencial para conseguir establecer con mayor precisión cuáles son las necesidades de capacitación del ingeniero de software.

Además de llenar los dos cuestionarios mencionados, la autoevaluación también considera la evaluación 360°, la cual consiste en que las personas con las que el ingeniero de software interactúa en su trabajo (dueños, jefes, clientes internos, clientes externos, otros ingenieros de software) contesten un conjunto de preguntas con la finalidad de evaluar su desempeño en el trabajo, esto permite al ingeniero de software darse cuenta cómo las demás personas perciben su desempeño.

Para realizar la autoevaluación es necesario que se haya elaborado el diccionario de competencias de la organización o en un momento dado se puede utilizar otro diccionario de competencias que no sea el oficial de la organización. Si se cumple lo anterior, la autoevaluación de competencias se puede realizar cuando el ingeniero de software quiera

conocerse mejor y determinar sus necesidades de actualización y desarrollo de competencias.

La autoevaluación es la puerta de entrada para realizar la actividad detectar necesidades de actualización y a su vez ésta para realizar el plan de desarrollo de competencias.

Actividad 2: Detectar necesidades de actualización

La detección de necesidades de capacitación es un método comparativo que consiste en identificar la brecha que existe entre lo que debe hacerse y lo que se hace realmente. Es decir se comparan las competencias que el ingeniero posee actualmente y las que requieren para desempeñar eficazmente su puesto de trabajo.

El proceso de detección de necesidades de capacitación lo realiza el ingeniero de software con el apoyo y asesoramiento del tutor asignado.

El método comparativo para detectar necesidades de capacitación se compone de cuatro fases:

- 1) Situación ideal.
- 2) Situación real.
- 3) Análisis comparativo.
- 4) Definición de necesidades.

Situación ideal

Tiene como propósito definir la manera ideal de desempeñar y los resultados que se deben obtener en un determinado puesto. Es necesario indicar cuáles son los conocimientos, comportamientos y actitudes que permiten lograr un desempeño satisfactorio de la actividad laboral. Para definir la situación ideal se requiere:

- Identificar los clientes internos y externos: es decir quiénes son personas con las que se interactúa, ya sea recibiendo u ofreciéndole un producto o servicio, al ejecutar las funciones laborales.

- Definir los requerimientos del cliente: cómo desea el cliente (interno o externo) que sea el producto o servicio que se le ofrece.
- Definir actividades para cubrir requerimientos: consiste en establecer cómo deben de hacerse las actividades de la función laboral, para que por un lado se cumplan los requerimientos del cliente y por otro cumplir con el estándar de calidad para dicha actividad.
- Identificar competencias: identificar las competencias que se requieren para ejecutar las actividades que se definieron en el paso anterior.
- Elaborar perfil de puesto ideal: consiste en definir las características ideales para el puesto de trabajo.

Como apoyo para la definición de la situación ideal se puede recurrir a documentos tales como descripción del perfil de puesto, manual de procedimientos de la organización, plan de expansión de la organización, nuevas o futuras necesidades de desempeño. El resultado debe ser un documento que contenga la información mencionada en cada uno de los pasos para el diseño de la situación ideal.

Situación real

Consiste en identificar cuál es la razón o razones por la que no es posible cumplir con la situación ideal definida anteriormente y clasificar estas razones en una de las tres posibilidades siguientes:

- No se puede: debido a que no existen los métodos, materiales o equipos necesarios para realizar eficazmente la función laboral.
- No se sabe: no se tienen los conocimientos necesarios para utilizar los métodos, materiales o equipos y cumplir con la función laboral.
- No se quiere: un problema relacionado con la actitud, motivación y los valores de la persona.

Para lograr mayor objetividad en la definición de la situación ideal se recurre a la autoevaluación de competencias que se elabora en el proceso de autoevaluación y a los

resultados de las evaluaciones de 360° que contestan los clientes internos y externos con los que interactúa el ingeniero de software. También son útiles reportes realizados por jefes y las quejas realizadas por los clientes, observación de situaciones tales como baja en la productividad, proyectos fuera de tiempo, conflictos con los compañeros, etc. El resultado es un documento en el que se especifica y clasifican las razones por las cuales no es posible cumplir con lo planteado en la situación ideal.

Análisis comparativo

El análisis comparativo consiste en analizar cuidadosamente los reportes obtenidos en la definición de la situación ideal y la situación real. Ésta puede convertirse en una tarea complicada y definitiva para detectar de manera efectiva las necesidades de capacitación. El análisis comparativo requiere que el ingeniero de software tenga la competencia de autocrítica que consiste en la capacidad de distinguir los propios defectos y de enfrentarlos, proponiéndose y comprometiéndose hacer lo mejor posible para que estos no sigan presentándose.

El análisis comparativo se realiza en dos dimensiones:

- *Análisis de tareas:* se analizan y comparan las tareas identificadas en la situación ideal con la situación real, se establece las similitudes y diferencias entre las tareas que deben realizarse y las que realmente se realizan.
- *Análisis de comportamientos:* se analizan y comparan los comportamientos identificados en la situación ideal, se establecen las similitudes y diferencias entre los comportamientos que deben mostrar y los que realmente se muestran.

El resultado del análisis comparativo es un reporte en el cual a manera de reflexión el ingeniero de software describe qué situaciones y qué actitudes son necesarias mejorar con el fin de cumplir con el desempeño eficaz de la función laboral.

Definición de necesidades de actualización

De acuerdo con el reporte del análisis comparativo, plan de carrera, diccionario de competencias y marco de competencias, se identifica y establece cuáles son las competencias que se requiere desarrollar para alcanzar un nivel que permita alcanzar el desempeño deseado.

Como resultado se debe obtener un reporte en el cual se especifique qué competencias se requiere desarrollar, la razón por la cual se debe desarrollar, de qué manera beneficia en su vida y en su trabajo al ingeniero de software desarrollar dicha competencia, que resultados se espera obtener, cómo se puede establecer si la competencia se ha desarrollado adecuadamente. Es importante indicar en el reporte de qué competencias son prioritarias de desarrollar.

La detección de necesidades de capacitación se realiza cuando el ingeniero de software quiera mejorar y después que haya realizado una autoevaluación. La finalidad de realizar el proceso de detección de necesidades de capacitación es definir cuáles son las competencias que se requiere desarrollar.

Actividad 3: Realizar plan de desarrollo de competencias

El plan de desarrollo de competencias es un instrumento de gestión que permite al ingeniero de software dirigir sus acciones de desarrollo de competencias. Éste describe de manera detallada un conjunto sistematizado y coherente de acciones de desarrollo de competencias en un determinado período de tiempo. Todas las acciones se deben derivar de la detección de necesidades de actualización.

El proceso de planeación del desarrollo de competencias consiste precisamente en elaborar el plan de desarrollo de competencias, definiendo las actividades, las estrategias, los cursos y otros aspectos que facilitan al ingeniero de software para lograr el desarrollo de las competencias.

El ingeniero de software es el encargado de elaborar su propio plan de desarrollo de competencias, en base a los resultados de la detección de necesidades de capacitación y la disponibilidad de cursos, tiempo, etc. Sin embargo, para que este plan sea válido es necesario que sea evaluado y aprobado por el tutor asignado.

La planeación del desarrollo de competencias se divide en dos fases principales:

- Definición de las acciones de desarrollo de competencias.
- Validación del plan de desarrollo de competencias.

Definición de las acciones de desarrollo de competencias

Consiste en identificar, seleccionar y priorizar las acciones que contribuyan a alcanzar el nivel de desarrollo de competencias deseado. La definición de cada una de estas acciones debe contener por lo menos la información que se muestra en el anexo B (formato para el plan personal de desarrollo de competencias).

Validación del plan de desarrollo de competencias

La validación del plan de desarrollo de competencias del ingeniero de software está a cargo del tutor asignado. El objetivo de evaluar el plan de desarrollo de competencias es para proporcionar sugerencias con respecto al contenido y el enfoque del plan. El tutor evalúa que la descripción de cada acción especifique lo que se indica.

La actividad realizar plan de desarrollo de competencias se ejecuta después de haber concluido la actividad detectar necesidades de actualización. No debe transcurrir más de un mes después de haber obtenido el reporte de detección de necesidades de actualización, en caso de suceder esto es necesario ejecutar de nuevo la actividad detectar necesidades de actualización.

El plan de desarrollo de competencias permite al ingeniero de software tener un mejor control sobre las acciones para el desarrollo de competencias y la certidumbre de que se van a cumplir los objetivos del desarrollo de competencias.

Actividad 4: Desarrollar competencias

Se ejecutan las acciones para el desarrollo de competencias, utilizando las técnicas que se plantearon en el plan de desarrollo de competencias. El ingeniero de software es responsable de cumplir lo planeado, sin embargo el tutor cumple un rol muy importante en esta actividad, ya que es el que se encarga de motivar al ingeniero de software hacia el aprendizaje, distribuir la enseñanza en el tiempo, asegurar el impacto, etc. Aunque esto depende en gran medida que se haya elegido el contenido de la enseñanza y los instructores con las características correctas.

El objetivo en esta actividad es alcanzar el nivel de desarrollo de competencia deseado. La adquisición de competencias se inicia justo después de la planeación del desarrollo, generalmente, la planeación se realiza para períodos de un año.

Actividad 5: Transferir competencias

Consiste en utilizar en las actividades de trabajo las competencias adquiridas durante las acciones de desarrollo de competencias. Esta actividad es vital para lograr que el desarrollo de la competencia sea efectivo, por lo que se requiere de una persona con un nivel mayor de desarrollo de esa competencia que supervise que la competencia se utilice adecuadamente en la función laboral, esta persona puede ser el tutor.

Las técnicas que se pueden utilizar para lograr el entrenamiento de las competencias adquiridas son simulaciones, participación en casos reales bajo supervisión, prácticas, etc.

El objetivo es reducir el tiempo que el ingeniero de software requiere para aprender a utilizar las competencias que adquiere en sus actividades de trabajo y hacer más eficiente el desarrollo de competencias.

La actividad transferir competencias se realiza durante y después de la actividad adquirir competencias y continúa hasta que el ingeniero consiga aplicar adecuadamente las competencias adquiridas en sus actividades de trabajo.

Actividad 6: Proporcionar seguimiento

La actividad proporcionar seguimiento es el proceso dinámico y participativo enfocado a la obtención de información actualizada acerca de las acciones de desarrollo de competencias, esta información resulta útil para la toma de decisiones para la planeación del desarrollo de competencias del ingeniero de software y para determinar si las acciones seleccionadas fueron las correctas o no, y por qué.

El responsable de la actividad proporcionar seguimiento al desarrollo de competencias es el tutor asignado al ingeniero de software. Puede utilizar técnicas como reuniones, correos electrónicos, cuestionarios, entrevistas o cualquier medio de comunicación que le permita verificar que el plan se está ejecutando adecuadamente. Algunos de los aspectos que debe asegurar es que se cumplan son:

- La asistencia del ingeniero de software al curso o cursos.
- La calidad del curso de capacitación al cual el ingeniero de software está asistiendo.
- La capacidad del instructor del curso.
- El contenido de los cursos.
- Los materiales utilizados para el aprendizaje.
- La motivación del ingeniero de software.
- El esfuerzo del ingeniero de software en las actividades de desarrollo.

La actividad proporcionar seguimiento se debe realizar a lo largo de la estrategia de adquisición de competencias. El objetivo es identificar ineficiencias en el proceso de desarrollo de competencias y asegurar la correcta ejecución del plan de desarrollo.

Actividad 7: Retroalimentar

La actividad retroalimentar consiste en informar al ingeniero de software sobre cómo está haciendo su trabajo, cómo puede mejorarlo, esto para saber qué competencias puede desarrollar. La retroalimentación también se proporciona con referencia a las actividades de desarrollo de competencias, esto le permite verificar que las competencias que adquiere son correctas.

Análisis y diseño de la herramienta de soporte

VI.1 Introducción

En este capítulo se describe el análisis y diseño de la herramienta de soporte, la cual se construye con el propósito de guiar al ingeniero de software durante el proceso de desarrollo de sus competencias, proceso que comprende las actividades: autoevaluar, detectar necesidades de capacitación, realizar plan de desarrollo de competencias, desarrollar competencias y transferir competencias.

La herramienta de soporte que se propone es un sistema basado en software. La razón por la cual se elige este tipo de sistema es porque el lugar de trabajo típico del ingeniero de software se caracteriza por tener como infraestructura principal una red de computadoras, ya que una de las herramientas de trabajo que utiliza el ingeniero de software es la computadora (para documentar, programar, diseñar, etc.). Así, la herramienta de soporte es en específico un sitio Web que se publica en el portal interno de la organización e incluso en Internet para que puedan acceder a él personas que están fuera de la red de la organización, por ejemplo candidatos a un puesto, personal trabajando en otro edificio, etc.

Otra razón, por la cual se elige un sistema basando en software es porque, en general, al ingeniero de software se le facilita más interactuar con la tecnología que con las personas. El perfil del ingeniero (en general) se orienta a trabajar con métodos, tecnologías, herramientas, etc., mientras que trabajar con persona es algo que normalmente se le complica.

La Figura 20 muestra la relación que tiene la herramienta de soporte con el resto de los componentes del modelo de desarrollo de competencias, se observa que tiene relación directa con el proceso de desarrollo de competencias, ya que el objetivo de la herramienta de soporte es precisamente guiar al ingeniero de software durante este proceso. En la Figura 20 se observa, en un recuadro con una línea más gruesa, la representación de la herramienta de soporte.

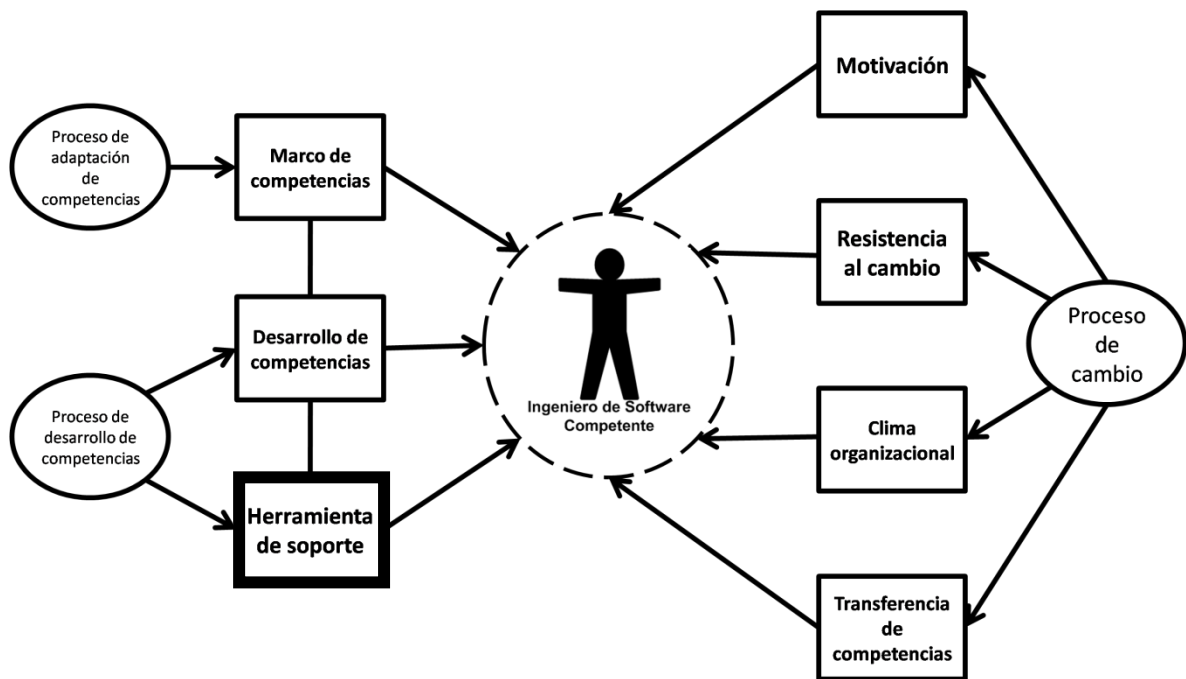


Figura 20. Representación de los componentes del modelo de desarrollo de competencias para ingenieros de software (Herramienta de soporte).

En los apartados siguientes se describen las características principales del análisis y diseño de la herramienta de soporte para el desarrollo de competencias del ingeniero de software. Se analizan las características de algunos sistemas similares que se identificaron a través de una búsqueda referencial, se definen los requerimientos que debe cumplir el sistema, se presenta el diseño que incluye diagrama de clases, diseño de la base de datos, descripción de la funcionalidad, entre otros y finalmente se describe los casos de pruebas para la herramienta de soporte.

VI.2 Análisis de herramientas similares

Al buscar (en Internet) una herramienta de soporte que guíe al usuario durante el desarrollo de competencias, no se encontró un sistema específico para esa tarea (esto no quiere decir que no exista), en cambio se identifican un conjunto de sistemas que se centran en administrar información que utilizan los departamentos de recursos humanos de las organizaciones. Hay que recordar que la capacitación y desarrollo del personal (en donde se aplica el desarrollo de competencias) en general se considera una tarea que se realiza en los departamentos de recursos humano de las organizaciones. En esta investigación se propone que el ingeniero de software sea el responsable de su propio desarrollo, claro que con apoyo y orientación de la organización, por esta razón es que el diseño de la herramienta de soporte se centra proporcionar soporte al ingeniero de software, al tiempo que dispone de la funcionalidad para permitir a la organización interactuar con el ingeniero de software a través de ella.

A continuación se analizan las características de algunos de los sistemas encontrados en la Internet, esto con la intención de conocer la funcionalidad que ofrecen y compararla con la funcionalidad que se implementa en la herramienta de soporte para el desarrollo de competencia (herramienta propuesta). Es importante mencionar que los sistemas que se describen aquí son de paga y no se encontró un sistema con un tipo de licencia que no requiera paga.

Cezanne Software

Cezanne Software es un sistema para la gestión de recursos humanos, el cual integra un conjunto de funciones orientadas a la mejora, gestión, recompensación y retención del personal. En la Tabla XX se describe algunas características generales de este sistema.

Tabla XX. Características generales del sistema Cezanne Software.

Nombre: <i>Cezanne Software.</i>	Compañía: <i>Cezanne Software.</i>
País: <i>Inglaterra.</i>	Idiomas: <i>inglés, italiano, español y francés.</i>
Web: <i>http://www.cezannesw.com</i>	Tipo de sistema: <i>Gestión de RH.</i>

No se encuentra (en la página web de la compañía) una versión de demostración del software, sin embargo, se menciona que tiene la funcionalidad siguiente:

- Gestión del rendimiento.
- Formación y desarrollo.
- Selección del personal.
- Planificación retributiva.
- Encuestas.
- Reportes.
- Planes de sucesión y carrera.
- Gestión del personal.
- Análisis salarial.
- Revisión salarial.
- Diseño de organigramas.

Además de esta funcionalidad, la compañía ofrece los servicios de formación, profesionales y de soporte con respecto al sistema.

Evacom

Evacom es un sistema que permite realizar evaluación de competencias en línea. En la Tabla XXI se pueden observar las características generales de esta herramienta.

Tabla XXI. Características generales del sistema Evacom.

Nombre: <i>Evacom.</i>	Compañía: <i>Grupo Empresarial Triple A.</i>
País: <i>Colombia.</i>	Idiomas: <i>español.</i>
Web: <i>http://evacom.aaa.co</i>	Tipo de sistema: <i>Evaluación de competencias.</i>

De acuerdo con la información que se presenta en la página web, la herramienta cuenta con la funcionalidad siguiente:

- Definición de diccionario de competencias.
- Competencias corporativas.
- Asignación de puntajes ideales.
- Matrices de evaluadores (metodología 360°).
- Competencias por cargo.
- Evaluación para cada competencia.

- Retroalimentación y seguimiento.
- Taller de aplicación.
- Formación en el uso del programa.

Además, ofrece los servicios de formación, acompañamiento y soporte con respecto a la herramienta.

SnowDrop

El software SnowDrop está diseñado para la gestión de recursos humanos, en particular para las actividades selección, formación y desarrollo del personal. Las características generales de este software se muestran en la Tabla XXII.

Tabla XXII. Características generales del sistema SnowDrop.

Nombre: <i>SnowDrop.</i>	Compañía: <i>SnowDrop Systems.</i>
País: <i>España.</i>	Idiomas: <i>inglés.</i>
Web: <i>http://www.snowdrop.es</i>	Tipo de sistema: <i>Gestión de RH.</i>

No se encontró una versión de prueba en el sitio web de la compañía, pero de acuerdo con la información publicada ahí, la funcionalidad que ofrece es la siguiente:

- Selección del personal.
- Gestión de registros del personal.
- Formación y desarrollo.
- Gestión del rendimiento.
- Intranet/Portal del empleado.
- Planes de sucesión
- Nóminas.
- Evaluación 360°.

Además de esta funcionalidad, también se ofrecen los servicios de acompañamiento y soporte del software.

TPMG

El TPMG es un software que ofrece funcionalidad que permite la medición del desempeño y desarrollo de los recursos humanos (retroalimentación, desempeño y encuestas). En la Tabla XXIII se muestra el resumen de las características generales de este software.

Tabla XXIII. Características generales del sistema TPMG.

Nombre: <i>TPMG.</i>	Compañía: <i>The Performance Management Group.</i>
País: <i>Inglaterra.</i>	Idiomas: <i>inglés.</i>
Web: <i>http://www.tpmg.com</i>	Tipo de sistema: <i>Evaluación del desempeño.</i>

En la página web de la compañía no se encontraron detalles sobre los servicios de acompañamiento y soporte del software, en cambio se indica que la funcionalidad que ofrece el software es la siguiente:

- Retroalimentación (metodología 360°).
- Encuestas sobre la actitud de los empleados.
- Administración del desempeño.

Winper

El Winper es un software que reúne un conjunto de herramientas orientadas a la gestión de los recursos humanos para actividades tales como remuneración, gestión del personal, selección, etc. En la Tabla XXIV se pueden ver las características generales de este software.

Tabla XXIV. Características generales del sistema Winper.

Nombre: <i>Winper.</i>	Compañía: <i>InnovaSoft.</i>
País: <i>Chile.</i>	Idiomas: <i>español.</i>
Web: <i>http://www.winper.cl</i>	Tipo de sistema: <i>Gestión de RH.</i>

La funcionalidad de esta herramienta, de acuerdo a información obtenida en la página web de la compañía, está organizada de la manera siguiente:

- Sistemas de remuneración.
- Recursos humanos.
- Personal.
- Gestión de personas.

La compañía ofrece los servicios de implantación y soporte del software.

Mid-market

El Mid-market es un sistema de software que permite la gestión de los recursos humanos, ya que ofrece funciones para realizar tareas tales como gestión de compensaciones, administración del desempeño, administración del talento, etc. En la Tabla XXV se indican las características generales de esta herramienta.

Tabla XXV. Características generales del sistema Mid-market.

Nombre: <i>Mid-market.</i>	Compañía: <i>Workstream Professional Series.</i>
País: <i>EUA-Cánada.</i>	Idiomas: <i>inglés.</i>
Web: <i>http://www.exceed.com</i>	Tipo de sistema: <i>Gestión de RH.</i>

Las actividades de gestión de recursos humanos que soporta el sistema se indica a continuación.

- Desempeño.
- Desarrollo.
- Reclutamiento.
- Evaluación 360°.
- Planes de sucesión.
- Compensación.

La compañía también ofrece los servicios de implantación del sistema, capacitación del personal y soporte.

RealPath

El RealPath es una herramienta que ofrece la funcionalidad necesaria para construir un enfoque de gestión del personal basado en competencias. Esta herramienta permite administrar un diccionario de competencias, reportes, planes de acción, guía para el líder, administración, etc. En la Tabla XXVI se observan las características generales de esta herramienta.

Tabla XXVI. Características generales del sistema RealPath.

Nombre: <i>RealPath.</i>	Compañía: <i>Real Time Performance.</i>
País: <i>Estados Unidos.</i>	Idiomas: <i>inglés.</i>
Web: <i>http://www.realtimperformance.com</i>	Tipo de sistema: <i>Modelo de competencias.</i>

En la página web de la compañía no se indican los servicios que ofrecen sobre el soporte para la herramienta, aunque indica que tiene la funcionalidad siguiente:

- Diccionario de competencias.
- Reportes.
- Planes de acción.
- Retroalimentación (metodología 360°).

