TESIS TESIS TESIS TESIS TESIS

CENTRO DE CIENCIAS BÁSICAS

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

TESIS

MODELO MOPROSOFT, UNA ADAPTACIÓN DE LA CATEGORÍA OPERACIÓN, PROCESO DESARROLLO Y MANTENIMIENTO DE SOFTWARE PARA LAS SALIDAS "ESPECIFICACIÓN DE REQUERIMIENTOS" Y "ANÁLISIS Y DISEÑO" AL PROCESO DE DESARROLLO DE SOFTWARE EMBEBIDO, CASO DE ESTUDIO: NI CRIO-9074 DE NATIONAL INSTRUMENT PARA PROYECTO FINAL VISION SYSTEM

PRESENTA

Vianney Sotelo Mauries

PARA OBTENER EL GRADO DE MAESTRÍA EN CIENCIAS CON OPCIÓN A LA COMPUTACIÓN

TUTOR

Dr. Francisco J. Álvarez Rodríguez

COMITÉ TUTORAL

Dr. José Manuel Mora Tavarez
Dr. Guillermo Ramírez Prado
Dr. Ángel Eduardo Muñoz Zavala
M. C. Pablo Espinosa Lepe (Colaborador)

Aguascalientes, Ags., 12 de diciembre de 2013.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

Asunto: Voto probatorio.

A QUIEN CORRESPONDA:

PRESENTE

Por medio de la presente hago de su conocimiento que la tesista:

Vianney Sotelo Mauries

La impresión de su documento de tesis con titulo: **Modelo MoProSoft, una Adaptación de la Categoría Operación, Proceso Desarrollo y Mantenimiento de Software para las Salidas "Especificación de Requerimientos" y "Análisis y Diseño" al Proceso de Desarrollo de Software Embebido, Caso de Estudio: NI cRIO-9074 de National Instrument para Proyecto Final Vision System,** ya que cumple con los requisitos de contenido y forma exigidas en la Universidad Autónoma de Aguascalientes.

Sin más quedó a sus órdenes para cualquier aclaración al respecto.

Atentamente,

Dr. Francisco Javier Alvarez Rodríguez.

Profesor Investigación.

Aguascalientes, Ags. 29 de Noviembre del 2013.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

Asunto: Voto probatorio.

A QUIEN CORRESPONDA:

PRESENTE

Por medio de la presente hago de su conocimiento que la tesista:

Vianney Sotelo Mauries

La impresión de su documento de tesis con titulo: **Modelo MoProSoft, una Adaptación de la Categoría Operación, Proceso Desarrollo y Mantenimiento de Software para las Salidas "Especificación de Requerimientos" y "Análisis y Diseño" al Proceso de Desarrollo de Software Embebido, Caso de Estudio: NI cRIO-9074 de National Instrument para Proyecto Final Vision System,** ya que cumple con los requisitos de contenido y forma exigidas en la Universidad Autónoma de Aguascalientes.

Sin más quedó a sus órdenes para cualquier aclaración al respecto.

Atentamente,

Dr. José Manuel Mora Tavares.

Profesor Investigación.

Aguascalientes, Ags. 10 de Diciembre del 2013.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

Asunto: Voto probatorio.

A QUIEN CORRESPONDA:

PRESENTE

Por medio de la presente hago de su conocimiento que la tesista:

Vianney Sotelo Mauries

La impresión de su documento de tesis con titulo: **Modelo MoProSoft, una Adaptación de la Categoría Operación, Proceso Desarrollo y Mantenimiento de Software para las Salidas "Especificación de Requerimientos" y "Análisis y Diseño" al Proceso de Desarrollo de Software Embebido, Caso de Estudio: NI cRIO-9074 de National Instrument para Proyecto Final Vision System,** ya que cumple con los requisitos de contenido y forma exigidas en la Universidad Autónoma de Aguascalientes.

Sin más quedó a sus órdenes para cualquier aclaración al respecto.

Atentamente,

Dr. Angel Eduardo Muñoz Zavala.

Profesor Investigación.

Aguascalientes, Ags. 10 de Diciembre del 2013.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

UNIVERSIDAD AUTONOMA DE AGUASCALIENTES

ANIVERSARIO UAA

Centro de Ciencias Básicas

ING. VIANNEY SOTELO MAURIES
ALUMNO (A) DE LA MAESTRÍA EN CIENCIAS
CON OPCIÓN A LA COMPUTACIÓN,
P R E S E N T E .

Estimado (a) alumno (a) Sotelo:

Por medio de este conducto me permito comunicar a Usted que habiendo recibido el voto aprobatorio de su tutor de tesis titulada: ***"MODELO MoProSoft, una Adaptación de la Categoría Operación, Proceso Desarrollo y Mantenimiento de Software para las Salidas "Especificación de Requerimientos" y "Análisis y Diseño" al Proceso de Desarrollo de Software Embebido, Caso de Estudio: NI Crio-9074 de National Instrument para Proyecto Final Vision System,*** hago de su conocimiento que puede imprimir dicho documento y continuar con los trámites para la presentación de su examen de grado.

Sin otro particular me permito saludarle muy afectuosamente.

A T E N T A M E N T E
Aguascalientes, Ags., 11 de diciembre de 2013
"SE LUMEN RROFERRE"
EL DECANO SUSTITUTO

M. en C. JOSÉ DE JESÚS RUIZ GALLEGOS

Universidad Autónoma de Aguascalientes
CENTRO DE CIENCIAS BÁSICAS

c.c.cp.- Interesado
c.c.p.- Secretaría de Investigación y Postgrado
c.c.p.- Jefatura del Depto. de Apoyo al Posgrado
c.c.p.- Jefatura del Depto. de Ciencias Computacionales
c.c.p.- Consejero Académico
c.c.p.- Minuta Secretario Técnico
JJRG,mjda

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

"Si añades un poco a lo poco
y lo haces así con frecuencia,
pronto llegará a ser mucho."
Hesíodo

Agradecimientos

Se requiere la misma capacidad analítica para escribir la tesis que para elegir las palabras adecuadas y poder agradecerle a aquellos que me han brindado el apoyo necesario y su sabiduría para concluir el presente proyecto.

A los contribuyentes de nuestro hermoso país, cuya parte de sus recursos son destinados a los programas de becas para posgrados del PNPC del Consejo Nacional de Ciencia y Tecnología (CONACYT) para fomentar el crecimiento profesional.

A mi asesor de tesis Doctor Francisco Javier Álvarez Rodríguez, por su enseñanza, paciencia, soporte y tiempo para poder realizar este proyecto.

Al Doctor José Manuel Mora Tavarez por su apoyo incondicional y ser mi guía en los momentos en que no todo era luz.

Al Doctor Guillermo Ramírez Prado por participar al inicio del proyecto ayudándome a definir mi tema de investigación.

Al Doctor Ángel Eduardo Muñoz Zavala le agradezco sus consejos y palabras de aliento en el camino.

A mis queridos colegas y amigos M.C. Pablo Espinosa Lepe, M.C. José Luis García Servín e Ing. Abimael Luna, así como al departamento de Manufacturing Technology Engineering de Flextronics Manufacturing dirigido por el Ing. Francisco Javier Ochoa. Sinceras gracias por su colaboración en el proyecto.

También deseo agradecer a personas que han sido importantes por haberme brindado su cariño y darle sentido a mi vida todo el tiempo. A mi familia, en especial a mi madre, Mireya Mauries Ruíz, por siempre haber creído en mí donde quiera que ahora se encuentre; a mi padre, Arturo Sotelo Mendoza, por su paciencia al tener que ausentarme de algunas labores profesionales para cumplir con esta meta; a mis tías Alma Patricia Mauries Ruíz y Ofelia Bermeo y sus hermosas familias

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

por ser de gran apoyo y soporte cuando tenía que ir a clases o necesitaba palabras sabias para continuar; a mi hija Alondra Janalthé Hinojo Sotelo por entender que el tiempo que sacrificábamos era para que cumpliera con mis tareas; a mi querida amiga y asesora para corrección de estilo en el presente trabajo, Ing. Mónica Brizuela Sandoval, quien me ayudo de manera profesional y emocional ofreciéndome siempre su apoyo incondicional. A Luis Fernando Torres Roncal que llegó en este justo instante a compartir parte de este logro sembrando ánimo con sus dulces miradas y hermosas sonrisas en (a veces) mi tan desganada y cansada alma.

Agradezco también infinitamente a los que de alguna manera han tocado mi vida dejándome consecuentemente el aprendizaje de alguna lección. Colegas, profesores y compañeros de trabajo y de espacio que al final se convirtieron en amigos junto con sus familias, acompañándome en muchos momentos especiales: Esmeralda Ventura Díaz, Juan Carlos Montaño Monroy, Gaddiel Hurtado Montiel, Edgar Jiménez Ocampo, Lizette Zamora Martínez, María Elena Nájera Medina, Isidro Hinojo Hernández, Rafael Molina Contreras, Arturo Bermúdez, Gustavo Sosa Serna, Marco Antonio Esquivias Mendoza, Carlos Enríquez, Lizbeth Gazano, Carmen Ayala, Lourdes Gutiérrez Calderón, Tere Montañez, Miktlan Ehekateotl Kwauhtlinxan, Paloma, Sergio A. Alcaraz, Eduardo Zamarripa, Guillermo Romo, Eduardo Romo y tantos amigos de años que quizá ya no estén tan cercanos y que por falta de espacio no puedo enunciar pero que llevo siempre en mi corazón.

Por todo lo compartido, enseñado y aprendido en su momento forjándome tal cual ahora soy:

Gracias,
Vianney Sotelo Mauries.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

Dedicatoria

"¿Cómo te amo?
Déjame contarte las maneras.
Te amo con la profundidad,
la anchura
y la altura
que mi alma puede alcanzar."

Elizabeth Barrett Browing

A mi hija
Alondra Janalthé Hinojo Sotelo
quien es, a hoy, el principal amor
y motor de mi vida.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

ÍNDICE GENERAL

.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

ÍNDICE DE TABLAS

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

ÍNDICE DE FIGURAS

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

ACRÓNIMOS

ADL:	Architecture Description Language.
ADT:	Abstract Data Types.
ASIC:	Application Specific IC.
CBS:	Clear Box Structure.
CMMI:	Capability Maturity Model Integration.
CPS:	Cyber-Physical System.
CRSE:	Cleanroom software Engineering.
DRAM:	Dynamic Random Access Memory.
DSP:	Digital Signal Processor.
E/S:	Entrada/Salida.
FAT:	Factory Acceptance Tool.
FPGA:	Field Programmable Gate Array.
FTP:	File Transfer Protocol.
FVS Proposal:	Final Vision System Proposal.
HIL:	Hardware in-the-loop.
HTTP:	Hypertext Transfer Protocol.
HW:	Hardware.
IDEAL:	Initiating, Diagnosing, Establishing, Action, Learning.
IEC:	International Electrotechnical Commission.
IR:	Ingeniería de Requisitos.
IRQ:	Interruption Request.
ISO:	International Organization for Standardization.
MNC:	Multinational Corporation.
MoProSoft:	Modelo de Procesos para la Industria de Software.
MPS.BR:	Mejora de Proceso del Software Brasileño.
MTE:	Manufacturing Technology Engineering.
OPE.X:	Nomenclatura del modelo MoProSoft para Categoría Operación.
PSP:	Personal Software Process.
RFQ:	Request for Quotation.
RTW:	Real time workshop.
RUP:	Rational Unified Process.
SCAMPI:	Standard CMMI Appraisal Method for Process Improvement.
SE:	Sistemas Embebidos.
SMEs:	Small and medium enterprise.
SOW:	Specification Of Work.
SPI:	Software Process Improvement.
SW:	Software.
TSP:	Team Software Process.
UML:	Unified Modeling Language.
VSE:	Very Small Entities.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

RESUMEN

El propósito principal de esta tesis fue llevar a cabo un esfuerzo de contribución para mostrar la adaptación de: "Requerimientos" y "Análisis y Diseño", concernientes a un Modelo de Madurez de Procesos, específicamente al desarrollo de Software Embebido.

El método de investigación presentado es Diseño Conceptual en conjunto con un Caso de Estudio, usando como base el Modelo por Niveles de Capacidad de Procesos para la Industria de Software: MoProSoft Ver. 1.3 Específicamente, se trabajó en la *Categoría Operación: Proceso OPE.2 Desarrollo y Mantenimiento de Software, salidas: "Especificación de requerimientos" y "Análisis y Diseño"*. Dicho modelo es la actual base de la Norma Mexicana para la Industria de Desarrollo y Mantenimiento de Software (NMX-059-NYCE-2005) y una parcial contribución de ISO (ISO/IEC 12207; ISO/IEC 29110). Adicionalmente se hace uso del marco de referencia "ciclos de vida" y técnicas para desarrollo de Sistemas Embebidos, enfocándonos al Software Embebido, buscando prácticas reportadas y herramientas que ayudarán a generar los productos.

El resultado final generado son los artefactos para documentación de salidas en OPE.2: "Especificación de requerimientos" y "Análisis y Diseño" aplicable a cualquier proyecto con Software Embebido implícito. La validación se realizó a través de su aplicación a un Caso de Estudio con NI cRIO-9074 para el Proyecto Final Vision System creado por el departamento Manufacturing Technology Engineering (MTE) en Flextronics Manufacturing.

Con esto se espera haber participado en una contribución innovadora a las propuestas del uso de Software Process Improvement (SPI), llevando la Ingeniería de Software en una pequeña porción al área de los Sistemas Embebidos, tema de interés en la literatura a nivel internacional que ha sido poco explorada.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

ABSTRACT

The main purpose of this thesis was to conduct an effort to show the contribution of adaptation, "Requirements" and "Analysis and Design", concerning to a Process Maturity Model, specifically at the development of embedded software.

The research method is presented as a Conceptual Design in conjunction with a Case of study using a model as a basis for Process Capability Levels for Industry Software: Ver.1.3 MoProSoft Specifically. The thesis is worked on the Operation Category: Process OPE.2 Software development and Maintenance, outputs: "Requirements Specification" and "Analysis and Design". This model is the current basis of Mexican Standard for Industry Development and Maintenance of Software (NMX-059-NYCE-2005) and a partial contribution of ISO (ISO/IEC 12207, ISO/IEC 29110). Further use of the framework "Life cycles" and techniques for developing embedded systems is made, focusing in Embedded Software, looking for reported practices and tools that help to generate the products.

The final results are generated as artifacts for documentation of the outputs OPE.2 "Requirements Specification" and "Analysis and Design" applicable to any project with implicit Embedded Software. The validation was performed through application to a Case of Study with NI cRIO-9074 for Final Draft Vision System created by the department Manufacturing Technology Engineering (MTE) at Flextronics Manufacturing.

With this thesis is expected to have participated in an innovative contribution to the proposed use of Software Process Improvement (SPI), leading software engineering in a small portion of the area of Embedded Systems, topic of interest in the international literature that has been little explored.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

INTRODUCCIÓN

El primer capítulo se compone por los elementos de la Introducción hacia el trabajo de investigación, los cuales dan a conocer la descripción del contexto de la situación problemática de investigación, la relevancia y justificación de la investigación, la descripción general del enfoque y métodos de investigación y la descripción de capítulos de la tesis.

El segundo capítulo explica la formulación de la investigación a través de la descripción del problema, la pregunta y el objetivo, la proposición y la metodología relacionados con la investigación.

El tercer capítulo es parte importante para entender el tema de la tesis, ya que explica el marco teórico primordial a través de los fundamentos teóricos base, resumiendo la descripción de los principales estudios relacionados, el análisis de las contribuciones y las limitaciones de los principales estudios relacionados. De esta manera, se plantea ya el modelo de investigación que se usará y se proporciona la descripción de los elementos que lo componen.

El cuarto capítulo de la tesis explica el desarrollo y la validación de la investigación que se llevó a cabo, usando el modelo de investigación planteado.

El quinto capítulo se centra en dar a conocer las contribuciones esperadas por medio de la explicación de los productos de investigación logrados, las contribuciones al conocimiento, las contribuciones a la práctica y las recomendaciones para investigaciones subsecuentes.

El sexto capítulo muestra las conclusiones sobre los resultados obtenidos, el aprendizaje personal y, finalmente, los productos obtenidos del trabajo de investigación.

Hasta aquí se tiene el bosquejo general del trabajo de investigación entregado para la obtención del grado.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

CAPÍTULO 1

PLANTEAMIENTO DEL PROBLEMA DE ESTUDIO

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

1. PLANTEAMIENTO DEL PROBLEMA DE ESTUDIO

1.1 DESCRIPCIÓN DEL CONTEXTO DE LA SITUACIÓN PROBLEMÁTICA DE INVESTIGACIÓN.

Referente al tema de Desarrollo y Mantenimiento de Software debemos enfatizar que, a través de los reportes vistos en la literatura, se denota una preocupación recurrente de colegas en el área acerca de la falta de metodologías, métodos o modelos para el desarrollo de software en general. Por otra parte también comentan que el uso de éstos y su práctica son escasos en el área del desarrollo de software.

Tomando el enfoque de los sistemas embebidos nos encontramos en la literatura que debido a su complejidad, es aún más escasa la práctica de métodos efectivos para su desarrollo, lo cual se expande en consecuencia al software embebido. Se han encontrado esfuerzos por implementar algunos modelos a los sistemas embebidos, enfocados a ambas partes de desarrollo (hardware y software), dejando muchos huecos sin cubrir en el apartado de software. Algunos ejemplos de éstos huecos son la carencia de herramientas que permitan la documentación del proceso, su evaluación en actividades concluidas y el garantizar la calidad y confiabilidad del producto final a través de la vinculación de los requerimientos, el diseño, el desarrollo, las pruebas, la evaluación y el mantenimiento de éste hasta su liberación.

La complejidad de crear un modelo de proceso de software aplicable a la medida al desarrollo de software embebido puede partir de inicio de su variedad por naturaleza dado el amplio ámbito de su aplicación y la gran diversidad de dispositivos electrónicos usados para el diseño de los sistemas embebidos que albergan al software. Ejemplos de ello son los microcontroladores, procesadores, DSPs y FPGAs por mencionar algunos. Otra problemática es el campo de acción en el cual son requeridos, ya sea desde el nivel industrial para la automatización de procesos o maquinaria, hasta el uso en la denominada domótica. Esta última se refiere a productos de uso doméstico, automatizados por un sistema embebido; tal es el caso de cafeteras, lavadoras, estéreos, televisiones, reproductores de video, cámaras, celulares, dispositivos de control, comunicación, etc. Adicionalmente están presentes en aplicaciones de uso médico, aeronáutico, espacial y militar.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

1.2 RELEVANCIA Y JUSTIFICACIÓN DE LA INVESTIGACIÓN.

A raíz de la problemática descrita, se considera que la proposición del uso de un Modelo de Procesos para el Desarrollo de Software Embebido es un área de interés y posibilidades para llevar a cabo un análisis y generar material, logrando la implementación de dicho modelo. En la presente investigación se ha optado por el uso del modelo *MoProSoft*. Particularmente, se seleccionó la Categoría Operación: Proceso *OPE.2 Desarrollo y Mantenimiento de Software*, haciendo uso de las Actividades *A2. Realización de la Fase de Requerimientos* en su salida *"Especificación de Requerimientos"* y de *A3. Realización de la Fase de Análisis y Diseño* en su salida *"Análisis y Diseño"* para ajustarlas a Software Embebido.

Como se ha citado en el apartado 1.1, la necesidad de generar un producto de calidad es muy importante, sobre todo si el software del sistema embebido corresponde a la aplicación en maquinaria, donde la seguridad del factor humano es prioridad o en el caso de aplicaciones como la aeronáutica, sistemas espaciales, control en plantas nucleares, hidroeléctricas, petroleras, etc. También en medicina tenemos a la instrumentación biomédica, donde es vital garantizar que el producto funcionará en forma eficaz y eficiente, pues de él dependen vidas humanas.

Y en esta misma línea no está exento el uso de dispositivos con software embebido para el hogar o la vida cotidiana. Es también importante garantizar su óptimo funcionamiento, tal como lo menciona Chandy, J. C. (2010) en uno de sus artículos: "Los consumidores no esperan que su televisor se bloquee y reiniciar el sistema; ellos cuentan con que los autos sean de alta fiabilidad, donde, de hecho, se utiliza un controlador computarizado para mejorar la confiabilidad y la eficiencia de los mismos" (p. 7).

Otro ejemplo es el caso de las comunicaciones o aplicaciones críticas en donde el tiempo real es siempre el talón de Aquiles para sistemas de control, tal como los sistemas que usan Ethernet. Una forma en la cual Berger, A., & Berger, A. H. (2002) lo explica es diciendo "Los sistemas embebidos tienen restricciones de tiempo real" (p. 10).

Por ello es importante en los sistemas embebidos la efectividad del software, la cual depende en cierta medida de

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

la calidad del software embebido que se ha programado para los dispositivos electrónicos.

En general, podemos asegurar que para los sistemas de control automático o semiautomático nunca es deseado un mal funcionamiento, desde el punto de vista del usuario final ni del fabricante. Esta irregularidad podría traducirse en reclamaciones continuas o, en casos extremos, hasta la pérdida de vidas humanas.

Todos estos detalles se consideran justificación suficiente para acreditar la necesidad de la implementación de un Modelo de Procesos. Dicho modelo estará orientado a garantizar la calidad del proceso para el desarrollo de Software Embebido, sin importar el tipo de aplicación que éste tenga.

A través de nuestra propuesta basada en el análisis y uso del Modelo de Procesos *MoProSoft*, Actividades *A2. Realización de la Fase de Requerimientos* en su salida *"Especificación de Requerimientos"* y de *A3. Realización de la Fase de Análisis y Diseño* en su salida *"Análisis y Diseño", está investigación* se enfoca a lograr la implementación en el desarrollo del Software Embebido. Lo anterior se comprobará bajo un Caso de Estudio para NI cRIO-9074 de National Instruments, correspondiente al Proyecto Final Vision System de Flextronics Manufacturing. Cabe mencionar que dicho proyecto fue realizado por el departamento de Manufacturing Technology Engineering (MTE) en Flextronics Manufacturing.

1.3 DESCRIPCIÓN GENERAL DEL ENFOQUE Y MÉTODOS DE INVESTIGACIÓN.

Cada uno de los Modelos de Madurez de Procesos citados en la literatura se enfoca en explicarle al usuario el "qué" debe de hacerse en la organización o grupo de trabajo para llevar a cabo el control del proceso de generación de software. Sin embargo, estos modelos no especifican el "cómo" llevar a la práctica dicho proceso.

En base a esa última premisa, se considera viable y factible elaborar una investigación con enfoque de Diseño Conceptual, aunado a un Caso de Estudio fundamentado en la cita de Mora T. M. (2009). A través de su manual para el Método de Investigación Conceptual, el cual "propone usar cuatro fases que están basadas en sugerencias generales reportadas por investigadores sobre la validación de modelos conceptuales (Sargent, 1999), principios para evaluar investigaciones

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

teóricas (Whetten, 1989) y procedimientos y técnicas empleadas en la Investigación Cualitativa de tipo Teoría Fundamentada en Datos (Myers, 2003; Strauss y Corbin, 1990)".

Básicamente, las fases sugeridas en el manual de Mora T. M. (2009) son:

a) Fase I, Formulación del Problema de Investigación.
b) Fase II, Análisis de Trabajos Relacionados.
c) Fase III, Aplicación o Diseño del Modelo Conceptual.
d) Fase IV, Validación del Modelo Conceptual aplicado o diseñado.

Esto significa, en el desarrollo del trabajo de investigación, primero hacer uso de literatura relacionada para conocer los Modelos de Madurez de Procesos reportados en la actualidad; segundo, realizar la validación de la decisión para llevar a cabo un análisis teórico del modelo MoProSoft para aplicarlo en el uso de desarrollo de Software Embebido. Adicionalmente, implica estudiar literatura enfocada a los sistemas embebidos y al software embebido y al caso de estudio NI cRIO-9074 de National Instruments. Finalmente, la aplicación práctica de la documentación del caso bajo el Modelo de Madurez propuesto con el desarrollo de dos artefactos, correspondientes a las salidas de las actividades A2 y A3 del Modelo MoProSoft.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

CAPÍTULO 2

FORMULACIÓN DE LA INVESTIGACIÓN

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

2. FORMULACIÓN DE LA INVESTIGACIÓN

2.1 PROBLEMA DE INVESTIGACIÓN.

Hasta el momento son pocos los buenos resultados reportados en la literatura acerca de los esfuerzos llevados a cabo para el uso de modelos de madurez de procesos en la micro o pequeña empresa en métodos o metodologías para el desarrollo de Software (Alarcón, A. A. C., etal 2011). Lo anterior afecta, como consecuencia, también al tema del Software Embebido. En este último, no hay herramientas concretas y sólidas que permitan garantizar en forma certera la calidad del mismo, independientemente de la aplicación a la que se destine el dispositivo que alberga al software embebido (Chandy, J. C. 2010).

En su artículo Hernández (2010), enfatiza que "El software como parte de un sistema embebido se utiliza para controlar los productos electrónicos a nivel lógico…. Desarrollarlo involucra retos completamente diferentes a los que la mayoría de los programadores de software están acostumbrados". Esto se refiere a que hay muchas áreas del conocimiento que debe ser capaz de digerir el Ingeniero de Diseño y conocer perfectamente el proceso para descifrar sus puntos vulnerables y prever cualquier comportamiento no deseado del sistema.