Existen otras herramientas más con características similares a las que se presentan en este apartado. El enfoque principal de estos sistemas está en administrar información y proporcionar automatización de tareas referentes a la gestión del personal. Una característica que debe tener la herramienta de soporte para el desarrollo de competencias (que se propone en esta investigación) es que el usuario principal de la misma es el propio ingeniero de software, aunque sea la organización quien se encarga de administrar cierta información de la herramienta.

Hay que destacar que el enfoque principal de la herramienta de soporte es guiar al ingeniero de software durante el proceso de desarrollo de competencias, de lo cual carecen las herramientas que aquí se presentan, ya que no se encontró una herramienta que cumpla con

los requerimientos que se implementan en la herramienta de soporte que se propone en esta investigación. En la Tabla XXVII se muestra la comparación entre las herramientas de soporte que se describen arriba y las características de la herramienta de soporte que se propone.

Tabla XXVII. Comparación de las características de los sistemas y el sistema que se propone.

Características	Cezanne	Mid-market	Snowdrop	TPMG	Winper	Evacom	Realpath	Propuesta
Evaluación del desempeño	√	√	√	√	√			
Selección de personal	√	√	√		√			
Planes de carrera	√	√	√		√			
Análisis salarial	√		√		√			
Compensaciones	√	√			√			
Portal para trabajadores			√					√
Formación y desarrollo	√	√	√		√	√	√	√
Administración	√	√	√		√	√	√	√
Sensibilización al cambio						√		√
Perfil de competencias					√	√		√
Diccionario de competencias					√	√	√	√
Identificación de competencias								√
Asignación de puntajes					√	√		√
Evaluaciones para cada competencia					√	√		√
Planes de acción						√	√	√
Retroalimentación y seguimiento		√		√		√	√	√
Foro de cooperación								√
Recursos para el desarrollo								√

Se observa en la Tabla XXVII que de las herramientas que se analizan solo SnowDrop maneja un portal para el trabajador, mientras que la herramienta de soporte que se propone está dirigida al trabajador (en este caso el ingeniero de software) y no a la organización o al departamento de recursos humanos. También se observa que únicamente otras dos herramientas (de las que se analizan) tiene la funcionalidad para evaluar competencias (Evacom y Winper). Otro punto importante es que la sensibilización al cambio es un factor

clave para lograr el éxito de un sistema como el que se propone, de la información que se obtuvo sobre las herramientas enlistadas arriba, la única que menciona que considera la sensibilización al cambio es Evacom, aunque ésta carece de dos funcionalidades importantes que se proponen en la herramienta de soporte que son: foro de cooperación y recursos para el desarrollo de competencias.

El foro de cooperación es un medio de comunicación asíncrona a través del cual ingenieros de software pueden consultar dudas, presentar casos, exponer experiencias, entre otras, referentes a las actividades del desarrollo de competencias. Los recursos para el desarrollo de competencias se refieren a documentos, enlaces web, videos, audios, cursos en línea y otros que sirven de apoyo al ingeniero de software para desarrollar competencias.

En los apartados siguientes se describen algunas características específicas sobre la herramienta de soporte. En la definición de los requerimientos se describe los requisitos que debe cumplir la herramienta de soporte y después se presenta el diseño de la misma, en este último apartado se encuentran cuestiones técnicas sobre la herramienta de soporte.

VI.3 Análisis de requerimientos

VI.3.1 Introducción

En este apartado se presenta un resumen de la definición de requerimientos que se encuentra en el reporte técnico [Rivera Ibarra *et al.*, 2008] sobre el análisis y diseño de la herramienta de soporte para el desarrollo de competencias, perteneciente también a esta investigación. A la herramienta de soporte se le asigna el nombre Sistema de Apoyo para el Autodesarrollo de Competencias (SAAC).

La estructura del documento de requerimientos (que contiene la definición de los requerimientos y que se encuentra en el reporte técnico) se realiza en base al estándar ANSI/IEEE Std 830-1993 de la IEEE, el cual propone una estructura estándar para el documento de requerimientos de sistemas basados en software. Se agrega al contenido sugerido por este estándar una hoja de control de versiones del documento y una guía para

la validación de los requerimientos. El presente documento no tiene la misma estructura que el estándar mencionado, únicamente se presenta un resumen del documento de requerimientos.

VI.3.2 Descripción general

VI.3.2.1 Usuarios del sistema SAAC

Se identifica que al sistema SAAC lo pueden utilizar diferentes tipos de usuarios: administrador (o administrador del sistema), ingeniero (o ingeniero de software) y tutor, los cuales tienen distintas finalidades como se explica a continuación:

Administrador del sistema

Tiene la responsabilidad de gestionar y mantener actualizada la información del sistema. Los objetivos principales del administrador al utilizar el sistema SAAC son:

- Administrar el sistema en general.
- Notificar al ingeniero las competencias que son recomendables desarrollar.
- Notificar al ingeniero sobre los cursos de capacitación disponibles.
- Encuestar a los ingenieros sobre políticas y cursos de capacitación.
- Administrar el conocimiento que los ingenieros generan en su trabajo.

Ingeniero de software

Utiliza el sistema SAAC para consultar información sobre el desarrollo de competencias. Los principales objetivos que puede alcanzar el ingeniero con ayuda del sistema SAAC son:

- Consultar información sobre las competencias sugeridas para el puesto de trabajo que desempeña.
- Realizar autoevaluaciones para conocer el estado (o nivel) de sus competencias.

- Colaborar con la organización en la planeación de su capacitación y desarrollo profesional.
- Administrar la información personal sobre competencias y desarrollo profesional.
- Colaborar compartiendo su conocimiento sobre las actividades de su trabajo y de desarrollo de competencias.

Tutor

Tiene la responsabilidad de revisar y aprobar el plan de desarrollo de competencias a través del sistema SAAC. El objetivo principal del tutor es:

- Colaborar con el ingeniero en la planeación de su capacitación y desarrollo profesional.

El ingeniero de software es el usuario principal del sistema SAAC, por lo que los requerimientos más importantes están enfocados en facilitarle tareas, administrar información y recursos sobre el desarrollo de competencias y su desarrollo profesional.

VI.3.2.2 Organización de la funcionalidad

La información y funcionalidad de la herramienta de soporte se divide en tres secciones: *principal*, *personal* y *administrativa*. Además, cada una de estas secciones está dividida en subsecciones que agrupan funciones que cumplen objetivos similares. En la Figura 21 se puede observar el mapa del sitio, es decir, la organización que tiene el sitio web para el desarrollo de competencias del ingeniero de software.

Sección principal

La información y funcionalidad presente en la sección principal está disponible para todos los usuarios con acceso al sistema SAAC, es decir, que todos los usuarios tienen permiso para consultar y utilizar el contenido en esta sección, con la excepción de que el administrador del sistema no puede participar ni en las encuestas ni en el foro de cooperación.

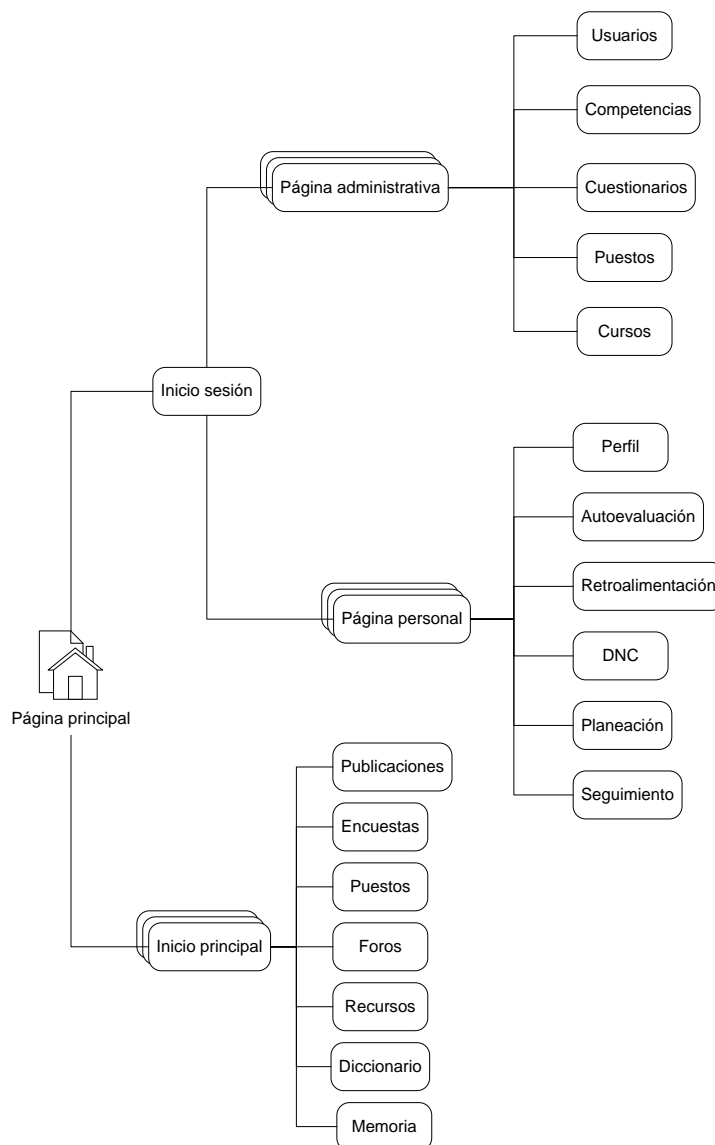


Figura 21. Mapa del sitio web (herramienta de soporte).

La sección principal se divide en varias subsecciones que se enlistan a continuación:

- Inicio: contiene la información general sobre el sistema y enlaces para acceso rápido a las distintas funciones del sistema.
- Noticias: contiene información y funcionalidad para buscar y consultar noticias publicadas en el sistema por los usuarios.

- Encuestas: contiene la información sobre las encuestas elaboradas por la empresa para conocer la opinión de los ingenieros de software sobre algún tema en específico y permite participar en ellas.
- Foros: se utiliza para expresar, compartir, discutir, opinar y debatir temas, experiencias o ideas, referentes al desarrollo profesional y las actividades laborales del ingeniero de software.
- Perfiles de puestos: contiene la información y funciones que muestran la descripción de los puestos de la empresa registrados en el sistema.
- Recursos para el autodesarrollo: contiene un conjunto de recursos que apoyan al ingeniero de software en el desarrollo de competencias.
- Diccionario de competencias: es una herramienta del sistema en el cual el usuario encuentra la funcionalidad necesaria para buscar y consultar las definiciones y ejemplos de las competencias registradas en el sistema. También, al consultar la información sobre una competencia, se despliega una lista con los enlaces a los recursos relacionados con esa competencia.

Sección personal

Para que un usuario pueda ingresar y consultar el contenido de la sección personal debe cumplir con lo siguiente:

- Tener una cuenta de usuario, la cual es asignada por la organización a través del administrador del sistema, esta cuenta incluye un identificador y una contraseña.
- Seleccionar el enlace correspondiente a la sección personal disponible en la sección principal del sistema.
- Realizar el proceso iniciar sesión utilizando el identificador y contraseña asignados.

Una vez que se cumple con los requisitos anteriores se puede consultar y utilizar la información y funciones disponibles en la sección personal. A esta sección pueden ingresar los tipos de usuarios ingenieros y tutores, las funciones son las mismas para ambos, excepto por la funcionalidad exclusiva para el usuario tutor. El contenido en esta sección está dividido como se muestra a continuación:

- Inicio: contiene la información general sobre la sección personal del sistema, así como los enlaces para acceso rápido a algunas funciones del sistema.
- Perfil personal: muestra la información personal del usuario, como el nombre, correo electrónico, nombre de usuario, etc. Además permite modificar esta información.
- Autoevaluar: contiene información y funciones que ayudan al ingeniero de software autoevaluar el estado de sus competencias.
- DNC: permite al usuario determinar sus necesidades de capacitación, de acuerdo a la información en su perfil personal y a la información sobre los perfiles de puestos.
- Plan: contiene la información y funciones que ayudan al ingeniero de software a diseñar un plan para el desarrollo de competencias.
- Retroalimentación: contiene la información y funciones necesarias para que el ingeniero de software reciba y proporcione retroalimentación de y hacia sus compañeros.

Para el caso del usuario tutor, se agrega a las subsecciones anteriores la siguiente:

- Revisar plan: contiene la información y funciones necesarias para administrar y aprobar el plan de desarrollo de un ingeniero de software asignado para su tutoría.

Sección administrativa

Para que el usuario pueda ingresar y consultar el contenido de la sección administrativa, debe cumplir con lo siguiente:

- Tener una cuenta de administrador del sistema, la cual se registra utilizando una cuenta de administrador principal, la cual está habilitada en el sistema desde su implementación, el identificador y contraseña se proporcionan a la organización.
- Seleccionar el enlace a la sección administrativa disponible tanto en la sección principal y personal.
- Realizar el proceso de autenticación de usuario utilizando el identificador y contraseña asignados.

Una vez que se satisfacen los requisitos anteriores es posible consultar y utilizar la información, opciones y funciones disponibles en la sección administrativa. En esta sección se concentra el contenido necesario para mantener actualizado el sistema. La funcionalidad de esta sección está dividida como se indica a continuación:

- Inicio: contiene la información general sobre la sección administrativa del sistema, así como los enlaces para acceso rápido a algunas funciones del sistema.
- Usuarios: contiene las funciones necesarias para administrar los usuarios del sistema.
- Competencias: contiene las funciones necesarias para administrar la información sobre competencias.
- Cuestionarios: contiene las funciones necesarias para administrar la información sobre los cuestionarios.
- Perfiles de puestos: contiene las funciones necesarias para administrar la información sobre los perfiles de puestos de la organización.

Para obtener mayor información sobre la descripción y contenido de cada una de las secciones es necesario consultar el reporte técnico sobre el diseño de la herramienta de soporte.

VI.3.3 Requerimientos específicos

VI.3.3.1 Requerimientos funcionales

Los requerimientos funcionales describen servicios que el sistema debe facilitar al usuario, indican la manera en que el sistema debe responder a cada entrada de información que el usuario hace y cómo debe responder bajo otras situaciones particulares. A continuación se describen de manera general algunos de los requerimientos funcionales más relevantes del sistema SAAC.

Requerimientos funcionales de perfil de puestos

El sistema debe tener la funcionalidad necesaria, en la sección principal, que permita al ingeniero de software consultar los perfiles de puestos de la organización. El administrador del sistema es el encargado de registrar, modificar y eliminar los perfiles de puestos de la organización desde la sección administrativa del sistema.

Requerimientos funcionales de encuestas

El sistema debe tener la funcionalidad necesaria para que el ingeniero de software pueda ver una lista de encuestas vigentes y sus respectivas descripciones, además puede responder la encuesta, ver los resultados y proponer un tema de encuesta. Las encuestas son generadas y administradas por el administrador del sistema desde la sección administrativa del sistema. Las encuestas deben aparecer en la sección principal del sistema y deben estar asociadas a una o varias competencias.

Requerimientos funcionales de foro de cooperación

El sistema debe tener la funcionalidad necesaria para que el ingeniero de software pueda consultar una lista de foros y sus respectivos temas de discusión, debe poder consultar los mensajes de un tema en específico, agregar un tema de discusión y agregar mensajes sobre un tema.

Requerimientos funcionales de recursos para el autodesarrollo

El sistema debe tener la funcionalidad necesaria para que el ingeniero de software pueda buscar, consultar, agregar, modificar y eliminar recursos para el autodesarrollo de competencias. Los recursos para el autodesarrollo deben estar disponibles en la sección principal del sistema, además deben estar asociados a una o más competencias.

Requerimientos funcionales de diccionario de competencias

El sistema debe tener la funcionalidad necesaria para que el ingeniero de software pueda buscar y consultar la definición de las competencias registradas en el diccionario de competencias. Además, puede proponer o solicitar al administrador del sistema la

definición de una competencia que no esté registrada. El administrador del sistema se encarga de agregar, modificar o eliminar las competencias que utiliza el sistema.

Requerimientos funcionales de datos personales

El sistema debe tener la funcionalidad necesaria para que el ingeniero de software pueda agregar o modificar sus datos personales. El ingeniero de software debe ser la única persona con el permiso para consultar y modificar la información personal, esta funcionalidad e información debe estar disponible en la sección personal.

Requerimientos funcionales de autoevaluación de competencia

El sistema debe tener la funcionalidad necesaria para que el ingeniero de software pueda realizar autoevaluaciones de competencia, además debe poder elegir una competencia para autoevaluar y consultar el resultado de dicha autoevaluación. La funcionalidad correspondiente a la autoevaluación de competencia debe estar disponible para el ingeniero de software desde la sección personal.

Requerimientos funcionales de plan de desarrollo de competencias

El sistema debe tener la funcionalidad necesaria para el ingeniero de software pueda diseñar, consultar, modificar, proponer y ver el estado de un plan para el desarrollo de competencias. También debe permitir al ingeniero de software enviar la información del plan de desarrollo de competencias al tutor asignado, lo anterior desde la sección personal. Adicionalmente, el tutor, desde su sección personal, debe disponer de la funcionalidad que le permita consultar la información sobre el plan de desarrollo de competencias que envía el ingeniero de software, el tutor debe poder enviar mensajes para solicitar correcciones al plan de desarrollo de competencias o en su efecto de aprobación del mismo. Cuando un tutor aprueba un plan de desarrollo de competencias de un ingeniero de software, el sistema debe cambiar el estado de dicho plan.

Requerimientos funcionales de administración de usuarios

El sistema debe tener la funcionalidad necesaria para que el administrador del sistema pueda registrar, buscar y eliminar usuarios del sistema. La funcionalidad para administrar usuarios debe ubicarse en la sección administrativa del sistema.

VI.3.3.2 Requerimientos no funcionales

A continuación se describen de manera general algunos de los requerimientos no funcionales más relevantes en el sistema.

Requerimientos de interfaz de usuario

- Adaptación de la información: el sistema debe tener la capacidad para adaptar la información que utiliza de manera que el usuario pueda visualizar y comprender el contenido.
- Organización de la información: el sistema debe organizar la información de manera que se agrupe de acuerdo a la funcionalidad y a los usuarios que usan ella.
- Información descriptiva: las secciones, botones, opciones, figuras, etc., que forman parte del sistema deben tener el texto que describa claramente su funcionalidad o contenido.
- Idioma de presentación de la información: el contenido de información debe presentarse en su totalidad en el idioma español.

Requerimientos ergonómicos

- Tamaño de la fuente: el sistema debe tener un tamaño de fuente tal que una persona con visión normal pueda a una distancia de medio metro del monitor leer el contenido.
- Contraste de contenido: el sistema debe tener los colores necesarios que faciliten la visualización de la información que se despliega.

VI.3.4 Validación de requerimientos

Existen ciertas características que la descripción de los requerimientos debe cumplir para que se considere válido por el cliente, usuario y desarrollador. A continuación se detallan algunas de las características que formaran parte de los requisitos para validar los requerimientos.

- **Completo:** la especificación del requerimiento está completa si no necesita ampliar detalles en su redacción, es decir, si proporciona la información suficiente para su comprensión.
- **Consistente:** la especificación del requerimiento es consistente si no existen requerimientos que se contradigan.
- **Correcto:** la especificación del requerimiento es correcta si precisa la funcionalidad que debe cumplir.
- **Modificable:** la especificación del requerimiento es modificable si se puede revisar y mantener un historial de los cambios hechos en el requerimiento.
- **Necesario:** debe documentar algo que el cliente realmente necesite, que sea para conformidad del sistema externo con el que se tenga interacción o para satisfacer un estándar.
- **No ambiguo:** todos los lectores a los cuales está dirigido el documento de requerimientos deben llegar a la misma y consistente interpretación del mismo.
- **Prioridad:** se debe asignar una prioridad para cada requerimiento que indique que tan esencial es el mismo para la realización del producto.
- **Realizable:** la especificación del requerimiento debe ser posible implementarse de acuerdo a las capacidades y limitaciones del sistema y el medio que lo rodea.
- **Verificable:** debe ser cuantificable de manera que permita de manera que permita hacer uso de los siguientes métodos de verificación: inspección, análisis, demostración o pruebas.

Las características anteriores son las que deben cumplir cada una de las especificaciones de requerimientos para el sistema. En la Tabla XXVIII se muestra un ejemplo de formato que se utiliza para validar cada una de las especificaciones de los requerimientos del sistema.

Tabla XXVIII. Ejemplo de formato para la validación de requerimientos.

Validación de requerimiento	
Identificador	RFPUB01
Nombre	Consultar publicación
Versión	1.0 (31-01-08)
Instrucciones	Marcar la casilla que indique la característica que la especificación del requerimiento indicado cumple.
Características	<input type="checkbox"/> Completo <input type="checkbox"/> Modificable <input type="checkbox"/> Prioridad <input type="checkbox"/> Consistente <input type="checkbox"/> Necesario <input type="checkbox"/> Realizable <input type="checkbox"/> Correcto <input type="checkbox"/> No ambiguo <input type="checkbox"/> Verificable

VI.4 Análisis de casos de uso

VI.4.1 Introducción

Los casos de uso se utilizan para especificar la comunicación y el comportamiento que debe tener el sistema en su interacción con los distintos usuarios o sistemas. Los casos de uso tienen el propósito de servir como un medio para especificar los requerimientos del sistema (mencionados en el apartado anterior) en un lenguaje comprensible tanto para el usuario (o cliente) y el desarrollador (o analista) del sistema.

Para realizar la representación de los casos de uso se utiliza el Lenguaje Unificado de Modelado (Unified Modeling Language, UML). A pesar que UML no tiene la categoría de estándar es muy reconocido y utilizado para realizar el análisis y diseño de sistemas basados en software. A continuación se presentan algunos de los principales casos de uso que se definen para el sistema SAAC.

VI.4.2 Representación de los casos de uso

La representación de los casos de uso se divide en tres partes: diagrama (gráfica), identificación y definición.

VI.4.2.1 Diagrama del caso de uso

Los diagramas o gráficas de casos de uso se utilizan para representar el comportamiento del sistema. Su función es dar una vista general y simple de la funcionalidad del sistema.

VI.4.2.2 Identificación del caso de uso

La identificación del caso de uso se utiliza para diferenciar los casos de uso que componen el análisis y diseño del sistema. La identificación del caso de uso se realiza a través de la siguiente información.

- **Identificador:** es una combinación de números y letras único para cada caso de uso. Se puede utilizar también un identificador jerárquico que tiene la siguiente forma: XXX.YYY, es decir una combinación de números y letras, que se divide en dos o más partes que se separan por un punto.
- **Nombre:** a cada caso de uso se le asigna un nombre que describa en pocas palabras la tarea que el usuario puede realizar con el sistema. Generalmente se compone de un verbo y un nombre sustantivo.
- **Historial:** se compone de la siguiente información:
 - **Creado por:** indica el nombre de la persona que documenta el caso de uso.
 - **Fecha de creación:** indica la fecha en que se elaboró la primera versión del caso de uso.
 - **Última actualización por:** indica el nombre de la persona que modifica la información de la definición del caso de uso.
 - **Fecha de última actualización:** Indica la fecha de la última vez que se modificó la información de la definición del caso de uso.

VI.4.2.3 Definición del caso de uso

La definición del caso de uso indica los detalles específicos de cada caso de uso. La información que se presenta en la definición del caso de uso es la siguiente.

- **Actor:** es una persona u otro sistema externo que interactúa con el sistema para ejecutar el caso de uso. El actor debe tener asignado un nombre que represente una

clase, rol o identificador. Un solo actor es el que inicia el caso de uso y otros actores pueden participar para ayudar a completar el mismo caso de uso.

- Disparador: se refiere a un evento que identifica o provoca el inicio del caso de uso.
- Descripción: es un texto corto que describe el objetivo del caso de uso. Puede indicar de manera general la secuencia de acciones y los resultados obtenidos en la ejecución del caso de uso.
- Precondiciones: son las condiciones o estados que se deben cumplir antes que se ejecute el caso de uso. Se debe mostrar una lista de las precondiciones y sus respectivas descripciones.
- Poscondiciones: son las condiciones o estados presentes una vez que se concluye la ejecución del caso de uso. Se debe mostrar una lista de las poscondiciones y sus respectivas descripciones.
- Flujo normal: indica con detalle la secuencia de acciones que el usuario debe realizar y la respuesta que obtendrá del sistema durante la ejecución normal del caso de uso.
- Flujo alternativo: describe una secuencia alternativa de acciones que el usuario puede realizar al presentarse una condición o evento que modifique el flujo normal de la ejecución del caso de uso.
- Excepciones: describe una condición o estado que provoca un error en la ejecución del caso de uso y especifica la manera en la cual el sistema responde a esa condición.
- Incluye: lista de otros casos de uso que se incluyen (utiliza) en el caso de uso que se describe.
- Prioridad: indica la prioridad de implementación de la funcionalidad que el caso de uso permite ejecutar.
- Frecuencia de uso: indica el número de veces por unidad de tiempo que se cree que el actor ejecutará el caso de uso.
- Reglas del negocio: lista de las reglas del negocio que afectan a este caso de uso.

- Requerimientos especiales: son requerimientos adicionales que podrían necesitar durante la implementación del caso de uso.
- Suposiciones: lista de suposiciones que se tomaron en cuenta durante el análisis y diseño del caso de uso.
- Notas: lista de cualquier comentario adicional acerca del caso de uso.

En el siguiente apartado se modelan algunos de los casos de usos más representativos del sistema.

VI.4.3 Casos de uso

VI.4.3.1 Caso de uso administrar usuarios

En la Figura 22 se muestra el diagrama del caso de uso administrar usuarios. Se observa que el actor es el administrador del sistema y que el caso de uso administrar usuario extiende a los casos de uso registrar usuario y consultar usuario, este último a su vez extiende a los casos de uso modificar usuario y eliminar usuario.

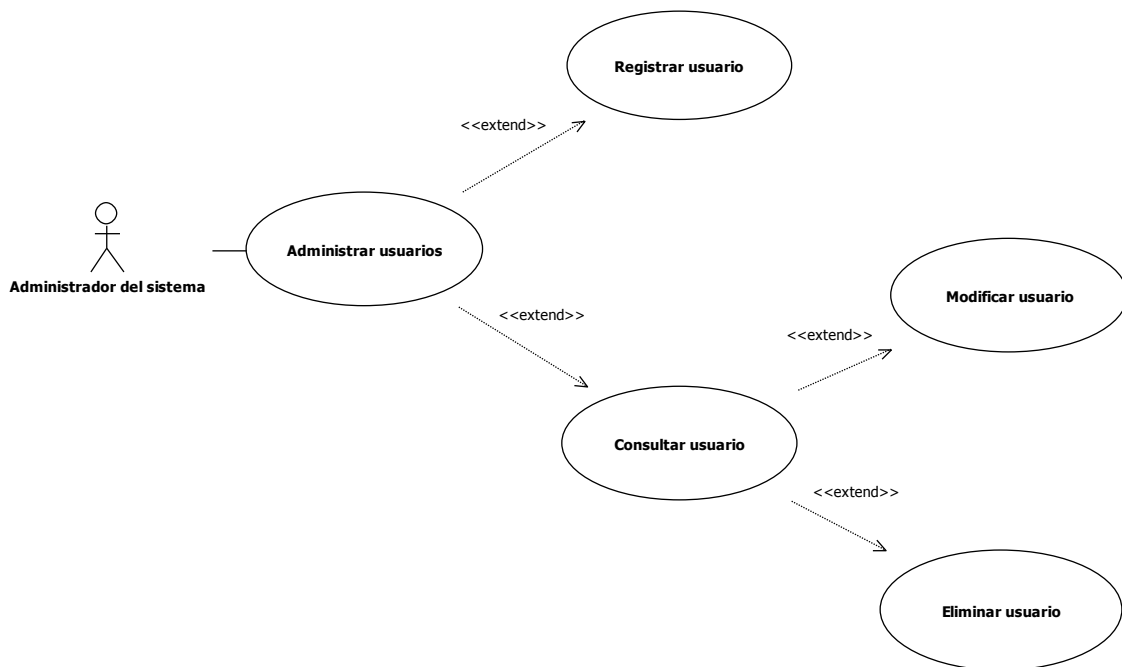


Figura 22. Diagrama del caso de uso administrar usuarios.

Identificador:	AUS		
Nombre:	Administrar usuarios.		
Creado por:	José G. Rivera I.	Modificado por:	
Fecha de creación:	24/mayo/2008	Última actualización:	

Actores:	Administrador del sistema
Descripción:	El sistema registra la información sobre los usuarios (nombre, identificador y contraseña) para que éstos puedan tener acceso a la funcionalidad del sistema. La administración de los usuarios incluye las tareas registrar, consultar, modificar y eliminar información sobre los usuarios.
Precondiciones:	1. El usuario debe iniciar sesión como administrador del sistema.
Flujo normal:	<ol style="list-style-type: none"> 1. El usuario selecciona la opción administrar usuarios del menú de la sección administrativa del sistema. 2. Si el usuario selecciona la opción “Registrar nuevo usuario” se ejecuta el caso de uso registrar usuario (AUS.R). 3. Si el usuario selecciona la opción “Consultar usuario” se ejecuta el caso de uso consultar usuario (AUS.C).
Incluye:	<ol style="list-style-type: none"> 1. Registrar usuario. 2. Consultar usuario. 3. Modificar usuario. 4. Eliminar usuario.
Prioridad:	Media.

VI.4.3.2 Caso de uso administrar competencias

En la Figura 23 se muestra el diagrama del caso de uso administrar competencias. Se observa que el actor es el administrador del sistema y que el caso de uso administrar competencias extiende a los casos de uso registrar competencias y consultar competencias, este último a su vez extiende a los casos de uso modificar y eliminar competencias. Este caso de uso permite ejecutar tareas similares al caso de uso anterior (administrar usuarios).

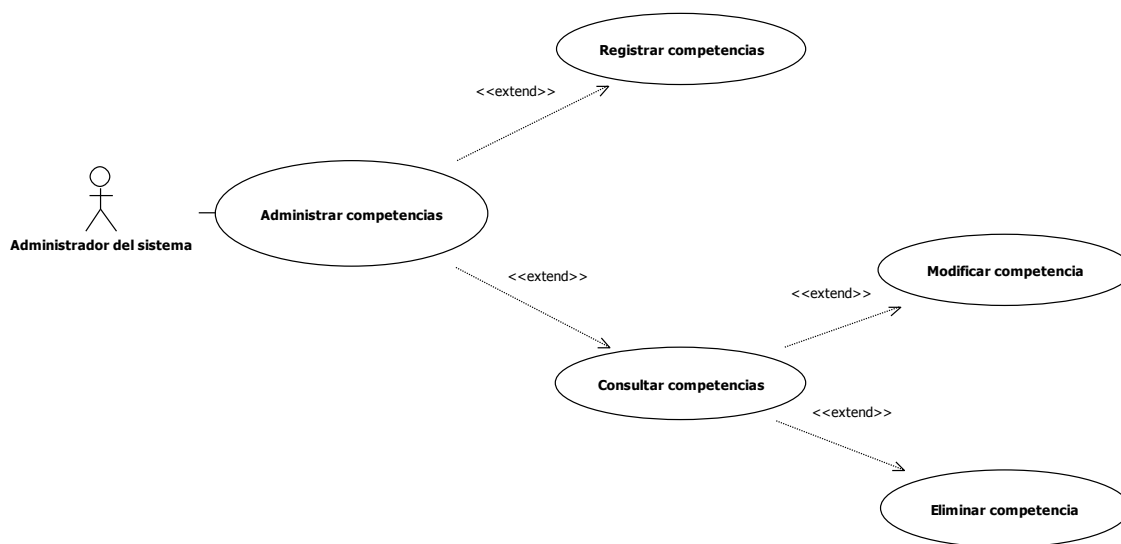


Figura 23. Diagrama del caso de uso administrar competencias.

Identificador:	ACO		
Nombre:	Administrar competencias.		
Creado por:	José G. Rivera I.	Modificado por:	
Fecha de creación:	24/mayo/2008	Última actualización:	

Actores:	Administrador del sistema
Descripción:	El sistema registra la información sobre las competencias (definición, criterios, etc.) para que éstas se puedan utilizar para ejecutar las diferentes funciones del sistema. Administrar competencias incluye las tareas registrar, consultar, modificar y eliminar competencias.
Precondiciones:	1. El usuario debe iniciar sesión como administrador del sistema.
Flujo normal:	<ol style="list-style-type: none"> 1. El usuario selecciona la opción administrar competencias del menú de la sección administración del sistema. 2. Si el usuario selecciona la opción “Registrar nueva competencia” se ejecuta el caso de uso registrar competencia (ACO.R). 3. Si el usuario selecciona la opción “Consultar competencia” se ejecuta el caso de uso consultar competencia (ACO.C).
Incluye:	<ol style="list-style-type: none"> 1. Registrar competencia. 2. Consultar competencia. 3. Modificar competencia. 4. Eliminar competencia.
Prioridad:	Alta.

VI.4.3.3 Caso de uso administrar cuestionarios

En la Figura 24 se muestra el diagrama del caso de uso administrar cuestionarios. Se observa que el actor es el administrador del sistema y que el caso de uso administrar cuestionario extiende a los casos de uso registrar cuestionario y consultar cuestionario, este último a su vez extiende a los casos de uso modificar, publicar, quitar, generar y eliminar cuestionarios.

VI.4.3.4 Caso de uso administrar planes

En la Figura 25 se muestra el diagrama del caso de uso administrar planes. Se observa que el actor es el ingeniero de software y que el caso de uso administrar planes extiende a los casos de uso registrar y consultar planes, este último a su vez extiende a los casos de uso modificar, revisar y eliminar planes. El caso de uso revisar planes lo realiza el ingeniero de software en colaboración con el tutor.

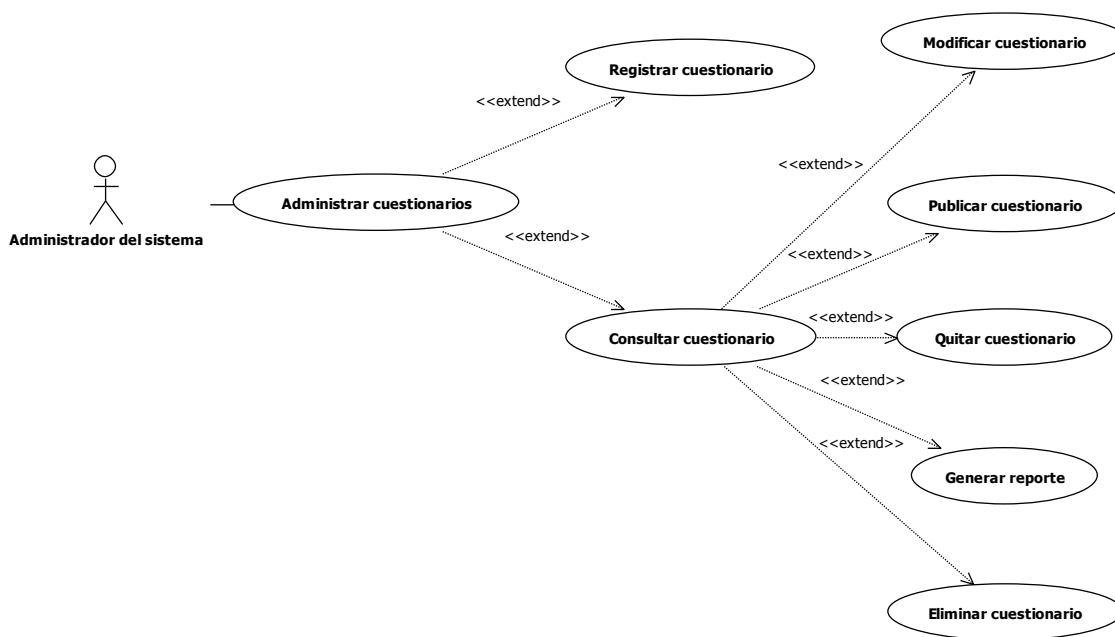


Figura 24. Caso de uso administrar cuestionarios.

Identificador:	ACU		
Nombre:	Administrar cuestionarios		
Creado por:	José G. Rivera I.	Modificado por:	
Fecha de creación:	24/mayo/2008	Última actualización:	

Actores:	Agente de cambio
Descripción:	El sistema registra la información sobre los cuestionarios (encuesta, evaluación, retroalimentación, etc.) para que éstos se puedan utilizar para ejecutar las diferentes funciones del sistema. Administrar cuestionarios incluye las tareas registrar, consultar, modificar, eliminar, publicar, quitar y generar reporte de los cuestionarios.
Precondiciones:	1. El usuario debe iniciar sesión como administrador del sistema.
Flujo normal:	1. El usuario selecciona la opción administrar cuestionarios del menú de la sección administración del sistema. 2. Si el usuario selecciona la opción “Registrar nuevo cuestionario” se ejecuta el caso de uso registrar cuestionario (ACU.R). 3. Si el usuario selecciona la opción “Consultar cuestionario” se ejecuta el caso de uso consultar cuestionario (ACU.C).
Incluye:	1. Registrar cuestionario. 2. Consultar cuestionario. 3. Modificar cuestionario. 4. Eliminar cuestionario. 5. Publicar cuestionario. 6. Quitar cuestionario. 7. Generar reporte de cuestionario.
Prioridad:	Alta.

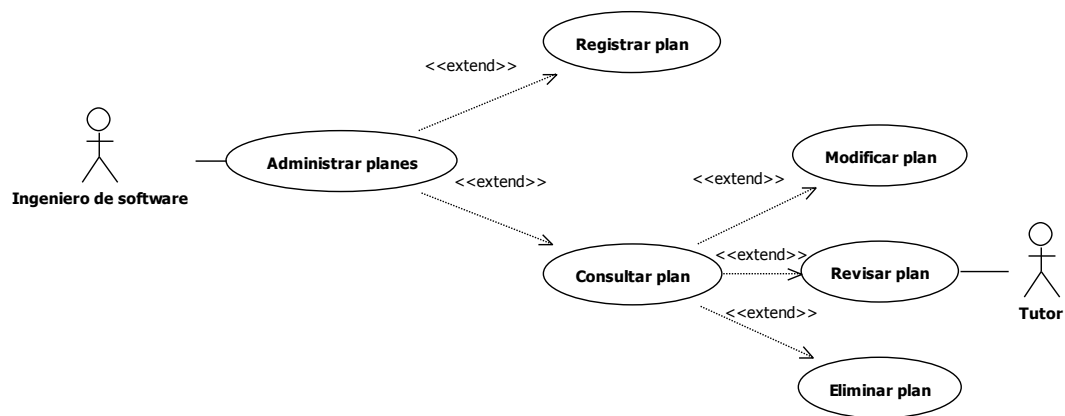


Figura 25. Caso de uso administrar planes.

Identificador:	APL		
Nombre:	Administrar planes		
Creado por:	José G. Rivera I.	Modificado por:	
Fecha de creación:	24/mayo/2008	Última actualización:	

Actores:	Ingeniero de software
Descripción:	El sistema registra la información sobre los planes para el autodesarrollo del ingeniero de software para que éste pueda utilizarlos en sus actividades de desarrollo profesional. El tutor es el responsable de revisar y aprobar los planes del ingeniero de software que le fue asignado. Administrar planes para el autodesarrollo incluye las tareas registrar, consultar, modificar, revisar y eliminar planes.
Precondiciones:	1. El usuario debe iniciar sesión como usuario del sistema.
Flujo normal:	1. El usuario selecciona la opción administrar planes del menú de la sección personal del sistema. 2. Si el usuario selecciona la opción “Registrar plan” se ejecuta el caso de uso registrar plan (APL.R). 3. Si el usuario selecciona la opción “Consultar plan” se ejecuta el caso de uso consultar recurso (APL.C).
Incluye:	1. Registrar plan. 2. Consultar plan. 3. Modificar plan. 4. Revisar plan. 5. Eliminar plan.
Prioridad:	Alta.

VI.5 Diseño del sistema

La herramienta de soporte es un sitio web que se publica en el portal interno de la organización (Intranet e incluso puede publicarse en Internet). A continuación se presenta brevemente algunas características del diseño del sistema como su arquitectura, diagramas de clases, diccionario de datos y diseño de la base de datos. En el reporte técnico se puede consultar mayor información acerca de estas características.

VI.5.2 Arquitectura del sistema

La arquitectura de un sistema basado en software es la organización fundamental de un sistema plasmado en sus componentes, la relación de cada uno de esos componentes con el resto, con el entorno y los principios que orientan su diseño y evolución.

Debido a que el sistema es un sitio web o sistema informático, la arquitectura más adecuada y común para este tipo de sistema se basa en el modelo cliente-servidor. El modelo cliente-servidor es un sistema de dispositivos conectados en red que permite compartir recursos y aplicaciones que se almacenan en una computadora central (servidor), la cual administra, ejecuta y atiende las peticiones de otras computadoras distribuidas en la red (cliente).

Una de las principales ventajas de utilizar esta arquitectura es que para realizar actualizaciones del sistema únicamente se requiere actualizar el software en el servidor y este cambio se ve reflejado en los clientes que accedan al servidor. Además, se reduce el tráfico en la red, ya que el cliente solicita información al servidor solamente cuando éste la necesita.

Los sistemas informáticos basados en el modelo cliente-servidor, se pueden apoyar en el patrón de diseño Modelo Vista Controlador (MVC), que tiene la característica de separar los datos, la interfaz de usuario y la lógica de control. Para una aplicación Web (como en el caso del sistema SAAC), el componente vista es la página HTML y el código que proporciona los datos dinámicos a la página, el componente modelo es el sistema de gestión de base de datos y la lógica del negocio, y finalmente el componente controlador es el responsable de recibir los datos de entrada desde el componente vista y decidir qué modelo y vista utilizar para presentar la información al usuario.

En la Figura 26 se representa la vista física de la arquitectura del sistema SAAC, en ella se observa el servidor web que es un programa instalado en una computadora que implementa el protocolo HTTP, el cual se utiliza para transferir páginas web por la red. El servidor web se conecta a un servidor de base de datos (BD), en éste se almacena la información que utiliza el sistema. Los clientes (computadoras, laptops y otros dispositivos) se conectan al servidor por medio de una red cableada (Ethernet) o inalámbrica (Wireless). Las

computadoras que se ubican fuera de la Intranet (red local) donde está instalado el sistema SAAC pueden (si se cuenta con la infraestructura y configuración pertinente) conectarse al sistema (servidor web) a través de Internet.

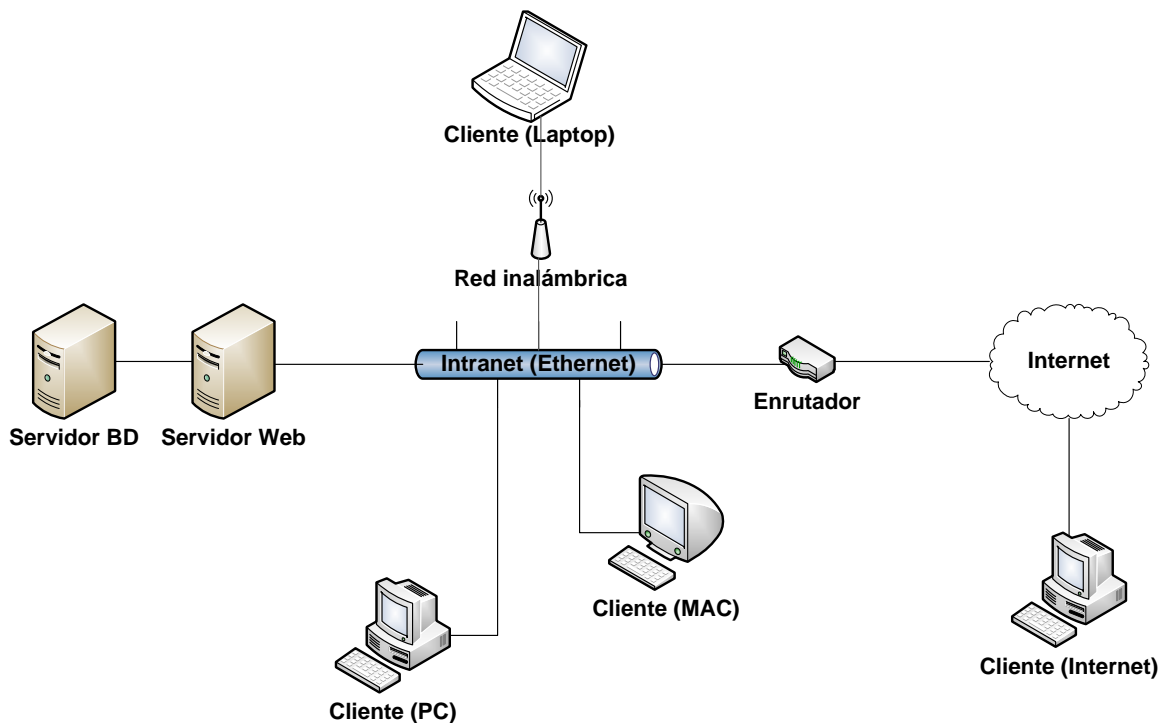


Figura 26. Diagrama de la arquitectura del sistema SAAC.

En la Figura 27 se muestran los componentes del MVC de la arquitectura del sistema SAAC. Se observa que el componente vista se implementa utilizando las tecnologías XHTML, CSS, Javascript y XML, la combinación de los dos últimos también se conoce como AJAX. Por otro lado, el componente controlador se implementa a través de Servicios Web (Web Services, WS), los cuales reciben datos desde el componente vista y determina qué acción (modelo) realizar. Finalmente el componente modelo se implementa utilizando PHP como lenguaje de programación y MySQL como gestor de base de datos. Una característica de las tecnologías que se proponen para implementar el sistema, es que las licencias de éstas permiten su uso sin que el usuario tenga que pagar por ello.

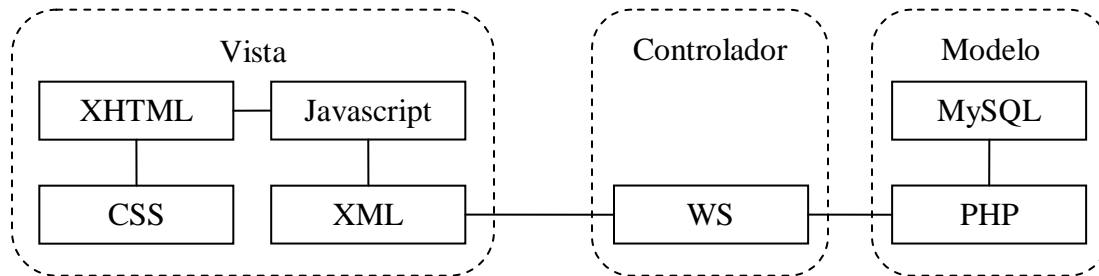


Figura 27. Diagrama de la arquitectura del sistema SAAC (Modelo Vista Controlador).

La funcionalidad del componente vista se ejecuta del lado del cliente, mientras que la funcionalidad del componente controlador y del componente modelo se ejecuta del lado del servidor. Para observar las páginas del sistema SAAC, el cliente debe tener instalado un programa navegador web³ (tal como Internet Explorer, Mozilla Firefox, Opera, Safari o NetScape). El navegador web instalado en el cliente se comunica con el servidor web (ver Figura 26) utilizando el protocolo HTTP perteneciente a la capa de aplicación del modelo de referencia OSI. La función básica del navegador web es solicitar la página web al servidor y desplegar su contenido, la página web es una hoja electrónica que contiene información descrita en el Lenguaje de Marcas de Hipertexto (HTML).

La página web puede tener asociado una o varias Hojas de Estilo en Cascada (Cascading Style Sheets, CSS), ésta se utiliza para definir la forma en que se presenta la información de una página web o HTML, la idea de utilizar CSS es separar la estructura del documento de su presentación visual (color, tamaño, tipo de fuente, etc.). Adicionalmente la página web, tiene incrustado (o embebido) código Javascript, el cual es un lenguaje de programación interpretado, es decir es un lenguaje que requiere de un intérprete para ejecutarse (este intérprete lo incluyen la mayoría de los navegadores web actuales), un intérprete es un programa capaz de analizar y ejecutar el código de otros programas, a este tipo de lenguajes también se le conoce como lenguaje script.