2.2 PREGUNTA Y OBJETIVO DE INVESTIGACIÓN.

Objetivo (O1).- Estudiar la problemática de falta de modelos, métodos y/o metodologías a través de la investigación conceptual, para adaptar el modelo de MoProSoft en la Categoría Operación OPE.2 Desarrollo y Mantenimiento de software para las Salidas de "Especificación de Requerimientos" y "Análisis y Diseño" al Software Embebido. Se validará a través de un caso de estudio: Desarrollo de software en NI cRIO-9074 para el Proyecto Final Vision System, buscando una contribución al Modelo de Madurez de Procesos aplicable a casos generales de desarrollo de software embebido en el proceso de Desarrollo y Mantenimiento

Pregunta.- ¿Es factible implementar un modelo conceptual específico, logrando el objetivo y cubriendo las restricciones de la Categoría Operación del modelo de MoProSoft para el desarrollo de software embebido?

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

2.3 PROPOSICIÓN DE INVESTIGACIÓN.

Al ser una investigación conceptual aplicada no experimental (Runeson, P., & Höst, M., 2009) tenemos la construcción de proposiciones de investigación, las cuales se denotan como:

P.O: El análisis de la Categoría Operación de MoProSoft, así como la búsqueda de técnicas en los sistemas embebidos enfocadas al software embebido y las herramientas en su desarrollo para llevarlo a la especificación como modelo general, no lograrán garantizar la calidad del producto final (software embebido) albergado en el dispositivo y usado para el control de un sistema embebido, facilitando la delimitación de los requerimientos y la documentación del análisis y diseño del Ingeniero desarrollador.

P.A: El análisis de la Categoría Operación de MoProSoft, así como la búsqueda de técnicas en los sistemas embebidos enfocadas al software embebido y las herramientas en su desarrollo para llevarlo a la especificación como modelo general, lograrán garantizar la calidad del producto final (software embebido) albergado en el dispositivo y usado para el control de un sistema embebido, facilitando la delimitación de los requerimientos y la documentación del análisis y diseño del Ingeniero desarrollador.

2.4 METODOLOGÍA DE INVESTIGACIÓN.

La forma en la cual se elige llevar a cabo la investigación, se basa en la matriz de clasificación para los principales tipos de metodologías de Investigación (Mora, T. M. 2011). Dicha matriz se muestra en la Figura 1. De esta manera, se propone el uso de una metodología de investigación tipo "Diseño Conceptual", usando como base el modelo MoProSoft.

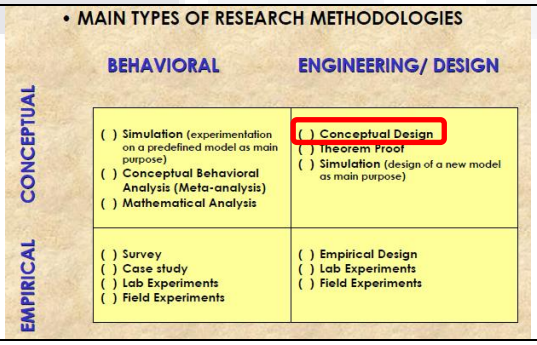


Figura 1. Principales tipos de metodología de Investigación (Mora, T. M. 2011).

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

A través del seguimiento del modelo de investigación propuesto, se pretende llegar a la generación de una salida de investigación de "Diseño Conceptual" (Mora, T. M. 2011), avalada por la clasificación de los principales tipos de resultados para investigación sugeridos y mostrados en la figura 2.

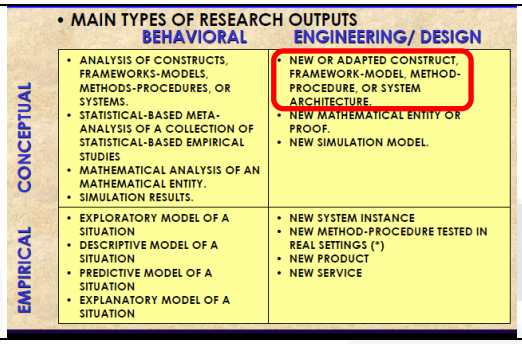


Figura 2. Principales tipos de salidas de Investigación (Mora, T. M. 2011)

La validación de la investigación se ejercerá a través de un Caso de Estudio o del Modelo Exploratorio de una situación. En específico, se llevará a cabo la aplicación de los productos de salida generados en el proyecto Final Vision System, del departamento de Manufacturing Technology Engineering (MTE) en Flextronics Manufacturing.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

CAPÍTULO 3

MARCO TEÓRICO FUNDAMENTAL

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

3. MARCO TEÓRICO FUNDAMENTAL

3.1 FUNDAMENTOS TEÓRICOS BASE.

El apartado del Marco Teórico Fundamental tiene como objetivo mostrar las bases conceptuales que sustentan el desarrollo de la investigación. Con la finalidad de organizar la información de manera fluida, dada la complejidad de estructura al usar tres áreas de conocimiento: Ingeniería de Software, Modelos de Procesos y Sistemas Embebidos, se considera pertinente subdividirlo en tres importantes temas y un subtema tal como se muestra a continuación:

- 3.1.1. Los Modelos para el Desarrollo de Software: En este apartado se explica qué son los modelos y las razones por las cuales se recomienda su uso.
- 3.1.2. Descripción del Modelo de MoProSoft: Explicación de qué es el modelo y su composición.
- 3.1.3. Sistemas Embebidos, enfoque en el módulo: Software Embebido: Se explica qué son los sistemas embebidos particularmente el software embebido.
  - 3.1.3.1. Introducción a NI cRIO-9074 y sus Aplicaciones.

De esta forma quedan claramente definidos los límites de los temas de interés relacionados directamente con el título de la tesis.

3.1.1 Los Modelos para el Desarrollo de Software.

Partiendo de la definición del término "Modelo" más adecuada a nuestro estudio por el Diccionario de la Real Academia de la Lengua Española (2001) tenemos que:

> Un modelo es un arquetipo o punto de referencia para imitarlo o reproducirlo. En las obras de ingenio y acciones morales es un ejemplo que se debe seguir o imitar por su perfección. Un modelo es también la representación en pequeño de alguna cosa. En empresas, una posición para indicar que lo designado por el nombre ha sido creado como ejemplar o se considera que puede serlo.

Por su parte Tardío, M. A. (2011) lo define como una estructura conceptual que sugiere un marco de ideas para un conjunto de descripciones que de otra manera no podrían ser sistematizadas. De ahí se parte a una explicación de los

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

varios modelos y estándares existente en los que se tratan temas para la generación de productos y/o servicios de software, tales como ISO 9001:2000; ISO/IEC 12207:2002, CMMI, MoProSoft o MPS.BR, entre otros.

Además de definir a un modelo, también es importante tener clara la idea de lo que es un proceso de desarrollo de software. Mishra, D. (2009) lo define de la siguiente manera: "… el proceso de software... es un ambiente de capacidades de recursos interrelacionados, administrando una secuencia de actividades usando métodos apropiados y prácticas para desarrollar un producto de software que está conformado para cumplir los requerimientos del cliente". Esto es que el proceso se ve afectado por los recursos con los que cuenta la empresa y con una cultura organizacional enfocada a la satisfacción del cliente.

Hasta este punto se consideran definidos a un modelo y el proceso de software, ahora es posible abordar la descripción de los Modelos de Calidad, definiéndolos en forma específica bajo la óptica de Paternina, P. K. et al.(2011). Dichas autoras definen a los Modelos de Calidad como herramientas que guían a las organizaciones a la mejora continua y la competitividad, dándoles especificaciones de qué tipo de requisitos deben de implementar para poder brindar productos y servicios de alto nivel. En referencia a la definición dada, las autoras alertan de que un Modelo de Calidad no es una metodología que resuelva la vida de forma sencilla y clara, haciendo énfasis en que éste más bien dirá "Qué hacer" y no "Cómo hacerlo", ya que este último cuestionamiento depende de la metodología a usar y de los objetivos de cada negocio.

Pero hablando de calidad en general, encontramos que en la industria se denotan un sin número de productos físicos, los cuales son creados bajo ciertos criterios o normas que garantizan dicha calidad del producto. Quizá es sencillo visualizar la calidad en productos tales como los alimentos, que deben pasar por normas estrictas de organismos de salubridad; productos electrodomésticos que cumplan las normas de seguridad eléctrica; incluso en el caso de la educación o los servicios se puede escuchar hablar de calidad. Visualizando la calidad desde la perspectiva de un ciudadano(a) consumidor(a), se puede percibir cómo cada uno de ellos prefiere adquirir diferentes clases de productos, que de un individuo a otro varía la elección dependiendo del gusto, costumbre, región o necesidad específica. Del mismo

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

modo el software, a pesar de no ser un producto físico palpable, para poder satisfacer las necesidades de su usuario debe cumplir con cierta "calidad". Por su parte, Paternina, P. K. (2011) menciona que la calidad en el proceso de generación de los productos de software se tiene una exigencia creciente, consecuencia del amplio uso del software en procesos críticos para las organizaciones o maquinaria.

La respuesta que se reporta para lograr software de calidad en forma consciente es la de usar Modelos de Madurez de Procesos, y en forma extra oficial sabemos que también está implícita la experiencia del desarrollador o desarrolladores del software.

La bibliografía consultada para analizar los modelos de procesos no define concisamente "Modelo de Madurez de Procesos". Sin embargo, de forma deductiva puede intuirse como una estructura que enmarca los elementos, acciones y actores esenciales que deben documentarse, ejecutarse y verificarse dentro de un proceso a través de métricas que permitan conocer el nivel de madurez que tiene dicho proceso u organización.

Partiendo de esta premisa, Paternina, P. K. (2011) define que "el modelo de madurez de procesos más popular es CMMI (Modelo de Capacidad y Madurez Integrado). Sin embargo, este modelo es complejo para implementar en empresas pequeñas". Mishra, D. (2009) escribe acerca de la calidad del software mencionando que depende en gran medida del proceso que se usa para crearlo.

Siguiendo la misma línea de explicación de modelos, Pino, F. (2007) y Mishra, D. (2009) hacen énfasis en compartir que las propuestas de mejora del SEI e ISO (como CMMI, IDEAL, SCAMPI, ISO 12207, ISO 15504) han sido creadas y están estructuradas para ser utilizadas por organizaciones grandes y difícilmente pueden ser aplicadas a organizaciones pequeñas. Esto es porque un proyecto de mejora supone gran inversión en dinero, tiempo y recursos, así como la alta complejidad de las recomendaciones y también porque el retorno de la inversión se produce a largo plazo y tal como es de esperarse, las pequeñas empresas desean la obtención de una alta calidad a un bajo costo (Mishra, D. 2009). Este tema es muy notorio en el contexto nacional, donde empresarios y emprendedores no cuentan con los recursos económicos al arrancar su proyecto y donde los programas de apoyo suelen ser procesos largos y complicados para obtener el acceso a dichos recursos.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

Desafortunadamente otro factor que es visible en la cotidianidad es que dichos negocios empezarán a verse redituables cuando menos cinco años, si los sobreviven como empresa, después de emprender.

Surgido de estas desventajas de los modelos Pino, F. (2007) comenta que a partir de finales de los años noventa ha tomado gran fuerza en la comunidad de Ingeniería de Software (industria e investigadores) la Mejora de Procesos de Software (SPI, Software Process Improvement) en pequeñas organizaciones, dedicadas a la creación de software, para resolver los inconvenientes de los modelos que ya se han mencionado. Esta opinión se refuerza en la opinión de Mishra, D. (2009) bajo la observación de que SPI ha sido reconocido como una forma efectiva en las compañías para la mejora de la calidad de software.

Para la adecuación de los modelos a las pequeñas organizaciones Pino, F. (2007) menciona que se requirió de una razón de impulso en esta área, la cual se especifica bajo la opinión de muchos autores acerca de que las características especiales de las pequeñas organizaciones de software hacen que los programas de mejora de procesos deban aplicarse de un modo particular y visiblemente diferente a como se hace en las grandes organizaciones y que esto no es tan sencillo como el hecho de considerar a dichos programas de mejora como versiones a escala de los usados en las grandes compañías. Esto se puede traducir en que muchas de las pequeñas organizaciones no poseen un gran número de recursos materiales o humanos para realizar su proceso de desarrollo de software, tal como lo hacen las grandes organizaciones, y en la mayoría de las ocasiones su estructura organizacional difiere de los grandes corporativos.

Cuando se habla de una pequeña empresa se hace referencia a la clasificación de una SME's, bajo el concepto inicial de Mishra, D. (2009), cuyo promedio de empleados es de 1 a 50 y se caracteriza por su insuficiencia de recursos humanos, la falta de desarrollo y medio ambiente de apoyo, la falta de presupuesto y su dependencia de las grandes organizaciones, por lo cual estas SME's encuentran a la mejora de proceso de software como un reto mayor. Del mismo modo Mishra, D. (2009) cita a Johnson and Brodman para definir que para ellos una pequeña organización, o haciendo la analogía una SME's, está compuesta por menos de 50 desarrolladores de software y con proyectos pequeños menores a 20 proyectos de desarrollo.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

Lo anterior no es el único reto a superar, también Tardío, M. A. (2011) habla de la importancia del problema cultural cuando se quieren importar y adoptar modelos definidos en otros países. Zahran (1998) indica que "Si el proceso no casa con la cultura de la organización será rechazado por el cuerpo organizacional como sucede en los trasplantes de órganos" y la Mishra, D. (2009) enfatiza que "... las organizaciones... no tienen una cultural de procesos y en una cultura de procesos también las costumbres de la gente y su conducta son influidas por la orientación del pensamiento y la administración de los principios del proceso". Es indiscutible que el factor cultural y también el temor de los individuos a romper paradigmas históricamente ha sido una lucha considerable siempre que se pretende introducir una nueva tecnología, método, proceso, idea o metodología.

Pero no hay que perder de vista que, tal como lo expresa Mishra, D. (2009), "hay evidencia de que la mayoría de las pequeñas organizaciones desarrolladoras de software no están adoptando los estándares existentes, esto derivado de que las percepciones negativas de las pequeñas firmas son principalmente conducidas por una vista negativa de los costos, la documentación y la burocracia". Empero también estas organizaciones pequeñas y medianas han reconocido la necesidad de mejorar sus productos de software y evaluarlo parece insuficiente ya que se sabe que su calidad depende del proceso usado para crearlo, por lo cual buscan la forma de evaluar sus procesos y productos.

En conclusión el análisis de estas investigaciones lleva a resumir que al día de hoy la industria que desarrolla software, que es generalmente una pequeña entidad, está interesada en mejorar sus procesos, especialmente considerando a SPI. También es destacable mencionar que este tipo de compañías son las que más aportan a la economía de cada país. Y quizá es posible aventurarse a pensar que, debido a su definición, son las que mayormente tienden a emerger como sector. La pregunta, consecuencia de estas investigaciones, es: ¿se puede trabajar en algún modelo que pueda cubrir las expectativas para la evaluación de sus procesos y productos como pequeñas empresas volviéndose en consecuencia competitivas? Ya sea en un contexto nacional o internacional. En respuesta a esta pregunta, aunque quizá con otras palabras otorgadas por la comunidad investigadora en conjunto con la industria, se expone por Ríos, B.L. (2008) el resumen de los resultados del interés que algunos países tienen en el correcto desarrollo y crecimiento de la

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

industria del software, apareciendo en consecuencia estándares como "Impact, Processus, Adept, Rapid, Agile SPI, Mares, MesoPyME, SIMEC-SW, MR-MPS, Competisoft" por mencionar algunos.

3.1.2 Descripción del Modelo de MoProSoft.

Después de haber consultado la literatura, parece importante mencionar lo que se consideran las raíces del Modelo MoProSoft, descrito en el artículo Oktaba, H., & Ibargüengoitia, G. G. (1998) explicando a través del modelado con objetos como estructurar y sistematizar los conceptos fundamentales del Proceso de Software con la finalidad de comprenderlo mejor a través de clases y relaciones entre estas. De esa manera incluye la forma en la cual se asocian las estructuras entre dependencias de los conceptos básicos tales como: fases, actividades, productos, roles y agentes para cada categoría del modelo. En la figura 3 se replica el principio del modelado.

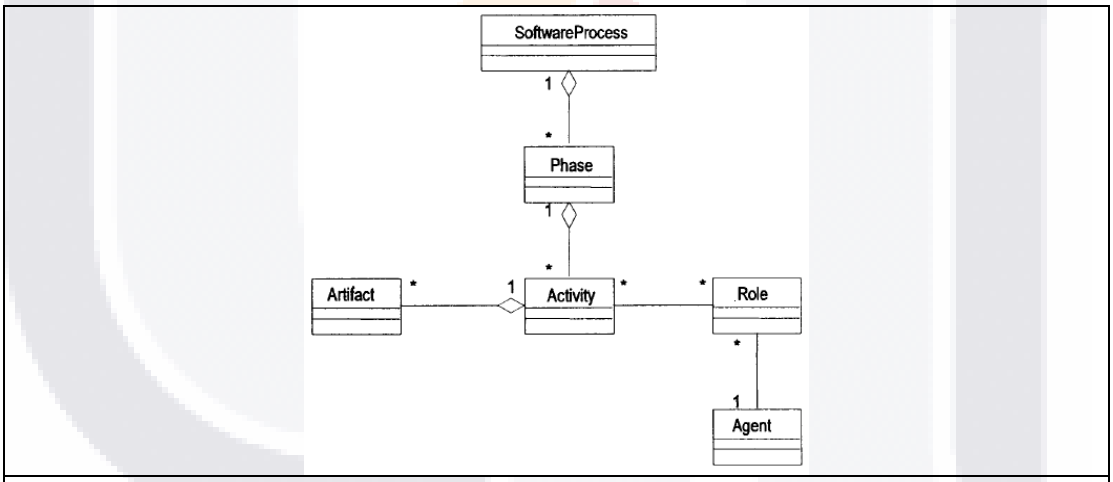


Figura 3. Diagrama de clases para proceso de software, extraído de Oktaba, H., & Ibargüengoitia, G. G. (1998)

Preliminar al trabajo de Oktaba, H., & Ibargüengoitia, G. G. (1998) existieron otros enfoques para el modelado del Proceso de Software y lo citan bajo la referencia de Huff (1996), en el mismo documento, clasificados como:

- Paradigma no ejecutable – Textual o gráfico (Ej. IDEF0).
- Paradigma basado en estados – Estados autómatas, Redes de Petri, Gramáticas formales.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

- Paradigma basado en reglas – Sistemas expertos, Prolog, Sistemas de planeación.
- Paradigma Imperativo – Ada como lenguaje de especificación.

Es por ello que al no existir trabajo previo con el uso del modelado orientado a objetos hacen su propuesta para explicar de manera fácil la complejidad de la estructura estática del Proceso de Software para un mejor entendimiento al explicarlo a la comunidad académica y grupos desarrolladores de software.

La generación del modelo parte de la descripción del Proceso de Software y sus componentes. Posteriormente llevaron a cabo la construcción del modelo abstracto haciendo una identificación de las clases básicas del proceso y sus relaciones, que ya han sido mostradas en la figura 3 bajo la notación UML.

Una vez que se tuvo el modelo general presentaron el diagrama de la especialización de las fases para ello se basaron en algunos modelos del ciclo de vida para el desarrollo de software y que son referenciados en el documento como: "modelo en cascada [Royce, 1970; Böhem, 1981], modelo de espiral [Böhem, 1981] y el modelo iterativo e incremental para el desarrollo orientado a objetos [Booch, 1994]. Encontrar los puntos comunes e importantes entre ellos permitió el modelado de la clase de Fases.

En segundo lugar desarrollaron la especialización de Actividades, dejándolas divididas en cuatro grupos básicos: producción, control, tecnología y comunicación. En el documento son definidas las características de cada una de estas haciendo énfasis en que dependerá de la aplicación del software desarrollado el elegir las técnicas, métodos y/o metodologías que se usen para llevar a cabo cada una de las tareas.

En otro apartado se detalla la especialización de Roles que bajo la concepción de las autoras refleja la clasificación de las actividades.

La especialización de artefactos también está directamente relacionada con las actividades al requerirse el control de las entradas y las salidas de información vital para los involucrados en el proceso.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

Por último se menciona que en la especialización de los agentes pueden ser humanos o herramientas, pero para fines del trabajo sólo se describen los humanos.

Al llegar a este punto ya hay un panorama más claro que permite explicar la composición del Patrón de Procesos que compone al modelo MoProSoft.

El manual Oktaba H., et al.(2005) define al Modelo MoProSoft como un Modelo de Procesos para la Industria de Software descrito por niveles de capacidad de procesos. Desarrollado a solicitud de la Secretaría de Economía para servir de base a la Norma Mexicana para la Industria de Desarrollo y Mantenimiento de Software bajo el convenio con la Facultad de Ciencias, Universidad Nacional Autónoma de México.

Está conformado por el concepto de Patrón de Procesos que es un esquema de elementos que sirve para la documentación de los procesos. Se constituye por tres partes, donde sus componentes se sintetizan de manera general en la tabla 1.

DEFINICIÓN GENERAL DEL PROCESO	**PRÁCTICAS**	**GUÍAS DE AJUSTE**
Nombre del proceso. Categoría a la que pertenece. Propósito. Descripción general de actividades. Objetivos. Indicadores. Metas cuantitativas. Responsabilidad y autoridad. Subprocesos (si existen). Procesos relacionados. Entradas. Salidas. Productos internos.	Roles involucrados y capacitación. Actividades que incluyen: los roles, la descripción y nombre de la actividad con un número consecutivo. Diagrama de flujo del trabajo. Verificaciones y validaciones. Productos que deben ser incorporados a la base del conocimiento. Recursos de Infraestructura. Mediciones. Capacitación. Situaciones excepcionales. Lecciones aprendidas.	Identificación de la guía: Estas serán las posibles modificaciones al proceso que no deben de afectar los objetivos del mismo.

Tabla 1. Resumen para el Patrón de Procesos de MoProSoft Oktaba, H., & Ibargüengoitia, G. G. (1998).

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

Éste Patrón de Procesos es el esquema para documentar los procesos de MoProSoft, pero el modelo puede ser adecuado a ciertas necesidades, modificado en cierta forma o en su defecto agregar procesos que no se contemplan, pero siempre respetando una serie de reglas descritas en el manual y preservando los objetivos, indicadores y metas cuantitativas.

Ya que se ha descrito el patrón, se procede a describir la estructura del modelo de Procesos MoProSoft, la cual se divide en tres categorías principales con sus respectivos procesos:

- Categoría de Alta Dirección (Procesos: Gestión de negocios).
- Categoría de Gerencia (Procesos: Gestión de Procesos, Gestión de Proyectos, Gestión de Recursos –Humanos y ambiente de trabajo, Bienes y servicios e Infraestructura, Conocimiento de la Organización-)
- Categoría de Operación (Procesos: Administración de Proyectos Específicos, Desarrollo y Mantenimiento de Software).

MoProSoft es un modelo de uso general, es decir, aplicable en cualquier PyME enfocada al desarrollo de software. Su objetivo es poder llevar a cabo la estandarización de los procesos, procedimientos y documentación en el ciclo de desarrollo del software, logrando así generar una base del conocimiento que se pueda someter a un análisis para lograr el mejoramiento continuo en todos los procesos involucrados en el desarrollo para minimizar esfuerzos en diseños futuros de software. El diagrama de categorías de procesos se reproduce en la figura 4.

Como puede observarse en el diagrama de categorías la Categoría Operación cuenta con dos procesos que han sido identificados con una abreviatura en el manual de MoProSoft como a continuación se describe:

1. OPE.1: Administración de Proyectos Específicos.
2. OPE.2: Desarrollo y Mantenimiento de Software.

El proceso de interés está enfocado en la categoría OPE.2: Desarrollo y Mantenimiento de Software, cuyo patrón y diagrama de flujo se describirá en el capítulo 4.1.1. a la par del desarrollo del trabajo de investigación.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

Es importante resaltar que para poder cubrir las actividades principales del modelo, y como consecuencia de cada proceso en cada categoría, se debe controlar la generación de entradas (provenientes de otro proceso o actividad), salidas (destinadas a otro proceso o actividad) y productos internos (monitoreo interno del proceso).