³ Es una aplicación de software que permite recuperar y visualizar documentos de hipertexto (HTML).

Javascript permite ejecutar programas del lado del cliente en lugar del lado del servidor, lo cual reduce la carga de procesamiento en el servidor y permite al usuario (cliente) tener respuesta más rápida de sus solicitudes.

Por otro lado, AJAX es una técnica de desarrollo web que permite crear páginas dinámicas e interactivas. La palabra AJAX proviene del acrónimo Javascript Asíncrono y XML (Asynchronous Javascript And Xml). AJAX es una técnica asíncrona en el sentido de que los datos que utiliza el cliente los solicita al servidor sin interferir con la visualización ni el comportamiento de la página web. XML es un Lenguaje de Marcas eXtensible (eXtensible Markup Language), en la técnica AJAX se utiliza el formato XML para transferir (en un formato estándar) los datos entre el cliente y el servidor.

Del lado del cliente se utilizan los servicios web (Web Services, WS) para realizar el intercambio de datos entre el cliente y el servidor. Los servicios web, al igual que la técnica AJAX, utilizan el formato XML para realizar el intercambio de datos, así la página web utilizan AJAX para solicitar alguno de los servicios web disponibles en el servidor. El servicio web realiza el control de la información que recibe y la información que envía desde y hacia el cliente, el servicio web al recibir los datos determina qué funciones ejecutar de acuerdo al componente modelo. Las funciones definen la lógica de las operaciones sobre los datos que utiliza el sistema, estas funciones están codificadas utilizando el lenguaje PHP, que de manera similar a Javascript es un lenguaje de programación interpretado, sin embargo éste opera del lado del servidor. Finalmente, los datos que se utilizan en la funcionalidad del sistema SAAC son almacenados en una base de datos utilizando el sistema de gestión de base de datos MySQL.

VI.5.3 Diagrama de clases

Para realizar el diseño del sistema se utiliza el lenguaje unificado de modelado (Unified Modeling Language, UML), el cual define, entre otras cosas, una serie de diagramas que permiten el diseño de sistemas. En este apartado se presenta el diagrama de clases del sistema. En la Figura 28 se pueden observar el diagrama UML de las clases que implementan la funcionalidad del sistema SAAC.

VI.5.4 Diccionario de datos

El diccionario de datos especifica las características lógicas de los datos que utiliza el sistema para funcionar. Entre la información que se incluye en el diccionario de datos se encuentran: nombre, descripción, alias, organización, entre otra, acerca de cada uno de los diferentes tipos de datos que utiliza el sistema para su funcionamiento normal. A continuación se enlista el nombre y descripción de los datos del sistema, para obtener más detalles sobre los mismos es necesario consultar el reporte técnico sobre la herramienta de soporte.

- *ACCIÓN_PLAN*: se refiere al conjunto de acciones que forman el plan de desarrollo de competencias del ingeniero de software.
- *ARCHIVO*: se refiere a un recurso digital que se almacena en una computadora y que tiene información relevante para el desarrollo de competencias.
- *CATEGORÍA_COMPETENCIA*: se refiere a un grupo de competencias que tienen características similares y que al utilizar en conjunto permiten al ingeniero de software mostrar competencia en una determinada situación o actividad.
- *COMPETENCIA*: se refiere al conjunto de conocimientos, habilidades y comportamientos que permiten el desempeño sobresaliente del ingeniero de software en las actividades contemporáneas del desarrollo de software.
- *COMPETENCIA_PERFIL_PERSONAL*: se refiere al conjunto de competencias que tiene el ingeniero de software y que se utilizan para describir sus capacidades como profesional a través del documento de perfil personal.
- *COMPETENCIA_PUESTO*: se refiere a las competencias que exige un rol o puesto al ingeniero de software para que éste logre el desempeño sobresaliente de las actividades correspondientes a ese rol o puesto.
- *CRITERIO*: se refiere a las destrezas o cualidades que el ingeniero de software debe demostrar para poder decir que tiene una determinada competencia.
- *CUESTIONARIO*: se conforma de una serie de preguntas que tiene un objetivo particular y que sirven para conocer la opinión o percepción del ingeniero de software sobre un determinado tema.

- *DNC*: significa Detección de Necesidades de Capacitación, es una técnica que permite determinar las competencias que requiere desarrollar el ingeniero de software para desempeñar un rol o puesto en ingeniería de software.
- *ENLACE*: indica la ubicación de un archivo digital.
- *ESCALA*: sirve para medir el nivel o grado en el cual el ingeniero de software cumple uno o varios de los criterios de una competencia.
- *ESTADO_PLAN*: se utiliza para indicar la situación de un plan de desarrollo de competencias, es decir, si éste está en desarrollo, revisión, ejecución, etc.
- *FORO*: es un medio de comunicación asíncrona que permite al ingeniero de software compartir ideas y opiniones sobre temas relacionados con el desarrollo de competencias.
- *GRUPO_RESPUESTA*: se refiere al conjunto de opciones de respuestas que tiene una pregunta.
- *IMAGEN_PUBLICACIÓN*: es una imagen que pertenece a una o varias publicaciones.
- *MENSAJE_TEMA*: es un texto que contiene la información que un ingeniero de software expresa sobre un tema en el foro de discusión.
- *NIVEL_COMPETENCIA*: expresión que indica si el ingeniero de software cumple con cierta cantidad de criterios para demostrar una competencia.
- *PERFIL_PERSONAL*: es un texto que indica las competencias que tiene un determinado ingeniero de software.
- *PLAN*: se refiere al conjunto de acciones distribuidas en un período de tiempo, a los criterios de evaluación de esas acciones y al objetivo o competencias que tiene como finalidad la ejecución de esas acciones.
- *PREGUNTA*: interrogación que se hace al ingeniero de software sobre un tema.
- *PUBLICACIÓN*: información acerca de un objeto o tema que se muestra a través del sistema.
- *PUESTO*: se refiere al conjunto de actividades relacionadas entre sí que cumple con objetivos bien identificados.

- *RECURSO*: se refiere al archivo digital que contiene información sobre el desarrollo de competencias.
- *RESPUESTA*: se refiere a una expresión que representa con cierto grado de exactitud la opinión o el conocimiento que tiene el ingeniero de software sobre una pregunta.
- *RESULTADO_CUESTIONARIO*: se refiere al conjunto de respuestas que proporciona el ingeniero de software a las preguntas que forman un cuestionario.
- *RESULTADO_RESPUESTA*: se refiere a una respuesta en específico que el ingeniero de software selecciona relativa a una pregunta.
- *TÉCNICA_DESARROLLO*: se refiere al conjunto de procedimientos, acciones y recursos que el ingeniero de software puede ejecutar para lograr desarrollar una determinada competencia.
- *TEMA_FORO*: se refiere a un asunto u objeto sobre el cual los ingenieros de software expresan su opinión (a través de mensajes) en un foro.
- *TIPO_ARCHIVO*: se refiere a las características que distinguen a un grupo de archivos.
- *TIPO_COMPETENCIA*: se refiere a las características que distinguen a un grupo de competencias.
- *TIPO_CUESTIONARIO*: se refiere a las características que distinguen a un grupo de cuestionarios.
- *TIPO_ENLACE*: se refiere a las características que distinguen a un grupo de enlaces.
- *TIPO_RESPUESTA*: se refiere a las características que distinguen a un grupo de respuestas.
- *TIPO_TÉCNICA*: se refiere a las características que distinguen a un grupo de técnicas.
- *TIPO_USUARIO*: se refiere a las características que distinguen a un grupo de usuarios.

- *USUARIOS*: se refiere a las personas que utilizan el sistema para el desarrollo de competencias.

VI.5.5 Diseño de la base de datos

Para almacenar y que se puedan volver a usar los datos que necesita el sistema para funcionar, se diseña una base de datos con una estructura que permita manejar los datos que se presentan en el apartado anterior. Así, el diseño de las tablas que almacenan los datos y que forman esta base de datos es el que se presenta en la Figura 29.

La Figura 29 representa el diagrama entidad/relación de la base de datos del sistema SAAC. En ella se observa cómo se almacena los datos agrupados en “tablas” y la relación entre ellas. El diagrama que se muestra en la Figura 29 permite dar al lector una perspectiva de la relación general que existe entre las tablas, para ver con mayor detalle las partes del diagrama entidad/relación, puede consultar el mismo diagrama pero con mayor resolución en el anexo C.

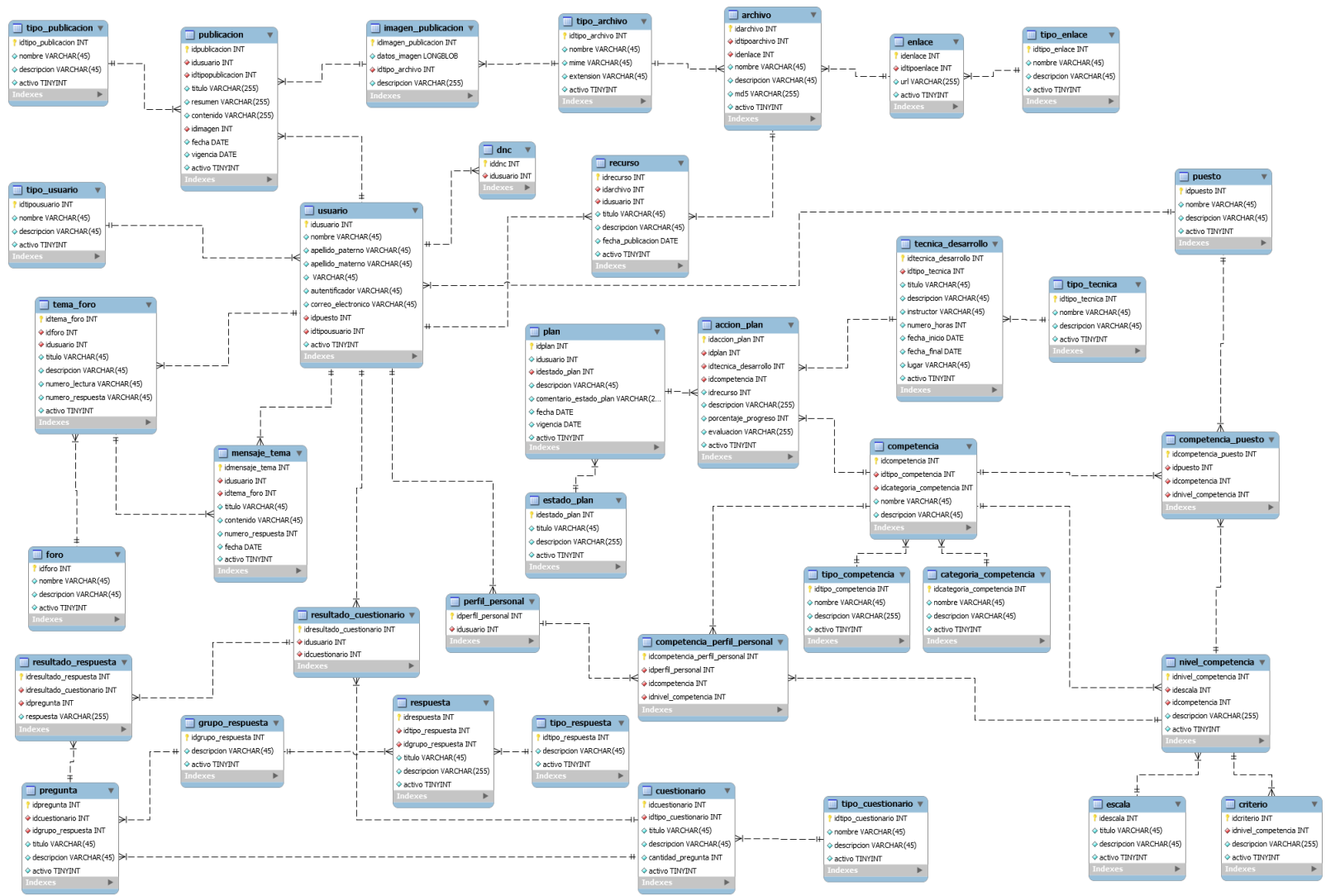


Figura 29. Diagrama Entidad/Relación de la base de datos del sistema SAAC.

VI.6 Descripción de la funcionalidad

El ingeniero de software es el usuario principal del sistema SAAC, sin embargo requiere de la colaboración de otros usuarios para ejecutar algunas de las funciones que implementa el sistema. A continuación se describe la interacción que realiza el ingeniero de software con otros usuarios a través del sistema.

Colaboración entre administrador del sistema e ingeniero de software

El administrador del sistema es el usuario que tiene el permiso necesario para actualizar la información que utiliza el sistema, en particular la información sobre las competencias que es clave en las diferentes funciones del sistema SAAC. El administrador del sistema provee la información sobre las competencias la cual de manera automática pasa a formar parte del diccionario de competencias. Para que el administrador del sistema pueda agregar una competencia es necesario que realice el proceso de inicio de sesión, ingrese a la sección administrativa, a la subsección de competencias (3) y presione el botón agregar nueva competencia (2), que se muestra en la Figura 30.



Figura 30. Agregar nueva competencia.

En esta misma página (que se muestra en la Figura 30) el administrador del sistema también puede realizar una búsqueda de alguna competencia que la hayan registrado

anteriormente, para realizar la búsqueda escribe una palabra relacionada con la competencias en el área de búsqueda (1), el resultado de la búsqueda aparece en (4) como se muestra en la Figura 30.

Al presionar el botón agregar nueva competencia (2) o seleccionar una competencia en el resultado de la búsqueda (4), aparece una nueva página en la cual se debe proporcionar la información sobre la competencia que desea agregar para el caso que se haya elegido agregar una nueva competencia o se muestra la información en modo de edición si se selecciona una competencia existente. En la Figura 31 se observa la página en la cual el administrador del sistema proporciona información sobre la competencia como el tipo (1), categoría (2), nombre (3) y descripción (4), en el área de datos generales.

The screenshot shows the SAAC system interface for managing competencies. At the top left is the SAAC logo with an owl icon. The main heading is "Consultar información de competencia". Below this, the "DATOS GENERALES" section contains the following fields:

- Tipo:** A dropdown menu with "Competencia personal" selected. A callout circle with the number 1 points to this field.
- Categoría:** A dropdown menu with "Manejar y resolver conflictos" selected. A callout circle with the number 2 points to this field.
- Nombre de la competencia:** A text input field containing "Análisis de problemas". A callout circle with the number 3 points to this field.
- Descripción:** A text area containing the text "Eficacia a la hora identificar un problema, buscar datos pertinentes al respecto, reconocer la información relevante y encontrar las posibles causas". A callout circle with the number 4 points to this field.

Figura 31. Agregar información sobre nueva competencia (datos generales).

Adicionalmente se puede proporcionar otra información como los criterios (se pueden agregar los criterios que sean necesarios), en (5) se escribe la descripción del criterio, se selecciona el nivel de competencia al que pertenece ese criterio (6), se presiona el botón guardar (7) y la información aparece en la tabla de criterios (8) como se muestra en la Figura 32. De manera similar como se agrega un criterio de competencia, es posible

también agregar ejemplos, estrategias y recursos para el desarrollo de dicha competencia, cada uno de los anteriores se puede guardar de manera independiente.

CRITERIOS

Descripción del criterio:

Nivel de competencia: 1 - Básico

Guardar

Tabla de criterios:

Descripción	Nivel
Descripción de uno de los criterios de la competencia.	1 - Básico

Figura 32. Agregar información sobre nueva competencia (criterios).

Una vez que el administrador guarda la información sobre una competencia, el ingeniero de software puede consultarla a través del diccionario de competencias. Para ingresar al diccionario de competencias el ingeniero de software selecciona el enlace correspondiente (1) desde la página principal del sistema SAAC como se muestra en la Figura 33.

En la subsección de diccionario de competencias el usuario puede buscar una competencia (1). El usuario escribe una palabra relacionada a la competencia que busca (esta palabra puede ser el nombre u otra en la descripción, criterios o ejemplos), presiona el botón aceptar y el resultado de la búsqueda que realiza el sistema se despliega en el área de resultado de búsqueda (2) como se muestra en la Figura 34. El usuario da clic en el nombre de la competencia de la cual desea obtener mayor información.

Una vez que el usuario da clic en la competencia que quiere consultar, el sistema despliega otra página en la cual se despliega la información sobre dicha competencia, como se muestra en la Figura 35.

SAAC 

[Inicio](#) [Noticias](#) [Encuestas](#) [Foros](#) [Perfiles](#) [Sesión](#)

El Sistema de Apoyo para el Autodesarrollo de Competencias (SAAC) es una herramienta de soporte para realizar planes de desarrollo profesional del Ingeniero de Software (SwE). Con ayuda del sistema SAAC, el SwE puede realizar autoevaluaciones que le permitan calcular el nivel (o grado) de desarrollo de sus competencias.

SwE Competente
MEJORA TU RENDIMIENTO 

El sistema SAAC dispone de un **Diccionario de Competencias** para poder consultar la información referente a una competencia.


[→Leer más](#)

1 →

[¿Cómo me mantengo actualizado?](#)


CONOCE TUS COMPETENCIAS

Figura 33. Enlace para ingresar al diccionario de competencias.

SAAC 

El **DICCIONARIO DE COMPETENCIAS** sirve de referencia tanto a la organización como al ingeniero de software. En él se define a las competencias, además para cada competencia se describen y enlistan los criterios y ejemplos sobre cómo el ingeniero de software demuestra que tiene una determinada competencia.

[→leermás](#)

Diccionario DE COMPETENCIAS 

El sistema SAAC dispone de un diccionario de competencias que consiste de una compilación de definiciones de competencias utilizadas en la ingeniería de software.

[→Leer más](#)

[¿Cómo me mantengo actualizado profesionalmente?](#)

1 →

Buscar competencia: [Aceptar](#)

Resultado de búsqueda: ← **2**

- Administración de proyectos
- Análisis de requerimientos
- Diseño de sistemas de software
- Programación
- Pruebas
- Administración de la configuración
- Calidad
- Validación y verificación

Figura 34. Subsección diccionario de competencias.



The image shows a screenshot of a web page titled 'Diccionario de Competencias'. At the top left is the SAAC logo, which includes the letters 'SAAC' in a bold, sans-serif font with a red lightbulb icon above the 'A', and a stylized owl illustration to its right. Below the logo, the page title 'Diccionario de Competencias' is centered in green. Underneath, a sub-header 'Competencias sociales -> Cooperar y trabajar en equipo' is displayed in red and green. The main content area is titled 'Deseos de contribuir' and contains a paragraph describing the goal of team work. Below this is a paragraph explaining the characteristics of people who work well in teams. The 'Lista de criterios' section is divided into two levels: 'Nivel maestro' and 'Nivel aceptable', each with a numbered list of four criteria.

SAAC 

Diccionario de Competencias

Competencias sociales -> Cooperar y trabajar en equipo

Deseos de contribuir

Estimular a los integrantes del equipo para que trabajen juntos de forma efectiva.

Las personas capaces de trabajar en equipo generan una atmósfera de colaboración amistosa y son un modelo de respeto, utilidad y cooperación. Son personas que saben conjugar el compromiso activo y entusiasta con el esfuerzo común y, en ese sentido, construir una identidad colectiva y alentar el espíritu de coros.

Lista de criterios

Nivel maestro:

1. Alienta y fomenta el espíritu en equipo, animando y motivando a los demás al cumplimiento de los objetivos de la organización.
2. Expresa satisfacción personal por el éxito de los demás, independientemente de si forma parte de su equipo de trabajo.
3. Toma los objetivos de la organización como propios, priorizándolos frente a los objetivos propios o de su grupo, y generando con su ejemplo el compromiso de los demás miembros del equipo.
4. Actúa para generar un ambiente amistoso, de buen clima y cooperación.

Nivel aceptable:

1. Solicita la opinión de los miembros del equipo.
2. Mantiene una actitud abierta para aprender de los demás.
3. Promueve la colaboración entre equipos.
4. Valora la contribución de ideas y experiencias ajenas, aún cuando se le planteen diferentes puntos de vista a los suyos.

Figura 35. Ejemplo de descripción detallada de competencia.

Los cuestionarios, perfiles de puesto y cursos de capacitación pueden tener asociadas a ellos, una o varias competencias. El administrador del sistema es la persona que tiene el permiso necesario para agregar la información de los cuestionarios, perfiles de puesto y cursos. Para agregar un nuevo cuestionario, el administrador del sistema debe realizar el proceso de inicio de sesión, ingresar a la sección administrativa y a la subsección cuestionarios (3), ahí debe presionar el botón nuevo cuestionario (2) que se ubica en el menú de la izquierda como se indica en la Figura 36. Además, en esta página se puede buscar un cuestionario en particular escribiendo una palabra relacionada con el cuestionario

en el área de búsqueda (1), el resultado de la búsqueda se despliega en (4) como se muestra en la Figura 36.




Figura 36. Agregar nuevo cuestionario.

Una vez que se presiona el botón nuevo cuestionario o se selecciona el nombre de un cuestionario en el resultado de la búsqueda, el sistema despliega una nueva página en la cual el administrador del sistema agrega la información que se requiere para registrar un nuevo cuestionario o modificar la información del cuestionario seleccionado. En la Figura 37 se observa un ejemplo de un caso en el cual se agrega un nuevo cuestionario. En el área de datos generales sobre el cuestionario (ver Figura 37) el administrador del sistema debe indicar el nombre (1) y descripción (2), posteriormente puede agregar las preguntas (3) (4) que sean necesarias, a cada una de ellas se le asigna un grupo de respuestas (5), las preguntas que se hayan agregado para ese cuestionario aparecen en la lista de preguntas (6) como se muestra en la Figura 37.

Para responder una encuesta, el ingeniero de software ingresa a la subsección de encuestas (1), entonces se despliega una página en la cual se encuentra la opción para buscar una encuesta (2), del resultado de la búsqueda (3) el ingeniero de software presiona el botón responder (4) de la encuesta en la que desea participar, tal como se muestra en la Figura 38.

Después de presionar el botón responder cuestionario el sistema despliega una nueva página en la cual se presentan las instrucciones para responder la encuesta, el sistema SAAC despliega las preguntas que el usuario debe responder. En la Figura 39 se muestra un ejemplo de cómo se despliegan las preguntas de una encuesta.

SAAC 

Consultar información de cuestionario

DATOS GENERALES

Nombre del cuestionario: ← 1

Descripción:
 ← 2

PREGUNTAS

Título de la pregunta : ← 3

Descripción:

Asignar grupo de respuesta: ← 5

Tabla de preguntas :

Título de la pregunta	Grupo respuesta
Título de una pregunta aquí .	Grupo de respuesta 1
Título de otra pregunta aquí .	Grupo de respuesta 2

Figura 37. Agregar información sobre nuevo cuestionario.

The screenshot shows the SAAC system interface. At the top left is the SAAC logo with an owl icon. A navigation menu contains buttons for 'Inicio', 'Noticias', 'Encuestas', 'Foros', 'Perfiles', and 'Sesión'. Callout 1 points to the 'Encuestas' button. Below the menu is a search bar with the text 'Buscar encuesta: horarios' and an 'Aceptar' button. Callout 2 points to the search bar. On the left, there is a section titled 'Encuestas SOBRE TUS COMPETENCIAS' with a person icon and a paragraph of text. On the right, there are two survey listings: 'Encuesta sobre salarios' and 'Encuesta sobre horarios'. Callout 3 points to the 'Encuesta sobre salarios' title, and callout 4 points to the 'Responder' button below it.

Figura 38. Subsección de encuestas del sistema SAAC.

The screenshot shows a survey question in the SAAC system. At the top left is the SAAC logo with an owl icon. The main heading is 'Cuestionario sobre resistencia al cambio'. Below this, it says 'Pregunta 1 de 26'. The question text is: 'Considere los dos enunciados que siguen. Elija la respuesta que refleje mejor su manera de reaccionar frente al cambio.' Two cases are provided: 'Caso A: en momentos turbulentos de cambios intensos me cuesta saber lo que tengo que hacer y no reacciono hasta al cabo de un cierto tiempo.' and 'Caso B: en momentos turbulentos de cambios intensos me doy cuenta enseguida de las oportunidades y de los peligros que surgen y actúo sin dilación.' Below the cases, it says 'Seleccione una respuesta:'. There are five radio button options: 'Igual o muy parecido a A.', 'Más parecido a A que a B.', 'Intermedio.', 'Más parecido a B que a A.', and 'Igual o muy parecido a B.'. The fourth option, 'Más parecido a B que a A.', is selected. At the bottom, there is an 'Enviar' button.

Figura 39. Responder una encuesta.

Otro tipo de cuestionario que el administrador del sistema agrega y que el ingeniero de software puede responder es la autoevaluación. Para realizar una autoevaluación es necesario que el ingeniero de software ejecute el proceso de inicio de sesión, ingrese a la subsección autoevaluación (1) y presione el botón iniciar (2) para responder la autoevaluación, tal como se muestra en la Figura 40.

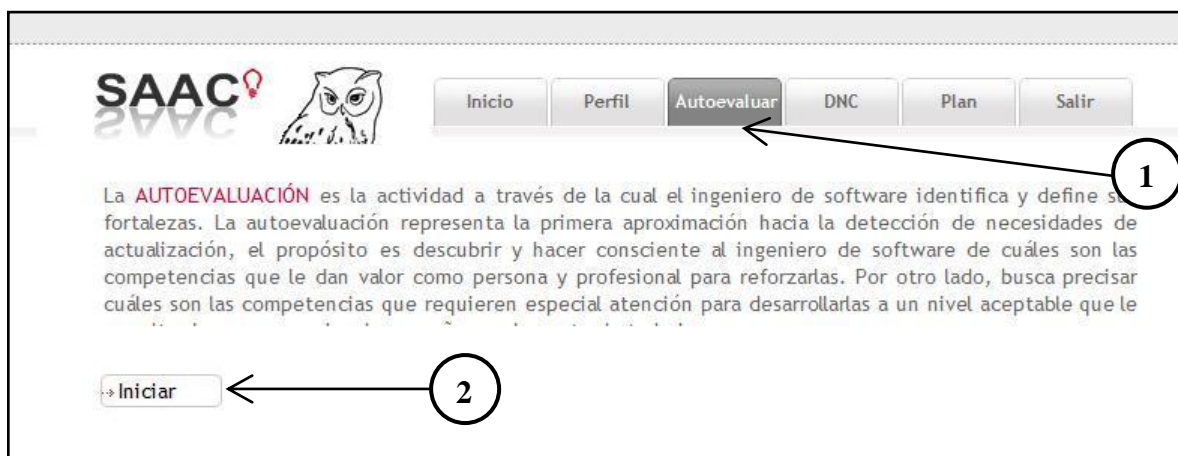


Figura 40. Subsección autoevaluar competencia.

La autoevaluación se divide en dos etapas, en la primera de ellas el ingeniero de software responde un cuestionario, en la Figura 41 se muestra un ejemplo del cuestionario de la primera etapa de una autoevaluación.

 The image shows a questionnaire titled 'Autoevaluación de competencia' in green text. Below the title, it says 'Por favor responde con sinceridad las preguntas siguientes:'. There is a horizontal line for a response. The question is '¿Con qué frecuencia ...?' followed by '1. Actúa para mantener las habilidades y conocimientos profesionales al día'. Below the question are five radio button options: 'nunca', 'casi nunca', 'de modo ocasional', 'a menudo', 'siempre', and 'no lo sé'. The 'a menudo' option is selected.

Figura 41. Responder autoevaluación de competencia.

En la segunda etapa de la autoevaluación de competencia, el ingeniero de software debe indicar, según su percepción, cuáles son las competencias que lo caracterizan, selecciona el tipo de competencia, la categoría, nombre y el nivel (1) que considera tener, además debe indicar cuáles de los criterios para esa competencias cumple (2), como se indica en la Figura 42.

SAAC 

Autoevaluación de competencia

Esta es la segunda parte de la autoevaluación de competencias. Las competencias se analizan o evalúan en función a comportamientos pasados, por ello no es necesario que usted se someta a ninguna prueba, siempre y cuando pueda analizar en forma objetiva sus comportamientos, tanto los exitosos como los no exitosos.
Por favor indique cuáles cree usted que son sus principales competencias:

Competencia 1

Tipo: Categoría: **1**

Competencia: Nivel:

Seleccione los criterios de la competencia que considera que cumple:

Descripción del criterio uno.

Descripción del criterio dos. **2**

Descripción del criterio tres.

Descripción del criterio cuatro.

Figura 42. Responder segunda etapa de autoevaluación.

Al finalizar las dos etapas de la autoevaluación de competencia, el sistema SAAC despliega la información sobre el resultado de la misma. En la Figura 43 se observa la gráfica en la cual se compara el nivel de competencia que el ingeniero de software considera tener, con los niveles de competencias que otros ingenieros de software, jefes y clientes consideran que el ingeniero de software tiene, esta información se obtiene a través de la evaluación

360° (o retroalimentación), que es una de las formas en la cuales el ingeniero de software puede interactuar con otras personas en la organización.

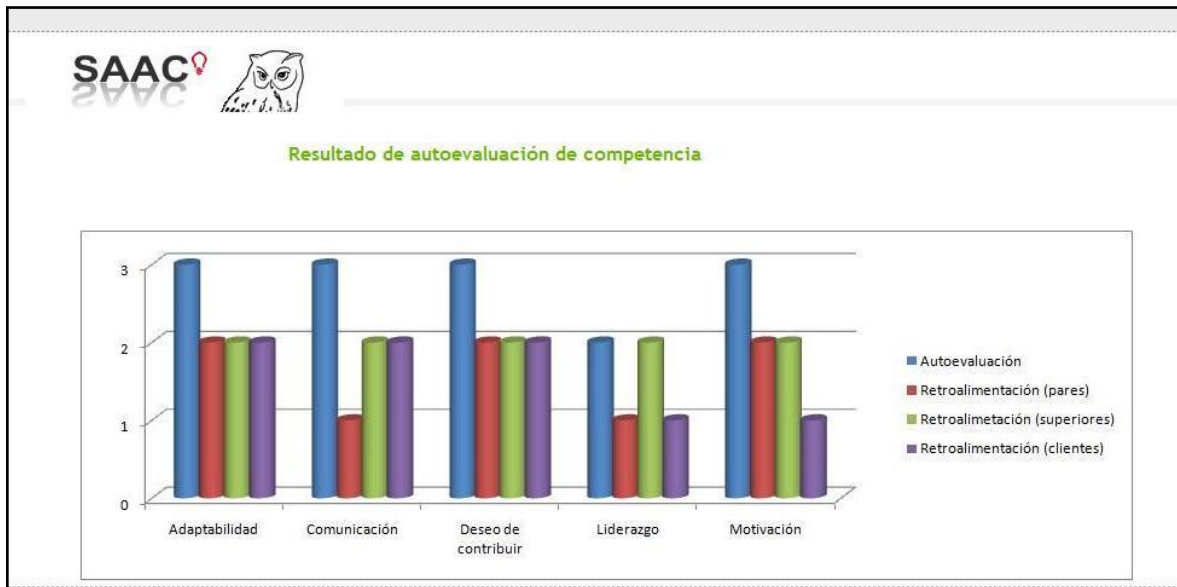


Figura 43. Despliegue del resultado de autoevaluación de competencia.

Otra manera en la cual el administrador del sistema colabora para que el ingeniero de software utilice el sistema SAAC, es a través de la actualización del perfil de competencias del puesto. Para que el administrador del sistema pueda agregar, actualizar o eliminar un perfil de competencias de un puesto, requiere realizar el proceso de inicio de sesión, ingresar a la sección administrativa, a la subsección perfiles (1), se despliega la página que se muestra en la Figura 44, en esta página tiene la opción de agregar un nuevo perfil de puesto (3) o seleccionar uno existente (2) para su consultar, actualización o eliminación.

Cuando el administrador del sistema presiona el botón agregar nuevo puesto, o en su caso la opción de modificar un puesto ya existente, se despliega una página similar a la que aparece en la Figura 45, se agregan los datos generales sobre el puesto (1) y las competencias (2) asociadas a ese puesto, las cuales aparecen en la lista (3) como se muestra en la Figura 46.

SAAC

Inicio Usuarios Competencias Cuestionarios **Perfiles** Salir

En esta sección del sistema SAAC puedes agregar, modificar y eliminar perfiles de puestos de la organización.

Perfiles
DEL SISTEMA SAAC

Para agregar un nuevo perfil de puesto del sistema SAAC presione el siguiente botón.

3 →

Da click en el nombre del puesto para ver más detalles.

Lista de perfiles de puesto:

Administrador de proyectos ← **2**
Analista
Programador

Figura 44. Subsección perfiles de competencias de puestos.

SAAC

DATOS GENERALES

Nombre del puesto:

Descripción:

1 ←

Figura 45. Agregar información sobre el perfil de competencias del puesto (datos generales).

COMPETENCIAS

Tipo: Competencia social ▼

Categoría: Cooperar y trabajar en equipo ▼

Competencia: Grupo respuesta 2 ▼

Nivel: 1 - Básico ▼

→ Guardar

Tabla de competencias :

Competencia	Nivel
Administración de proyectos	1 - Básico
Programador	2 - Aceptable

Figura 46. Agregar información sobre el perfil de competencias del puesto (competencias).

El ingeniero de software utiliza la información sobre el perfil de competencias del puesto para comparar las competencias que tiene él y las competencias que requiere para el puesto que labora actualmente o laborará en el futuro. En la Figura 47 se muestra un ejemplo en el cuál el sistema SAAC despliega de manera gráfica la diferencia que existe entre las competencias que posee el ingeniero de software (1) y las que se sugieren para ejecutar el puesto (2).

Colaboración entre ingenieros de software.

Anteriormente en este apartado se describe que una forma de colaboración a través del sistema SAAC que ocurre entre los ingenieros de software es la evaluación 360° (o retroalimentación). Además, existen otras formas, como el foro de cooperación y publicación de noticias, en las cuales los ingenieros de software pueden participar para colaborar en el desarrollo de competencias.

Para participar en el foro de cooperación, el ingeniero de software debe ingresar a la subsección foros (1) que se encuentra en la sección principal del sistema SAAC, en ella se despliega una lista con el nombre y descripción de los foros de cooperación disponibles (2), como se muestra en la Figura 48.

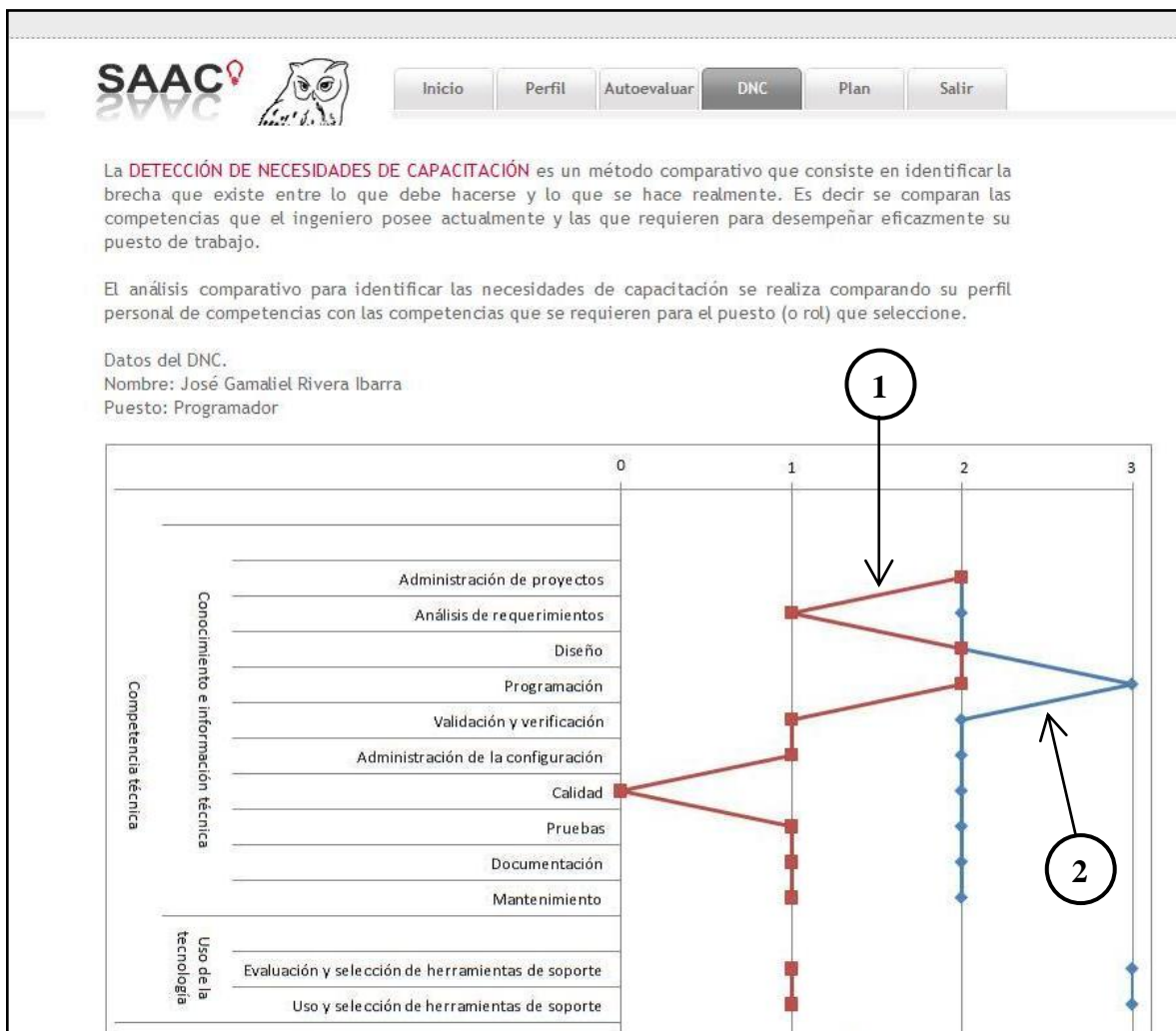


Figura 47. Subsección detección de necesidades de capacitación del sistema SAAC.

Una vez que el ingeniero de software presiona el botón correspondiente al foro (3) (de la Figura 48) en el cual desea participar, se despliega una nueva página en la cual se despliega una lista de los temas que se discuten en el foro seleccionado, como se muestra en la Figura 49. Si el ingeniero de software no realiza el proceso de inicio de sesión con anticipación, el sistema SAAC solicita que se ejecute este proceso para poder ver los mensajes de los temas del foro. En la Figura 49 se puede observar los campos para iniciar sesión (1) y la lista de temas (2) que conforman el foro de cooperación.



Figura 48. Subsección de foros del sistema SAAC.



Figura 49. Información sobre los temas del foro de cooperación seleccionado.

Una vez que el ingeniero de software inicia sesión e ingresa al tema en el cual desea participar se despliega una página similar a la que se muestra en la Figura 50. Si el ingeniero de software desea responder a un mensaje que envía otro usuario (1), debe

presionar el botón responder (2), el sistema SAAC despliega la página que se muestra en la Figura 51, en la cual se despliega el mensaje que se responde (1) y los campos para agregar la información que se requiere para responder dicho mensaje (2), una vez que el usuario haya llenado los campos, presiona el botón agregar (3), que se muestran en la Figura 51, después de presionar el botón agregar, el mensaje se despliega en el tema del foro correspondiente.



Figura 50. Despliegue de los mensajes de un tema de un foro de cooperación.

Colaboración entre el tutor y el ingeniero de software.

La colaboración entre el tutor y el ingeniero de software a través del sistema SAAC se da cuando el ingeniero de software realiza un plan para el desarrollo de competencias y lo envía a revisión, el tutor recibe en su cuenta de usuario en el sistema SAAC la información sobre el plan de desarrollo de competencias. Para que el ingeniero de software agregue un plan requiere realizar el proceso de inicio de sesión, ingresar a la subsección plan (1) y presionar el botón agregar (3), como se muestra en la Figura 52. Además de agregar planes, puede modificar o enviar a revisión planes que haya realizado anteriormente (2).

The image shows a forum interface for SAAC. At the top left is the SAAC logo with a lightbulb and an owl. Below it is a horizontal line, followed by the text "Titulo del tema" in green. Another horizontal line separates the header from the message content. The message content includes: "Fecha: 22 de agosto de 2008, 15:29 hrs.", "Titulo del mensaje : del tema fulanito.", "Autor del mensaje : fulanito.", and "Texto del mensaje aquí.". Below this is another horizontal line. The reply form consists of a "Titulo del mensaje :" label followed by a text input field containing "yo opino que". Below the input field is a larger text area containing "yo opino que una opción es". At the bottom left of the form is a button labeled "→Agregar". Three numbered callouts (1, 2, and 3) with arrows point to the message title, the reply title input field, and the "Agregar" button, respectively.

Figura 51. Responder un mensaje en un tema de un foro de cooperación.

Al presionar el botón agregar nuevo plan de desarrollo de competencias, o elegir la opción modificar plan, se despliega una página similar a la que aparece en la Figura 53. En ella se observan los campos en los cuales el ingeniero de software escribe la información necesaria para elaborar el plan de desarrollo de competencias, como el nombre (1), descripción (2), entre otros, una vez que agrega esta información presiona el botón guardar (3) para almacenar la información.

SAAC 

Inicio Perfil Autoevaluar DNC **Plan** Salir

1

EL **PLAN DE DESARROLLO DE COMPETENCIAS** es un instrumento de gestión que permite al ingeniero de software dirigir sus acciones de desarrollo de competencias. Éste describe de manera detallada un conjunto sistematizado y coherente de acciones de desarrollo de competencias en un determinado período de tiempo. Todas las acciones se deben derivar de la detección de necesidades de actualización.


El proceso de planeación del desarrollo de competencias consiste precisamente en elaborar el plan de desarrollo de competencias, definiendo las actividades, las estrategias, los cursos y otros aspectos que facilitan al ingeniero de software para lograr el desarrollo de las competencias.

Planes de desarrollo de competencias:

Plan para el desarrollo de la competencia comunicación. [Ver](#)
 Plan para el desarrollo de la competencia adaptabilidad. [Revisar](#) ← **2**
 Plan para el desarrollo de la competencia liderazgo. [Modificar](#)

→ Agregar ← **3**

Figura 52. Subsección plan del sistema SAAC.

SAAC 

EL **PLAN DE DESARROLLO DE COMPETENCIAS** es un instrumento de gestión que permite al ingeniero de software dirigir sus acciones de desarrollo de competencias. Éste describe de manera detallada un conjunto sistematizado y coherente de acciones de desarrollo de competencias en un determinado período de tiempo. Todas las acciones se deben derivar de la detección de necesidades de actualización.

Competencia: Liderazgo ↓ ← **1**

Descripción: ← **2**
 Dentro de mi decisión de especializarme en la gestión de proyectos, tengo que desarrollar mis competencias en la dirección de equipos y capacidad de liderazgo, cualidad fundamental que me servirá no solo en mi trayectoria profesional sino también en mi vida personal

→ Guardar ← **3**

Figura 53. Agregar información sobre el plan de desarrollo de competencias.

VI.7 Casos de prueba

Los casos de prueba son la especificación de un conjunto de datos de entrada, condiciones de ejecución y resultados esperados, que se identifican con el propósito de realizar la evaluación de aspectos particulares del sistema [Leffingwell *et al.*, 2003]. Los casos de prueba se utilizan, por un lado, para validar si la implementación del sistema cumple con los requerimientos que se plantean en la etapa de diseño del mismo, y por otro lado, para detectar errores en la implementación antes que éste comience a operar.

En este apartado de la investigación se presentan los casos de pruebas que se utilizan para evaluar a la herramienta de soporte, esto con el propósito de detectar errores en la implementación de sus funciones, en la interfaz de usuario, en la estructura de los datos, entre otros. También, los casos de prueba se ejecutan con el propósito de validar que la funcionalidad cumple los requerimientos propuestos en su diseño.

VI.7.1 Diseño de los casos de prueba

El diseño de los casos de prueba se realiza siguiendo los pasos que proponen Dean Leffingwell y Don Widrig [Leffingwell *et al.*, 2003], señalan que para especificar los casos de pruebas se pueden seguir los pasos siguientes:

- Identificar los escenarios de casos de uso.
- Para cada escenario, identificar uno o más casos de prueba.
- Para cada caso de prueba, identificar las condiciones para ejecutarlo.
- Realizar los casos de prueba agregando datos.

En la Figura 54 se observa la relación que existe entre casos de uso, casos de prueba y el plan de pruebas del sistema.

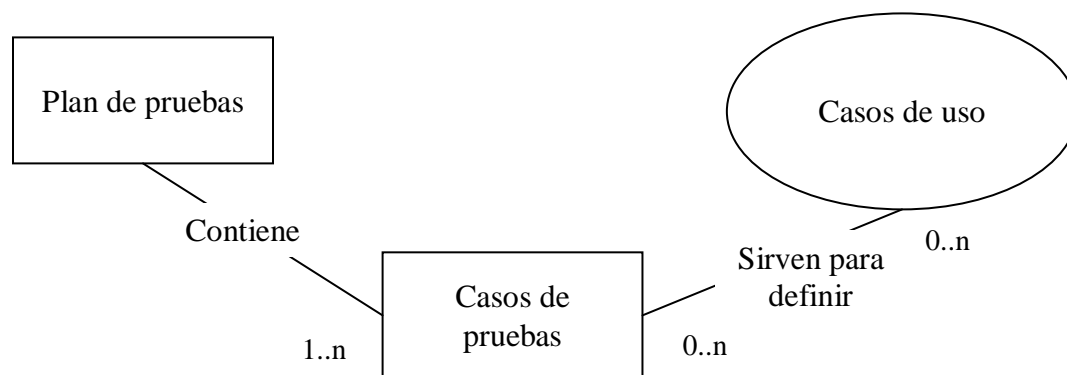


Figura 54. Relación entre casos de uso, casos de prueba y el plan de pruebas (versión original en Leffingwell *et al.*, 2003).

VI.7.1.1 Escenarios de casos de uso

En base a la especificación de los casos de uso del sistema y en las formas de uso del modelo de desarrollo de competencias (que se presentan en el capítulo III), se identifican un conjunto de escenarios en los cuales el usuario puede utilizar el sistema. Un escenario de caso de uso es una instancia de un caso de uso, es decir, es la ejecución de un caso de uso donde un usuario específico ejecuta el caso de uso de una forma determinada.

De cada caso de uso se pueden derivar varios escenarios, cada uno de los posibles caminos que puede seguir el usuario al ejecutar un caso de uso representa un escenario. Resulta complicado abarcar todos los escenarios posibles, a continuación se enlistan los casos de uso y algunos de los escenarios que se derivan de ellos, estos últimos se utilizan para definir los casos de prueba del sistema.

- **Caso de uso administrar usuarios.**
 - El administrador del sistema registra a un nuevo usuario del sistema.
 - El administrador del sistema busca y modifica la información de un usuario del sistema.
 - El administrador del sistema elimina a un usuario del sistema.
- **Caso de uso administrar competencias.**
 - El administrador del sistema registra una nueva competencia.

- El administrador del sistema busca y modifica la información de una competencia.
- El administrador del sistema elimina la información de una competencia.
- **Caso de uso administrar perfiles de puesto.**
 - El administrador del sistema registra la información del perfil de puesto.
 - El administrador del sistema busca y modifica la información de un perfil de puesto.
 - El administrador del sistema elimina la información de un perfil de puesto.
- **Caso de uso administrar cuestionarios.**
 - El administrador del sistema registra un nuevo cuestionario.
 - El administrador del sistema busca y modifica la información de un cuestionario.
 - El administrador del sistema elimina la información de un cuestionario.
 - El administrador del sistema publica un cuestionario.
 - El administrador del sistema quita un cuestionario.
- **Caso de uso administrar perfil personal.**
 - El ingeniero de software modifica su perfil personal.
- **Caso de uso proporcionar retroalimentación.**
 - El ingeniero de software proporciona retroalimentación a otro usuario.
 - El ingeniero de software consulta la retroalimentación recibida.
- **Caso de uso realizar autoevaluación de competencia.**
 - El ingeniero de software realiza la autoevaluación de su competencia.
- **Caso de uso realizar detección de sus necesidades de capacitación.**
 - El ingeniero de software realiza la detección de necesidades de capacitación.
- **Caso de uso administrar planes.**
 - El ingeniero de software realiza un nuevo plan de desarrollo de competencia.
 - El ingeniero de software manda a revisión un plan de desarrollo de competencias.
 - El ingeniero de software modifica la información de un plan de desarrollo de competencias.