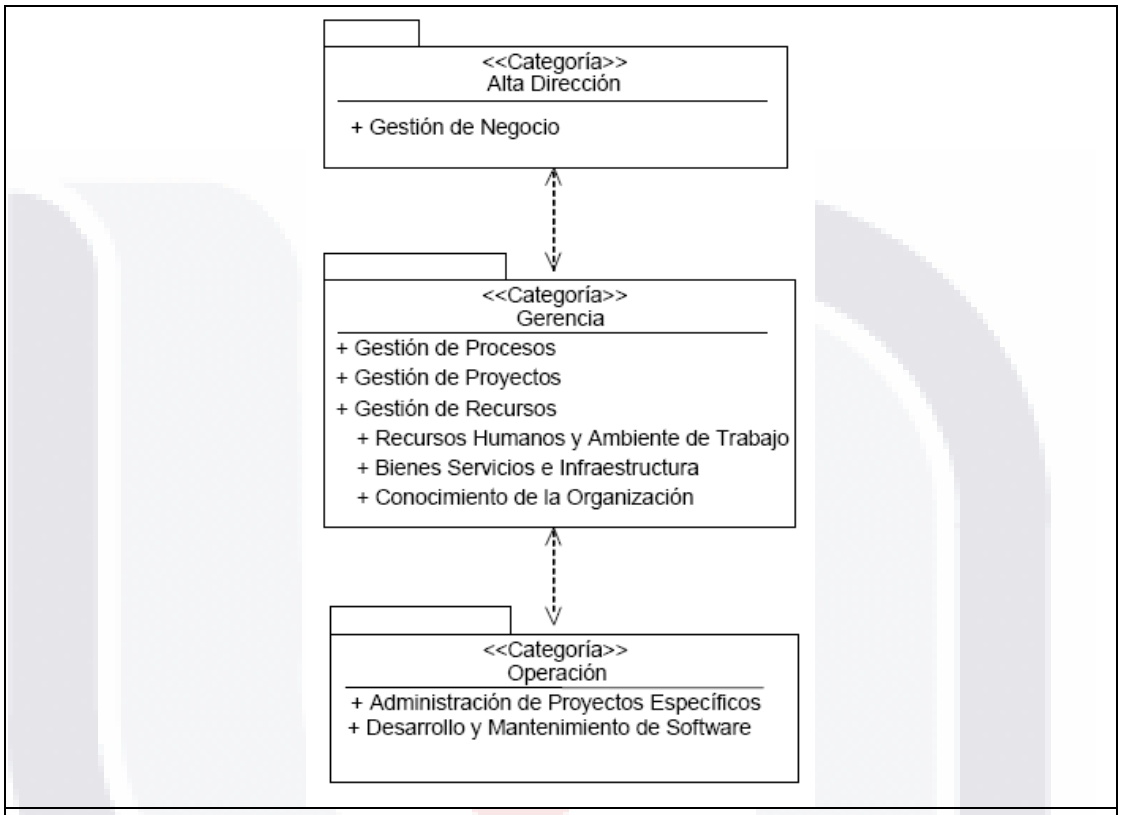


Figura 4. Categorías de Procesos MoProSoft, extraído de Oktaba H., et al.(2005)

La versión del manual de MoProSoft usado es la versión coloreada, lo cual permite determinar el nivel de madurez de los procesos en la empresa o área de software donde se aplica el modelo después de la validación bajo la siguiente consideración:

Nivel	Capacidad de proceso	Color
1	Realizado	amarillo
2	Gestionado	azul
3	Establecido	verde
4	Predecible	rosa
5	Optimizado	ninguno

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

Se hace énfasis en el uso de los colores dado que durante la descripción del desarrollo en el capítulo 4.1.1. se usarán para especificar el nivel que le corresponde a las actividades y productos.

Y continuando con la evolución del modelo, tenemos referencia de cómo ha sido encaminado a la norma NMX-I-059/04 descrita brevemente en el siguiente apartado.

3.1.2.1 MoProSoft como una norma nacional hacia un mapeo internacional.

Para responder a la pregunta ¿Cómo evolucionó el modelo de MoProSoft a una norma nacional?, se tiene como marco de referencia la importancia del modelo en la proyección internacional, relatado en Oktaba, H. (2007).

El modelo de MoProSoft fue aprobado como norma por NYCE el 5 de julio de 2005 y publicada el 15 de agosto bajo la notación NMX-I-059/04-NYCE-2005 Tecnología de la Información-Software-Modelos de procesos y evaluación para desarrollo y mantenimiento de software. Está integrada por cuatro partes listadas y definidas a continuación:

Parte 01: Definición de conceptos y productos.
Parte 02: Requisitos de procesos (MoProSoft) -> Norma ISO
Parte 03: Guía de implantación de procesos.
Parte 04: Directrices para la evaluación (Eval Prosoft)

Es importante mencionar que la parte 02 fue considerada para el mapeo a la norma ISO/IEC 12207 por el interés del delegado de Finlandia, Timo Varkoi en 2007 bajo una reunión en Moscú (Oktaba, H. 2007).

Al ser necesaria la evaluación del modelo con formalidad se desarrolló la norma NMX-I-006/02-NYCE-2006, que está orientada a la evaluación de procesos y a la aplicación de la evaluación para la mejora y determinación de la capacidad.

Dejando hasta este punto la forma en la cual el trabajo con MoProSoft ha ido trascendiendo, se desea incluir que a nivel de Ingeniería de Software en los estándares de ISO se menciona también al ISO/IEC 90003:2004 que proporciona una guía para organizaciones en la aplicación de ISO 9001:2000 a la adquisición, provisión, desarrollo, operación y mantenimiento de software computacional y servicios de soporte relacionado. Se hace énfasis en que ISO/IEC

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

90003:2004 no agrega o de ninguna forma cambia los requerimientos de ISO 9001:2000 y sus directrices proporcionadas no se intentan usar como criterio de valoración en la calidad del sistema de gestión de registro/codificación (ISO/IEC 90003:2004).

La aplicación de ISO/IEC 90003:2004 es apropiada al siguiente tipo software que:

1. Es parte de un contrato comercial con otra organización.
2. Un producto disponible para un sector del mercado.
3. Usado para apoyar los procesos de una organización.
4. Embebido en un producto de hardware o
5. Relacionado a servicios de software.

El detalle con este documento es que se visualiza bajo su abstract como independiente de la tecnología, los modelos del ciclo de vida, el desarrollo de procesos, la secuencia de actividades y la estructura organizacional propia de una organización trasladando al final como referencia para ello a guías adicionales y referencias enfocadas a ISO/IEC JTC 1/SC 7 que son estándares para la ingeniería de software en este rubro proporcionados para proporcionar asistencia en la aplicación del ISO 9001:2000, en particular al ISO/IEC 12207, ISO/IEC TR 9126, ISO/IEC 14598, ISO/IEC 15939 and ISO/IEC TR 15504.

Como se podrá observar nuevamente llegamos al ISO/IEC 12207, donde ya fue mostrada la participación del modelo MoProSoft.

3.1.3 Sistemas Embebidos, enfoque en el módulo: Software Embebido

Para llevar a cabo la apertura con una definición clara, se ha recurrido a un glosario de términos industriales obtenido en Netrino (2011), cuya traducción se cita a continuación:

"Sistema Embebido: Una combinación de hardware computacional y software, y quizá mecánica de manera adicional u otras partes, diseñadas para ejecutar una función dedicada. En algunos casos, los sistemas embebidos son parte de un sistema o producto más largo, como en el caso de un sistema de antibloqueo de freno en un carro. Algunos ejemplos son: hornos de microondas, celulares, calculadoras, relojes digitales, VCR's, misiles cruise, receptores GPS, monitores de corazón, impresoras láser, armas de radar, controladores

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

de motor, cámaras digitales, semáforos, controles remotos, máquinas tostadoras, máquinas de fax, localizadores, cajas registradoras, transportadores, bombas de gas, lectores de tarjetas de crédito/débito, termostatos, marcapasos, monitores de gas en sangre, analizadores de grano y gazillion, otros."

Otra definición citada en la literatura García, R. C. I. (2001) define que "el software embebido se refiere a los sistemas de cómputo que reside en muchos casos, sin que el usuario se entere, dentro de estos productos. Éste software forma parte de un sistema embebido el cual podemos entender como un subsistema electrónico de procesamiento, programado para realizar una o pocas funciones para cumplir con un objetivo específico. Generalmente es parte integral de un sistema heterogéneo mayor, que puede incluir partes mecánicas, eléctricas y/o electromecánicas. La historia de los sistemas embebidos se remonta a comienzos de los años 60, cuando dispositivos basados en microprocesadores y microcontroladores comenzaron a emplearse en el control de tareas aeronáuticas y espaciales. Las limitaciones de alto costo y diseño de estos primeros dispositivos provocaron una espera hasta 1992 en el que se creó el consorcio PC/104 Forward, A., & Lethbridge, T. C. (2008), formado por Ampro, RTD y otros fabricantes."

Algunas otras definiciones encontradas en sitios no formales lo refieren de manera muy similar, por lo cual se resume en conclusión que: Un sistema embebido es el conjunto de hardware y software usado para realizar una tarea de automatización, cálculo o procesamiento, y por consecuencia el software embebido es aquel código almacenado en un dispositivo del hardware para cumplir las funciones deseadas. También se usa el término sistema empotrado (Posadas Cobo, H. 2011) o sistema ciber-físico (Ting, J. S., 2011) para referirse a éstos.

De ahí sobreviene una pregunta en un sistema embebido: ¿dónde se aloja el software embebido? La respuesta obvia es que en algún elemento de hardware en el sistema, pero ¿cuál es ese hardware? Al respecto se puede decir que hay una gran variedad de dispositivos que pueden almacenar software embebido para una aplicación y que su elección depende de las características (o requerimientos generales) del sistema embebido y su finalidad. Algunos dispositivos usados para el almacenamiento de software embebido se citan en por Espino, E., & Torres, D. (2013) cuando menciona las ventajas de un

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

FPGA. Llevando a cabo una comparación de los FPGA con microprocesadores, microcontroladores, procesadores digitales de señales (DSPs) para aplicaciones de control y cómputo.

Bueno pues esos precisamente son los elementos hardware de un sistema embebido que usualmente contiene al software embebido. Nótese que se muestran de forma enunciativa mas no limitativa, es decir que quizá existan otros o hayan sido desarrollados durante el proceso de la tesis nuevas tecnologías. Por el momento se deja esta lista con propósito de referencia.

A continuación se resumen las características de los sistemas embebidos tomados de Pérez, A. David A. (2009), quien considera los más relevantes citados en Vahid, F. & Givargis, T. (2002):

1. *Funcionamiento específico.* Un sistema embebido usualmente ejecuta un programa específico de forma repetitiva. En contraste, un sistema de escritorio ejecuta una amplia variedad de programas, además nuevos programas son añadidos frecuentemente. Puede haber excepciones, podría ocurrir que el programa del sistema embebido fuese actualizado a una nueva versión.
2. *Fuertes limitaciones.* Todos los sistemas de computación poseen limitaciones en sus métricas de diseño, pero en los sistemas embebidos son muy fuertes. Una métrica de diseño es una medida de algunas características de implementación, como: costo, tamaño, desempeño, y consumo de energía. Los sistemas embebidos generalmente deben ser poco costosos, poseer un tamaño reducido, tener un buen desempeño para procesar datos en tiempo real, y además consumir un mínimo de energía para extender el tiempo de vida de las baterías o prevenir la necesidad de elementos adicionales de enfriamiento.
3. *Reactivos y tiempo real.* Muchos sistemas embebidos deben ser reactivos o reaccionar ante cambios en el ambiente, además de realizar algunos cálculos en tiempo real sin ningún retraso, es decir, se deben tener resultados en tiempos fijos ante cualquier eventualidad. Por ejemplo, el módulo de control de viaje de un automóvil continuamente monitorea la velocidad y los sensores de frenos, reaccionando ante cualquier eventualidad. Ante un estímulo anormal, el módulo de control debe realizar los cálculos de forma precisa y acelerada para garantizar la entrega de los resultados dentro de un tiempo límite, una violación en este tiempo podría ocasionar la pérdida del control del automóvil. En contraste,

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

un sistema de escritorio se enfoca en realizar cálculos con una frecuencia no determinada y la demora de los mismos no produce fallas en el sistema.

Así como se tienen características específicas para los sistemas embebidos, también se encuentran enunciadas características del software embebido en Hernández, V. J. I. (2010) y son tres importantes atributos típicos que se deben tener para el desarrollo del software embebido: "confiabilidad, limitaciones en recursos de hardware y respuesta en tiempo real Obregón, H. (2007),". La figura 5 ilustra y explica estos atributos en forma de esquema.

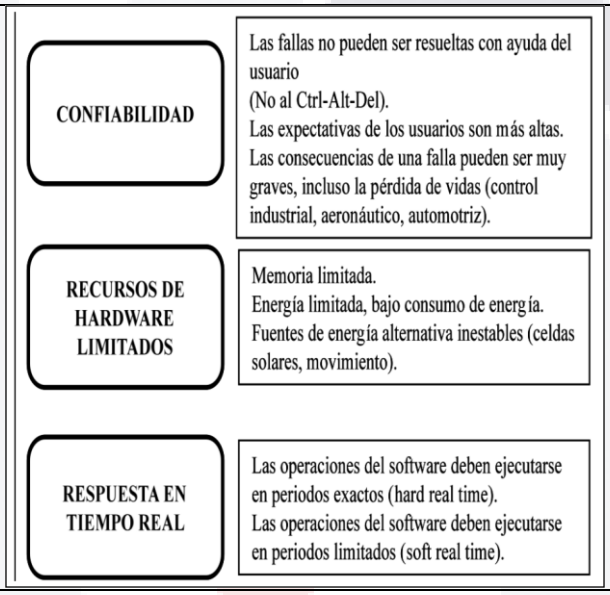


Figura 5. Atributos del software embebido, extraída de Hernández, V. J. I. (2010)

Se considera importante listar las características de los sistemas embebidos y del software embebido para notar su similitud más no su igualdad, es decir, no se habla de lo mismo al mencionar Sistema Embebido que al referirse al Software Embebido. La reflexión viene en el sentido de tener siempre en mente que el Software Embebido y el hardware en donde se aloja son subconjuntos del Sistema Embebido.

El párrafo anterior lleva al punto de buscar algunos diagramas que ilustren el lugar que ocupa el software embebido dentro de un sistema embebido. Algunas representaciones encontradas en la literatura son las siguientes.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

En Hernández, V. J. I. (2010) se presenta el diagrama reproducido en la figura 6, la cual muestra la forma en la que se relacionan otros sistemas con el sistema embebido y el lugar que ocupa el software embebido dentro del sistema.

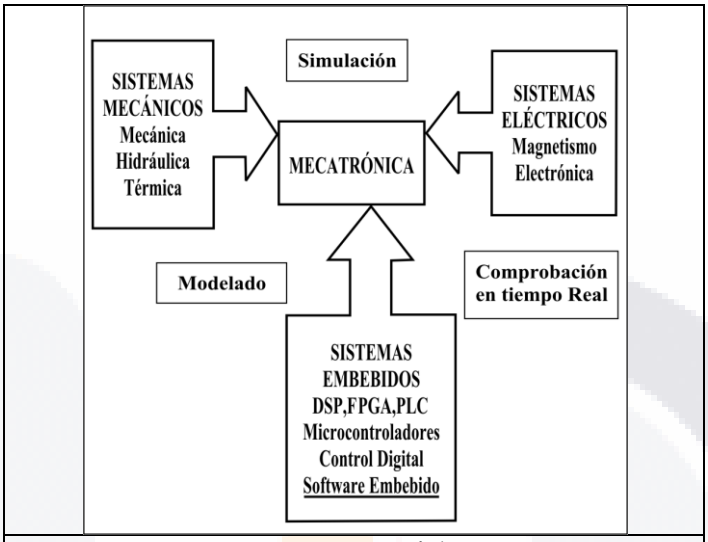


Figura 6. Relación de la mecatrónica, sistemas relacionados y el software embebido, extraído de Hernández, V. J. I. (2010)

Otro diagrama retomado de Pérez, A. David A. (2009) ilustrado en la figura 7 muestra un sistema embebido desde el enfoque técnico de aplicación para las comunicaciones.

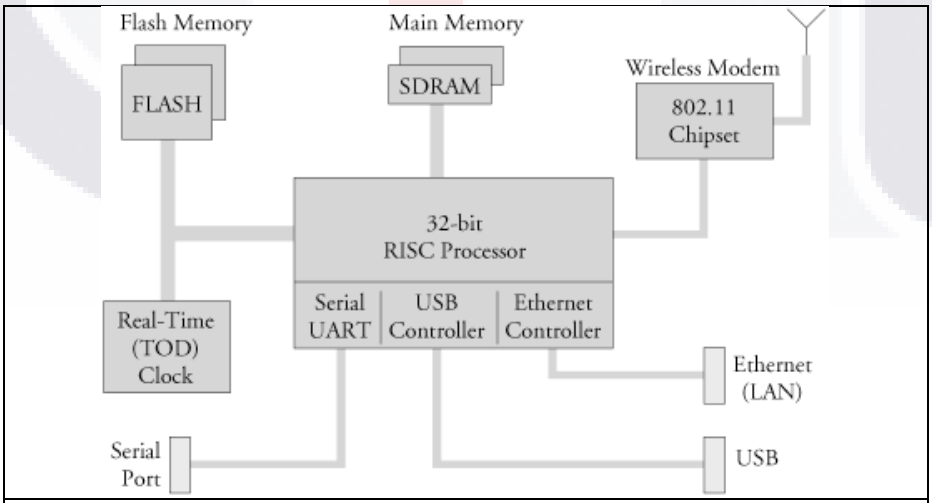


Figura 7. Ejemplo de sistema embebido, extraído de Pérez, A. David A. (2009)

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

En el trabajo de Núñez, G. O. R. N. (2011) se encuentra la figura 8 ilustrando con un diagrama a bloques la tarjeta NI Single Board RIO visualizada como un sistema embebido para adquisición de datos y control:

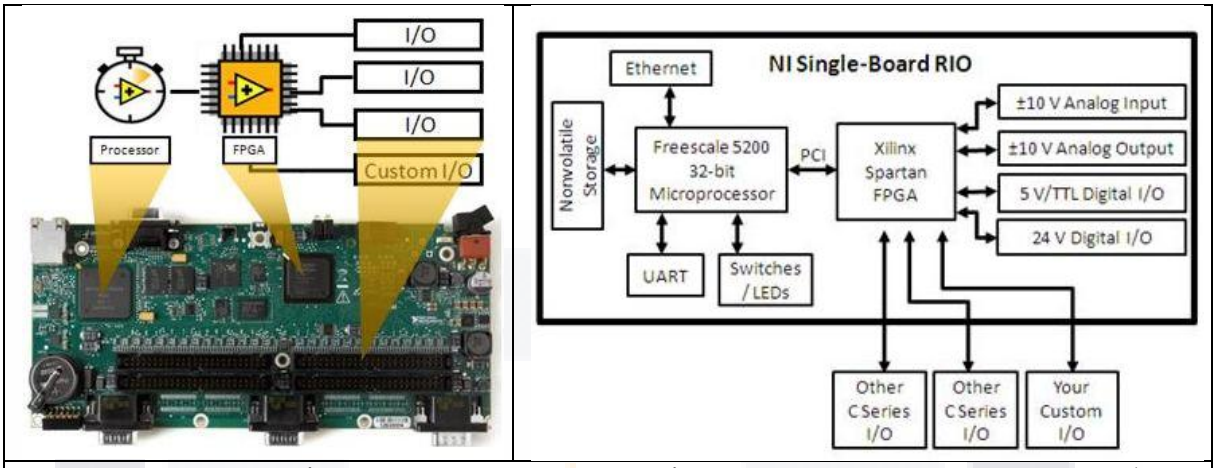


Figura 8. NI Single Board RIO y diagrama a bloques, extraída de Núñez, G. O. R. N. (2011).

Como se puede notar no hay un estándar en los diagramas o representaciones encontradas, pero es posible llevar a cabo una aportación de creación propia al capítulo, consecuencia de la lectura revisada para escribir el marco teórico de sistemas embebidos y su software. Para ello se elabora una síntesis, mostrada en la figura 9, con información de componentes generales que lo integran.

Una vez que se ha dejado claro lo que es un sistema embebido y sus componentes, es importante hacer referencia al dispositivo electrónico que se usará en el caso de estudio que se presentará en el capítulo 4.2. Por dicha razón el siguiente apartado muestra una reseña de lo que un cRIO es y las aplicaciones que tiene.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

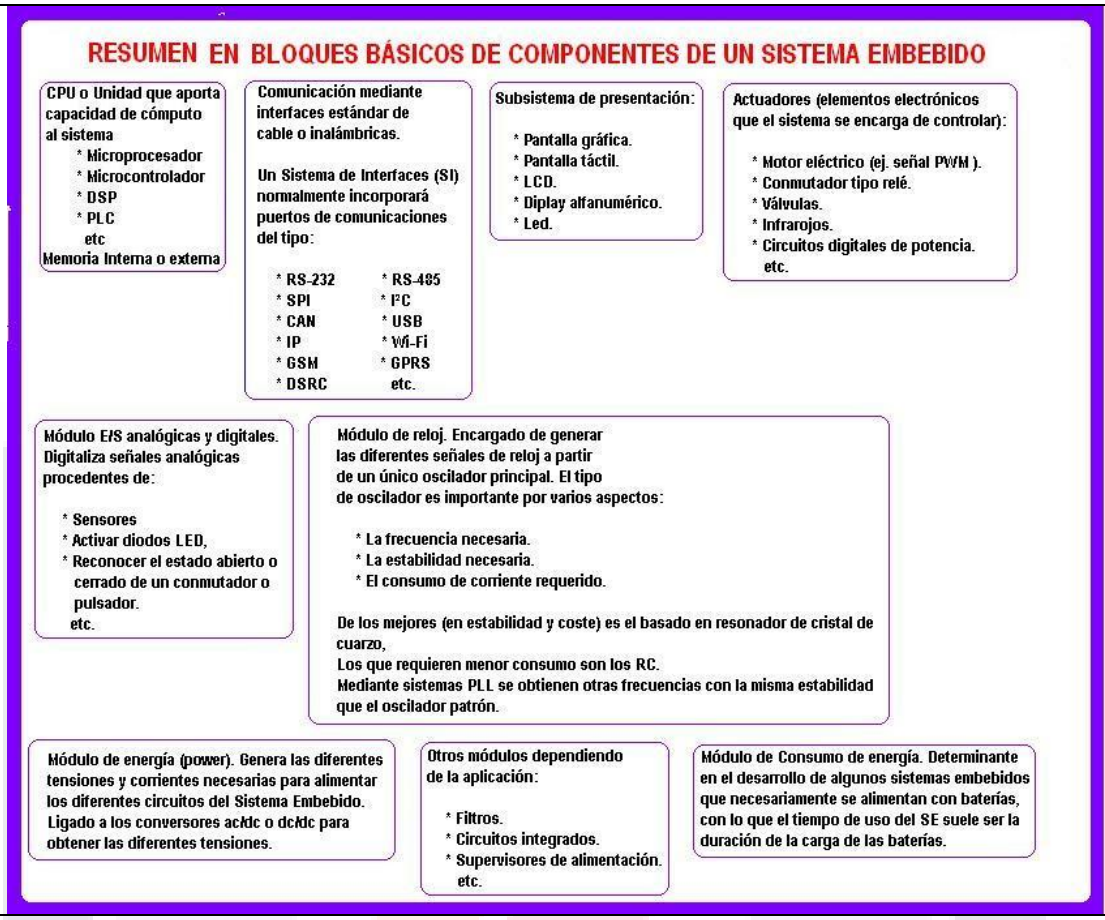


Figura 9. Componentes de un sistema embebido, aportación.

3.1.3.1 Introducción a NI cRIO-9074 y sus Aplicaciones.

Para justificar el caso de estudio de la presente tesis es necesario mencionar brevemente detalles de la empresa creadora del controlador integrado NI cRIO-9074: National Instruments (NI), así como enfatizar las importantes aplicaciones que actualmente usan este tipo de equipos y software.

El sitio oficial National Instruments (2013, A), indica que desde 1976 se ha comprometido con los ingenieros y los científicos equipándolos con herramientas que aceleren la productividad, la innovación e inventiva. Se explican que el diseño de los sistemas gráficos de National Instruments está enfocado a la integración del software y las plataformas de hardware para simplificar el desarrollo de cualquier sistema que necesite medición y control. Tanto ingenieros como científicos usan estas plataformas desde el diseño hasta el

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

uso en las líneas de producción en múltiples industrias, investigaciones avanzadas y proyectos académicos.

Su cofundador, presidente y CEO, Dr. James Truchard junto con su cofundador y miembro de Tecnología de Negocios, Jeff Kodoskynos, comparten que la visión a largo plazo y el objetivo de esta compañía es mejorar la sociedad a través de su tecnología, lo cual le ha dado la fortaleza para clientes, empleados, proveedores y accionistas.

National Instruments reporta en el 2012 Ingresos por $1.14 MM, oficinas en más de 40 países, aproximadamente 6.850 empleados y como clientes a 35.000 empresas más los que se suman atendidos anualmente.

Dentro de la amplia gama de productos que ofrece National Instruments se encuentra NI cRIO-9074, que es un controlador Integrado en Tiempo Real de 400 MHz y FPGA de 2M de Compuertas.

Las especificaciones proporcionadas por el documento pdf de National Instruments (2013, B) indican que el sistema integrado cRIO-9074 combina un procesador en tiempo real y arreglos de compuertas programables en campo (FPGA's) reconfigurables en el mismo chasis para aplicaciones embebidas de monitoreo y control de máquinas. Integra un procesador industrial en tiempo real de 400 MHz con un FPGA con compuertas de 2M, y tiene ocho ranuras para módulos de E/S de la Serie C de NI. Para aplicaciones robustas, el cRIO-9074 ofrece un rango de temperatura de operación de -20 a 55 °C junto con un rango de entrada de suministro de potencia de 19 a 30 VDC. Este sistema tiene 128 MB de DRAM para operación embebida y 256 MB de memoria no volátil para registro de datos.

Figura 10. NI cRIO-9074, tomado de National Instruments (2013, C).

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

El cRIO-9074 tiene dos puertos 10/100 Mb/s Ethernet que se pueden usar para llevar a cabo comunicación programática en la red y Web integrada (HTTP) y servidores de archivos (FTP) así como para añadir expansión y E/S distribuida al sistema. Por ejemplo, se puede usar un puerto Ethernet para comunicación en red a un servidor o sistema empresarial y el otro puerto para expansión de E/S (conectando fácilmente otro sistema CompactRio u otro dispositivo basado en Ethernet para E/S adicional).

El sistema operativo compatible para estas unidades es VxWorks y el Software compatible para su programación es LabVIEW, LabVIEW FPGA Module, LabVIEW Professional Development System y LabVIEW Real-Time Module

En referencia al software embebido de este sistema, National Instruments (2013, C) indica que se puede sincronizar el código embebido en ejecución a una interrupción (IRQ) en el FPGA o en un milisegundo interno de la fuente del reloj en tiempo real. El sistema operativo de LabVIEW en tiempo real (RTOS) proporciona confiabilidad y simplifica el desarrollo de aplicaciones embebidas complejas incluido el tiempo crítico de control y la adquisición en los bucles dando prioridades a estos lazos para su procesamiento posterior en los registros de datos y la comunicación Ethernet/serial. Empotrando funciones de I/O elementales tales como funciones de lectura y escritura de FPGA proporcionan una interface de comunicación altamente optimizada a los circuitos reconfigurables del FPGA. Los valores de los datos son leídos del FPGA en un formato de entero y son entonces convertidos a unidades de escalas ingenieriles en el controlador.

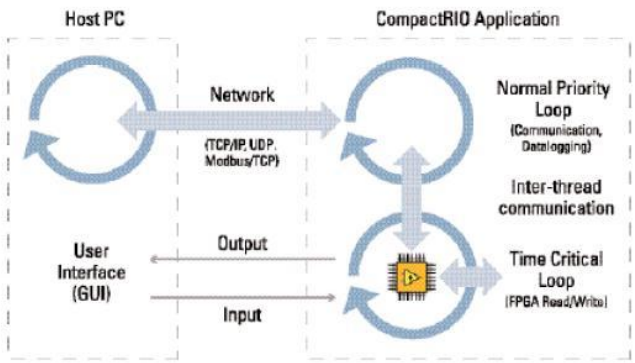


Figura 11. Arquitectura de Software CompactRIO, extraído de National Instruments (2013,C)

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

La figura anterior muestra el diagrama de la Arquitectura de Software usada por el NI cRIO-907x.

En conclusión National Instruments (2013) menciona ofrecer herramientas de sistemas gráficos para ingenieros y científicos que desarrollan sistemas de control y monitoreo de la próxima generación en industrias como energía, control industrial, ciencias biológicas y transportación. Con el hardware de E/S Reconfigurable, RIO, y el software NI LabVIEW de diseño de sistemas, los equipos pequeños de diseño pueden generar prototipos y desplegar más rápido sistemas de control y monitoreo embebidos.

Pero con un enfoque a las aplicaciones más comunes listadas por National Instruments (2013) resumimos el siguiente listado:

1. **Energía Renovable y Eléctrica:** Los ingenieros y científicos en todo el mundo utilizan sistemas embebidos para producir un impacto positivo en el ecosistema global. Los sistemas de hardware robustos y de alto rendimiento y el software resuelven cualquier tarea de control y monitoreo en la industria de la tecnología limpia, incluyendo sistemas de monitoreo en línea para aplicaciones eólicas y solares, control en tiempo real para potencia de electrónica y sistemas embebidos de medidas y análisis para calidad de la potencia.
2. **Petróleo y Gas:** Monitoreo y control avanzados para todos los segmentos de la industria de petróleo y gas son ampliamente usados; desde tecnologías de extracción y producción hasta fugas en tubería, fallas, seguridad e inspección. A través de una sola arquitectura, las compañías de exploración y producción (E&P) pueden tener una vista sistemática de la operación y atender los diferentes requerimientos de datos de varios departamentos y funciones en el trabajo.
3. **Maquinaria Industrial y Control:** Al diseñar máquinas de alta precisión, los equipos tienen el reto de reducir el costo de desarrollo, incrementar la productividad de las máquinas y acortar los tiempos de diseño. Hoy en día, los equipos líderes en diseño adoptan hardware de E/S reconfigurable y software de desarrollo para integrar E/S, visión y movimiento sincronizados en una plataforma de diseño que les ayuda generando máquinas más inteligentes.
4. **Monitoreo y Control para Transportación y Equipo Pesado:** Varias aplicaciones en las industrias de transportación

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

requieren algún tipo de monitoreo y control embebido. Puede requerir usar la rápida generación de prototipos de control para unidades de control de motores (ECU), monitoreo de condición de máquinas de los componentes de vehículos, simulación hardware-in-the-loop (HIL) y registro de datos portátil. El hardware robusto y modular ofrece la flexibilidad para cumplir con las necesidades específicas del proyecto de una manera confiable y eficiente.

5. . **Monitoreo y Control para Ciencias Biológicas:** La diversidad del campo de las ciencias biológicas se refleja en una amplia variedad de aplicaciones. Ingenieros y científicos usan hardware y software para desarrollar soluciones altamente flexibles, escalables y rentables; desde biotecnología hasta pruebas de dispositivos médicos y desde instrumentación analítica hasta monitoreo fisiológico. Con estas herramientas los científicos pueden realizar la investigación a tiempo y los ingenieros pueden reducir el tiempo en el mercado al rápidamente generar complejos prototipos de diseños de dispositivos.
6. . **Monitoreo y Control para Monitoreo de Condición:** El monitoreo de condición de máquinas puede ayudar a su organización a evitar interrupciones inesperadas, optimizar el rendimiento de máquinas y reducir tiempo de reparación y costos de mantenimiento. Con los sistemas de monitoreo basados en registro de condición embebida en línea se puede comprender mejor la condición operativa y vitalidad del equipo. El hardware y software son usados en sistemas de monitoreo de condición desplegados en una variedad de turbinas, compresores, generadores y otras máquinas industriales.
7. **Monitoreo y Control para Robótica:** Robótica es una de las áreas de ingeniería más complejas y de mayor crecimiento. Una plataforma de desarrollo intuitiva para diseñar desde vehículos autónomos hasta brazos industriales de base fija es bastante útil. Al diseñar aplicaciones complejas de robótica, se es más productivo usando entornos de desarrollo gráficos con un alto nivel de abstracción para la comunicación de sensores, algoritmos de cinemática y autonomía, control de motores y más.

A través de estos ejemplos de aplicaciones proporcionados por National Instruments (2013) es posible percatarse de la importancia del desarrollo de sistemas embebidos eficientes, ya sea bajo los estándares y productos generados por años en

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

National Instruments o bien en general para la creación de nuevos proyectos y arquitecturas en el campo del control y la adquisición de datos con sistemas embebidos.

Empero es importante destacar que a pesar de la amplia información de los productos, herramientas mencionadas y soporte otorgados para el uso de estos, poco es mencionado en la página oficial de NI acerca del proceso de requerimientos del sistema o de alguna guía que funja como soporte al desarrollo del proyecto ayudando a documentar también la arquitectura y el diseño.

Al respecto se elabora una encuesta breve a los ingenieros de diseño en el grupo de Manufacturing Technology Engineering (MTE) para verificar las herramientas, métodos, técnicas y/o metodologías usadas en sus trabajos específicamente para el control del proceso de desarrollo de software usando el cRio, y en general su panorámica en el desarrollo de software embebido basados en su experiencia.

Los resultados obtenidos se mostrarán a continuación usando el formato de pregunta y su respectiva semblanza de respuestas otorgadas por cada uno de los Ingenieros involucrados en el equipo de trabajo (tres integrantes) con dominio en el área de software y hardware respetando el orden en el cuál fueron recibidos los cuestionarios y cabe hacer énfasis en que cada ingeniero respondió de manera independiente su cuestionario y en diferente periodo.

1. Hablando de cRIO, ¿Considera que "las herramientas" proporcionadas para "la exclusiva programación del dispositivo (FPGA integrado en cRIO)" están sólo enfocadas al desarrollo y diseño del software embebido y/o interfaces o también contienen apartados para llevar a cabo el control de versiones, modificaciones, mantenimiento, debug, test y/o control de requerimientos del software. En caso de no saberlo o no usarlas, no omita la información.

Respuesta Ingeniero A:	National Instruments cuenta con un control de versiones de software, pero es pobre en sus características. Se ha intentado en el área de MTE usar software del sistema pero no se han aplicado formalmente para el control

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

	de versiones.
Respuesta Ingeniero B:	El software que utilizamos para la programación de este dispositivo es Labview, el cual cuenta con control de versiones, modificaciones, mantenimiento y debug, en cuanto al control de requerimiento, desconozco si lo tenga.
Respuesta Ingeniero C:	Las herramientas proporcionadas para la programación del dispositivo FPGA no consideran las opciones de control de versiones, modificaciones , debug, test y control de requerimientos dentro del ambiente de programación del FPGA, estas se encuentran disponibles como herramientas opcional que pueden ser adquiridas de manera separada según las necesidades.

2. Cuando es requerido el desarrollo de una herramienta que lleva implícito el desarrollo de software embebido ¿En qué se basa para la selección del tipo de dispositivo electrónico que se va a programar: En la petición del cliente, en base a un cuestionario de requerimientos del proyecto o sistema, basado en la experiencia profesional u otro?

Respuesta Ingeniero A:	La selección del dispositivo está basada en la experiencia e importancia para el Ingeniero desarrollador. Se toma en consideración, bajo la experiencia, verificar la conectividad, los módulos que se requieren para entradas y salidas calculando una holgura de expansión a futuro en un 25%, selección de herramientas a usar para la programación de National Instruments como VHDL para

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

	FPGA. Se calcula, basado en la experiencia, la cantidad de memoria que podría consumirse al desarrollar el programa. Y aunque al parecer NI tiene una herramienta para medir los tiempos de ejecución en el procesador me baso en mi propia experiencia de medición de los ciclos en otros programas desarrollados para determinar en forma más confiable y real el tiempo de ejecución que tendría en el nuevo proyecto.
Respuesta Ingeniero B:	Se utiliza un RFQ con el cliente en cuanto a restricciones mecánicas (espacio), o de costo, también se lleva a cabo un cuestionario de los requerimientos de funcionamiento del equipo. Paso siguiente nos apoyamos en los proveedores que tenemos relación de trabajo para elegir el mejor equipo de acuerdo al requerimiento proporcionado.
Respuesta Ingeniero C:	El orden es: Petición del cliente (si esta existe), Cuestionario o documentación generada para el proyecto, experiencia profesional

3. ¿En su área se usa actualmente algún modelo de procesos o formato para el seguimiento a los requerimientos del proyecto, la documentación de los requerimientos y la arquitectura del desarrollo del software (más que de hardware)?

Respuesta Ingeniero A:	Específicamente para el software no. Pero para los requerimientos del sistema completo se usa una R.F.Q. (Request for Quotation) para darle inicio al requerimiento

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

	del proyecto. Sólo que no es muy funcional porque casi siempre el cliente la llena mal, omite información dejándola escueta o en el peor de los casos el cliente ni siquiera contesta todo el documento porque le parece largo o porque no entiende algunas preguntas.
Respuesta Ingeniero B:	Se utiliza un SOW para el seguimiento del requerimiento y una FAT para comprobar junto con el cliente que se logró lo requerido.
Respuesta Ingeniero C:	No, es de nuestro conocimiento la existencia del software Requirements Gateway de National Instruments sin embargo esta herramienta no es aplicada en nuestros desarrollos.

4. En referencia con otros equipos de trabajo con los que haya tenido la oportunidad de compartir experiencias de desarrollo de software embebido ¿han compartido y/o comparado los modelos/metodologías/métodos/documentación que ellos usan con los propios usados en el área donde participa para el control de requerimientos y arquitectura de software? ¿estos modelos/metodologías/métodos/documentación cumplen algún estándar internacional o nacional?

Respuesta Ingeniero A:	Para los requerimientos de software, ya sea para un fixture complejo o simple, hechos al departamento de MTE no hay un proceso rigurosamente formal de llevarlos. Al considerarnos el cliente como los "expertos", nos permiten que el Ingeniero a cargo proponga la mejor solución al problema. A pesar de que no se comparte información con otros equipos de trabajo en

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

	cuanto a técnicas o metodologías, durante mi desarrollo profesional he clasificado de cada proyecto la información vital y tomado las mejores prácticas ya sea de clientes o proveedores armando un formato (FVS Proposal) para proyectos grandes solamente. Si el proyecto es pequeño basta llenar otro formato (T.A.), sobre todo si el cambio es sólo para el módulo de software. O al menos recibo un correo donde me explican que desean que cambie. Lo que a este punto me ayudaría en la parte de software es encontrar alguna forma de hacer mandatorio un "Guide line" del software diseñado para dejar muy claro que todo parámetro modificable no debe de ser hired wired, ya que esto en ocasiones se vuelve un dolor de cabeza al momento de una modificación.
Respuesta Ingeniero B:	No en ambas preguntas.
Respuesta Ingeniero C:	No, únicamente la información que se ha compartido con otros equipos de trabajo ha sido únicamente conversaciones técnicas al respecto.

Al realizar una comparación a las respuestas es notable que la forma de trabajar de los miembros del equipo es similar, bajo el mismo entendimiento del proceso para los requerimientos del diseño, pero posterior a ello cada uno se basa en su experiencia para llevar adelante el proyecto y su experiencia les ha llevado a observar cuidadosamente las herramientas disponibles en cuanto a requerimientos y diseño de software en forma diferente pero funcional.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

3.2 DESCRIPCIÓN DE PRINCIPALES ESTUDIOS RELACIONADOS.

Con la finalidad de entender en paralelo los estudios relacionados en ambas áreas de conocimiento, se presentarán dos apartados: 3.2.1 abarcará las investigaciones en el área de ingeniería de Software, con el uso de modelos de procesos, mientras el apartado 3.2.2 hablará de las investigaciones relacionadas con técnicas o métodos usados para el desarrollo de software embebido.

3.2.1 Investigaciones relacionadas a la aplicación de estándares para Modelos de Procesos de Software en MiPyME, PyME o VSE.

En la actualidad se ha encontrado reportado en la literatura el problema de adaptación de los estándares para Madurez de Procesos y Mejoras al Proceso de Software en las pequeñas, medianas y micro empresas. En la búsqueda de soluciones a este problema se le ha dado seguimiento a varias implementaciones de perfiles bajo proyectos piloto basados en guías de estándares como los de ISO/IEC 29110, JTC1/SC7 WG24 (O'Connor, R. V., & Laporte, C. Y. 2011) que han sido completados en varios países tales como Canadá (4 sujetos), Bélgica (25 sujetos), Francia (14 sujetos), Irlanda, también hay aún varios en curso en todos los países citados. La investigación reportada por O'Connor, R. V., & Laporte, C. Y. (2011) se llevó a cabo para mostrar los beneficios potenciales del uso de estándares, los cuales están específicamente diseñados a las necesidades de industrias pequeñas. A estas industrias se les denomina VSE (Very Small Entities) y las guías de los estándares están enfocadas a cubrir las necesidades de estas, a través del trabajo y el esfuerzo que se está realizando por el grupo desarrollador de guías en ISO/IEC, JTC1/SC7 y WG24 enfocadas a las pequeñas empresas para aplicar sus estándares para el desarrollo de software.

En O'Connor, R. V., & Laporte, C. Y. (2010).se encuentra expuesto el seguimiento que se le da a los proyectos pilotos desarrollados y en curso para obtener información de las pequeñas empresas y saber cómo debe llevarse la implementación de estándares en estas industrias (como ISO 12207 y 29110). Los grupos piloto de pruebas en Canadá, Irlanda, Francia y Bélgica principalmente muestran que hay una clara necesidad de integrar a las pequeñas empresas o VES´s a procesos estándar para la creación de software ya que se les considera como industrias que aportan a la economía de

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

un país en forma significante. El uso de guías, llamadas reportes técnicos, que son monitoreadas por paquetes de implementación desarrollados para el ISO 29110 se enfoca a cubrir esa necesidad.

Bajo la definición formal de lo que es una VSE: "Empresa constituida por no más de 25 empleados", tomada de O'Connor, R. V., & Laporte, C. Y. (2010, June), se exponen algunas de las razones por las cuales no adoptan modelos como CMMI las VSE, intentando ofrecer una alternativa a través de un estándar más ligero con ayuda de los paquetes de implementación.

Se presentaron nuevos comentarios ad hoc en el proyecto de Laporte, C. Y., et al. (2008) que intenta facilitar el acceso hacia la utilización y la utilización misma de los estándares de ingeniería de software de ISO/IEC JTC1/SC7 en la pequeña industria (VSE). La investigación fue desarrollada con la colaboración de empresas desarrolladoras de software en países de Latinoamérica, América del Norte y Europa. El compromiso hecho en mayo del 2004 en una junta en Canadá donde se planteó la problemática de adopción de estándares en la pequeña industria de desarrollo de software, dio pauta a la inquietud de conocer los motivos de la renuencia de adopción de los estándares. Dada la importancia que la pequeña industria tiene en la generación de software y el impacto que pueden llegar a tener en manufactureras que usan sus servicios, se está de acuerdo en que debe garantizarse la calidad y estandarización en la generación de software en las pequeñas empresas (VSE).

La investigación en Ribaud, V., (2010) trata la adaptación de los estándares a las VSE enfocándose en explicar los resultados de la pregunta específica de los programas piloto desarrollados:

"E-44 ¿Qué procesos de infraestructura son apropiados para soportar las nuevas tecnologías e ingeniería concurrente?"

A través de la aplicación del cuestionario en una empresa Canadiense con trece personas laborando en ella, seleccionada por que pidió soporte para mejorar su proceso de desarrollo de software. Se probaron dos hipótesis: la primera referente a que se puede llevar a cabo un auto entrenamiento en algunas actividades del proceso y segundo que el estándar 29110 requiere de más documentación de soporte. En este caso práctico se ha aplicado el análisis de mapeo de los

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

estándares ISO enfocados a la Infraestructura y Proceso de Soporte de la empresa para lograr llevar a cabo la construcción de una tabla en la cual se resume una propuesta para dicha Infraestructura y Proceso de Soporte para cualquier VSE que se dedique a la generación de software.

Uno de los modelos presentados por Laporte, C. Y., etal (2008) es MoProSoft que se enfoca en tratar de resolver esta problemática en la ciudad de México. Lo cual alienta a mencionar su importancia.

Oktaba, H., etal (2007) se refiere a la historia y trayecto que ha seguido el proyecto de MoProSoft para llegar a lo que se conoce como CompetiSoft. Usa estudios de Laport para evidenciar y sustentar terminología como VES y puntos que hacen que las empresas rechacen los estándares. Menciona que CompetiSoft está usando EvalProSoft para evaluar que el modelo propuesto con MoProSoft da resultados en la industria. La investigación obtiene cooperación de países como Australia, Bélgica, Canadá, República Checa, Finlandia, India, Irlanda, Italia, Japón, Corea, Luxemburgo, México, Sudáfrica, Tailandia y el Reino Unido. La autora actualmente trabaja en México con el monitoreo en seis industrias nacionales. El proyecto nació de la necesidad de implementar estándares en las pequeñas empresas bajo normas y costumbres nacionales pero basándose en las mejores prácticas internacionales de desarrollo de software. El modelo de MoProSoft tiene vínculo con otros estándares, relación con CompetiSoft y EvalProSoft con lo que da a conocer la estructura básica y datos importantes alrededor de este trabajo. MoProSoft ha logrado implementarse como una norma mexicana NMX-059-NYCE-2005 para trascender como una incorporación a la norma internacional ISO/IEC. Que funge como base del proyecto de CompetiSoft que se encuentra en desarrollo.

En forma resumida el objetivo en el cuál coinciden estos trabajos de investigación es en llevar a cabo esfuerzos conjuntos desde 2004 a la fecha para tomar lo mejor de los estándares internacionales, enfocándolos a las necesidades de las VSE, dejando de ser costosos, difíciles de implementar y laboriosos o burocráticos, facilitando los documentos en forma gratuita.

Sin embargo hay algunas áreas que no se han investigado, a continuación se redactan algunas de las observaciones hechas por algunos autores.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

Iniciado con el trabajo de O'Connor, R. V., & Laporte, C. Y. (2011) donde mencionan que hay mucho aun por aprender de los proyectos piloto. Principalmente que existe la necesidad de una ligera y flexible aproximación de una valoración de proceso actualmente en construcción sobre un mecanismo de valoración para ISO/IEC 29110. Siempre bajo el objetivo de asegurar que las VSE's no requieran invertir de ninguna manera en forma similar en términos de tiempo, dinero y otros recursos en el proceso de valoración, tal como lo hacen las SME (Small and Médium Enterprises), o las MNC (Multinational Corporation). Se sugiere que sean autoevaluadas con posibilidad de tener soporte por Internet basándose en herramientas. Repensar en el proceso de valoración o evaluación con nuevos métodos e ideas para el proceso en las VSE's sería un campo de innovación según el autor.

Por otra parte O'Connor, R. V., & Laporte, C. Y. (2010) mencionan tener mucho por hacerse en el estándar ISO 29110, ya que es completamente nuevo y tiene áreas no exploradas tanto para la adaptación a otros perfiles, áreas y detalles.

En el caso de Laporte, C. Y., etal (2008) se habla de que aun hay perfiles de VSE que no han sido cubiertos del todo y que hay mucho por completar en los paquetes de implementación que se están desarrollando a través del grupo de WG24.

Finalmente en Oktaba, H., etal (2007) se muestra que el trabajo fututo pretendido es la retroalimentación de las empresas que están usando el modelo MoProSoft se puedan mejorar los procesos y completar si hay aun algún hueco por cubrirse.

3.2.1.1. Reportes comparativos de modelos.

Una de las literaturas que vuelve a marcar la importancia de generar modelos para la pequeña industria de software es Mishra, D., & Mishra, A. (2009) donde los autores toman resultados de una encuesta basada en la comparación de seis metodologías SPI (Software Process Improvement) aplicada en SME's, y que se resumirán a continuación:

1. *Modelo OWPL:* Un enfoque gradual para la mejora en procesos de Software en SME's. Tiene 3 estados a cubrir:

    1) Micro encuestas;
    2) Evaluación OWPL:
    3) SPICE valoración.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

En general se enfoca en hacer un diagnóstico por encuestas de desarrollo de procesos y determinar lo que debe trabajarse para acercarse a una certificación.

2. *Modelo Software Process Matrix (SPM):* Este modelo es el armado de matrices partiendo de los requerimientos del cliente, así mismo ver la relación que hay entre los procesos y prácticas.

3. *Aproach for Software Process Establishment in Micro and Small Companies (ASPE-MSC):* Respecto al enfoque que se maneja para el establecimiento en micro y pequeñas compañías está definido por la integración y adaptación de enfoques existentes. Las principales fases de la valoración son:

i) Planeación:
   (a) Fase 1: Diagnóstico.
   (b) Fase 2: Análisis Estratégico.
   (c) Fase 3: Definición.
ii) Monitoreo y Control.
iii) Tratamiento después de terminado el proceso.

4. *PRISMS:* Esté enfoque de mejora en los procesos de software para pequeñas y medianas empresas, es una acción de proyecto de investigación de tres investigadores de la Universidad Leed Metropolitan. Se trabaja en conjunto con desarrolladores y administradores de proyectos en varias compañías en la parte de planeación e implementación de programas para el mejoramiento de proceso de software. Por un periodo de tres años. Los puntos clave son:

1) Examinar proceso informal para crear modelo.
2) Definición de metas del negocio por el administrador.
3) Ejercicio de consulta con todos los miembros del equipo para adaptar un SPI.
4) Se consulta CMM pero sólo para definir las KPAS para la mejora.
5) De esa consulta para KPAS se da prioridad e identifica lo que es útil basado en las metas del negocio.
6) Medir es parte integral de SPI, se puede hacer una métrica en forma de atributos de las metas.
7) El plan SPI se revisa periódicamente.

5. iFLAP: Definido como el mejoramiento del marco de trabajo utilizando asignación de pesos y planeación de mejoras, se

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

usa para evaluar una sola área de proceso o a todas, son 3 pasos principales:

Paso 1 Selección.
Paso 2 Asignación.
Paso 3 Mejoras en la Planeación.

6. MESOPYME: Es un modelo general de SPI definido por ISPI Ribaud, V., (2010). en cuatro estados cuyos objetivos son similares a los modelos Ideales de SEI. Los puntos importantes son:

Stage 1: Consigna para la Mejora.
Stage 2: Análisis de Proceso de Software (CMM Capability Maturity Model).
Stage 3: Soluciones de Mejora.
Stage 4: Institucionalización.

En el artículo de investigación de Paternina P., K., etal (2011), también tenemos una comparativa que aborda las características generales del modelo más famoso: CMMI (Capability Maturity Model Integration) creado por el SEI de la Universidad Carnegie Mellon, pero a pesar de ser un modelo bien definido y de que se ha probado su efectividad se considera un modelo robusto nada fácil de implementar en la pequeña o mediana industria. El modelo en general consta de 5 niveles de madurez con 5 niveles de capacidad. El objetivo de CMMI es lograr la mejora de calidad en procesos y productos diciendo Que hacer, pero no Cómo hacerlo. El modelo se basa en las constelaciones de CMMI para implementarse, estas son:

1. CMMI-DEV publicada en agosto del 2006 y sirve como guía para medir, monitorear y administrar el proceso de desarrollo y mantenimiento de productos y servicios.
2. CMMI-ACQ (Acquisition) publicada en noviembre del 2007 y sirve como guía para mejorar el proceso de adquisición de productos y servicios.
3. CMMI-SVC (Services) publicada en febrero del 2009 y sirve como guía para proporcionar servicios internos en una organización y a clientes externos.