- **Caso de uso revisar plan de desarrollo de competencia.**
 - El tutor revisa y solicita modificación de un plan de desarrollo de competencias.
 - El tutor revisa y aprueba un plan de desarrollo de competencias.
- **Caso de uso participar en foro de cooperación.**
 - El ingeniero de software consultar los mensajes de un tema de discusión.
 - El ingeniero de software agrega un mensaje en un tema de discusión.
- **Caso de uso responder encuesta.**
 - El ingeniero de software responde una encuesta.

VI.7.1.2 Especificación de los casos de prueba

Para cada uno de los escenarios de casos de uso se definen un conjunto de casos de pruebas, la definición de cada caso de prueba abarca la descripción, objetivos, condiciones, procedimientos, datos de entrada y resultados esperados en cada prueba. En este apartado se presenta de manera resumida los casos de prueba al sistema SAAC, en el reporte técnico se puede consultar mayor detalle sobre los casos de prueba.

- **Agregar un nuevo usuario:**
 - El sistema no debe permitir agregar datos no válidos.
 - El sistema no debe registrar un nombre de usuario repetido.
 - El sistema agrega el nuevo usuario cuando todos los datos son válidos.
- **Buscar un usuario:**
 - El sistema no debe regresar nombres de usuario cuando se utiliza una cadena de búsqueda no existente.
 - El sistema debe regresar los nombres de usuario que coinciden con la cadena de búsqueda existente.
- **Agregar una nueva competencia:**
 - El sistema no debe permitir agregar datos no válidos.
 - El sistema no debe registrar un nombre de competencia repetido.
 - El sistema agrega la nueva competencia cuando todos los datos son válidos.

- **Buscar una competencia:**
 - El sistema no debe regresar competencias cuando se utiliza una cadena de búsqueda no existente.
 - El sistema debe regresar las competencias que coinciden con la cadena de búsqueda existente.
- **Eliminar una competencia:**
 - El sistema no debe permitir eliminar una competencia cuando ésta está siendo utilizada por un usuario.
- **Agregar nuevo perfil de puesto:**
 - El sistema no debe permitir agregar datos no válidos.
 - El sistema no debe registrar un nombre de perfil de puesto repetido.
 - El sistema agrega el nuevo perfil de puesto cuando todos los datos son válidos.
- **Eliminar un perfil de puesto:**
 - El sistema no debe permitir eliminar un perfil de puesto cuando éste está siendo utilizado por un usuario.
- **Agregar un nuevo cuestionario:**
 - El sistema no debe permitir agregar datos no válidos.
 - El sistema no debe registrar un título de cuestionario repetido.
 - El sistema agrega un nuevo cuestionario cuando todos los datos son válidos.
- **Publicar un cuestionario:**
 - El sistema debe poner disponible el cuestionario para los usuarios indicados en la sección correspondiente.
- **Quitar un cuestionario:**
 - El sistema no debe poner disponible el cuestionario.
- **Modificar perfil personal:**
 - El sistema no debe permitir agregar datos no válidos.
 - El sistema actualiza el perfil personal cuando todos los datos son válidos.
- **Agregar un nuevo plan:**
 - El sistema no debe permitir agregar datos no válidos.

- El sistema agrega un nuevo plan cuando todos los datos son válidos.
- **Solicitar revisión de plan:**
 - El sistema debe enviar la solicitud de revisión al usuario correspondiente.
- **Participar en foro de cooperación:**
 - El sistema no debe permitir ingresar a un tema de discusión si el usuario no ha iniciado sesión.

Para ver mayor detalle sobre los casos de prueba es necesario consultar el plan de pruebas al sistema SAAC, que se encuentra en el reporte técnico. Para cada caso de prueba se determina si es correcto o fallido, para los casos de pruebas fallidos se debe determinar la causa de la falla y realizar las correcciones necesarias.

Diseño del experimento

VII.1 Introducción

En este capítulo se describe el diseño de un experimento para saber si el modelo propuesto ayudará a desarrollar las competencias del ingeniero de software, para esto es necesario realizar los experimentos que generen la información que permita comprobar que lo que se propone es cierto. Hay que tener en cuenta que un experimento consiste en un estudio de investigación en el que se manipulan deliberadamente una o más variables independientes para analizar las consecuencias que la manipulación tiene sobre una o más variables dependientes [Hernández Sampieri et al., 2005].

La aplicación del experimento se conoce como prueba o ensayo. De las pruebas se debe obtener la información necesaria para aceptar o rechazar la o las hipótesis que se definen en el diseño del experimento. Las pruebas pueden ayudar también a identificar otros factores que no están contemplados al inicio del experimento.

En los siguientes apartados se describen los objetivos del experimento, las variables dependientes e independientes, la validez interna del experimento, las hipótesis, las características de la muestra, el diseño del instrumento para la valoración de la intención de uso del modelo propuesto y el análisis o tratamiento de los datos obtenidos del experimento.

VII.2 Objetivos del experimento

En este apartado se definen los objetivos del experimento.

- Diseñar un experimento que permita generar la información suficiente para valorar la intención de uso del modelo de desarrollo de competencias propuesto.
- Diseñar un experimento que permita generar la información suficiente para evaluar el desarrollo de una competencia utilizando el modelo que se propone.
- Diseñar un experimento que permita generar la información suficiente para evaluar la sensibilización al cambio hacia el desarrollo de competencias utilizando el modelo que se propone.

VII.3 Variables

En este experimento la variable independiente es el uso o no del modelo de desarrollo de competencias propuesto y la variable dependiente es el nivel de desarrollo de una competencia, de esta manera se puede explicar el nivel de desarrollo de la competencia (variable dependiente) por el uso o no del modelo propuesto (variable independiente).

A continuación se describe a las variables independientes y dependientes.

VII.3.1 Variables independientes

La variable independiente es el uso del modelo de desarrollo de competencias propuesto (MDC). Esto implica que el ingeniero de software conozca y utilice el sistema SAAC y participe en el proceso de cambio durante el desarrollo de una determinada competencia.

VII.3.2 Variables dependientes

Para este experimento se tomarán en cuenta dos variables dependientes:

- El nivel de desarrollo de la competencia por parte del ingeniero de software (DC).
- El nivel de sensibilización al cambio hacia el desarrollo de competencias del ingeniero de software (SC).

VII.4 Validez interna

Las fuentes de invalidación interna son aquellos factores que atentan contra la validez interna del experimento. La invalidación interna se refiere al grado de confianza que se tienen en que los resultados del experimento sea posible interpretarlos y éstos sean válidos. La validez interna se relaciona con la calidad del experimento y se logra cuando hay control sobre el mismo [Hernández-Sampieri et al., 2001]. A continuación se describe las fuentes de invalidación interna que se consideran en el desarrollo de las pruebas.

Historia

Se refiere a todos aquellos hechos que ocurren durante la ejecución del experimento y que de alguna forma influyen sobre la variable dependiente [Hernández-Sampieri et al., 2001]. Por ejemplo, si durante la ejecución del experimento el facilitador (persona que aplica el cuestionario) deliberadamente persuade a los participantes para que consideren el uso del modelo de desarrollo de competencias. Para controlar esto se toman las medidas siguientes:

- Los participantes no deben conocer ni las hipótesis (en el caso que existan) ni las condiciones experimentales para evitar sesgo en sus respuestas.
- El facilitador no debe conocer la finalidad del experimento, simplemente recibe instrucciones precisa sobre qué decir a los participantes y cómo puede apoyarlos, sin afectar a la variable dependiente.

Maduración

Se refiere aquellos procesos internos de los participantes que como consecuencia del exceso de tiempo que se requiere para realizar el experimento influyen en el resultado del mismo, por ejemplo el cansancio, hambre, aburrimiento y cuestiones similares [Hernández-Sampieri et al., 2001]. Para controlar esto se toman las medidas siguientes:

- El experimento completo no debe sobrepasar el tiempo de una hora para evitar aburrimiento y cansancio.

- De ser necesario proporcionar un refrigerio al participante al inicio para evitar que se distraiga por hambre.

Inestabilidad

Se refiere a las fluctuaciones en las personas seleccionadas o componentes del experimento, o desequilibrio autónomo de mediciones repetidas aparentemente “equivalentes” [Hernández-Sampieri et al., 2001]. Para controlar esto se toman las medidas siguientes:

- La aplicación del instrumento a los participantes, para todos debe ser en el mismo horario.
- Evitar días feriados o coincidir con otros eventos que pueden distraer a los participantes.

Para lograr la validez interna del experimento se debe tener conocimiento y control sobre las diferentes variables que intervienen en el fenómeno, para así poder eliminar todas las explicaciones alternas que se deriven de las condiciones en las que se realiza el experimento. En este experimento se tomarán en cuenta y controlaran las variables que se muestran en la Tabla XXIX y que están clasificadas en tres categorías: organizacionales, personales y otras.

Las variables organizacionales están relacionadas con las características particulares de la organización, la premisa es que estas variables deberían afectar por igual a todos los ingenieros de software en esa misma organización. Las variables personales están relacionadas con las características particulares de la persona, en este caso las variables pueden presentarse con diferentes grados de afectación para cada ingeniero de software, aunque ellos estén en la misma organización. La otra variable que puede afectar es la calidad del curso de capacitación, esto en el caso que los ingenieros de software asistan a diferentes cursos.

Por lo descrito anteriormente, una forma de minimizar o incluso eliminar el posible efecto de las variables organizacionales, es que el experimento se aplique utilizando como objetos

de estudio a ingenieros de software en la misma organización y que asistan al mismo curso de capacitación.

Tabla XXIX. Variables no tomadas en cuenta por el experimento.

Variables organizacionales:
Clima organizacional sobre capacitación.
Políticas.
Apoyo organizacional para la capacitación.
Variables personales:
Compromiso organización-ingeniero de software.
Satisfacción laboral del ingeniero de software.
Seguridad de empleo.
Retribución económica.
Motivación por el desarrollo de competencias del ingeniero de software.
Otras variables:
Calidad de los cursos de capacitación.

Es posible que existan otras variables más, pero resulta complicado tomar en cuenta a cada una de ellas. Otras variables que pueden afectar el experimento y no se tomaran en cuenta son: la participación del ingeniero de software en uno o varios proyectos (las diferentes etapas), los recursos materiales disponibles para el desarrollo de competencias, la disponibilidad de tiempo, las costumbres, la edad y el estado civil del ingeniero de software.

VII.5 Hipótesis

La información que se obtenga de las pruebas que se realizarán, servirá para comprobar las hipótesis que se plantean a continuación.

Hipótesis nula

H₁₀: *El uso del modelo propuesto no incrementará el nivel de desarrollo de la competencia.*

$$\mathbf{H1_0} = DC_{MDC} = DC \quad (1)$$

Donde,

DC_{MDC} se refiere al nivel de desarrollo de la competencia utilizando el modelo propuesto.

DC se refiere al nivel de desarrollo de la competencia sin utilizar el modelo propuesto.

Hipótesis de investigación

H1: *El uso del modelo de desarrollo de competencia incrementará el nivel de desarrollo de la competencia.*

$$\mathbf{H1} = DC_{MDC} > DC \quad (2)$$

Donde,

DC_{MDC} se refiere al nivel de desarrollo de la competencia utilizando el modelo propuesto.

DC se refiere al nivel de desarrollo de la competencia sin utilizar el modelo propuesto.

Hipótesis nula

H2₀: *El uso del modelo propuesto no incrementará el nivel de la sensibilización al cambio hacia el desarrollo de competencias.*

$$\mathbf{H2_0} = SC_{MDC} = SC \quad (3)$$

Donde,

SC_{MDC} se refiere al nivel de sensibilización al cambio hacia el desarrollo de competencias utilizando el modelo propuesto.

SC se refiere al nivel de sensibilización al cambio hacia el desarrollo de competencias sin utilizar el modelo propuesto.

Hipótesis de investigación

H2: *El uso del modelo de desarrollo de competencias incrementará el nivel de la sensibilización al cambio hacia el desarrollo de competencias.*

$$\mathbf{H2} = SC_{MDC} > SC \quad (4)$$

Donde,

SC_{MDC} se refiere al nivel de sensibilización al cambio hacia el desarrollo de competencias utilizando el modelo propuesto.

SC se refiere al nivel de sensibilización al cambio hacia el desarrollo de competencias sin utilizar el modelo propuesto.

De acuerdo a los resultados arrojados por el proceso de validación se puede establecer el grado de verdad de las hipótesis.

VII.6 Características del experimento

VII.6.1 Unidades experimentales

Para realizar las pruebas es necesario seleccionar dos grupos: el experimental y el de control. Al grupo experimental se le aplicará un tratamiento que consiste en conocer y utilizar el modelo de desarrollo de competencias, mientras que el grupo de control no se aplicará nada. Ambos grupos se deben someter al mismo fenómeno, en este caso el desarrollo de una competencia (también se puede conocer como proceso de capacitación, entrenamiento o desarrollo profesional).

Para que una persona sea candidata a sujeto de estudio para este experimento debe cumplir lo siguiente:

- Ser ingeniero de software de profesión y estar laborando en una organización desarrolladora de software realizando alguno de los roles de la ingeniería de software.
- Tener contrato por tiempo indefinido y más de un año laborando en la misma organización. Esta condición se establece porque se considera que si el ingeniero de software está contratado por tiempo definido este no encontrará atractivo un programa que le ayude en su capacitación y desarrollo profesional.

VII.6.1.1 Delimitación de la población

Se indica cuál va ser la población estudiada y sobre la cual se pretende generalizar los resultados. La muestra se define como un subgrupo de la población que se delimita.

La población del presente experimento abarca a todos aquellos ingenieros de software que laboran en el análisis, diseño, implementación y mantenimiento de sistemas basados en software.

VII.6.2 Intención de uso

La intención de uso del modelo de desarrollo de competencias consiste en un conjunto de constructores que tienen la finalidad de medir el grado en el cual el modelo de desarrollo de competencias será o no utilizado por los ingenieros de software. Para medir la intención de uso se realiza la adaptación del modelo para medir la intención de uso diseñado por Medha Umarji y Carolyn Seaman [Umarji et al., 2005] el cual lo utilizan para predecir el grado de aceptación que tendrá la mejora del proceso de software en una determinada organización.

El modelo para medir la intención de uso que presentan Umarji y Seaman [Umarji et al., 2005] está diseñado en base al Modelo de Aceptación de Tecnología (Technology Acceptance Model, TAM) y la Teoría del Comportamiento Planificado (Theory of Planned Behavior, TPB), además de algunos constructores adicionales que proponen las propias autoras y que toman en cuenta la cultura organizacional, el impacto de los cambios provocados por la mejora del proceso de software y las características particulares de los ingenieros de software.

Umarji y Seaman [Umarji et al., 2005] señalan que al combinar los constructores de diferentes modelos el resultado de la predictibilidad es mejor en comparación a utilizar a cada uno de los modelos por separado.

A continuación se describen algunas de las características que se consideran similares entre la mejora del proceso de software (MPS) y el modelo de desarrollo de competencias, esta comparación se realiza para justificar la adaptación del modelo que proponen Umarji y Seaman para medir el grado de aceptación del modelo de desarrollo de competencias.

- a) *Objetivos similares*: la MPS tiene como objetivo implementar procesos que permitan producir software de alta calidad, reducir costos y tiempo e incrementar la productividad [Hyde et al., 2004]. Por otro lado, aunque en principio el modelo de desarrollo de competencias tiene como objetivo principal precisamente desarrollar la competencia del ingeniero de software, al desarrollar ésta lo que se desea es incrementar sus capacidades lo que debe conducir a mejorar la calidad del producto de software e incrementar la productividad.
- b) *Uso de recursos*: para alcanzar los objetivos de la MPS las organizaciones utilizan técnicas como inspección de código, uso de herramientas CASE, métodos de desarrollo (método ágil) y otras iniciativas. Mientras que al utilizar el modelo de desarrollo de competencias se exige al ingeniero de software aprender el proceso de desarrollo de competencias, utilizar la herramienta de soporte y otras actividades extras que requieren de tiempo y esfuerzo.
- c) *Uno puede considerarse parte del otro*: se puede considerar al modelo de desarrollo de competencias como una iniciativa que busca la MPS como el estudio que realizan Eugene McGuire y Kim Randall sugiere [McGuire et al., 1998]. Ellos efectúan un estudio para determinar, según la percepción de los ingenieros de software, qué competencias se requiere desarrollar para lograr la MPS.
- d) *Ambos provocan cambios*: en general la implementación de una estrategia de MPS exige el aprendizaje de nueva tecnología, cambios en las prácticas de trabajo y carga adicional de trabajo. Entre las actividades que exigen las propuestas de MPS se requiere proporcionar cierta información sobre las actividades laborales para medir el progreso del proyecto, los recursos utilizados, entre otras cosas, sin embargo esto a veces es interpretado por el ingeniero de software como una manera para medir su rendimiento, por lo cual no siempre se muestran entusiastas por participar en este tipo de iniciativas [Umarji et al., 2005]. De manera similar, el modelo de desarrollo de competencias requiere de agregar nuevos hábitos de trabajo como la autoevaluación de competencias y el diseño de un plan de capacitación, los cuales también demandan que el ingeniero de software proporcione cierta información sobre sus competencias.

- e) *Beneficios intangibles*: tanto para las iniciativas de MPS como para el desarrollo de competencias, los resultados al utilizarlos no siempre son visibles y el retorno de la inversión puede ser largo.

Las similitudes que se describen arriba permiten justificar la adaptación del modelo que proponen Umarji y Seaman [Umarji et al., 2005] para predecir el grado de aceptación que tendrá por parte del ingeniero de software el modelo de desarrollo de competencias. Por esto se diseña un modelo de intención de uso el cual utiliza constructores que se obtienen de diferentes modelos para la adopción de tecnologías.

VII.6.2.1 Constructores

En este apartado se describen los constructores (o variables independientes) que permiten predecir el grado (o nivel) de intención de uso del modelo de desarrollo de competencias. En este modelo se utilizan algunos de los constructores que utilizan Umarji y Seaman [Umarji et al., 2005] y se agrega un constructor (impacto en la trayectoria profesional) que no está considerado en su modelo.

A continuación se define a cada uno de los ocho constructores que se utilizan para predecir la intención de uso del modelo de desarrollo de competencias.

Utilidad

Se refiere a la percepción que tiene el ingeniero de software sobre el grado en el cual el modelo de desarrollo de competencias le proporcionará lo necesario para mejorar sus prácticas actuales de trabajo.

Facilidad de uso

Se refiere a la percepción que tiene el ingeniero de software sobre el grado en el cual cree que el modelo de desarrollo de competencias será fácil de utilizar.

Norma subjetiva

Se refiere al grado de influencia sobre el ingeniero de software que tiene la opinión de los demás hacia el modelo de desarrollo de competencias.

Impacto en la trayectoria profesional

Se refiere a la percepción que tiene el ingeniero de software sobre el grado en el cual el modelo de desarrollo de competencias tiene un impacto a largo plazo en su trayectoria.

Miedo a las consecuencias adversas

Se refiere a la percepción que tiene el ingeniero de software sobre el buen uso que se hace de la información que proporciona al utilizar el modelo de desarrollo de competencias.

Autoeficacia

Se refiere a la percepción que tiene el ingeniero de software sobre si será capaz de ejecutar las actividades correspondientes al modelo de desarrollo de competencias.

Grado de control

Se refiere a la percepción que tiene el ingeniero de software sobre el grado de disponibilidad por parte de la organización para recibir sugerencias de cambios y así poder involucrarse en el uso del modelo de desarrollo de competencias.

Cantidad de conocimientos requeridos

Se refiere a la percepción que tiene el ingeniero de software sobre la cantidad de conocimientos y de entrenamiento que requiere para utilizar el modelo de desarrollo de competencias

La variable dependiente o el parámetro a medir es la intención de uso del modelo de desarrollo de competencias que se propone. La intención de uso se refiere al grado en el cual el ingeniero de software tiene la intención de utilizar el modelo de desarrollo de competencias.

VII.6.3 Instrumentos de medición

VII.6.3.1 Diseño del instrumento para valorar la intención de uso

Se utilizará la técnica de aplicación de cuestionario para obtener la información que se requiere para realizar un análisis cuantitativo que permita valorar la intención de uso del ingeniero de software. Los constructores que se describen anteriormente sirven para diseñar un instrumento (cuestionario) que ayuda a predecir la probabilidad que existe de que el ingeniero de software utilice el modelo de desarrollo de competencias. El instrumento que se diseña está formado por 20 afirmaciones, cada una corresponde a uno de los constructores como se indica en la Tabla XXX.

Cada una de las afirmaciones se valora utilizando la escala de tipo Likert de 5 puntos, en la cual el 1 representa una respuesta negativa y el 5 una respuesta positiva, como se indica a continuación:

- 1) Totalmente en desacuerdo.
- 2) En desacuerdo.
- 3) Neutro.
- 4) De acuerdo.
- 5) Totalmente de acuerdo.

En el anexo B se puede observar el instrumento que se diseña para medir la intención de uso del modelo de desarrollo de competencias por parte del ingeniero de software.

Por otro lado, para medir las variables dependientes se requiere de un conjunto de instrumento de valoración:

- Instrumento para valorar el nivel de desarrollo de la competencia.
- Instrumento para valorar el nivel de sensibilización al cambio hacia el desarrollo de competencias.

Tabla XXX. Descripción de las afirmaciones del instrumento agrupadas por constructor.

Utilidad
Creo que desarrollar mis competencias me ayudará a incrementar la calidad con la que hago mi trabajo.
Creo que el desarrollo de mis competencias contribuirá a que pueda realizar más rápido mi trabajo.
Creo que el modelo de desarrollo de competencias me permitirá darme cuenta si necesito capacitarme y en qué.
Creo que el modelo de desarrollo de competencias me permitirá darme cuenta de cuáles son mis fortalezas y debilidades como ingeniero de software.
Creo que el modelo de desarrollo de competencias me permitirá tener un mejor control sobre mi desarrollo profesional.
Facilidad de uso
Creo que me resultaría complicado aprender a utilizar el modelo de desarrollo de competencias.
Creo que el modelo de desarrollo de competencias me podría demandar una gran cantidad de tiempo para aprender a utilizarlo.
Creo que el modelo de desarrollo de competencias es bastante complejo de utilizar.
Norma subjetiva
Creo que al darme cuenta que mis compañeros utilizan el modelo de desarrollo de competencias podría animarme a utilizarlo.
Creo que si no utilizo el modelo de desarrollo de competencias mi jefe y colegas criticaran el rendimiento que tengo en mi trabajo.
Creo que desarrollar mis competencias hará que mis colegas consideren que soy un ingeniero de software competente.
Impacto en la trayectoria profesional
Creo que desarrollar mis competencias me permitirá aspirar a un mejor puesto de trabajo.
Creo que desarrollar mis competencias me permitirá tener mayor participación en las diferentes actividades del desarrollo de software.
Miedo a las consecuencias adversas
Creo que el modelo de desarrollo de competencias permitirá a mi jefe darse cuenta que no soy competente para realizar mi trabajo.
Creo que utilizar el modelo de desarrollo de competencias permitirá a los demás darse cuenta que tengo un bajo rendimiento.
Autoeficacia
Creo que podría llegar a ser un experto en el uso del modelo de desarrollo de competencias.
Grado de control
Creo que el modelo de desarrollo de competencias es para beneficio de la organización y yo sólo proporcionaré la información que necesitan.
Creo que la organización no me permitirá hacer sugerencias sobre la forma de uso del modelo de desarrollo de competencias.
Cantidad de conocimientos requeridos
Creo que necesitaré estudiar muy bien el modelo de desarrollo de competencias para poder utilizarlo.
Creo que no se requiere conocer mucho sobre el modelo de desarrollo de competencias para poder utilizarlo.