La versión actual de la suite CMMI es la versión 1.3, liberada el 1 de noviembre de 2010.

Posteriormente se menciona el apartado que explica IT-Mark, que es el primer modelo de calidad internacional diseñado específicamente para las empresas del sector informático o

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

TIC (Empresas Desarrolladoras de Software). Orientado a la pequeña y la micro empresa de TI, pretende además de la mejora de procesos y productos, lograr la introducción de la empresa al mercado internacional garantizando su competitividad. Consta de 3 niveles de madurez:

1. I.T. Mark acredita a una empresa que es consciente de los problemas relacionados con la gestión técnica, de seguridad y del negocio, y que los mantiene habitualmente bajo control.
2. I.T. Mark Premium acredita a una empresa que ha conseguido una Buena Madurez en sus procesos de trabajo técnico, seguridad y del negocio. En este caso los niveles necesarios son considerablemente superiores a los descritos anteriormente, exigiéndose que todos los procesos evaluados desde los tres puntos de vista están razonablemente desarrollados.
3. I.T. Mark Elite acredita a una empresa que ha conseguido un nivel Superior en la Definición e Institucionalización de sus procesos de trabajo técnico, de seguridad y de negocio, por lo que se confía en que la calidad de sus productos sea buena, debido a la madurez de sus procesos y a la mejora continua.

Al final tiene un apartado en el que compara el modelo de CMMI contra IT Mark, mostrando el nivel de madurez similar que tienen. Después se define en otro apartado PSP (Personal Software Process) y TSP (Team Software Process), donde se menciona que estos están enfocados a las personas y los equipos para lograr el mejoramiento en el monitoreo del proceso.

Uno de los datos estadísticos presentados que resulta interesante se refiere a cómo se ha incrementado la certificación de industrias del 2003 al 2010 por apoyo de los programas que el artículo de investigación menciona. Así mismo es interesante que de las seis industrias certificadas con nivel cinco por CMMI, tres sean colombianas (la mitad). Aunque son más las certificadas con IT Mark.

3.2.2. Investigaciones relacionadas a la aplicación de estándares para Modelos de Procesos de Software en el módulo de Software en los Sistemas Embebidos.

Recordando que un sistema embebido es aquel que usa un dispositivo electrónico (microprocesador, microcontrolador,

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

FPGA, DSP, etc.) para almacenar un software que ejecuta una o varias tareas en base a entradas de los sensores o información manifestándose en los actuadores o salidas, es obvio visualizar que esto implica necesariamente buscar el tipo de software que irá empotrado (software embebido) en el dispositivo electrónico según la aplicación a la cual se destine el sistema.

Y agregando a la información que ya se había presentado en el marco teórico que al ser los sistemas embebidos conformados por hardware y software, Axelsson, J. (2011) discute en su investigación el tópico de la incertidumbre en el contexto de la arquitectura del software embebido y su sistema. Este trabajo presentan un vínculo entre la complejidad y la incertidumbre y las clasifica en diferentes tipos, además explica por qué la incertidumbre se incrementa en la arquitectura de sistemas con software intensivo y presenta 10 diferentes tácticas que pueden ser empleadas para lidiar con la incertidumbre y mitigar los riesgos asociados.

Bajo estos precedentes es requerido exponer los trabajos de investigación que indican los retos que se han tenido en el uso de métodos o metodologías para el desarrollo de sistemas embebidos y software embebido.

3.2.2.1 Uso de Metodologías y Planteamientos de Problemas para Creación de Sistemas Embebidos y Desarrollo de Software.

En el artículo de investigación elaborado por Chandy, J. C. (2010) se habla acerca de los sistemas ciber-físicos (o CPS, equivalentes a un sistema embebido) mencionando los componentes que lo conforman: computadores embebidos, monitoreo de redes y control de procesos físicos además de la forma en la cual se comunican para mantener su función a través de la retroalimentación para poder llevar a cabo cálculos. En este trabajo lo interesante es el análisis que se hace de los desafíos en el diseño de los sistemas ciber-físicos a través de un planteamiento de si la informática y las tecnologías de redes actuales proporcionan una base adecuada para ellos. La conclusión de esta investigación se resumen en que para mejorar los procesos de diseño de los sistemas ciber-físicos no será suficiente con elevar el nivel de abstracción o verificar, formalmente o no, los diseños en los que se basan las abstracciones de hoy, además se incluye la recomendación de que para aprovechar todo el potencial de los CPS se deben reconstruir los procesos de las

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

abstracciones informáticas y de las redes, y los procesos se deberán acoger en pleno a los principios de las dinámicas físicas y de la computación. Esto abre una ventana de posibilidades para considerar que hay modelos, métodos, metodologías o procesos esperando poder ser implementados en los sistemas ciber-físicos o embebidos.

En el caso e Ting, J. S. (2011) tenemos la continuación al tema de los CPS, pero en esta ocasión el enfoque es hacia el planteamiento de grandes desafíos en el diseño de software. Hace mención de no sólo diseñar un sistema en torno a los plazos de ejecución, sino también considerar el maximizar la utilización de los recursos. Explica la propuesta de un sistema de base para la arquitectura de software en el que los servicios se puedan diseñar e implementar y componer fácilmente de acuerdo con la demanda y mediante aplicaciones individuales, de manera que satisfagan requisitos específicos de confianza, seguridad, eficiencia, confiabilidad y previsibilidad, considerando siempre los límites de las capacidades del hardware determinado. Toca temas referentes al futuro del software en sistemas ciber-físicos como el superar los vacíos en la semántica, una propuesta para la arquitectura de software para que el sistema esté estructurado como una colección de servicios de componentes, que puedan ser aislados a través de la utilización de alguna técnicas de hardware y/o de software o combinados en un sólo espacio de dirección de acuerdo con requisitos bien definidos. Considera como las principales limitaciones de los sistemas ciber-físicos de hoy, desde una perspectiva del software de sistemas a que:

> 1) Hay una inconsistencia entre las necesidades de aplicación y los aportes de servicio del sistema debido a API's inadecuadas, lo que dificulta la posibilidad que las aplicaciones especifiquen con precisión el servicio que desean;
> 2) Los servicios del sistema tienden a ser aplicaciones agnósticas, ya que por lo general se centra en la equidad y la eficiencia en lugar del tiempo, la confianza, la seguridad, la fiabilidad y las limitaciones;
> 3) Los sistemas actuales son poco flexibles para ser fácilmente extendidos con servicios específicos para aplicaciones destino.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

Por otro lado en la conclusión del trabajo hace énfasis en los desafíos que enfrentan los sistemas de apoyo para CPS futuros como:

1) El diseño de una arquitectura de software subyacente que se organice a sí misma con los métodos más adecuados de comunicación y el aislamiento entre servicios;
2) La composición automática de servicios para satisfacer las limitaciones de la aplicación, teniendo en cuenta las limitaciones del hardware subyacente;
3) El diseño cuidadoso de API's y la consideración de la heterogeneidad de hardware en la generación y verificación de un sistema software para una aplicación dada.

Quizá el punto en este trabajo de investigación es que no hay una metodología definida, es más una guía de cuáles son los problemas que pueden ser atacados a través del diseño o propuesta de una metodología o modelo, lo cual da una vez más pauta a pensar que proponer la adaptación del modelo de MoProSoft en una categoría es viable.

Se ha visto hasta este punto la problemática que genera el no seguir un modelo o método para el diseño de sistemas ciber-físicos o embebidos, pero es importante denotar de que éste problema es grande y complejo para ser abarcado en toda su extensión. Así que centrando la atención en los trabajos desarrollados para el subsistema de software embebido tenemos las siguientes citas que se enfocan en este tema.

En el artículo presentado por González, P. L., & Ulrrego, G. G. (2008) se propone un modelo de requisitos como apoyo para la construcción de sistemas embebidos, bajo la consideración de que en la actualidad, las metodologías de Ingeniería de Requisitos propuestas para este dominio no establecen continuidad en su proceso de desarrollo, por poseer una fuerte orientación a la etapa de diseño y un énfasis más débil en la etapa de análisis. La falta de propuesta de herramientas; para la obtención de los requisitos da pauta a crear su modelo de requisitos a través de un caso práctico. En la Introducción mencionan las carencias encontradas en las metodologías de ingeniería de requisitos para sistemas embebidos, por lo cual se sugieren una serie de materiales y métodos presentando los conceptos básicos que permiten entender la estructura del modelo con la visión en dos

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

grandes áreas: la ingeniería de requisitos (IR) y los sistemas embebidos (SE). Reportan los pasos seguidos (identificación de las características propias de los sistemas embebidos, construcción de un modelo de interacciones de agentes, transformación y adaptación del modelo de requisitos Con base en las interacciones típicas) y prosiguen al caso de aplicación (aplicación un sistema de sensado de movimiento, inmerso en un sistema de alarma para hogar, que fue utilizado en la metodología ECSAM, propuesta en Lavi et al. 2005). El logro fue explotar y especializar las categorías del modelo de la metodología ABC-Besoins, y mostrando que a través de la obtención de subcategorías de requisitos propias obtenidas para los sistemas embebidos se da mayor expresividad al modelo y constituye una guía útil para la captura de los requisitos específicos de los sistemas embebidos. El resultado es otorga una nueva metodología denominada ABC-Besoins-SEM en la cual continúan trabajando al momento de reportar.

En este caso la contribución que se da es muy similar a nuestra propuesta, la diferencia es el modelo base que usa y que está enfocada específicamente a requerimientos pero de todo el sistema embebido, lo cual en nuestro caso no es de prioridad, ya que deseamos solo adaptar la parte de software embebido bajo el modelo de MoProSoft.

Otro caso que reporta aportaciones al desarrollo de sistemas embebidos está ubicada en Gastaldi, G. G., etal (2002, Abril)en donde presenta el uso de la herramienta de trabajo Matlab y su entorno de trabajo con metodología de Codiseño Hardware Software, cuyo uso es para calculo y programación en distintas disciplinas científicas y en particular en la electrónica. En él describen por apartados las características generales, herramientas y conceptos más importantes en la utilización de Codiseño HW/SW, obteniendo los criterios para una evaluación. También realiza un estudio de Matlab, su entorno de trabajo Simulink y herramientas de uso (tool boxes), describiendo las posibilidades de su utilización. El artículo agrupa una serie de características como rasgos del sistema y describe los controladores embebidos y su flujo, los niveles de abstracción relacionados con el desarrollo de sistemas digitales y puntos de vista o dominios de descripción del diseño, los cuales determinarán la forma en la cual se describen los componentes del sistema embebido. Basados en este marco teórico y aunado a la definición formal de codiseño HW / SW que dice "... es el proceso de diseño de un sistema que combina las perspectivas

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

hardware y software desde los estados primarios, para aprovechar la flexibilidad del diseño y la localización eficiente de las funciones." Se presenta una prueba conceptual para el diseño de un sistema embebido con Matlab. La contribución ayudaría en la propuesta de tesis a fijar un punto de partida para adaptar las tareas en una forma enfocada al diseño de desarrollo del software embebido identificando en el diagrama el punto de inicio ya que, como menciona el artículo, el codiseño HW / SW permite optimizar un diseño partiendo el sistema en componentes hardware y algoritmos software, considerando determinados factores de calidad partiendo de una especificación y diseñando de forma independiente a la arquitectura para posteriormente implementar el sistema. Algunas limitantes para este tipo de herramienta (Codiseño) son las limitaciones debido a indisponibilidad de herramientas adecuadas para el desarrollo de las distintas etapas pero que en nuestro caso no repercutirán por el perfil del tema a desarrollar en la propuesta.

En el caso de Gil, R. A. C. (2006) se propone el uso del método RUP de IBM basado en el método de espiral pero para desarrollo de software en general, se pueden tomar de él algunos puntos de cómo determinar la forma en la cual se debe empatar el desarrollo específico de software embebido y el modelo MoProSoft. Además de dar una recomendación de no usar modelo en cascada porque los problemas que presenta este modelo es que al ser una secuencia de grandes etapas que requieren como hitos la documentación completa antes de continuar con la siguiente etapa, ven necesario solucionarlo haciéndose uso de métodos iterativos e incrementales unidos a otras prácticas claves como la orientación al manejo de riesgos y la planeación adaptable para guiar en una forma más adecuadamente y natural el proceso de desarrollo de software. Tomando en consideración este detalle y al ser MoProSoft un modelo enfocado a las PyME o grupos pequeños de trabajo, consideramos una buena aportación a nuestro desarrollo.

Con base en esto podemos iniciar el siguiente apartado para conocer ahora que es lo que se tiene hecho para cubrir estos huecos en el desarrollo del software embebido mostrando los esfuerzos de colegas a través de su experiencia y análisis.

En el caso de Hernández, V. J. I. (2010) se presenta al software embebido como un tipo de aplicación muy particular en la ingeniería de software cuya aplicación en las diversas actividades del ser humano es notable en nuestros tiempos y

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

que posee características específicas que se demandan al construirlo. A través del artículo se realiza un análisis de los retos que se presentan durante su proceso de desarrollo tomando en cuenta factores que repercutan en una mala calidad del producto y del software embebido implementado, donde la confiabilidad es fundamental. Además se habla de las tendencias que se tienen a futuro en la mejoría de sus herramientas de diseño, así como en la cobertura de sus aplicaciones. Temas como la definición de software embebido, sus categorías, los ámbitos de aplicación, sus características, los retos al desarrollarlo, las tendencias en el desarrollo son abordados de una forma amena y general para dar una panorámica de la situación real acerca del software embebido.

También hace énfasis en que la industria del software y los sistemas embebidos cada vez tienen mayor auge y que mantienen una creciente presencia en las diferentes actividades del ser humano. Menciona que algunas de sus particularidades son su alta confiabilidad en su operación, recursos de hardware limitado y respuestas en tiempo real, lo que da como consecuencia aplicarse en la búsqueda de mejores, métodos, técnicas, herramientas y proceso de desarrollo que garanticen productos de calidad y con una amplia gama de aplicaciones y desarrollos tecnológicos. Hace hincapié en aunar esfuerzos entre ingenieros de software y de hardware al ser un trabajo interdisciplinario que incluye otras ramas de la ingeniería como eléctrica, electrónica, mecánica, mecatrónica, biología, etc. La recomendación es alentar la búsqueda de mejores herramientas de diseño, ampliar la cobertura de aplicación y tendencia a mejorar las prácticas para su construcción.

En Ruiz, M. B. (2010) se propone el uso de una herramienta para generar software embebido bajo la óptica de que cada vez es más frecuente el uso de modelos para describir el software de los sistemas embebidos. Aplicando un caso práctico de sistema ejemplifica que en sí los modelos no están pensados para visualizar código sino para representar un sistema con un nivel de abstracción superior al de los lenguajes de programación. El artículo indica que la migración de una metodología de programación hacia el desarrollo de software basado en modelos, supone un incremento en nivel de abstracción y en productividad similar al producido al cambiar desde lenguaje de bajo nivel hacia lenguaje de alto nivel. Hace también énfasis en cómo los modelos ayudan a comprender el sistema a diseñar y cómo se favorece el intercambio de ideas entre los miembros del equipo, de diseño

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

y sus clientes, con la ventaja de que algunos modelos permiten simular el funcionamiento de un sistema desde las primeras etapas del diseño abriendo la posibilidad de generar automáticamente el código fuente y la documentación del proyecto. Esto garantiza a lo largo del desarrollo la conexión entre modelo, código y documentación. También menciona que la práctica del uso de modelos para el desarrollo de software no está muy extendida, lo cual da como resultado el afán de esforzarse en que se impulse el despegue de metodologías de programación, como la que propone a través del uso de statecharts y el software basado en UML IAR Visual-State que es un entorno gráfico para diseño, verificación e implementación de sistemas embebidos basados en máquinas de estados jerárquicas o statecharts. El artículo explica en forma secuencial lo que es un modelo y las características generales que se esperan de este basado en estudios de las mejores prácticas entre ingenieros y técnicos resumiendo que deben ser abstractos, comprensibles, precisos, predictivos y económicos. Por otro lado indica en caso reportado que enfoques actuales en el desarrollo de software basado en modelos coinciden en: utilizar una representación gráfica del sistema a desarrollar, describir el sistema con un cierto grado de abstracción y generar código ejecutable para el sistema embebido partiendo del propio modelo. Se muestran como ventajas del diseño dirigido por modelo los statecharts por ayudar a construir modelos gráficos que describan con precisión el comportamiento de los sistemas. Indican que esto ayuda a obtener el funcionamiento deseado de un sistema y que aunado a la representación gráfica del modelo se obtiene en consecuencia comunicación e intercambio de ideas entre los clientes y el equipo de desarrollo del sistema, independientemente del tipo de formación que posean y sin necesidad de construir un prototipo hardware.

Para la aplicación que hacen del caso práctico hacen uso de seis pasos determinando así las especificaciones del sistema, las cuales son enumeradas a continuación:

1. Identificar los eventos y las acciones
2. Identificar los estados
3. Agrupar los estados por jerarquías
4. Agrupar por concurrencia
5. Añadir las acciones y transiciones
6. Añadir las sincronizaciones

Aunque no queda muy claro si está hecho bajo la metodología de UML, pero si menciona que para el desarrollo de sistemas

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

hasta ese momento se exige la utilización de herramientas de diseño de elevada productividad como: lenguajes de alto nivel, asistentes o bibliotecas para la inicialización de periféricos, statecharts, sistemas de generación automática de código, middleware, entre otros. El hecho de que se eligiera para el desarrollo del artículo a los statecharts se debe a que están situados en un nivel de abstracción superior al de los lenguajes de programación, logrando trasladar el diseño al dominio de la aplicación usando sus gráficos con enorme capacidad descriptiva que los hacen fáciles de interpretar que los listados de código y de compartir con el equipo de desarrollo independientemente del perfil de este.

En el caso de la investigación presentada por Prakash, V. C., Sastry, etal (2011) proponen el modelado de software embebido a través del uso de Clear Box Structure (CBS) proveniente de Cleanroom software Engineering (CRSE). Basados en los reportes generados de otras técnicas tales como: modelo basados en generación de código, aproximaciones por implementación de concurrencia, el Framework StreamIT basado en flujo de datos, el Real time workshop (RTW) que considera la generación de código basado en un modelado de sistema de control, propuestas de evaluación parcial de métodos que ayuden a la automatización del diseño y generación de código, el uso de métodos para optimización de la generación de código basado en el modelo de tarjetas de código preservando el modelo semántico, la generación de un código C basado en los modelos, uso del método de SystemC TLM Model especificaciones de UML, uso del sistema SustemC TLM Model, SystemC RTL Model más enfocado a código FPGA RTL en lenguaje VHDL para generar netlist ASIC y la propuesta de uso de diagramas de estado. Concluyen que ninguno de estos logra enfocar consideraciones para software y hardware en sistemas embebidos en lo que respecta a generación documentada de código. A través de un caso práctico de simulación muestran el uso de CBS propuesto ilustrando como el flujo es considerado para el avance de desarrollo paralelo de software y hardware necesario para la creación de software embebido.

En Scaife, N., etal (2008, Abril) se presenta el planteamiento de una metodología llamada “Hume” para el desarrollo de software que es un lenguaje experimental, pero se enfoca a los costos por la construcción para el diseño y verificación en aplicaciones embebidas. Mas no menciona un modelo a seguir para el desarrollo, lo que se pretende es mostrar cómo este lenguaje apoya al diseño del software embebido siendo un lenguaje de alto nivel y permitiendo notar

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

la disminución de costos y llevar el análisis correcto para la implementación del sistema en tiempo y espacio que es un factor clave de control en los sistemas embebidos.

Se ha dejado al último el análisis del libro de Berger, A. H. (2002) dada su relevancia en nuestra investigación. El libro en general se presenta como una guía basada en la experiencia del autor como Ingeniero en la IBM desarrollando sistemas embebidos y con la cual pretende facilitar el estudio de los mismos tanto por alumnos como por Ingenieros en campo.

Berger, A. H. (2002) considera que las diferencias entre un sistema computacional y un sistema embebido se resume en que los sistemas embebidos:

1. Son dedicados a tareas específicas.
2. Soportados por un amplio arreglo de procesadores y arquitecturas de procesadores.
3. Son usualmente sensibles al costo.
4. Tienen restricciones de tiempo real (externo) [restricciones sensibles, restricciones críticas].
5. Usan sistemas operativos en tiempo real (RTO's).
6. Las fallas de software son más severas que en sistemas de escritorio.
7. Tienen restricciones de alimentación (calor, ventilación, CPU, RAM, ROM, I/O).
8. Continuamente deben operar en extremas condiciones ambientales.
9. Tienen mucho menos recursos de sistema que los sistemas de escritorio.
10. Continuamente almacenan todo su código objeto en ROM.
11. Requieren herramientas especializadas y métodos para ser diseñados eficientemente.
12. Continuamente tienen un circuito debbuging dedicado.

Un dato interesante mencionado es que la primera tarea de un estudiante de sistemas embebidos es buscar literatura y propósito de procesadores. En la publicación recuentan 140 microprocesadores diferentes y más de 40 vendedores.

Importante es mencionar que la experiencia mostrada en el trabajo de Berger, A. H. (2002) indica que el desarrollo de un sistema embebido requiere de siete fases en su ciclo y de manera enunciativa estas son:

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

Fase 1 – Especificaciones del Producto.
Fase 2 – Partición entre Hardware y software.
Fase 3 – Iteración e Implementación.
Fase 4 – Diseño detallado de Hardware y Software (Camino de actividades de diseño de Hardware y Camino de actividades de diseño de Software).
Fase 5 – Integración de Hardware y Software.
Fase 6 – Pruebas de aceptación.
Liberación de Producto
Fase 7 – Mantenimiento y actualización.

En la figura 12 se muestra el modelo que ilustra las fases que se han mencionado.

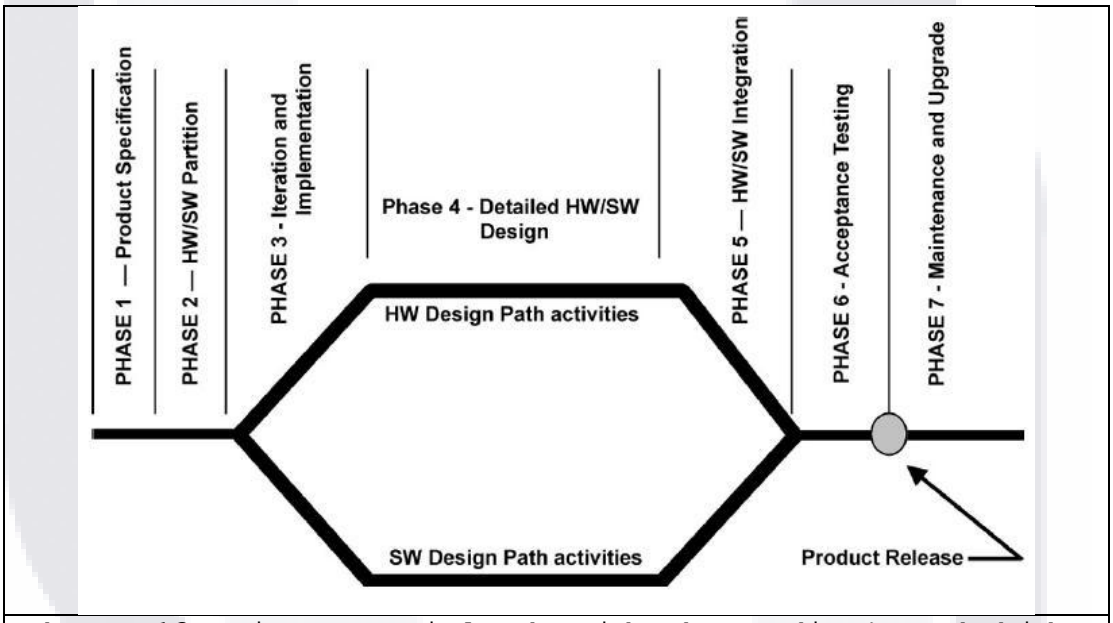


Figura 12. Diagrama Ciclo de vida de un diseño embebido, extraida de Berger, A. H. (2002).