VII.6.4 Procedimiento experimental

Las pruebas que permiten obtener la información del experimento deben tener una duración suficiente para que el ingeniero de software participe primero en el proceso de sensibilización al cambio hacia el desarrollo de competencias y segundo en el curso de capacitación de la competencia. Se deben obtener mediciones de la competencia a desarrollar, al inicio y final de curso de capacitación. De manera similar, se debe valorar el nivel de sensibilización al cambio hacia el desarrollo de competencias antes y después del proceso de sensibilización.

VII.6.5 Plan de ejecución de la prueba para valorar la intención de uso

Se refiere al procedimiento que se debe seguir para aplicar el instrumento de valoración a los participantes. Los pasos que se sugiere seguir se describen a continuación.

- 1) *Descripción del experimento*: el facilitador explicará a los participantes cómo será su participación en el experimento y el rol que juega él en la ejecución del mismo.
- 2) *Entrega de materiales*: el facilitador entrega a cada uno de los participantes los materiales que necesitar para llevar a cabo el experimento.
- 3) *Descripción del instrumento*: el facilitador describe a los participantes las partes que forman al instrumento de valoración.
- 4) *Descripción de condiciones*: el facilitador explica a los participantes las condiciones bajo las cuales se ejecutará el experimento.
- 5) *Descripción del modelo de desarrollo de competencias*: el facilitador describe a los participantes las principales características del modelo de desarrollo de competencias, los participantes pueden seguir al facilitador leyendo y observando las figuras presentes en la primera parte del instrumento.
- 6) *Sección de dudas*: el facilitador responde las dudas de los participantes antes de comenzar a responder el instrumento de valoración.
- 7) *Aplicación del instrumento*: el facilitador da la señal a los participantes para que comiencen a responder el instrumento de valoración.

- 8) *Finalización del experimento*: el facilitador agradece la cooperación de los participantes.

VII.7 Análisis de los datos

Una vez que se aplican los instrumentos y se codifican los datos, se obtendrán dos mediciones de cada variable dependiente por ingeniero de software. Una medición corresponde a la aplicación del instrumento al inicio del experimento y la otra al final del experimento. La diferencia entre el nivel inicial y el nivel final representará el nivel de desarrollo de la competencia y de la sensibilización al cambio, respectivamente. Con los datos obtenidos se aplica la prueba t para determinar si existe una diferencia significativa entre las medias de los dos grupos (experimental y de control) y en base al resultado realizar la comprobación de las hipótesis.

Discusión, aportaciones y trabajo futuro

VIII.1 Discusión

En este trabajo se realizó el análisis y diseño de un modelo de desarrollo de competencias para el ingeniero de software. Este modelo es una guía que señala diferentes aspectos que se deben tomar en cuenta al momento de desarrollar las competencias del ingeniero de software. El modelo que se propone además intenta cubrir algunas de las carencias y problemas que se señalan en la literatura sobre algunos modelos de mejora como CMMI, P-CMM y PSP, algunos ejemplos de estos problemas son la falta de modelos prescriptivos, el desarrollo de habilidades blandas y métodos para manejar el cambio que producen las iniciativas de mejora continua.

La motivación por realizar este trabajo surge en primer lugar de la inquietud del autor por explorar el factor humano en la ingeniería de software y en segundo lugar de la necesidad por desarrollar investigación sobre el desarrollo de competencias en el ingeniero de software que permitan aumentar su productividad en el desarrollo de software. Se puede constatar a través de la revisión literaria que existe relativamente poca investigación que se enfoque al desarrollo de las competencias del ingeniero de software, el proyecto más importante en este aspecto es P-CMM, que está respaldado por el SEI.

A continuación se examinan las preguntas de investigación que se plantean al inicio del trabajo y la manera en la cual a través del desarrollo de la investigación se responde a ellas.

¿Cómo puede saber el ingeniero de software si necesita actualizarse profesionalmente?

En esta investigación se analizó la literatura sobre métodos de capacitación y desarrollo de competencias, para conocer y seleccionar un proceso de desarrollo de competencias. Este proceso está conformado por un conjunto de actividades que se enfocan en determinar si el ingeniero de software requiere actualizarse, qué competencias actualizar y qué acciones llevar a cabo para lograr la actualización de dichas competencias. Para facilitar la ejecución de este proceso, sus características se implementan en la herramienta de soporte (el sistema SAAC), de esta manera el ingeniero de software voluntariamente puede ingresar al sistema SAAC y realizar la autoevaluación de su competencia para responder la pregunta ¿necesito actualizarme?, y la detección de las necesidades de capacitación para saber qué competencias actualizar.

De lo anterior surge la inquietud por saber si realmente el ingeniero de software voluntariamente utilizará el sistema SAAC para obtener soporte en el desarrollo de sus competencias. Para esto, el proceso de manejo del cambio (que forma parte de la estrategia) juega un papel importante, ya que este proceso considera actividades que intenta incidir en la percepción del ingeniero de software sobre la importancia de su desarrollo profesional. Para que el ingeniero de software tenga la motivación y la voluntad de actualizar sus competencias, utilizar el sistema SAAC y colaborar con la organización en actividades que tengan como objetivo su crecimiento profesional, es necesario que éste conozca y valore la importancia de mantenerse actualizado (entre otras cosas), esto es uno de los objetivos del proceso de cambio.

¿Qué competencias debe desarrollar el ingeniero de software para lograr el desempeño sobresaliente de las actividades de la ingeniería de software?

En esta investigación se realizó el análisis y diseño de un marco de competencias para ingenieros de software, en él se especifican qué competencias y a qué nivel se sugiere que el ingeniero de software desarrolle de acuerdo a su rol en el proceso de desarrollo de software. Estas competencias se seleccionan a través del análisis de la literatura y de diversos estudios empíricos realizados por la Dra. Josefina Rodríguez.

El marco de competencias que se propone puede ser utilizado por las organizaciones como referencia para determinar qué competencias quieren que desarrollen sus ingenieros de software. De ser necesario se pueden agregar o quitar competencias de acuerdo a los objetivos estratégicos y definición de los perfiles de puestos de cada organización.

En la estrategia que forma parte del modelo se considera un proceso que indica qué actividades son necesarias para desarrollar o adaptar el marco de competencias que se propone. Este proceso se definió en base a estudios disponibles en la literatura sobre técnicas para el desarrollo de marcos de competencias.

¿Qué actividades se deben realizar para desarrollar las competencias del ingeniero de software?

El proceso de desarrollo de competencias que forma parte de la estrategia indica los pasos necesarios para desarrollar las competencias del ingeniero de software. El proceso se basa en los métodos para el desarrollo de competencias que se abstraen de la literatura. Para facilitar la ejecución de este proceso, sus características se implementan en la herramienta de soporte (el sistema SAAC). El proceso de cambio considera actividades de entrenamiento para enseñarle al ingeniero de software cómo ejecutar el proceso de desarrollo de competencias y especialmente utilizando el sistema SAAC como apoyo.

¿Qué características debe tener una herramienta que proporcione soporte al ingeniero de software durante el desarrollo de sus competencias?

Uno de los resultados de este trabajo de investigación es un reporte técnico [Rivera-Ibarra *et al.*, 2008] en el cual se incluye el análisis y diseño de la herramienta de soporte, en él se presenta el análisis de la especificación de los requerimientos que debe cumplir una herramienta que permita proporcionar soporte al ingeniero de software para el desarrollo de sus competencias.

¿Qué factores (o situaciones) pueden obstruir o dificultar el desarrollo de competencias del ingeniero de software?

El modelo que se propone describe algunos de los factores que se considera (desde el particular punto de vista de los autores) pueden obstruir el desarrollo de competencias del ingeniero de software. El modelo indica únicamente algunos de los múltiples factores que pueden existir. Los factores que se muestran en el modelo surgen del análisis de los estudios realizados por diversos investigadores expertos en el área de comportamiento organizacional a falta de estudios al respecto en el área de ingeniería de software.

El modelo considera un conjunto de actividades, que se especifican en el proceso de cambio, las cuales tienen como objetivo identificar la presencia de estos factores y planear las actividades necesarias que permitan disminuir el efecto negativo que pudieran causar al momento de realizar el desarrollo de las competencias del ingeniero de software. Estas actividades pueden variar dependiendo de las características de cada factor que esté presente y las características de las personas que forman la organización, por esto es importante contar con un experto capaz de planear las acciones que permitan mitigar la obstrucción de las actividades de desarrollo de competencias.

De acuerdo a lo que se describe arriba, se puede constatar que en el trabajo de investigación se propone una posible respuesta para cada una de las preguntas de investigación que se plantearon al inicio del desarrollo de la misma.

Por otro lado, el modelo que se propone cubre ciertas características o requerimientos que se encontró son un problema o una carencia en otros modelos similares (como CMMI, P-CMM y PSP). Estos problemas se plantean al inicio de la investigación y son los que se enlistan a continuación:

- Mejora basada en las necesidades.
- Desarrollo de habilidades blandas.
- Manejo del cambio.
- Modelos prescriptivos.

Para lograr la mejora basada en las necesidades del individuo, el modelo que se propone sugiere que las actividades de autoevaluación y la detección de necesidades de capacitación se realicen de manera individual, para que así sea posible conocer qué competencias requiere desarrollar cada ingeniero de software y trabajar específicamente en esas competencias y no otras que posiblemente no le benefician para realizar sus actividades laborales. En algunos casos la detección de necesidades de capacitación se realiza a nivel de grupos de personas o incluso del análisis de las tareas, por ejemplo la detección de necesidades de capacitación para el personal del departamento de informática, sin embargo cada individuo puede tener necesidades diferentes.

El desarrollo de habilidades blandas está considerado en el modelo que se propone porque el marco de competencias (que forma parte del modelo) especifica que se deben desarrollar competencias de tres tipos: técnicas, sociales y personales, de las cuales las competencias que están clasificadas como del tipo social y personal son conocidas también como habilidades blandas.

El manejo de cambio se implementa a través de diversas actividades que se especifican en el proceso de cambio que forma parte de la estrategia del modelo que se propone. El manejo de cambio se basa principalmente en el trabajo realizado por la Dra. Josefina Rodríguez [Rodríguez-Jacobo, 2003] quien desde 1994 a la fecha trabaja en la formación de grupos de desarrollo de software. En su trabajo propone una estrategia para manejar el cambio en las organizaciones desarrolladoras de software.

El modelo que se propone además de servir como referencias (modelo descriptivo) también propone una estrategia que especifica un conjunto de procesos los cuales tienen la finalidad de indicar cómo aplicar o utilizar el modelo en una organización, por esta razón se considera que el modelo se puede clasificar como prescriptivo. Sin embargo, es necesario realizar experimentos para conocer si las personas que utilizarían el modelo que se propone lo consideran lo suficientemente prescriptivo.

VIII.2 Aportaciones

Como resultado de esta investigación se obtienen las siguientes aportaciones:

- 1) La ingeniería del software es un área multidisciplinaria que intenta tomar las mejores prácticas de otras áreas del conocimiento para producir software de calidad. El SWEBOK es una guía creada para representar cada una de las áreas del conocimiento que forman la taxonomía de la ingeniería de software. Los aspectos relacionados con la organización y la persona caen dentro de las áreas de investigación del SWEBOK en la cual la investigación aún es intermitente [Umaji *et al.*, 2005b]. Uno de los principales aportes de este trabajo de investigación es aplicar el conocimiento de otras áreas como la psicología, comportamiento organizacional y recursos humanos con la intención de incrementar las capacidades de los ingenieros de software y por consiguiente la calidad del software.
- 2) Otro aporte significativo del trabajo es la investigación sobre el estado del arte acerca de la generación de capacidades del ingeniero de software.
- 3) El producto principal de este trabajo de investigación es el análisis y diseño de un modelo de desarrollo de competencias para ingenieros de software, éste es otra de las aportaciones generadas durante el desarrollo de la tesis.
- 4) En este trabajo de investigación se realiza el análisis y diseño de un marco de competencias para ingenieros de software. El marco de competencias propone un conjunto de competencias claves que al desarrollar el ingeniero de software podrían ayudar a que éste aumente su rendimiento en el trabajo. Este marco de competencia fue aceptado y publicado como artículo de investigación en el Software Engineering Symposium 2008, dentro del marco del Encuentro Nacional de Computación 2008 realizado en Mexicali Baja California, del 6 al 7 de octubre.
- 5) Como parte del marco de competencias, se propone un conjunto de habilidades blandas (competencias sociales y personales), éstas generalmente no se toman en cuenta en los procesos de formación y capacitación del ingeniero de software, pero son importantes para su buen desempeño laboral.

- 6) En este trabajo de investigación se realiza el análisis y diseño de una herramienta que proporciona soporte al ingeniero de software para el desarrollo de sus competencias. La mayoría de los sistemas que utilizan competencias, están orientados a la gestión de recursos humanos y no al desarrollo de competencias.
- 7) Además se realiza el análisis de los posibles factores que pueden obstruir el desarrollo de competencias del ingeniero de software. A falta de estudios en el área de ingeniería de software relacionado con este tema, se recurre a otras áreas de conocimiento para analizar los factores identificados.

VIII.3 Trabajo futuro

Es necesario desarrollar más investigación para realizar un modelo de desarrollo de competencias para ingenieros de software más robusto. El resultado que se alcanzó en este trabajo de investigación establece una base firme y un camino a seguir para realizar las pruebas necesarias que proporcionen retroalimentación sobre la idea que se propone y se realicen los cambios que se consideren necesarios para cumplir con el objetivo del modelo.

A continuación se describen algunos de los principales temas que requieren un mayor número de estudios y experimentos.

- Implementar el sistema SAAC, además sería interesante agregar la funcionalidad para apoyar en las actividades de la gestión de recursos humanos.
- Analizar qué áreas de proceso del P-CMM se pueden implementar con la ayuda del modelo de desarrollo de competencias que se propone.
- Realizar estudios de campo para evaluar si las competencias que se proponen en el marco de competencias son las que realmente requiere el ingeniero de software.
- Realizar estudios de campo para evaluar la veracidad y confiabilidad del modelo propuesto.
- Elaborar diversos instrumentos para valorar las competencias que se proponen en el marco de competencias.
- Desarrollar un modelo matemático que en base a la evaluación de las competencias permita determinar el nivel o grado de competencia.

Referencias

- Abran, A., Moore, J., Bourque, P., Dupuis, R., y Tripp, L. 2005. *Guide to the Software Engineering Body of Knowledge - 2004 Version – SWEBOK*, IEEE-Computer Society Press, 200 p.
- Acuña, S.T. y Ferré X. 2001. Software Process Modelling. *Multi-Conference on Systemics, Cybernetics and Informatics, ISAS-SCI (1)*, Julio 22-25, Orlando, Florida, Estados Unidos, pp. 237-242.
- Acuña, S. T., Juristo, N., Moreno, A. M., and Mon, A. 2005. *A Software Process Model Handbook for Incorporating People's Capabilities*. Springer-Verlag New York, Inc. 324 p.
- Aguado, D., y Arranz, V. 2005. Desarrollo de competencias mediante blended learning: un análisis descriptivo. *Artículo publicado en Píxel-Bit, Revista de Medios y Educación*, (26): 79-88.
- Alex, L. 1991. Descripción y registro de cualificaciones: el concepto de la cualificación. *Revista Formación Profesional*, (2): 23-27.
- Alles, M. 2005. *Desarrollo del talento humano: Basado en competencias*. Publicado por Ediciones Granica S.A., 356 p.
- Allison, I. and Merali, Y. 2007. Software process improvement as emergent change: A structurational analysis. *Information and Software Technology*, 49 (6): 668-681.
- Aragón Sánchez, A., Tenorio Ronda, J., Pérez Rodríguez, M., Sabater Sánchez, R., Sánchez Marín, G., Sánchez Quiros, I. y Sanz Valle, R. 2004. *Fundamentos de dirección y gestión de recursos humanos*. Cengage Learning Editores, 1^{ra} Edición, 312 p.
- Armstrong, M. 2003. *A Handbook of Human Resource Management Practice*. Kogan Page, 9th Edition, 979 p.
- Baldwin, T. T., Magjuka, R. J. y Loher, B. T. 1991. The perils of participation: Effects of choice on trainee motivation and learning. *Personnel Psychology*, 44: 51-65. (Reprinted in International Library of Management, Spring, 1994).

- Baumgartel, H., y Jeanpierre, F. 1972. Applying new knowledge in the Back-Home Setting: a study of Indian Managers' adoptive efforts. *Journal of Applied Behavioral Science*, 8 (6): 674-696.
- Berger, L. A., y Berger, D. R. 2003. *The Talent Management Handbook: Creating Organizational Excellence by Identifying, Developing, and Promoting Your Best People*. McGraw-Hill Professional, 1st Edition, 450 p.
- Biasca, R. E. 2005. *Gestión de cambio: El modelo Biasca*. Organizational Improvement and Chage. Outskirts Press, Inc., 700 p.
- Blanton, J. E., Schambach, T., y Trimmer, K. J. 1998. Factors affecting professional competence of information technology professionals. *SIGCPR Comput. Pers.* 19 (3): 4-19.
- Bohlander, G., Sherman, A. y Snell S. 2001. *Administración de recursos humanos*. Editorial Thomson International, 706 p.
- Bunk, G. P. 1994. La transmisión de las competencias en la formación y perfeccionamiento profesionales de la RFA. *Revista Europea de Formación Profesional*, 1: 8-14.
- Castillo Aponte, J. 2006. *Administración de personal: un enfoque hacia la calidad*. ECOE Ediciones, 378 p.
- CMMI. 2006. *CMMI for Development, Version 1.2*. Software Engineering Institute, CMU/SEI-2006-TR-008. Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 573 p.
- Cotton, P. 2004. Desarrollando un clima organizacional óptimo. *Sydney: II Simposio hacia lugares de trabajo más seguros en Australia*.
- Cruz Elvira, M., y Prieto Ramiro, R. 1999. *¿Cómo desarrollar la empleabilidad?*. Cáritas Española, 109 p.
- Curtis, B., Hefley, W. y Miller, S. 2001. *People Capability Maturity Model (P-CMM), Version 2.0*. Software Engineering Institute, CMU/SEI-2001-MM-001. Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 735 p.
- Damien J. y Soon A. 2001. The threat-rigidity model of professional obsolescence and its impact on occupational mobility behaviors of IT professionals. *Twenty-Second International Conferences on Information Systems*, Diciembre, New Orleans, Estados Unidos, pp. 567-573.

- Davis, F. 1989. Perceived Usefulness, Perceived Ease Of Use, And User Acceptance Of Information Technology. *MIS Quarterly*, 13 (3): 319-340.
- Davis, K. y Newstrom, J. 2007. “Naturaleza del cambio organizacional”. Nov. 2007; Fuente: LosRecursosHumanos.com <http://www.losrecursoshumanos.com/contenidos/1849--naturaleza-del-cambio-organizacional.html>, (último acceso: octubre de 2008).
- Dawes, S. S. y Helbig, N. 2007. Building government IT workforce capacity: a competency framework. In *Proceedings of the 8th Annual international Conference on Digital Government Research: Bridging Disciplines & Domains* (Philadelphia, Pennsylvania, Mayo 20 – 23, pp. 254-255.
- DeMarco, T. y Lister, T. 1999. *Peopleware (2nd Ed.): Productive Projects and Teams*. Dorset House Publishing Co., Inc. 261 p.
- Disney, A. M. y Johnson, P. M. 1998. Investigating data quality problems in the PSP. *SIGSOFT Softw. Eng. Notes* 23 (6): 143-152.
- Downey, J. y Ali Babar, M. 2008. On identifying the skills needed for software architects. In *Proceedings of the First international Workshop on Leadership and Management in Software Architecture*, Mayo 11, Leipzig, Alemania, pp. 1-6.
- Dubois, D. 1993. *Competency-based Performance Improvement: A Strategy for Organizational Change*. Human Resource Development Press, p. 372.
- Echeverría, S. B. 2002. Gestión de la competencia de acción profesional. *Revista de Investigación Educativa*, 20 (1): 7-43.
- Fernández Losa, N. 2002. El desarrollo profesional de los trabajadores como ventaja competitiva de las empresas. *Cuadernos de Gestión*, 2 (1): 65-92.
- Fernández Sánchez, N., y Gamboa Méndez, M. 2001. Detección de necesidades de capacitación y educación continua: un apoyo para el diagnóstico de necesidades de capacitación. <http://www.tuobra.unam.mx/publicadas/021123233415.html>, (último acceso: octubre de 2008).
- Fossum J., Arvey R., Paradise C. y Robbins, N. 1986. Modeling the skills obsolescence process: a psychological/economic integration. *Academy of Management Review*, 11 (2): 362-374.
- Frías Fernández, P. 2001. *Desafíos de modernización de las relaciones laborales: hacia una nueva cultura y concertación empresarial*. Lom Ediciones, 392 p.

- Glass, R. 2002. *Facts and Fallacies of Software Engineering*. Addison-Wesley, 195 p.
- Goldstein, I. L. 1991. Training in the work organizations. In M. D. Dunnette & L. M. Hough (Eds), *Handbook of industrial and organizational psychology*, 2nd ed, 2: 507-619.
- Green Gina C., Collins Rosann W. y Hevner Alan R. 2005. The impacts of quality and productivity perceptions on the use of software process. *Information and Software Technology*, 47 (8): 543-553.
- Hanne, T. y Neu H. 2004. Simulating human resources in software development processes. *In proceeding of the European Simulation and Modelling Conference (ESM) 2003, Napoli*. pp. 83-87.
- Hansen, E. 2006. *Orientación profesional: Un manual de recursos para países de bajos y medianos ingresos*, Montevideo: CINTERFOR/OIT. EMP/SKILLS, 137 p.
- Hernández Sampieri, R., Fernández Collado, C. y Baptista Lucio, P. 2001. *Metodologías de la investigación*. Editorial McGrawHill, 2da. edición. 501 p.
- Humphrey, W. 1989. *Managing the Software Process*. Addison-Wesley, 494 p.
- Humphrey, W. 1995. *A Discipline for Software Engineering*. Addison-Wesley, 816 p.
- Humphrey, W., 1999. *Pathways to Process Maturity: The. Personal Software Process and Team Software Process*. SEI. Interactive, 2 (2): 1-17.
- Hyde, K y Wilson, D. 2004. Intangible benefits of CMM-based software process improvement. *Software Process: Improvement and Practice*. 9 (4): 217-228.
- ISFOL. 1995. Competenze trasversali e comportamento organizzativo: le abilità di base per il lavoro che cambia.
- Kandeel, H., y Wahba, K. 2001. Competency Models for Human Resource Development: Case of Egyptian Software Industry. *IRMA 2001 International Conference*, Toronto, Ontario, Canada, IDEA Group Publishing, pp. 117-121.
- Khamisani, V.A., Siddiqui, M.S. y Bawany, M.Y. 2006. Analyzing Soft Skills of Software Engineers using Repertory Grid. *In IEEE Conference Proceedings of the Multitopic Conference, 2006. INMIC '06. IEEE*, pp. 259-264.

- Klappholz, D., Bernstein, L., y Port, D. 2003. Assessing Attitude Towards, Knowledge of, and Ability to Apply, Software Development Process. In *Proceedings of the 16th Conference on Software Engineering Education and Training*, Marzo 20 - 22. CSEET. IEEE Computer Society, Washington, DC, 268 p.
- Klein, J., Spector, M., y Grabowski, B. 2004. *Instructor Competencies: Standards for Face-To-Face, Online, and Blended Settings*, Ileana de la Teja, 164 p.
- Klein, J. D., y Richey, R. C. 2005. Improving individual and organizational performance: The case for international standards. *Performance improvement*, 44 (10): 9-14.
- Le Boterf, G. 1991. *Ingeniería y evaluación de los planes de formación*. Ediciones Deusto.
- Leffingwell, D., y Widrig, D. 2003. *Managing Software Requirements: a use case approach*. Addison-Wesley, 502 p.
- Marelli A., Tondora, J., y Hoge, M. 2005. Strategies for developing competency models. *Administration and Policy in Mental Health*, 32 (5): 533-561.
- Mathiassen, L., Ngwenyama, O. K., y Aaen, I. 2005. Managing Change in Software Process Improvement. *IEEE Softw.* 22 (6): 84-91.
- Mathieu, J. E., Tannenbaum, S. I. y Salas, E. 1990. A causal model of individual and situational influences on training effectiveness measures. *Paper presented at the 5th annual conference of the Society for Industrial and Organizational Psychology*. Miami 46 (1): 125-147.
- McCall, M. 1997. *High flyers: developing the next generation of leaders*. Harvard Business Press, 254 p.
- McGuire, E. G. y Randall, K. A. 1998. Process improvement competencies for IS professionals: a survey of perceived needs. In *Proceedings of the 1998 ACM SIGCPR Conference on Computer Personnel Research*. Marzo 26-28, Boston, Massachusetts, Estados Unidos. R. Agarwal, Ed. SIGCPR '98. ACM, New York, NY, pp. 1-8.
- McLagan, P. A. 1997. Competencies: the next generation. *Training & Development*, pp. 40-47.
- Méndez Álvarez, C. E. 2006. *Clima organizacional en Colombia: El IMCOC: un método de análisis para su intervención*. Universidad del Rosario, 143 p.