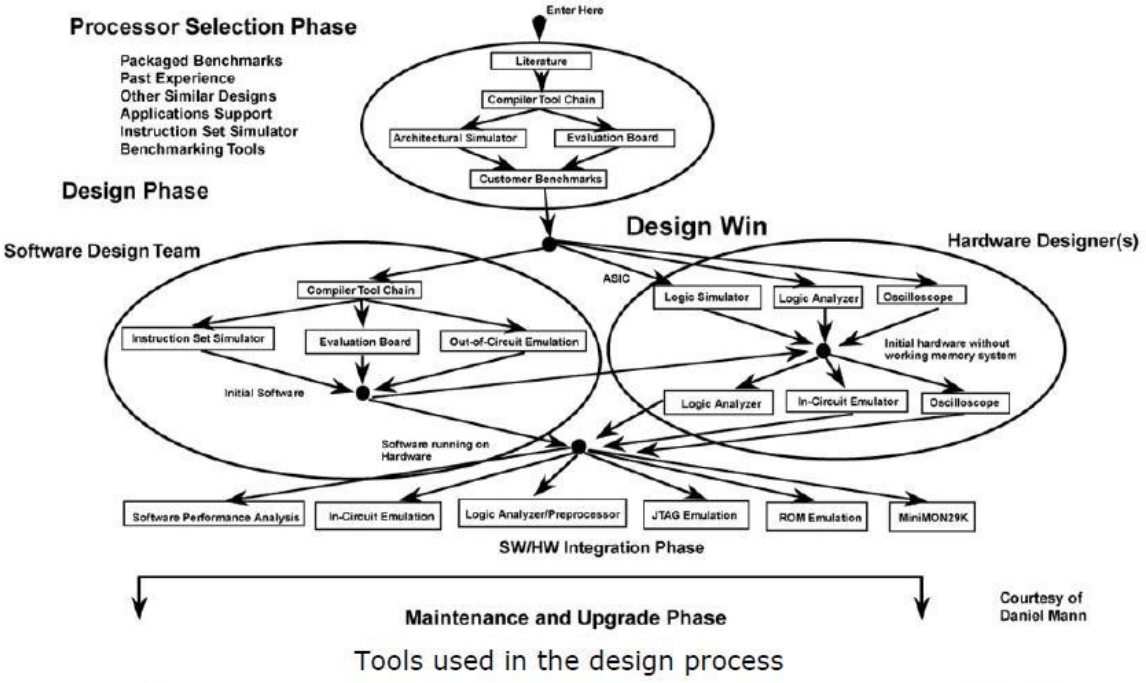
Los consejos a considerar en el ciclo de vida son:

- Recordar continuas iteraciones (es iterativo).
- Revisar las herramientas basadas en la lista del ciclo de desarrollo propuesto, considerado bajo el modelo del Dr. Daniel Mann, Avanced Micro Devices (AMD) figura 13.
- Considerar de principal importancia la "Selección de Procesador Correcto".
- Especificar el compilador.
- Considerar el costo.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

- Considerar el rendimiento de ejecución.
- Considerar las herramientas para mantenimientos y actualizaciones.



Tools used in the design process

The embedded design cycle represented in terms of the tools used in the design process (courtesy of Dr. Daniel Mann, AMD Fellow, Advanced Micro Devices, Inc., Austin, TX).

Figura 13. Modelo del Dr. Daniel Mann mostrando las herramientas usadas en el proceso de diseño , extraída de Berger, A. H. (2002)

Acerca de la fase de especificación del producto, que es donde se realiza la selección del procesador, con posterioridad se verá reflejado en el producto de Especificación de Requerimientos bajo las siguientes consideraciones mencionadas en la literatura:

- Puntos de referencia del empaquetado (montaje superficial, pin u otro).
- Basarse en la experiencia de proyectos anteriores.
- Recuperar información de diseños similares.
- Soporte de aplicaciones.
- Simulador del conjunto de instrucciones.
- Herramientas como punto de referencia.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

A pesar de que el trabajo abarca todas las fases del ciclo de desarrollo de un sistema embebido, por cuestiones de objetividad sólo se enfoca la atención en las primeras tres fases que son las consideradas con relación directa al desarrollo del trabajo de tesis.

Bajo esta observación iniciaremos la descripción a groso modo de lo que implica cada una de estas fases, pero se insta al lector en caso de desear conocer las otras fases a consultar el libro para un panorama amplio de las mismas.

Fase 1, Especificación de Producto: Aquí se generan los requerimientos para el equipo de trabajo y Las consideraciones a tomar en cuentas son:

- Usar métodos para requerimientos tales como cuestionarios y grupos enfocados.
- Organizar un equipo de requerimientos recomendado en tres participantes.
- Los roles de integrantes de equipo cambian.
- El cuestionario se elabora en dos semanas con preguntas de composición abierta.
- Al terminar el cuestionario los miembros hacen otro cuestionario preguntándose:
    1. ¿Qué escucho cada miembro?
    2. ¿Qué está en estado explícito?¿Qué está en estado implícito?
    3. ¿Les gustó lo que se tiene o sólo son amables?
    4. ¿Hubo alguien muy activo?
    5. ¿Se necesita refinar la presentación o la forma del cuestionario?
    6. ¿Hablaron con la gente correcta?
- Al final e equipo debe escribir un sumario de los resultados de la visita al cliente.

En esta misma fase se tienen observaciones a considerar al momento de realizar los requerimientos y las especificaciones del diseño, tales como:

- Tener un objetivo: Convertir un concepto en un producto.
- Comprender las necesidades.
- Hacer preguntas correctas.
- Determinar el punto de precio.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

- Considerar que el cliente quiere todo para ayer sin tener que pagar por ello.
- Nunca preguntar cosas que no son necesarias para el producto (para que ofrecer un avión si desean un bote).
- Enfocarse en que desea el cliente y extender su requerimiento al producto. Así se destilan las necesidades en la abstracción del diseño.
- Compartir una visión común de equipo de diseño, esto implica que cada miembro deba dar siempre la misma descripción en cuanto a metas y en consecuencia mantener el costo y ejecución.
- También especificar en el producto las herramientas requeridas para el diseño y desarrollo del producto con la finalidad de minimizar el riesgo de expectativas irreales en las metas.
- Llevar a cabo juntas que son para acordar: lo que debe tener, lo que se quiere tener y siempre el equipo debe de estar de acuerdo y enfocado. No se debe de avanzar hasta llegar a un consenso.

Fase 2, Partición de hardware y software: Considerada como una etapa crítica y en la cual el equipo de desarrollo para hardware y software debe mantener el máximo interés, pues el éxito del proyecto dependerá en gran medida de los acuerdos que se tomen en esta fase. Una omisión podría tener consecuencias al momento de la fase de integración del sistema. Las recomendaciones a seguir son:

- Llevar a cabo la decisión de partición: ¿qué problema se resolverá en hardware y qué problema se resolverá en software?.
- Recordar que los desarrolladores de aplicación trabajan con recursos predefinidos de hardware.
- Hacer diagramas de bloques puede ayudar a definir la partición.
- Es muy importante definir el proceso de funcionamiento del producto para hacer el diagrama de bloques. Un flow chart ayudaría a dar secuencia al proyecto. En esta parte se da el análisis de costo y componentes se va definiendo lo que es más óptimo para particionar tareas, hardware y software.
- Investigar los métodos de diseño para pasarlos a soluciones de hardware.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

- Es más fácil reparar defectos en software que en hardware. Poner mucha atención a la partición y si el hardware en un ASIC (Application Specific IC).
- El paso anterior lleva a la consideración de administración de riesgos, ya que se debe decidir y eso consume tiempo de análisis para ello.
- Si se requiere velocidad de procesamiento considerarlo en el procesador de acuerdo a costo, nuevas herramientas, nuevas tarjetas, datos y complejidad.
- Las mejoras de ejecución implican análisis acerca de:
  a) Ajuste fino a ejecución de procesador para minimizar las necesidades de hardware especializado.
  b) Usar hardware especializado y no formar equipo ASIC de diseño para resolver el problema.
- Se encuentra en esta fase un problema de optimización complejo basado en:
  1. Diseño de sistema embebido sensible al precio.
  2. Los principales diseñadores de vanguardia.
  3. Falta de estándares.
  4. Mercado competitivo.
  5. Propietario.
- Estos requerimientos conflictivos dificultan crear un diseño óptimo para el producto embebido.
- Fuerte consideración parte del tipo de procesador a usar:
  a) Microprocesador.
  b) Microcontrolador.
  c) ASIC núcleos personalizados.
- Basarse en experiencias aguda a la elección.

Fase 3, Iteración e Implementación: A esta etapa la considera el autor como la última en la cual los equipos de hardware y software aún estarán en comunicación y representa una zona un tanto borrosa entre la aplicación y la partición de hardware y software en la cual sus direcciones divergirá tal como se mostró en la figura 12 para el ciclo de vida de un sistema embebido. Esta fase representa el trabajo inicial de diseño antes de que los equipos de hardware y software construyan "un muro" entre ellos. Las consideraciones generales para esta etapa son:

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

- En el modelo de Mann se considera a la etapa como parte del proceso de selección (Fase de iteración).
- El diseñador de Hardware deberá usar las herramientas de simulación (simulador de arquitectura).
- El diseñador de software debe usar corrida de código con puntos de referencia y usar tarjetas micro computadas para debug antes de correr en el sistema real, esto asegurará la disminución de riesgos.

Hasta este punto hemos dejado descritas las fases del ciclo de vida del sistema embebido que nos interesan y ahora sólo como referencia general se describirá en qué consiste el resto de las fases.

Fase 4, Diseño detallado de hardware y software: Aquí se deben de considerar el uso de las herramientas para diseño de hardware y de software que se seleccionaron antes de dejar de tener contacto los equipos.

Fase 5, Integración de hardware y software: En esta etapa se usan herramientas y métodos para manejar la complejidad, se realizan los debbugging y descubrimientos de quizá algunos malos entendidos en la información (ej. Síndrome BigEndian/LittleEndian

Fase 6, Prueba del producto y liberación: En esta fase el equipo se encarga de asegurarse de que el software no colapsará en un momento crítico mediante el "testing" hecho por alguien ajeno a los equipos de ingenieros y técnicos para evitar resultados erráticos.

Fase 7, Mantenimiento y actualización de productos existentes: Dado que bajo registros de la realidad alrededor del 60% de los diseñadores da mantenimiento y actualización a productos existentes más que hacer nuevos se debe considerar investigar algunas herramientas de ingeniería inversa o similares para llevar a cabo esta fase en forma óptima. Cabe mencionar que usualmente los ingenieros que realizan esta labor no formaron parte del equipo original, lo cual implica esfuerzo para comprender y mantener el producto.

Una vez que han quedado claro algunos de los conceptos necesarios para el desarrollo del trabajo de investigación, es requerido presentar los contrastes encontrados en las investigaciones consultadas.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

En el siguiente capítulo se encontrará el análisis de contribuciones y limitaciones de dichos estudios.

3.3 ANÁLISIS DE CONTRIBUCIONES Y LIMITACIONES DE PRINCIPALES ESTUDIOS RELACIONADOS.

El objetivo del presente apartado es presentar en las siguientes tablas a tres columnas un resumen del análisis de la literatura ordenado en el mismo sentido de sistemas embebidos, software y software embebido.

Tabla 2. Literatura Chandy, J. C. (2010) Sistemas ciber-físicos (CPS), análisis de los desafíos en su diseño; Ting, J. S. (2011) Seguimiento a los CPS, planteamiento de grandes desafíos en el diseño de software.

Aportación	Puntos no explorados	Consideraciones
• Informa falta de métodos, metodologías y modelos para la creación de sistemas embebidos. • Hace énfasis en usar abstracciones de disciplinas para llevar a cabo una metodología. • Propuestas de cómo diseñar un sistema en torno a los plazos de ejecución incluyendo el maximizar la utilización de los recursos. • Requisitos específicos para estos sistemas: confianza, seguridad, eficiencia, confiabilidad y previsibilidad, límites de las capacidades del hardware. • Limitaciones de los sistemas CPS: inconsistencia entre necesidades de aplicación y aportes de servicio del sistema, los servicios del sistema tienden a centrarse en la equidad y la eficiencia en lugar del tiempo, la confianza, la seguridad, la fiabilidad y las limitaciones; falta de flexibilidad en los sistemas.	• No reportan ninguna técnica, método, metodología o modelo a seguir para el desarrollo, diseño o implementación de los sistemas embebidos.	• Las observaciones son generales a un sistema embebido, abarcando unidad de hardware y software. • Se da pauta a llevar a cabo esfuerzos que propongan métodos, modelos o metodologías para cumplir con los requisitos planteados para el desarrollo de los sistemas embebidos.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

Tabla 3. Literatura González, P. L., & Ulrrego, G. G. (2008) Modelo de requisitos para sistemas embebidos.

Aportación	Puntos no explorados	Consideraciones
• Propone un modelo de requisitos como apoyo para la construcción de sistemas embebidos. • Considera que las metodologías de Ingeniería de Requisitos propuestas para este dominio no establecen continuidad en su proceso de desarrollo, por poseer una fuerte orientación a la etapa de diseño y un énfasis más débil en la etapa de análisis. • Reporta falta de propuesta de herramientas para la obtención de los requisitos • Ejecuta un caso práctico para proponer este modelo de requisitos • Usa para la estructura del modelo la visión en dos grandes áreas: la ingeniería de requisitos (IR) y los sistemas embebidos (SE). • El logro fue explotar y especializar las categorías del modelo de la metodología ABC-Besoins (Metodología de ingeniería de requisitos por metas para web) para los sistemas embebidos. El resultado otorga una nueva metodología denominada ABC-Besoins-SEM.	• Documentación de los procesos. • Referencia a herramientas o artefactos usados para garantizar el cumplimiento de las etapas. • Validación y verificación del estado correcto de la etapa.	El modelo y ejemplo ayudan a considerar al artículo como una muestra de cómo llevar a cabo la adaptación del modelo, pero para MoProSoft se deberá pensar en ir cubriendo los apartados de validación de las actividades, generación de documentación y proponiendo herramientas para cada actividad enfocándolas al software embebido.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

Tabla 4. Literatura Gastaldi, G. G., etal (2002, Abril) Evaluación de programas de cálculo en ingeniería como herramienta de desarrollo para Codiseño Hardware/Software.

Aportación	Puntos no explorados	Consideraciones
• Aportaciones al desarrollo de sistemas embebidos con metodología de Codiseño Hardware Software, • Descripción de las características generales, herramientas y conceptos más importantes en la utilización de Codiseño HW/SW, obteniendo los criterios para una evaluación. • Cita características de los sistemas embebidos • Describe los controladores embebidos y su flujo • Cita los niveles de abstracción relacionados con el desarrollo de sistemas digitales. • Lista los dominios de descripción del diseño. • Cita al codiseño de HW/SW como el proceso de diseño de un sistema que combina las perspectivas hardware y software desde los estados primarios, para aprovechar la flexibilidad del diseño y la localización eficiente de las funciones." • Para codiseño las limitaciones son debidas a indisponibilidad de herramientas adecuadas para el desarrollo de las distintas etapas.	• Documentación de los procesos. • Referencia a herramientas o artefactos usados para garantizar el cumplimiento de las etapas. • Validación y verificación del estado correcto de la etapa.	• El diagrama "Y" que resume los niveles y dominios del desarrollo de un sistema embebido puede servir al momento de sugerir ciertas técnicas, herramientas o métodos a implementar en la categoría de Operación para MoProSoft.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

Tabla 5. Literatura Gil, R. A. C. (2006) Estructura básica del proceso unificado de desarrollo de software

Aportación	Puntos no explorados	Consideraciones
• Propone el uso del método RUP de IBM basado en el método de espiral para desarrollo de software en general. • Recomienda no usar modelo en cascada por los problemas que presenta al ser una secuencia de grandes etapas que requieren como hitos la documentación completa antes de continuar con la siguiente etapa • Sugiere el uso de métodos iterativos e incrementales • Sugiere uso de prácticas claves como orientación al manejo de riesgos y planeación adaptable para guiar en una forma más adecuadamente y natural el proceso de desarrollo de software.	• Detallar o ejemplifica las etapas de RUP en forma específica indicando herramientas para cada apartado o validaciones de cumplimiento.	• Apoya la teoría de usar un modelo esbelto pero efectivo como MoProSoft orientado a PyME o grupos pequeños de trabajo.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

Tabla 6. Literatura Hernández, V. J. I. (2010) El Software Embebido y los Retos que Implica su Desarrollo

Aportación	Puntos no explorados	Consideraciones
• Presenta al software embebido como un tipo de aplicación muy particular en la ingeniería de software • Indica que posee características específicas que se demandan al construirlo. • Análisis de los retos que se presentan durante su proceso de desarrollo • Tendencias que se tienen a futuro en la mejoría de sus herramientas de diseño y cobertura de sus aplicaciones. • Hace énfasis en que la industria del software y los sistemas embebidos cada vez tienen mayor auge y que mantienen una creciente presencia en las diferentes actividades del ser humano. • Sus particularidades son su alta confiabilidad en su operación, recursos de hardware limitado y respuestas en tiempo real. • Recomienda aplicarse en la búsqueda de mejores, métodos, técnicas, herramientas y proceso de desarrollo que garanticen productos de calidad y con una amplia gama de aplicaciones y desarrollos tecnológicos.	• No hace mención de herramientas o esfuerzos de métodos aplicables en el diseño o desarrollo.	• Apoya la necesidad de creación o adaptación de modelos, métodos o metodologías para el desarrollo de software. Embebido.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

Tabla 7. Literatura Ruiz, M. B. (2010) Desarrollo de Software Basado en Modelos para Sistemas Embebidos

Aportación	Puntos no explorados	Consideraciones
• En se propone el uso de la herramienta statechart para generar software embebido • Aplican un caso práctico de sistema ejemplificando el uso de la herramienta • Enuncia la ventaja de que algunos modelos permiten simular el funcionamiento de un sistema desde las primeras etapas del diseño abriendo la posibilidad de generar automáticamente el código fuente y la documentación del proyecto. • También menciona que la práctica del uso de modelos para el desarrollo de software no está muy extendida, lo cual da como resultado el afán de esforzarse en que se impulse el despegue de metodologías de programación, • Usa IAR Visual-State para reportar el caso práctico (entorno gráfico para diseño), incluyendo verificación e implementación de sistemas embebidos basados en máquinas de estados jerárquicas o statecharts. • Usa seis pasos para determinar las especificaciones del sistema: 1. Identificar los eventos y las acciones 2. Identificar los estados 3. Agrupar los estados por jerarquías 4. Agrupar por concurrencia 5. Añadir las acciones y transiciones 6. Añadir las sincronizaciones	• Dejar en claro de que metodología o modelo obtiene los seis pasos que determinan las especificaciones del sistema.	• Aportación a la forma de abordar el tema para llevar el caso práctico con el software embebido en la tarjeta Ethernet y reportarlo aunado al modelo de MoProSoft.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

3.4 MODELO O ESQUEMA GENERAL DE LA INVESTIGACIÓN.

Partiendo de la teoría presentada en el capítulo 2.4 acerca de la metodología elegida bajo clasificaciones en (Mora, T. M. 2011), se considera demostrada la validez para el desarrollo de la investigación y el uso del modelo.

El proceso seguido para la investigación se resume, como lo menciona el documento del Instituto Internacional de Integración (S/F), en:

1. Definición del problema.
2. Diseño de trabajo.
3. Recogida de datos.
4. Análisis de datos.
5. Informe y validación de la información.

Para apoyar el desarrollo del proceso de investigación se ha usado como apoyo el libro Eco, U. (1977) para organizar la información y redactar el documento final. Cabe mencionar que como resultado de esta consulta se ha hecho un esfuerzo de corrección en la marcha hacia algunas omisiones que se dieron al inicio de la investigación.

A este punto consideramos conveniente presentar el modelo de investigación propuesto y en el apartado siguiente llevar a cabo la descripción de los constructos y variables. En la figura 14 se muestra el modelo con los constructos del lado izquierdo y la variable del lado derecho.

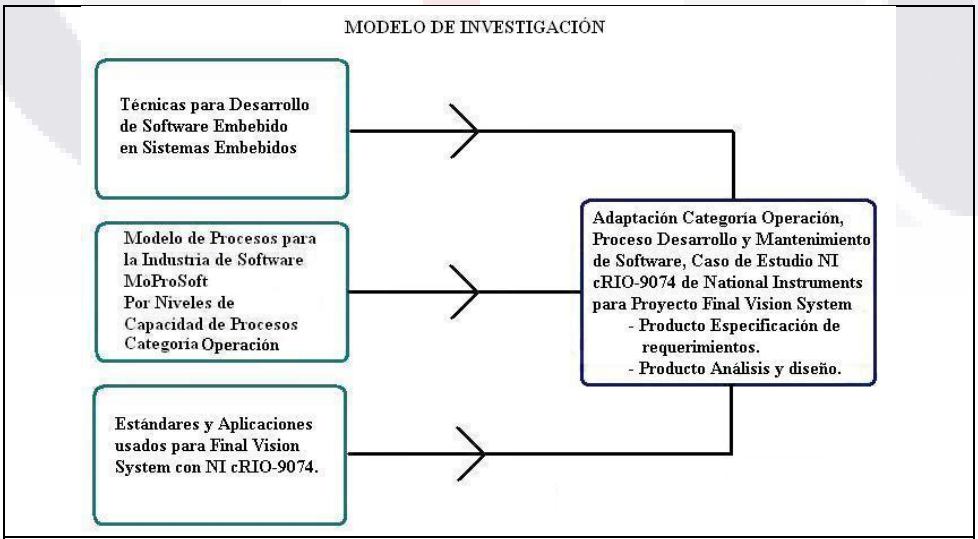


Figura 14. Modelo de investigación propuesto.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

A continuación se dará una definición de los elementos que integran el modelo:

Técnicas para Desarrollo de Software Embebido en Sistemas Embebidos: Se refieren a modelos, métodos o metodologías que los sistemas embebidos usan en la actualidad para generar su software y que puedan contribuir a cumplir las actividades propuestas por la categoría operativa de MoProSoft.

Modelo de Procesos para la Industria de Software MoProSoft por Niveles de Capacidad de Procesos Categoría Operación: En el modelo es la categoría que se encarga de la Administración de Proyectos Específicos y Desarrollo y Mantenimiento de Software.

Estándares y Aplicaciones usados para Final Vision System con NI cRio-9074: Se consideran dentro de este apartado a los elementos del proyecto proporcionados por el departamento de Manufacturing Technology Engineering (MTE) para generar la documentación desde los requerimientos hasta la liberación del producto.

Resultado de productos a presentar:

Adaptación Categoría Operación Proceso Desarrollo y Mantenimiento de Software, Caso de Estudio NI cRio-9074 de National Intruments para Proyecto Final Vision System: Será la presentación de los productos: Especificación de Requerimientos y Análisis y Diseño para probar la aplicación del modelo en el caso de estudio para el software embebido en el desarrollo del proyecto

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

CAPÍTULO 4

DESARROLLO Y VALIDACIÓN

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

4. DESARROLLO Y VALIDACIÓN

4.1 DESARROLLO DE INVESTIGACIÓN.

Tal como se explicó en el apartado 3.1.2. Definiciones del Modelo de MoProSoft Categoría Operación, una de las ventajas de usar el modelo de MoProSoft en la parte de Desarrollo y Mantenimiento del Software Embebido es que tiene como esencia y objetivo ser usada por grupos pequeños.

El trabajo de investigación realizado durante el desarrollo de la presente tesis tiene por objeto la generación de dos productos aplicables en el proceso de desarrollo y mantenimiento de software embebido usando los lineamientos del modelo de procesos para la industria del software MoProSoft. Sin embargo, estos productos no se han mantenido solamente a nivel teórico, sino que se han aplicado a un caso de estudio.

Como resultado se obtiene el contraste de un apartado del instrumento usado inicialmente para el desarrollo del caso de estudio y los productos desarrollados como esfuerzo de la presente tesis. Los productos de salida elaborados han sido nombrados: OPE.2_DOC1_EDR_v1.0.0.2012 para el caso de especificación de requerimientos y OPE.2_DOC2_AYD_v1.0.0.2012 para Análisis y Diseño.

Para presentar el trabajo se comenzará con una visión general de la categoría de operación (OPE), con objeto de enunciar los dos procesos que la componen: Administración de Proyectos Específicos (OPE.1) y Desarrollo y Mantenimiento de software (OPE.2), dando énfasis a describir brevemente OPE.2. Esto permitirá mostrar una visión de conjunto a partir de la cual se expondrán y desglosarán los detalles de cada actividad y salidas para dicha actividad.

Siguiendo el orden del modelo de procesos se irán presentando los elementos estudiados en referencia a los sistemas embebidos y software embebido para mostrar cómo se fueron integrando cada uno a las estructuras específicas de cada producto.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

4.1.1 La categoría de operación (OPE) del Modelo MoProSoft.

Como ya se mencionó y especificó en el capítulo anterior, el modelo de procesos MoProSoft está compuesto por tres categorías principales:

- Categoría de Alta Dirección (DIR).
- Categoría de Gerencia (GER/GES).
- Categoría de Operación (OPE).

El trabajo de tesis se enfoca en la tercera categoría del modelo: Categoría de Operación (OPE). Por dicha razón a continuación se explica su función y composición.

La Categoría de Operación se encarga de abordar las prácticas de los proyectos de desarrollo y mantenimiento de software. Las actividades que deben ser realizadas en esta categoría se llevan a cabo de acuerdo a los elementos proporcionados por la Categoría de Gerencia. Al finalizar la Categoría de Operación deberá de entregarle la información y los productos generados a la Categoría de Gerencia (Oktaba H., et al., 2005).

La Categoría de Operación se subdivide en dos procesos:

- Administración de Proyectos Específicos (OPE.1).
- Desarrollo y Mantenimiento de software (OPE.2).

El proceso de Administración de Proyectos Específicos (OPE.1) tiene como propósito el establecer y llevar a cabo sistemáticamente las actividades que permitan cumplir con los objetivos de un proyecto en tiempo y costo esperados (Oktaba H., et al., 2005).