- Mertens, L. 1996. *Competencia Laboral: sistemas, surgimiento y modelos*. Montevideo, Cinterfor/OIT, 119 p.
- Michaels, E., Handfield, H., y Axelrod, Beth. 2004. La guerra por el talento: principios para atraer, desarrollar y retener a Gerentes altamente talentosos. *Editorial Norma*, 232 p.
- Mochi Alemán, Prudencio O. 2006. *La industria del software en México en el contexto internacional y latinoamericano*. Publicado por UNAM, 261 p.
- Moitra, D. 1998. Managing Change for Software Process Improvement Initiatives: A Practical Experience-Research Section based Approach. *Software Process Improvement and Practices 4*, pp. 199-207.
- Morneau, K. A. y Talley, S. 2007. Architecture: an emerging core competence for IT professionals. In *Proceedings of the 8th ACM SIGITE Conference on information Technology Education*. SIGITE '07. ACM, New York, NY, pp. 9-1.
- Mouthaan, T. J., Olthuis, W., y Vos, H. 2003. Competence-Based EE-Learning: (How) Can We Implement It?. In *Proceedings of the 2003 international Conference on Microelectronics Systems Education*. MSE. IEEE Computer Society, Washington, DC, 33 p.
- Muchinsky, P. M. 2002. *Psicología aplicada al trabajo*. Editorial Thomson International, 570 p.
- Niazi M., Wilson, D. y Zowghi, D. 2006. Critical Success Factors for Software Process Improvement: An Empirical Study. *Software Process Improvement and Practice Journal*, 11 (2): 193-211.
- Pérez Roquela, M. 2002. “Crear un clima de creación”. Tecla: sitio de los periodistas cubanos para el debate y reflexión sobre temas teórico-profesionales, http://www.latecla.cu/bd/dentro/clima_pperez.htm, (último acceso: octubre de 2008).
- Perry, D. E., Staudenmayer, N., y Votta, L. G. 1994. People, Organizations, and Process Improvement. *IEEE Softw.* 11 (4): 36-45.
- Qin, S. 2007. Managing process change in software organizations: Experience and reflection: Research Sections. *Software Process*, 12 (5): 429-435.
- Quiñones, M.A. 1995. Pre-training context effects: Training assignment as feedback. *Journal of Applied Psychology*, 80, pp. 226-238.

- Rainer, A., y Hall, T. 2002. Key success factors for implementing software process improvement: a maturity-based analysis. *Journal of Systems and Software* 62(2): 71-84.
- Rebeil Corella, M. y Ruiz Sandoval, R. 2006. *El poder de la comunicación en las organizaciones*. Plaza y Valdés Editores/Universidad Iberoamericana.
- Reig, E., Fernández, J., y Jauli, I. 2003. *Los recursos humanos en las organizaciones*. Editorial Paraninfo, 250 p.
- Rodríguez Jacobo, J. 2003. “Una estrategia de cambio basada en las personas en una empresa de desarrollo de software”. Tesis doctoral, Instituto de Terapia Gestalt Región Occidente.
- Rodríguez Jacobo, J. 2004. *Formación de grupos de desarrollo de software: un enfoque psicosocial*. Editorial Yoltéolt.
- Rothwell, W. J. 2005. *Effective Succession Planning: Ensuring Leadership Continuity and Building Talent from Within*. New York: AMACOM, American Management Association, 2005, 3rd Edition, 400 p.
- Rychen, D. 2002. Key competencies for the knowledge society. *Education-Lifelong Learning and the Knowledge Economy Conference*, Octubre 10-11, Stuttgart, Alemania, 13 p.
- Sawyer, S., Coopriider, J., y Guinan, P. 2008. Social Interactions of Information Systems Development Teams: A Performance Perspective, *Information Systems Journal*.
- Sharp, H. y Robinson, H. 2005. Some social factors of software engineering: the maverick, community and technical practices. In *Proceedings of the 2005 Workshop on Human and Social Factors of Software Engineering*. HSSE '05. ACM, New York, NY, pp. 1-6.
- Sinnott, G., Madison, G., y Pataki, G. 2002. Competencies: report of the competencies workgrup. <http://www.cs.state.ny.us/successionplanning/workgroups/competencies/competencies.html> (último acceso: octubre de 2008).
- Spencer, L. M., y Spencer, S. M. 1993. *Competence at work: models for superior performance*. John Wiley & Sons, Inc, 384 p.
- Stelzer, D., y Mellis, W. 1998. Success factors of organizational change in software process improvement. *Software Process Improvement and Practice*, 4 (4): 227-250.

- Tannenbaum, S. I., y Yukl, G. 1992. Training and development in work organizations. *Annual Review of Psychology*, 43: 399-441.
- Tsai, H. P., Compeau, D., y Haggerty, N. 2004. A cognitive view of how IT professionals update their technical skills. In *Proceedings of the 2004 SIGMIS Conference on Computer Personnel Research: Careers, Culture, and Ethics in A Networked Environment*. Abril 22-24, Tucson, Arizona, Estados Unidos. SIGMIS CPR '04. ACM, New York, NY, pp. 70-73.
- Tucker, A. B. 2004. *Computer Science: Handbook*. CRC Press.
- Turley, R. T. y Bieman, J. M. 1995. Competencies of exceptional and nonexceptional software engineers. *J. Syst. Softw.* 28 (1): 19-38.
- Umarji, M. y Seaman, C. 2005. Predicting acceptance of Software Process Improvement. *SIGSOFT Softw. Eng. Notes* 30 (4): 1-6.
- Umarji M. y Seaman, C. 2005. Conceptualizing Software Engineering People issues. *Workshop on Ontology, Conceptualizations and Epistemology for Software and Systems Engineering*, Junio 9-10, Madrid, España.
- Vakalahti, Pasi. 1997. Process Improvement Frameworks - a Small Case Study with People Capability Maturity Model, *Software Process Improvement and Practice*, 3: 225-234.
- Valdez, B., Svolou, A., y Valdez, F. 2008. A Holistic Approach to Process Improvement Using the People CMM and the CMMI-DEV: Technology Process, People, & Culture, The Holistic Quadripartite. *Software Engineering Information Repository*. SEI and CMU, 2008; <https://seir.sei.cmu.edu/seir/> (último acceso: octubre de 2008).
- van Loo J., Grip A., y Steur M. 2001. Skills obsolescence: causes and cures. *International Journal of Manpower*, 22(1): 121-138.
- Vakaslathi P. 1997. Process improvement frameworks - a small case study with People Capability Maturity Model, *Software Process: Improvement and Practice*, 3 (4): 225-234.
- Wayne M. R. 2005. *Administración de recursos humanos: una visión panorámica*. Pearson Educación, 314 p.

Anexo A

Formato para el plan de desarrollo de competencias

José Gamaliel Rivera Ibarra

Nombre del Ingeniero de Software

Plan para el desarrollo de la competencia

Liderazgo

Justificación

Dentro de mi decisión de especializarme en la administración de proyectos de software, tengo que desarrollar mis competencias en la dirección de equipos y capacidad de liderazgo. Esta competencia es fundamental para el éxito tanto a nivel profesional como a nivel personal.

Objetivo

- Conocer, desarrollar y utilizar la competencia liderazgo en mi actividad laboral.

Acciones

- Investigar y estudiar las características que debe tener un buen líder.
- Observar y aprender de las personas que tiene la competencia de liderazgo.
- Asistir al curso de liderazgo.
- Practicar el uso de la competencia en mi trabajo con la ayuda de mi tutor.
- Realizar una simulación de un caso en la cual debo aplicar la competencia liderazgo.

Evidencia

- Recibir retroalimentación positiva sobre el desarrollo de la competencia liderazgo.
- Haber mejorado mi conocimiento y habilidad para coordinar y dirigir equipos de trabajo.
- Ser capaz de explicar las características de la competencia liderazgo.
- Ser capaz de afrontar situaciones en las cuales se exige la competencia liderazgo.

Seguimiento

- El día 18 de agosto realizaré una evaluación para demostrar que conozco y sé cómo aplicar la competencia liderazgo en mi trabajo.
- El día 23 de agosto asistiré a la plática sobre el liderazgo en las empresas.
- El día 1 de septiembre realizaré una sesión de entrenamiento con mi tutor.
- El día 28 de septiembre presentaré un simulacro en el cual aplicaré la competencia liderazgo.

Instrumento de valoración de la intención de uso

Instrumento de valoración sobre la intención de uso del modelo de desarrollo de competencias para ingenieros de software

Estimado colega ingeniero de software,

De antemano gracias por tu participación en esta encuesta sobre el uso del modelo de desarrollo de competencias para ingenieros de software. En los siguientes párrafos se describe brevemente qué es el modelo de desarrollo de competencias, se explica para qué sirve y cómo puede ayudarte en tu desarrollo profesional.

Se te pide que sigas atentamente las instrucciones proporcionadas por el facilitador. Una vez que el facilitador lo indique, puede pasar a la siguiente página y responder el cuestionario.

El cuestionario está formado por 20 declaraciones y cada una de ellas tiene 5 opciones de respuesta, que son:

[1] Totalmente en desacuerdo.

[2] En desacuerdo.

[3] Neutro.

[4] De acuerdo.

[5] Totalmente de acuerdo.

Por favor marca solamente una de las opciones en cada una de las declaraciones, elige aquella opción que concuerde más con tu percepción sobre la declaración correspondiente.

1. Creo que desarrollar mis competencias me ayudará a incrementar la calidad con la que hago mi trabajo.

[1]	[2]	[3]	[4]	[5]
Totalmente en desacuerdo	En desacuerdo	Neutro	De acuerdo	Totalmente de acuerdo

2. Creo que el desarrollo de mis competencias contribuirá a que pueda realizar más rápido mi trabajo.

[1]	[2]	[3]	[4]	[5]
Totalmente en desacuerdo	En desacuerdo	Neutro	De acuerdo	Totalmente de acuerdo

3. Creo que el modelo de desarrollo de competencias me permitirá darme cuenta si necesito capacitarme y en qué.

[1]	[2]	[3]	[4]	[5]
Totalmente en desacuerdo	En desacuerdo	Neutro	De acuerdo	Totalmente de acuerdo

4. Creo que el modelo de desarrollo de competencias me permitirá darme cuenta de cuáles son mis fortalezas y debilidades como ingeniero de software.

[1]	[2]	[3]	[4]	[5]
Totalmente en desacuerdo	En desacuerdo	Neutro	De acuerdo	Totalmente de acuerdo

5. Creo que el modelo de desarrollo de competencias me permitirá tener un mejor control sobre mi desarrollo profesional.

[1]	[2]	[3]	[4]	[5]
Totalmente en desacuerdo	En desacuerdo	Neutro	De acuerdo	Totalmente de acuerdo

6. Creo que me resultaría complicado aprender a utilizar el modelo de desarrollo de competencias.

[1]	[2]	[3]	[4]	[5]
Totalmente en desacuerdo	En desacuerdo	Neutro	De acuerdo	Totalmente de acuerdo

7. Creo que el modelo de desarrollo de competencias me podría demandar una gran cantidad de tiempo para aprender a utilizarlo.

[1]	[2]	[3]	[4]	[5]
Totalmente en desacuerdo	En desacuerdo	Neutro	De acuerdo	Totalmente de acuerdo

8. Creo que el modelo de desarrollo de competencias es bastante complejo de utilizar.

[1]	[2]	[3]	[4]	[5]
Totalmente en desacuerdo	En desacuerdo	Neutro	De acuerdo	Totalmente de acuerdo

9. Creo que al darme cuenta que mis compañeros utilizan el modelo de desarrollo de competencias podría animarme a utilizarlo.

[1]	[2]	[3]	[4]	[5]
Totalmente en desacuerdo	En desacuerdo	Neutro	De acuerdo	Totalmente de acuerdo

10. Creo que si no utilizo el modelo de desarrollo de competencias mi jefe y colegas criticaran el rendimiento que tengo en mi trabajo.

[1]	[2]	[3]	[4]	[5]
Totalmente en desacuerdo	En desacuerdo	Neutro	De acuerdo	Totalmente de acuerdo

11. Creo que desarrollar mis competencias hará que mis colegas consideren que soy un ingeniero de software competente.
- | | | | | |
|--------------------------|---------------|--------|------------|-----------------------|
| [1] | [2] | [3] | [4] | [5] |
| Totalmente en desacuerdo | En desacuerdo | Neutro | De acuerdo | Totalmente de acuerdo |
12. Creo que desarrollar mis competencias me permitirá aspirar a un mejor puesto de trabajo.
- | | | | | |
|--------------------------|---------------|--------|------------|-----------------------|
| [1] | [2] | [3] | [4] | [5] |
| Totalmente en desacuerdo | En desacuerdo | Neutro | De acuerdo | Totalmente de acuerdo |
13. Creo que desarrollar mis competencias me permitirá tener mayor participación en las diferentes actividades del desarrollo de software.
- | | | | | |
|--------------------------|---------------|--------|------------|-----------------------|
| [1] | [2] | [3] | [4] | [5] |
| Totalmente en desacuerdo | En desacuerdo | Neutro | De acuerdo | Totalmente de acuerdo |
14. Creo que el modelo de desarrollo de competencias permitirá a mi jefe darse cuenta que no soy competente para realizar mi trabajo.
- | | | | | |
|--------------------------|---------------|--------|------------|-----------------------|
| [1] | [2] | [3] | [4] | [5] |
| Totalmente en desacuerdo | En desacuerdo | Neutro | De acuerdo | Totalmente de acuerdo |
15. Creo que utilizar el modelo de desarrollo de competencias permitirá a los demás darse cuenta que tengo un bajo rendimiento.
- | | | | | |
|--------------------------|---------------|--------|------------|-----------------------|
| [1] | [2] | [3] | [4] | [5] |
| Totalmente en desacuerdo | En desacuerdo | Neutro | De acuerdo | Totalmente de acuerdo |
16. Creo que podría llegar a ser un experto en el uso del modelo de desarrollo de competencias.
- | | | | | |
|--------------------------|---------------|--------|------------|-----------------------|
| [1] | [2] | [3] | [4] | [5] |
| Totalmente en desacuerdo | En desacuerdo | Neutro | De acuerdo | Totalmente de acuerdo |
17. Creo que el modelo de desarrollo de competencias es para beneficio de la organización y yo sólo proporcionaré la información que necesitan.
- | | | | | |
|--------------------------|---------------|--------|------------|-----------------------|
| [1] | [2] | [3] | [4] | [5] |
| Totalmente en desacuerdo | En desacuerdo | Neutro | De acuerdo | Totalmente de acuerdo |
18. Creo que la organización no me permitirá hacer sugerencias sobre la forma de uso del modelo de desarrollo de competencias.
- | | | | | |
|--------------------------|---------------|--------|------------|-----------------------|
| [1] | [2] | [3] | [4] | [5] |
| Totalmente en desacuerdo | En desacuerdo | Neutro | De acuerdo | Totalmente de acuerdo |
19. Creo que necesitaré estudiar muy bien el modelo de desarrollo de competencias para poder utilizarlo.
- | | | | | |
|--------------------------|---------------|--------|------------|-----------------------|
| [1] | [2] | [3] | [4] | [5] |
| Totalmente en desacuerdo | En desacuerdo | Neutro | De acuerdo | Totalmente de acuerdo |
20. Creo que no se requiere conocer mucho sobre el modelo de desarrollo de competencias para poder utilizarlo.
- | | | | | |
|--------------------------|---------------|--------|------------|-----------------------|
| [1] | [2] | [3] | [4] | [5] |
| Totalmente en desacuerdo | En desacuerdo | Neutro | De acuerdo | Totalmente de acuerdo |

Diagrama entidad relación de la base de datos

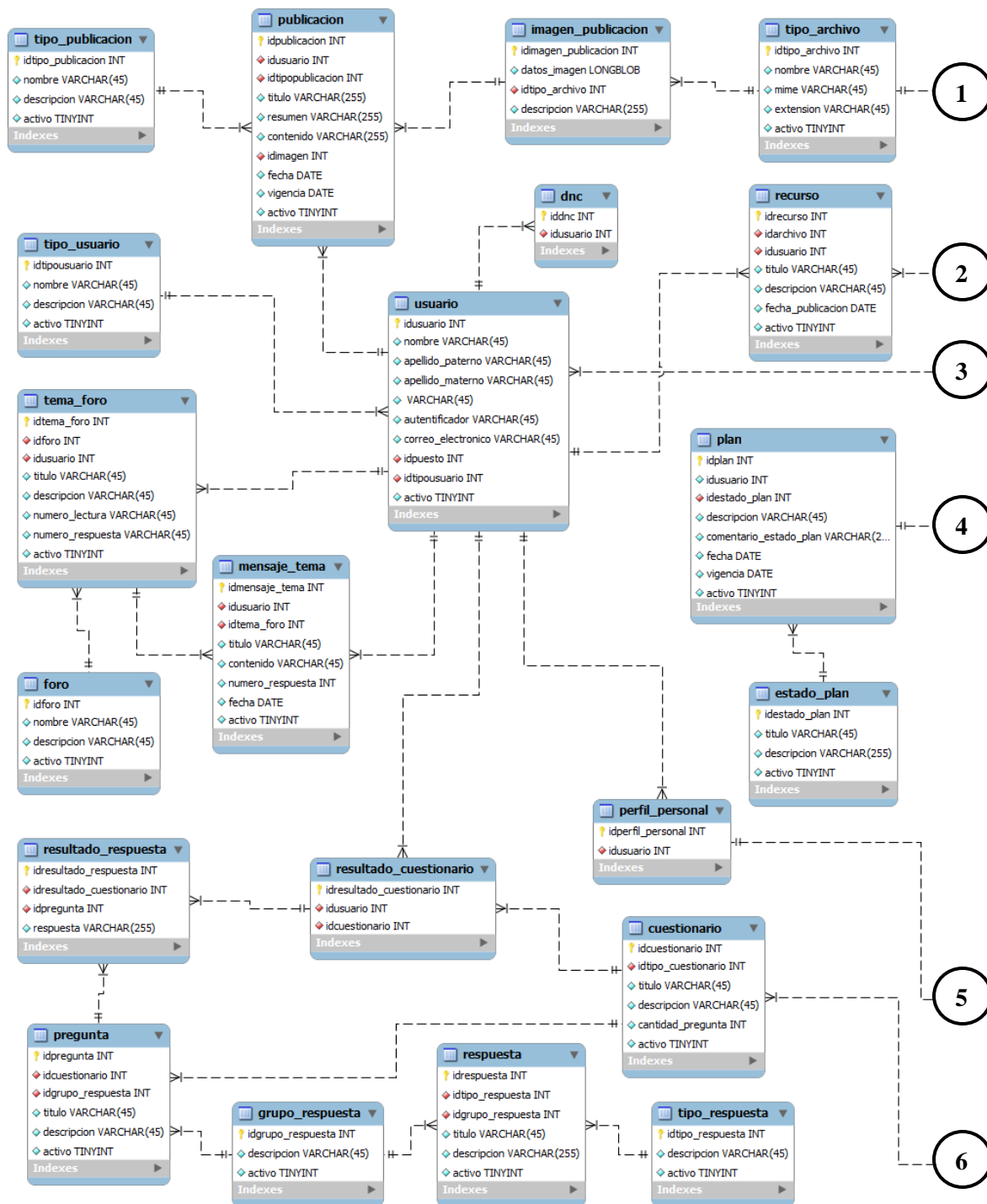


Diagrama 1. Diagrama Entidad/Relación de la base de datos del sistema SAAC (parte 1 de 2).

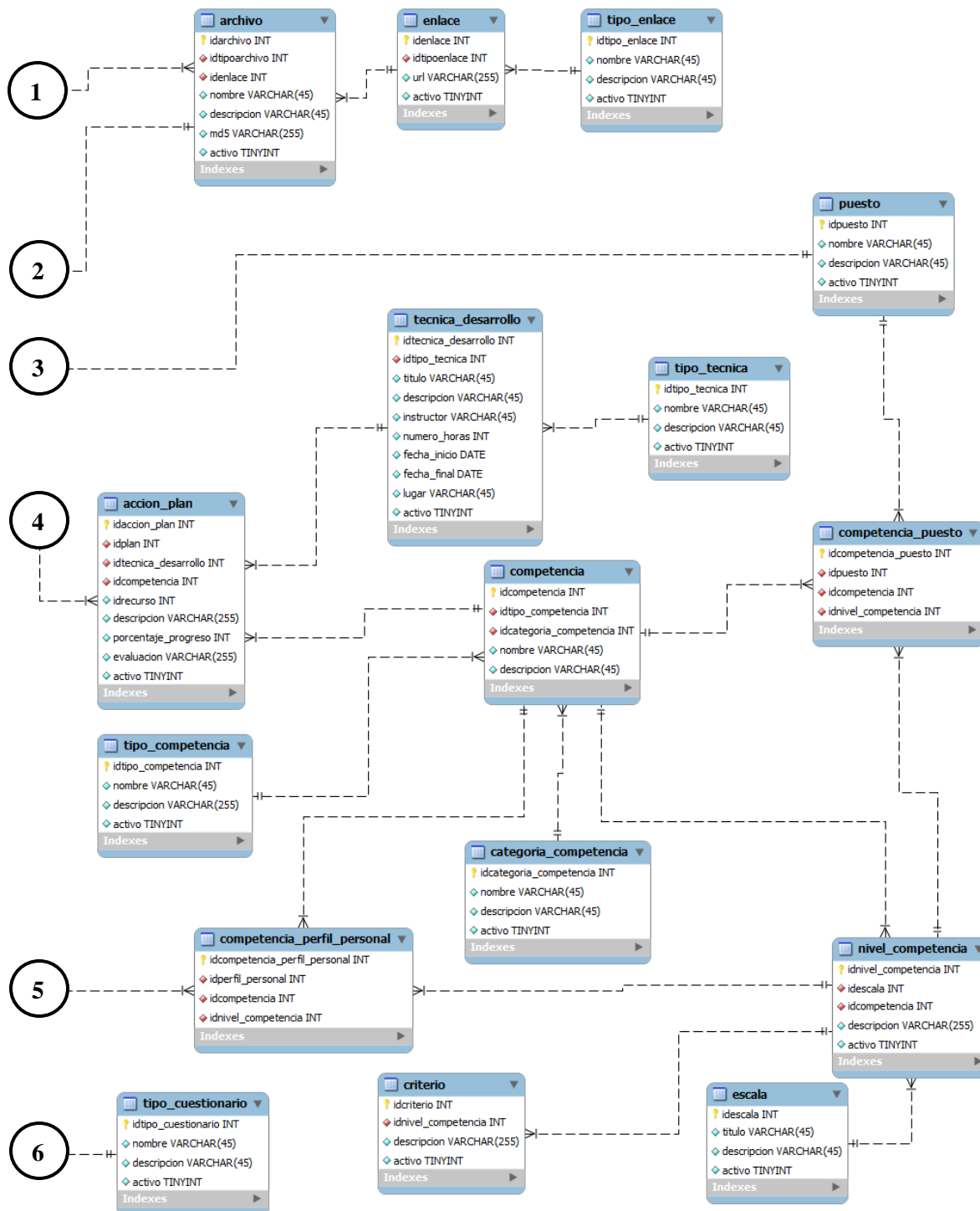


Diagrama 2. Diagrama Entidad/Relación de la base de datos del sistema SAAC (parte 2 de 2).