El propósito del proceso Desarrollo y Mantenimiento de Software (OPE.2) es la realización sistemática de las actividades de análisis, diseño, construcción, integración y pruebas de productos de software nuevos modificados cumpliendo con los requerimientos especificados (Oktaba H., et al., 2005). El diagrama de flujo para este proceso se puede revisar en la Figura 15.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

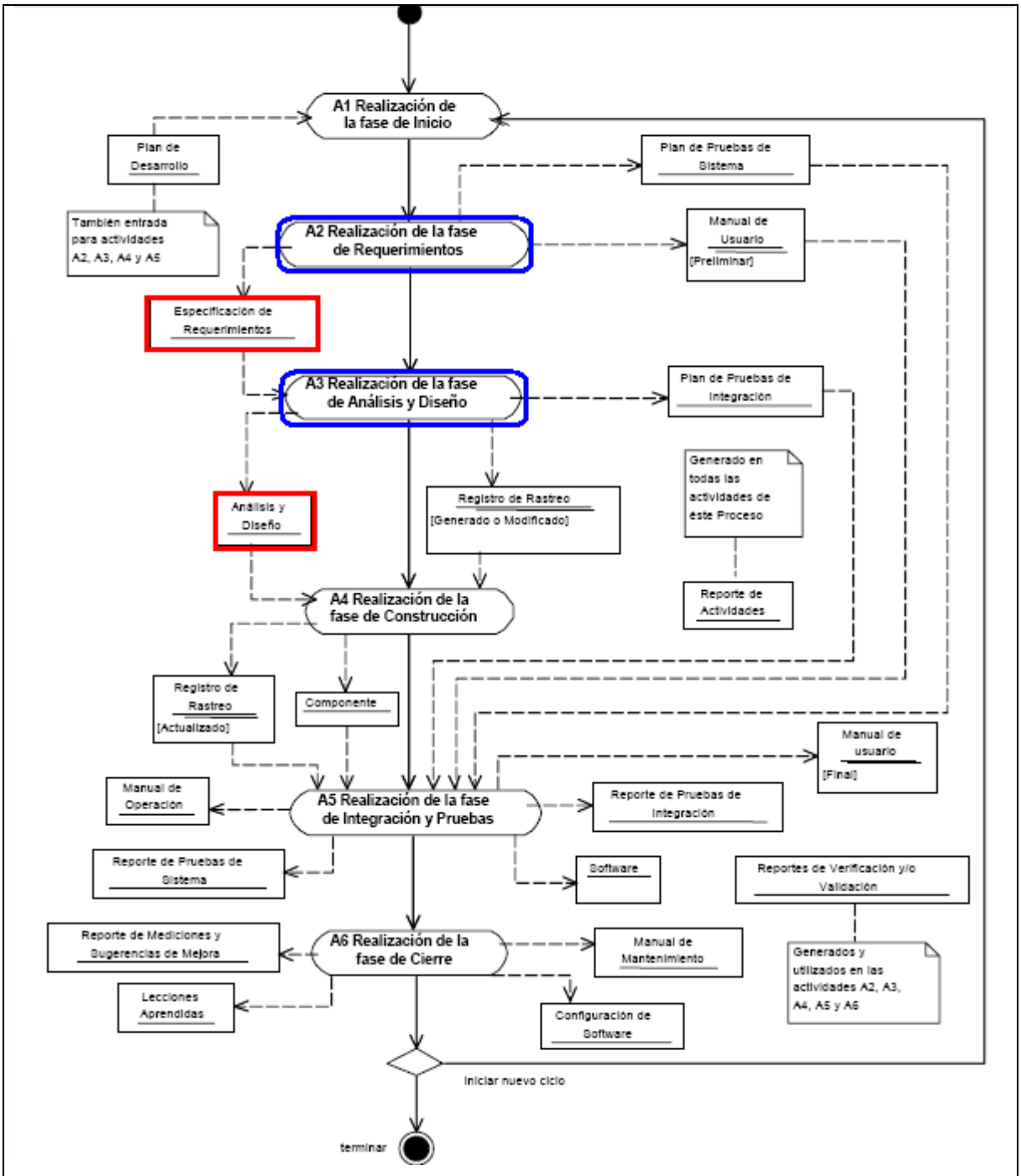


Figura 15. Diagrama de flujo de trabajo Desarrollo y Mantenimiento de Software MoProSoft.

Retomando la información descrita en el capítulo 3.1.2 acerca de la descripción del patrón de proceso usado para MoProsoft y los componentes del mismo, hablaremos del elemento *actividades*.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

Las actividades, como se recordará, son tareas de verificación y validación que se relacionan a los objetivos y los roles responsables. Éstas se visualizan en conjunto a través de los diagramas de flujo de trabajo indicando también los productos relacionados a cada una de ellas.

En la tabla 8 se listan las actividades y se contabilizan las sub actividades que se deben llevar a cabo en los procesos: Administración de Proyectos Específicos (OPE.1) y Desarrollo y Mantenimiento de Software (OPE.2).

Administración de Proyectos Específicos (OPE.1)	**Desarrollo y Mantenimiento de Software (OPE.2)**
• Planificación (18 subactividades). • Realización (11 subactividades). • Evaluación y control (3 subactividades). • Cierre (4 subactividades).	• Inicio (2 subactividades). • Requerimientos (14 subactividades). • Análisis y Diseño (11 subactividades). • Construcción (6 subactividades). • Integración y pruebas (12 subactividades). • Cierre (7 subactividades).

Tabla 8. Actividades y sub actividades de los Procesos involucrados en la Categoría de Operación.

Hasta este punto se tiene el panorama general de la categoría y los subprocesos que la componen, es por ello importante iniciar la delimitación de las actividades de interés.

En el capítulo 3.1.3 referente a los Sistemas Embebidos, se hace énfasis acerca de que el éxito en el Desarrollo de Software Embebido depende básicamente de dos actividades primordiales: la selección del procesador y la forma en la cual se planean resolver las tareas del sistema (hardware o software).

La pregunta en este punto es ¿En qué parte del proceso de OPE se encontrarán definidas estas dos actividades primordiales?

Para dar respuesta a esa interrogante inicialmente se llevó a cabo un mapeo en un documento de los productos de entradas, salidas e internos relacionados con el proceso OPE.2, manteniendo el modelo de colores que nos dan el nivel de

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

capacidad. La finalidad fue comprender de manera más visual la forma en la cual se hiciera posible localizar las actividades primordiales.

En dicho documento se llevaron a cabo anotaciones usadas como guías de relación y se le agregaron columnas de control. Los elementos integrados a este mapa son:

- **Hecho:** Se marca la casilla para llevar un control de los materiales que se ha completado de diseñar, haciendo un poco de referencia a la forma de registro de actividades de PSP.
- **Nombre de Producto:** Alberga el nombre que se le da al producto que controla la actividad de la categoría. La formulación del nombre se hace basado en la teoría de control de versiones usada para el software (Ríos, C. A., 1998) y se compone por OPE.2, que es la categoría a la cual pertenece en el proceso de software; DOCX, el número de documento que le corresponde en base al número de actividad, la X es el número; las iniciales de la actividad; una "v" seguida de la numeración inicial del producto en 3 niveles separadas por un punto y al final el año de aprobación de este producto (ejemplo 1.0.0.2012).
- **Descripción:** En esta columna se coloca la explicación de los elementos que debe de contener la categoría y nos ayudará a elaborar el producto aplicando los elementos encontrados en la literatura que se enfocan a los sistemas embebidos.
- **Destino:** Este apartado nos indica la categoría, actividad o proceso a la cuál irán destinados los productos finales para su seguimiento en el modelo de MoProSoft.

Se pretende que al terminar de llenar esta matriz y se completen los productos, ésta nos ayude a la construcción de diagrama de entradas y salidas propuesto como trabajo futuro.

A continuación se muestra la versión final del mapeo:

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

Nivel	Capacidad de proceso	Color
1	Realizado	amarillo
2	Gestionado	azul
3	Establecido	verde
4	Predecible	rosa
5	Optimizado	ninguno

ENTRADAS

NOMBRE	FUENTE
Plan de desarrollo: • Descripción del Producto. • Entregables. • Proceso Específico. • Equipo de trabajo. • Calendario.	Administración de Proyectos Específicos

SALIDAS

HECHO	NOMBRE DE PRODUCTO	NOMBRE	DESCRIPCIÓN	DESTINO
OK	OPE.2_DOC1_EDR_v1.0.0.2012	1. Especificación de Requerimientos. Es un documento.	Introducción: Descripción general del software y su uso en el ámbito de negocio del cliente. Descripción de requerimientos:	Administración de Proyectos Específicos

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

			Funcionalidad: Necesidades establecidas que debe satisfacer el software cuando es usado en condiciones específicas. Las funcionalidades deben de ser adecuadas, exactas y seguras. *Interfaz con usuario:* Definición de aquellas características de la interfaz de usuario que permiten que el software sea fácil de entender, aprender, que genere satisfacción y con el cuál el usuario pueda desempeñar su tarea eficientemente. Incluyendo la descripción del prototipo de la interfaz. *Interfaces externas:* Definición de las interfaces con otro software o con hardware. *Confiabilidad:* Especificación del nivel de desempeño del software con respecto a la madurez, tolerancia a fallas y recuperación. *Eficiencia:* Especificación del nivel de desempeño del software con respecto al tiempo y a la utilización de recursos.	

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

			Mantenimiento: Descripción de los elementos que facilitarán la compresión y la realización de las modificaciones futuras del software. *Portabilidad:* Descripción de las características del software que permitan su transferencia de un ambiente a otro *Restricciones de diseño y construcción:* Necesidades impuestas por el cliente. *Legales y Reglamentarios:* Necesidades impuestas por leyes, reglamentos, entre otros.	
OK	OPE.2_DOC2_AYD_v1.0.0.2012	2.- Análisis y Diseño Es un documento.	Descripción textual y gráfica de: Arquitectónica: Contiene la estructura interna del sistema, es decir la descomposición del sistema en subsistemas. Así como la identificación de los componentes que integran los subsistemas y las relaciones de interacción entre ellos. Detallada: Contiene el detalle de los componentes que permita de manera evidente su construcción y prueba en el ambiente de programación.	Administración de Proyectos Específicos.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

		3.- Componente	Conjunto de unidades de código relacionadas.	Administración de Proyectos Específicos.
		4.- Software	Sistema de software, destinado a un cliente o usuario, constituido por componentes agrupados en subsistemas, posiblemente anidados.	Administración de Proyectos Específicos.
		5.- Configuración de Software Conjunto de documentos y productos.	Conjunto consistente de productos de software. Matriz de consistencia (requerimientos) Especificación de Requerimientos: Análisis y Diseño: Software: Registro de Rastreo: Plan de pruebas de sistema: Reporte de Pruebas de Sistema: Plan de pruebas de integración: Reporte de Pruebas de Integración: Manual de Usuario: Manual de Operación. Manual de Mantenimiento:	Administración de Proyectos Específicos.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

		6.- Manual de Usuario Es un documento.	Puede ser electrónico o impreso. Describe: Forma de uso del software en base a interfaz de usuario. Redacción en términos comprensibles a los usuarios.	Administración de Proyectos Específicos.
		7.- Manual de Operación Es un documento.	Puede ser electrónico o impreso. Contendrá: Información indispensable de instalación. Información indispensable de administración del software. Ambiente de operación (sistema operativo, bases de datos, servidores, etc.) Redacción en términos enfocados a personal responsable de la operación.	Administración de Proyectos Específicos.
		8.- Manual de Mantenimiento Es un documento.	Puede ser electrónico o impreso. Describe: La configuración del Software. El ambiente usado para el desarrollo (compiladores, herramientas de análisis y diseño, construcción y pruebas). El ambiente usado para pruebas (compiladores, herramientas de análisis y diseño, construcción y pruebas). Redacción en términos	Administración de Proyectos Específicos.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

			comprensibles por personal de mantenimiento.	
OK	OPE.2_DOC9_RDA_V1.0.0.2012	9.- Reporte de Actividades Producto de Registro	Producto con: Actividad incluye: Fecha de Inicio Fecha final Responsables Mediciones: Tiempo de producción, corrección, verificación y validación; defectos encontrados en verificación, validación y prueba; Tamaño de productos.	Administración de Proyectos Específicos.
		10.- Lecciones aprendidas Producto de Registro	Registro de: Mejores prácticas en ciclo de desarrollo: Problemas recurrentes en ciclo de desarrollo: Experiencias exitosas en solución de problemas del ciclo de desarrollo: Mejores prácticas en ciclo de mantenimiento: Problemas recurrentes en ciclo de mantenimiento: Experiencias exitosas en solución de problemas en ciclo de mantenimiento:	Conocimiento de la organización.
		11.- Reporte de mediciones y sugerencias de mejora Producto de Registro	Producto con: a) Mediciones de los indicadores del proceso de Desarrollo y	Administración de Proyectos Específicos.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

			Mantenimiento de Software b) Sugerencias de mejora al proceso de Desarrollo y Mantenimiento de Software: Métodos, Herramientas, Formatos, Estándares, Etc.	
		12.- Registro de Rastreo.	Relación entre: Requerimientos; Elementos Análisis; Elementos Diseños; Componentes de Prueba; Planes de Prueba.	Administración de Proyectos Específicos.
		13.- Plan de Pruebas de Sistema.	Identificación de pruebas requeridas para el cumplimiento de los requerimientos especificados.	Administración de Proyectos Específicos.
		14.- Reporte de Pruebas de Sistema.	Registro de: Participantes. Fecha. Lugar. Duración. Defectos encontrados.	Administración de Proyectos Específicos.
		15.- Plan de Pruebas de Integración Es un documento.	Describe: Orden de integración de los componentes o subsistemas, guiado por la parte arquitectónica del Análisis y Diseño. Pruebas que se aplicarán para verificar la interacción ente los componentes.	Administración de Proyectos Específicos.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

		16.- Reporte de Pruebas de Integración. Es un registro.	Registrar: Participantes. Fecha. Lugar. Duración. Defectos encontrados.	Administración de Proyectos Específicos.

PRODUCTOS INTERNOS

NOMBRE	**DESCRIPCIÓN (ENFOCADA Sw.E.)**
17.- Reporte(s) de verificación. Registro.	Registrar: Participantes. Fecha. Lugar. Duración. Defectos encontrados.
18.- Reporte(s) de Validación. Registro.	Registrar: Participantes. Fecha. Lugar. Duración. Defectos encontrados.

TESIS TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

Este ejercicio da como fruto el localizar a la actividad primordial de identificación del procesador en la descripción para la salida "Especificación de requerimientos" y para la resolución de tareas del sistema embebido la salida que se refiere al "Análisis y Diseño".

Posteriormente se requirió ubicar en el diagrama de flujo de trabajo referente al proceso las actividades que están relacionadas a dos productos de salida: "Especificación de requerimientos" y "Análisis y Diseño".

En la figura 15 donde se presentó el diagrama de flujo puede observarse marcados en rojo los productos de salida donde se ubicaron las tareas primordiales para el desarrollo del sistema embebido y en azul las actividades que contienen al desarrollo de los productos de salida en MoProSoft.

De esta manera queda delimitada el área de trabajo para el desarrollo de la investigación que da lugar a la presente tesis.

Para cerrar el apartado referente a la delimitación del trabajo desde el enfoque del Modelo de Procesos se dará la descripción de las actividades del modelo denotadas en azul, tal como las describe el manual de MoProSoft, con la finalidad de dejar clara su descripción y hacer énfasis que se toma como referencia para le generación del producto de salida propuesto en la tesis. Posteriormente se explicará el análisis que se realizó a la literatura de los sistemas embebidos para hacer la sinergia de las dos teorías y resolver la creación de los productos de salida.

ROL	**DESCRIPCIÓN**
A2. Realización de la fase de Requerimientos (O1,O3)	
RDM AN	A2.1. Distribuir tareas a los miembros del equipo de trabajo según su rol, de acuerdo al *Plan de Desarrollo* actual.
AN CL US DU	A2.2. Documentar o modificar la *Especificación de Requerimientos*: • Identificar y consultar fuentes de información (clientes, usuarios, sistemas previos, documentos, etc.) para obtener nuevos requerimientos. • Analizar los requerimientos identificados para delimitar el alcance y su factibilidad, considerando las restricciones del ambiente del negocio del cliente o del proyecto. • Elaborar o modificar el prototipo de la interfaz

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

	con el usuario. • Generar o actualizar la *Especificación de Requerimientos*.
RE	A2.3. Verificar la *Especificación de Requerimientos* (**Ver1**).
AN DU	A2.4. Corregir los defectos encontrados en la *Especificación de Requerimientos* con base en el *Reporte de Verificación* y obtener la aprobación de las correcciones.
CL US RPU	A2.5. Validar la *Especificación de Requerimientos* (**Val1**).
AN DU	A2.6. Corregir los defectos encontrados en la *Especificación de Requerimientos* con base en el *Reporte de Validación* y obtener la aprobación de las correcciones.
RPU AN	A2.7. Elaborar o modificar *Plan de Pruebas de Sistema*.
RE	A2.8. Verificar el *Plan de Pruebas de Sistema* (**Ver2**).
RPU	A2.9. Corregir los defectos encontrados en el *Plan de Pruebas de Sistema* con base en el *Reporte de Verificación* y obtener la aprobación de las correcciones.
RM	A2.10. Documentar la versión preliminar del *Manual de Usuario* o modificar el manual existente.
RE	A2.11. Verificar el *Manual de Usuario* (**Ver3**).
RM	A2.12. Corregir los defectos encontrados en el *Manual de Usuario* con base en el *Reporte de Verificación* y obtener la aprobación de las correcciones.
RDM	A2.13. Incorporar *Especificación de Requerimientos*, *Plan de Pruebas de Sistema* y *Manual de Usuario* como líneas base a la *Configuración de Software*.
RDM	A2.14 Elaborar el Reporte de Actividades registrando las actividades realizadas, fechas de inicio y fin, responsables por actividad y mediciones requeridas.

ROL	**DESCRIPCIÓN**
A3. Realización de la fase de Análisis y Diseño (O1,O3)	
RDM AN DI	A3.1 Distribuir tareas a los miembros del equipo de trabajo según su rol, de acuerdo al *Plan de Desarrollo* actual.
AN DI DU	A3.2 Documentar o modificar el *Análisis y Diseño*: • Analizar la *Especificación de Requerimientos* para generar la descripción de la estructura interna del sistema y su descomposición en subsistemas, y éstos

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

	a su vez en componentes, definiendo las interfaces entre ellos. • Describir el detalle de la apariencia y el comportamiento de la interfaz con base en la *Especificación de Requerimientos* de forma que se puedan prever los recursos para su implementación. • Describir el detalle de los componentes que permita su construcción de manera evidente. • Generar o actualizar el *Análisis y Diseño*. • Generar o modificar el *Registro de Rastreo*.
RE	A3.3 Verificar el *Análisis y Diseño* y el *Registro de Rastreo* **(Ver4)**.
AN DI DU	A3.4 Corregir los defectos encontrados en el *Análisis y Diseño* y en el *Registro de Rastreo* con base en el *Reporte de Verificación* y obtener la aprobación de las correcciones.
CL RPU	A3.5 Validar el *Análisis y Diseño* **(Val2)**.
AN DI DU	A3.6 Corregir los defectos encontrados en el *Análisis y Diseño* con base en el *Reporte de Validación* y obtener la aprobación de las correcciones.
RPU	A3.7 Elaborar o modificar *Plan de Pruebas de Integración*.
RE	A3.8 Verificar el *Plan de Pruebas de Integración* **(Ver5)**.
RPU	A3.9 Corregir los defectos encontrados en el *Plan de Pruebas de Integración* con base en el *Reporte de Verificación* y obtener la aprobación de las correcciones.
RDM	A3.10 Incorporar *Análisis y Diseño*, *Registro de Rastreo* y *Plan de Pruebas de Integración* como líneas base a la *Configuración de Software*.
RDM	A3.11 Elaborar el *Reporte de Actividades* registrando las actividades realizadas, fechas de inicio y fin, responsable por actividad y mediciones requeridas.

De las actividades descritas nos aplicaremos a proporcionar los productos para las sub actividades A2.2 y A3.2 que se encuentran en el nivel de capacidad "Realizado".

4.1.2 El ciclo de vida de un Sistema Embebido.

Una vez que se ha comprendido el modelo de MoProSoft e identificado los puntos en los cuales se trabaja, es necesario sustentar la relación de dicho modelo con la aplicación en el desarrollo de software de los sistemas embebidos.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

Dado este requisito, en la figura 16 se muestra la ruta usada del modelo del ciclo de vida para el desarrollo de un sistema embebido, sugerencia tomada de Berger, A. H. (2002) como se vio en el capítulo 3.2.2.1. De las 7 fases necesarias para completar el ciclo de vida de un sistema embebido.

Recordaremos que en el modelo presentado para el desarrollo de sistemas embebidos, tal como lo menciona la categoría OPE.2, se pueden realizar uno o varios ciclos iterativos para lograr un buen resultado cumpliendo las especificaciones de los requerimientos.

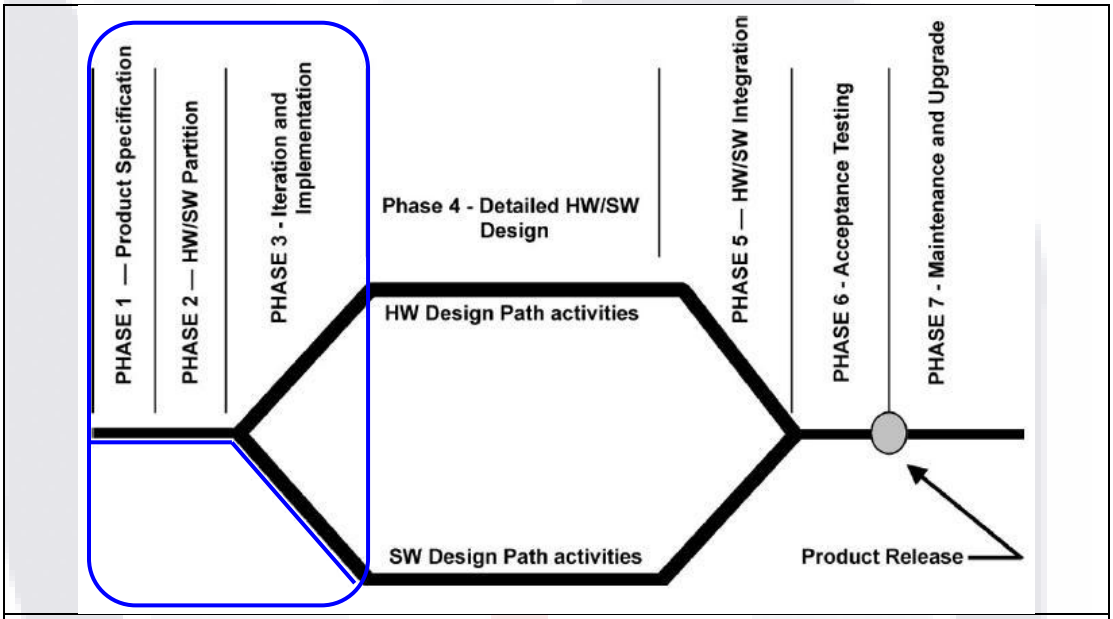


Figura 16. Diagrama Ciclo de vida de un diseño embebido marcando el áre de trabajo usado para la investigación, extraida de Berger, A. H. (2002).

Recordando las fases para el desarrollo de los sistemas embebidos listamos:

Fase 1 – Especificaciones del Producto.
Fase 2 – Partición entre Hardware y software.
Fase 3 – Iteración e Implementación.
Fase 4 – Diseño detallado de Hardware y Software (Camino de actividades de diseño de Hardware y Camino de actividades de diseño de Software).
Fase 5 – Integración de Hardware y Software.
Fase 6 – Pruebas de aceptación.
Liberación de Producto
Fase 7 – Mantenimiento y actualización.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

Al llegar a este punto se considera que el ciclo enmarcado por MoProSoft para Desarrollo y Mantenimiento de Software y el ciclo de vida para Sistemas Embebidos contienen similitudes en la forma de llevar el proceso.

Nótese que al final de la fase 2 del ciclo de vida para un sistema embebido, Partición entre Hardware y Software, se torna la división para el desarrollo del Software, lo cual resulta bastante positivo y adecuado para hacer la adaptación del modelo de MoProSoft desde la fase 1 hasta la fase 7 del ciclo de vida del sistema embebido siguiendo sólo la línea de actividades de diseño de Software en trabajos futuros. Por ello se ha llevado a cabo la tabla 9 que nos muestra la visión comparativa entre el modelo MoProSoft y el ciclo de vida del sistema Embebido.

MoProSoft, proceso Desarrollo Y Mantenimiento de Software.	**Ciclo de Vida, Sistema Embebido**
A.1 Realización de la fase de Inicio.	
A.2 Realización de la fase de Requerimientos.	**Fase 1** – Especificaciones del Producto. **Fase 2** – Partición entre Hardware y software.
A.3 Realización de la fase de Análisis y diseño.	**Fase 3** – Iteración e Implementación.
A.4 Realización de la fase de Construcción.	**Fase 4** – Diseño detallado de Hardware y Software (Haciendo énfasis en el camino de actividades de diseño de Software, pero considerando al hardware). **Fase 5** – Integración de Hardware y Software.
A.5 Realización de la fase de Integración y Pruebas.	**Fase 6** – Pruebas de aceptación. **Fase 7** – Mantenimiento y actualización.
A.6 Realización de la fase de la fase de cierre.	(Liberación de Producto)

Tabla 9. Comparativa entre el modelo MoProSoft y el ciclo de vida del sistema embebido, propuesta propia.

Una vez demostrada la empatía entre los modelados, se denota en azul el punto donde se trabaja para la realización de

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

productos de salida, haciendo uso de las aportaciones respectivas de las áreas.

Los siguientes dos apartados redactarán las bases usadas para la construcción de cada producto así como describir cada uno de estos.

4.1.3 Desarrollo de producto OPE.2_DOC1_EDR_v1.0.0.2012.

El producto *OPE.2_DOC1_EDR_v1.0.0.2012*, presentado en el apéndice, ha sido creado para cubrir la documentación que solicita la actividad "A2. Especificación de Requerimientos" de la Categoría Operación OPE.2 del Modelo de MoProSoft.

La estructura del producto está pensada en la adaptación del mapeo para la actividad específica de la salida "Especificación de Requerimientos" tomando un enfoque hacia el sistema embebido y su software embebido.

A continuación se explica cada uno de los apartados:

1) **Control de cambios al documento:** Éste es un recuadro donde se visualiza la información general tal como: Título del producto, Nombre de la compañía, Logotipo de la empresa, Nombre del proyecto, Categoría MoProSoft que cubre, Nombre del proceso al que pertenece el producto, Consecutivo de la revisión (manejado por el alfabeto consecutivo de la A-Z), Fecha de inicio de elaboración de los requerimientos, Área o Departamento responsable del levantamiento de los requerimientos, Nombre de quien elaboró el documento de los requerimientos, Nombre de quien revisó el documento de los requerimientos, Nombre de quien aprobó el documento de los requerimientos.
2) **Introducción:** Apartado generado para definir las características iniciales del producto que se va a desarrollar. Se considera primordial especificar el nivel de riesgo del sistema para sustentar la base de los estándares que habrán de requerirse en a la hora del diseño y los componentes del sistema embebido (hardware y software)
3) **Descripción de requerimientos:** En este apartado se describe de manera más específica los detalles del sistema embebido y su software. Este apartado estará compuesto por:
   a) *Funcionalidad:* Este apartado contiene dos preguntas. La pregunta denotada con el reactivo "a" busca obtener

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

información acerca de la tarea específica del sistema embebido. En el caso del reactivo "b" la información que se pretende obtener es la función específica del software. Estas dos preguntas deben responderse de manera breve pero muy precisa para evitar ambigüedades u omisiones en el resto del proyecto.

b) *Interfaz con usuario:* Está compuesto por tres incisos. El inciso "a" agrupa los elementos de interacción más comunes entre el sistema y el usuario humano, del mismo modo da la oportunidad de especificar si se carece de interfaz o en su defecto si se desearan agregar más opciones se pueden incluir en modo de listado. En el inciso "b" con la intención de verificar el conocimiento de los componentes, sus especificaciones, dimensiones, planos y generalidades se solicita listar a los componentes y sus documentos, el resultado será poner énfasis en la información que aún no se tiene disponible para conseguirla desde inicio y evitar su omisión a futuro. Por último en el inciso "c" se solicita una descripción visual inicial del sistema usando bloques o diagramas para hacer mucho más tangible al producto.

c) *Interfaces externas:* En ésta parte del producto se solicita listar mediante una tabla a dos columnas las relaciones que tendrá el sistema embebido con otros componentes de software o hardware independientes al sistema embebido que será diseñado. Este apartado requirió un ajuste ya que inicialmente no se consideraba si el sistema recibiría o enviaría información o interacción. Se le agregó una división a cada columna con la finalidad de clasificar cuales serían las entradas que recibiría el sistema embebido y cuáles las salidas que este entregaría y a que modulo.

d) *Confiabilidad:* Este apartado se basa en la teor+ia detrás de la Ingeniería de la Calidad del Software para poder llevar a cabo una especificación del nivel de desempeño del software con respecto a la madurez, tolerancia a fallas y recuperación. Se compone de tres incisos. En el inciso "a" se solicita la información para generar un perfil de operación, que no es otra cosa que una lista a conciencia de las tareas del sistema organizadas en grado de importancia y asignándoles un valor de probabilidad para poder ser cuantificadas en caso de ser requerido a través de algún método estadístico a lo largo del proyecto. Una vez concluido ese trabajo se debe desarrollar el inciso "b" que se refiere a llevar a cabo una lista de severidades de fallas en orden creciente con una descripción de la

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

misma (que pasa en el sistema cuando sucede la falla) y por último que parte del sistema se ve afectado con dicha falla. Por último en el inciso "c" se solicita especificar la herramienta de simulación o estadística para verificar los resultados de la confiabilidad propuesta considerando los dos incisos anteriores de este apartado.

e) *Eficiencia:* En este apartado se solicita la especificación del nivel de desempeño del software con respecto al tiempo y a la utilización de recursos. Se proponen dos incisos para realizar esta tarea. EL inciso "a" nos solicita el nombre de alguna herramienta o método estadístico que evalúe los parámetros de tiempo y/o recursos del sistema. Mientras que en el inciso "b" se presenta un listado de posibles recursos consumidos por el procesador al momento de ejecutar las funciones de software. Cabe mencionar que la lista es enunciativa, mas no limitativa, por lo cual en el último inciso se pregunta por cualquier otro parámetro que no esté considerado en la lista pero que se pretenda usar para la medición. La lista fue creada considerando las recomendaciones en Berger, A. H. (2002).
f) *Mantenimiento:* Para este apartado se ha simplificado la teoría en Berger, A. H. (2002) para resolver las tareas del mantenimiento futuro al software y de esa manera tener una descripción clara de los elementos que facilitarán la compresión y la realización de las modificaciones futuras según lo solicita el manual del Modelo de Procesos.
g) *Portabilidad:* Dados los rápidos avances tecnológicos y cambios constantes en los procesadores, en este apartado se considera propio verificar la compatibilidad que tendrá el software en caso de cambiar un procesador, el sistema operativo en tiempo real o en su defecto al modificar alguna interface del sistema. Un supuesto al momento de tener que darse una escalación del sistema debe ser definido en este apartado al describir los requisitos mínimos y parámetros que se deben tomar en cuenta para no afectar o comprometer la integridad del sistema ante un cambio.
h) *Restricciones de diseño y construcción:* Al estar el proyecto limitado por un tope de inversión y por la información que el cliente tiene del proceso para el cual se desarrollará el sistema, es requerido en este apartado contemplar las necesidades impuestas por el cliente para evitar inconvenientes en la instalación del sistema y un exceso en los costos del desarrollo del

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

proyecto. La tabla solicita una descripción de cada rubro y especificar si se cuenta con el mismo, se solicitara o deberá desarrollarse.

i) *Legales y Reglamentarios:* Todos los desarrollos de sistemas embebidos se ven sujetos a cumplir con estándares, ya sea nacionales o internacionales, en su parte eléctrica, mecánica, ambiental, de patentes, entre otros. Por este motivo es necesario contemplar las leyes, reglamentos y/o estándares que el cliente desea preservar en su producto final. Para este apartado se tienen tres incisos que abarcan este tópico. El inciso "a" se refiere a las licencias que deben adquirirse para el desarrollo y también para lograr la ejecución del software en el sistema embebido abarcando desde el inicio del proyecto hasta su entrega y funcionamiento. El inciso "b" verifica si existe al momento del desarrollo en el mercado alguna patente o norma que se vaya a usar (ej. algoritmo patentado, configuración o arreglo de hardware). Por último en el inciso "c" se hace énfasis en la solicitud de estándares o protocolos que deba de considerar el equipo de desarrollo al crear el nuevo sistema.

El producto *OPE.2_DOC1_EDR_v1.0.0.2012* sufrió algunos ajustes basados en la experiencia y colaboración del equipo de MTE dadas las diferencias que existen entre la teoría, la práctica y la experiencia de un desarrollador. En el capítulo 4.2 se dará una semblanza comparativa del actual RFQ manejado por el equipo de MTE y el producto propuesto para verificar y completarlo con la aportación propuesta en la tesis.

4.1.4 Desarrollo de producto OPE.2_DOC2_AYD_v1.0.0.2012

El producto OPE.2_DOC2_AYD_v1.0.0.2012, presentado en el apéndice, ha sido creado para cubrir la documentación que solicita la actividad "A3. Análisis y Diseño" de la Categoría Operación OPE.2 del Modelo de MoProSoft.

La estructura del producto está pensada en la adaptación del mapeo para la actividad específica de la salida "Análisis y Diseño" tomando un enfoque hacia el sistema embebido y su software embebido.

De forma inicial se considerara seguir la propuesta en el indicada por la recomendación en Banks, S., etal. (1994,

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

June) partiendo de tres componentes esenciales para describir una arquitectura:

(1) La estructura del software del sistema.
(2) La definición de los componentes de la arquitectura.
(3) Las conexiones entre los componentes.

El reto para construir el producto OPE.2_DOC2_AYD_v1.0.0.2012 se deriva de la amplia gama de aplicaciones de los sistemas embebidos y en consecuencia de su software. El desear proponer una arquitectura única y a la vez abarcar todo tipo de aplicaciones se hace viable si se decide usar una propuesta general enunciativa, más no limitativa, para contener una estructura interna del sistema, es decir lograr la descomposición del sistema en subsistemas, identificando los componentes y relaciones e interacción que componen al conjunto. Para ello se hace uso de propuestas de investigaciones reportadas en casos específicos, el documento VDD Template usado por el equipo de MTE e ideas tomadas de la práctica para ser trasladadas.

Siguiendo la recomendación hecha en Capilla, R. (2009, September) "tal como Bosch declaró en 2004 [3], una arquitectura de software debe ser vista "como el resultado de un conjunto de decisiones de diseño más que un conjunto de componentes y conectores", se tiene que al momento de armar el producto se considera inherente considerar la parte de requerimientos donde se decidió la partición de hw/sw.

Por otra parte al inicio del inciso de Arquitectónica se propone una guía para llevar a cabo un diagrama de arquitectura, la cual toma la perspectiva dada en Gannod, G. C., etal. (2001, April) donde define a una arquitectura como una configuración de componentes y conectores, especificando que un componente es una encapsulación de una unidad computacional y tiene una interfaz que especifica las capacidades que el componente puede proveer; mientras que los conectores encapsulan la forma en la que los componentes interactúan; y por último la interconexión de los componentes con conectores determina la topología de la arquitectura y proporciona una vista estructural y sistemática de un sistema, donde las semánticas están proporcionadas por las especificaciones individuales de los componentes y conectores.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

La propuesta al momento de la construcción también toma a consideración el resumen hecho en Kang, B., etal. (2005, August) generalizando las fases básicas para el desarrollo de software embebido como sigue:

1. **Diseño de la Arquitectura de Software:** Es la estructura total del Sistema que es descrita por los componentes y las conexiones en medio de los componentes. Los estilos de arquitectura de software categorizan arquitecturas basadas en lo especifico a la composición estructural, tal como datos compartidos, datos de tipo abstracto, invocación implícita, pila y filtro.
2. **Definición del Subsistema de Software:** Un Sistema puede ser dividido en uno o más subsistemas. Cada subsistema es una unidad lógica o física a ser desarrollada. Un subsistema contiene un tipo de características que el sistema necesita soportar.
3. **Diseño de Bloques Funcionales:** Cada subsistema está dividido en uno a más bloques funcionales basados en características y tamaño manejable. Un bloque contiene varias tareas que acompañan las funciones características de un procesador.
4. **Características y Tarea de Implementación:** Diseñar una tarea significa que todas las interfaces entre las tareas deben de estar bien definidas. Todos los parámetros de los mensajes y los cálculos de tiempo deben de ser definidos en esta fase. Las tareas son implementadas por un lenguaje de programación, y entonces los programas fuente son almacenados en archivos de computadora.

Tomando esta secuencia en consideración, los elementos para armar la guía que generará el diagrama de la arquitectura tratan de contener toda la información que solicita. Aquí se requiere de la inventiva, ingenio, experiencia y pericia del desarrollador para dejar lo suficientemente clara su arquitectura.

Otra de las inclusiones en el desarrollo del producto es la consideración del trabajo de investigación presentado en Banks, S., etal. (1994, June) donde menciona que Architecture Description Language (ADL) proporciona un lenguaje preciso e inequívoco para describir y especificar los componentes arquitectónicos, interconexiones, restricciones y características dinámicas.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

Trabajos como los presentados en Russell, J. T., & Jacome, M. F. (2003, May) y Lakhani, F., & Pont, M. J. (2012, June) pudieron proporcionar ideas de cómo manejar al producto más no así elementos sustanciales de inclusión.

A continuación se definen concretamente cada uno de los elementos que componen el producto:

1) **Control de cambios al documento:** Éste es un recuadro donde se visualiza la información general tal como: Título del producto, Nombre de la compañía, Logotipo de la empresa, Nombre del proyecto, Categoría MoProSoft que cubre, Nombre del proceso al que pertenece el producto, Consecutivo de la revisión (manejado por el alfabeto consecutivo de la A-Z), Fecha de elaboración del documento, Área o Departamento responsable del análisis y diseño, Nombre de quien elaboró el documento de análisis y diseño, Nombre de quien revisó el documento de análisis y diseño, Nombre de quien aprobó el documento de análisis y diseño.

2) **Arquitectónica:** También conocida como Arquitectura del sistema debe de contener información acerca de la estructura interna del sistema (descomposición del sistema en subsistemas). Una identificación de los componentes que integran los subsistemas y las relaciones de interacción entre ellos debe ser representada. Para cubrir el apartado se presenta la sugerencia de una guía de símbolos que cubren el esquema recomendado en Banks, S., etal. (1994, June) y que consiste en los siguientes objetos:

    a) *Elemento:* Es la entidad básica dentro de una arquitectura, como mínimo tiene un mensaje de entrada o salida y también describe la descomposición estructural.
    b) *Conexión:* Define la topología de la arquitectura al definir qué elementos se comunican con otros, también describe en forma concisa las interconexiones específicas, define las formas de intercambio de información a través de la conexión, especifica los protocolos para las conexiones usadas y finalmente define las relaciones de control entre los componentes (ej. síncrona, asíncrona, etc.).
    c) *Tipo de conexión:* Define una clase e conexiones a ser usadas en la arquitectura.
    d) *Mensaje:* Es la unidad de arquitectura para la comunicación. Cada mensaje contiene tipos específicos

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

de información y es transmitida entre elementos específicos.

e) *Información:* Es el dato y su tipo contenido en un mensaje.

f) *Capacidad:* Es la cantidad permitida a usar para transporte o almacenamiento de la información. Debe especificarse la unidad de medición.

g) *Restricciones de plataforma:* Es una forma de registro de cliente o requerimientos derivado para la plataforma de entrega que pertenece a la arquitectura; esto incluye el procesador físico, conectores, periféricos y su topología de conexión.

h) *Recuadro de arquitectura:* Para llevar un control de la secuencia del sistema se plantea el uso de espacios que alberguen el diseño de la arquitectura del sistema general hacia los subsistemas que lo integran. Para organizar y controlar las secuencias se debe manejar un título por cada espacio de diseño usando la nomenclatura: Diagrama [A..Z].[0..N].ver[a..z]; fecha elaboración [SS/MM/AAAA]; responsable []. La información entre corchetes deberá cambiar de manera progresiva conforme se integren subsistemas al diseño de la arquitectura. Cabe mencionar que la letra A siempre corresponderá a la arquitectura del sistema general.

3) *Detallada:* Una vez que se ha descrito gráficamente la arquitectura a través de los diagramas de diseño en este apartado debe registrarse el detalle de los componentes para que permita de manera evidente su construcción y prueba al programar. La estructura de diseño consiste en describir los siguientes objetos recomendados por Banks, S., etal. (1994, June):

a) *Modulo:* Representa una elección de diseño para implementar y empaquetar una o más funciones que se incluyen y son usadas en la arquitectura. También lista los requerimientos que este módulo suministrará y las operaciones que son necesarias para este módulo.

b) *Operación:* Es una descripción formal de una función abstracta y consiste de los componentes parametrizados los cuales pueden ser variados para permitir el mismo tipo de operación pero en diferentes entidades de dominio.

c) *Datos:* Describe una pieza específica de dato a medida que se utiliza en una aplicación particular.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

El formato y contenido del dato está dado por su tipo de dato, también el dato puede ser instanciado y usado en muchos lugares para muchas aplicaciones.

d) *Tipo de dato:* Describe la definición del dato el cuál puede ser usado en muchos lugares para muchas aplicaciones. Describe la definición del dato la cual puede ser reusada a través de la instanciación en muchos lugares. Básicamente representan los Abstract Data Types (ADT).
e) *Archivo:* Describen el formato físico del dato en un disco físico.
f) *Comunicación:* En diseño las comunicaciones son muy similares en contenido a las conexiones en la arquitectura. Definen la topología del diseño al enunciar cuales operaciones se comunican mutuamente. Describen el mecanismo por el cual el dato obtiene desde una operación a otra y también describen las propiedades generales de la comunicación.
g) *Tipos de comunicación:* Define una clase de comunicación a ser usada en el diseño y son muy similares en contenido a los tipos de conexión de la arquitectura. También contienen los siguientes dos modelos: Cooperación y comunicación.
    I. Modelos de cooperación: Describen la relación entre dos operaciones.
    II. Modelos de comunicación: Describen como dos operaciones se comunican mutuamente.

Esta tabla de información debe de llenarse por cada una de los diagramas de arquitectura propuestos, es decir a cada sistema y subsistema le corresponde una de estas tablas, motivo por el cual debe preservarse la nomenclatura en la parte superior: Diagrama [A..Z].[0..N].ver[a..z]; fecha elaboración [SS/MM/AAAA]; responsable [].

4.2 VALIDACIÓN DE INVESTIGACIÓN.

Para la validación de la investigación se ha propuesto una semblanza y comparación entre los documentos generados para el proyecto de Final Vision System desarrollado en Flextronics Manufacturing por el equipo de MTE y los productos de la tesis generados.

Dado el carácter delicado de la información del proyecto y las políticas internas de confidencialidad, los datos específicos se han omitido así como el nombre del cliente

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

para el cuál fue desarrollado, garantizando la política interna pero proporcionando información suficiente para el ejercicio académico de la presente tesis.
Se han considerado los puntos mencionados por el modelo de MoProsoft para hacer la semblanza a través de la tabla 10 de semblanza y comparación donde en el encabezado tenemos el nombre del producto evaluado y marcado con una X los requisitos que cumple cada uno conforme al modelo.

Los siguientes documentos fueron proporcionados por el equipo de MTE de Flextronics Manufacturing y son incluidos en la tabla 10 de semblanza y comparación:

- R. F. Q.: Request for Quotation (for Project).
- F. V. S.: Proposal (for Project).
- V. D. D.: Version Description Document (for Software Design).

Por otra parte están los productos generados en la presente tesis:

- Especificación de Requerimientos: OPE.2_DOC1_EDR_v1.0.0.2012
- Análisis y Diseño: OPE.2_DOC2_AYD_v1.0.0.2012

A continuación se presentan los resultados obtenidos basados en los documentos proporcionados por MTE para documentar el proyecto de Final Vision System y los productos generados en la tesis, los cuales en el encabezado solo son mencionados por el número de documento y el acrónimo de la actividad por cuestiones de espacio. El enfoque tomado para la semblanza y comparación es desde la perspectiva de desarrollo del software, aunque en los casos que lo ameritan se toman consideraciones de hardware tal como en las interfaces externas por ejemplo.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

A2. ESPECIFICACIÓN DE REQUERIMIENTOS					
Elementos de la actividad bajo MoProSoft	**Proyecto Final Vision System**			**Productos de Tesis**	
	R.F.Q.	**F.V.S.**	**V.D.D.**	**DOC1_EDR**	**DOC2_AYD**
Introducción.	X		X	X	
Funcionales.		X		X	
Interfaz con usuario.			X	X	
Interfaz externa.	X	X	X	X	
Confiabilidad.	X			X	
Eficiencia.		X	X	X	
Mantenimiento.			X	X	
Portabilidad.				X	
Restricciones de diseño y construcción.	X	X		X	
Legales y Reglamentarios.	X			X	
A3. ANÁLISIS Y DISEÑO					
Arquitectónica (Elemento, Conexión, Tipo de conexión, Mensaje, Información, Capacidad y Restricciones de plataforma).				X	X
Detallada (Modulo, Operación, Dato(s), Tipo de dato, Archivo, Comunicación, Tipos de comunicación).			X*	X	X

Tabla 10. Semblanza y comparación de productos.

*Sólo algunos elementos son descritos

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

CAPÍTULO 5

CONTRIBUCIONES ESPERADAS

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

5. CONTRIBUCIONES ESPERADAS.

5.1 PRODUCTOS DE INVESTIGACIÓN LOGRADOS.

En este apartado se da a conocer la evolución que ha tenido el trabajo de investigación, mostrando desde el planteamiento inicial hecho a la comunidad interesada en el área de Ingeniería de Software, hasta los últimos resultados de aportaciones por desarrollado a lo largo del periodo de investigación como productos logrados.

Artículo 1: En ese orden de ideas primeramente se presentó a revisión para la convocatoria de Conisoft 2012 el artículo: *"Esfuerzo de Implementación de la Categoría de Operación – Administración de Proyectos Específicos – del Modelo MoProSoft"*, enfocado al Desarrollo de Software Embebido usando el modelo MoProSoft en Categoría de Operación.

El artículo plantea y sustenta, en el marco teórico, que actualmente existe una problemática respecto a la escasa información y práctica de cómo usar Ingeniería de Software - modelos y metodologías- en el desarrollo de software en general, tendencia que se extiende en consecuencia al Software Embebido. Así mismo expone de manera general tres contribuciones al uso del modelo aplicado al Software Embebido.

Artículo 2: El artículo presentado en Conisoft 2013 lleva por título "Modelo MoProSoft, una Adaptación de la Categoría Operación, Proceso Desarrollo y Mantenimiento de Software en "Especificación de Requerimientos" al Proceso de Desarrollo de Software Embebido, Caso de Estudio: NI cRIO-9074 de National Instrument para Proyecto Final Vision System de Flextronics Manufacturing".

La investigación explica a través del enfoque la Ingeniería de Software que en la actualidad, existe una cantidad importante de trabajos que mencionan modelos de uso internacional para la estandarización de procesos de desarrollo de software, sin embargo la implementación de éstos implica la inversión a largo plazo en tiempo, recursos y esfuerzo, resultado costeables únicamente para las organizaciones grandes, es decir con más de 50 empleados o más de 20 proyectos.

También plantea en forma complementaria a través de su marco teórico que ha sido difícil el uso de modelos en la industria

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

para aplicaciones en los Sistemas Embebidos que implican desarrollo de Software Embebido. Reportadas se encontraron menos prácticas debido a la amplia variedad de aplicaciones que manejan y al nivel complejo que en ocasiones llega a tener su construcción.

Una vez que se unificó y justifico la visión de estas dos premisas se presentó el resultado del desarrollo de la adaptación del modelo de procesos de software MoProSoft en su categoría OPE.2, para el producto de la salida en la actividad A2 "Especificación de Requerimientos" para desarrollo del software embebido usando pruebas de ajuste para su validación con el proyecto Final Vision System de Flextronics Manufacturing a través de la colaboración del equipo de trabajo en el departamento Manufacturing Technology Engineering (MTE), que por sus características se considera un grupo pequeño enfocado al desarrollo de herramientas automatizadas que incluyen software embebido.

5.2 CONTRIBUCIONES AL CONOCIMIENTO.

Artículo 1: Se considera una buena aportación el visualizar el uso del modelo de MoProSoft como una metodología de apoyo al desarrollo de Software Embebido, porque el mismo modelo desde sus inicios contiene la visión de estar enfocado a industrias PyME con la ventaja de poder adaptarse de manera rápida a procesos no muy grandes. Sustentado en el marco teórico del artículo está que el software embebido, a pesar de su complejidad, se desarrolla en equipos pequeños, considerándose entonces bastante positivo el esfuerzo de adaptar MoProSoft en este tipo de proyectos.

Por otra parte cabe mencionar que en la ponencia del Congreso Conisoft 2012, especialistas en el área mostraron su entusiasmo en darle seguimiento a la adaptación del Modelo para proyectos de Software Embebido.

Artículo 2: La principal contribución al conocimiento es la aportación a través de la investigación conceptual para plantear la integración de dos áreas del conocimiento: la Ingeniería de Software y el desarrollo del Software Embebido en los Sistemas Embebidos bajo modelos para ser llevados al contexto real del desarrollo de esas aplicaciones a través de la generación de productos que están incluidos en normas o modelos.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

5.3 CONTRIBUCIONES A LA PRÁCTICA.

Artículo 1: Los pasos recomendados en la contribución parten primero de indicar como delimitar el proyecto. En nuestro caso se uso la delimitación del campo de acción en un subproceso, pensado de la siguiente manera:

> **Proyecto:** Sistemas embebidos.
> **Proceso:** Diseño y desarrollo de Software Embebido, módulo de CPU.
> **Subproceso:** Programación de microcontrolador.

En segundo lugar se plantea que se deben conocer las características del proceso OP1 Administración de Proyectos Específicos de la Categoría de Operación del Modelo de MoProSoft, el cual tiene como propósito establecer y llevar a cabo sistemáticamente las actividades que permitan cumplir con los objetivos de un proyecto en tiempo y costos esperados (Oktaba H., Alquicira E. C., Su R. A., et al. 2005). Esto llevó a la segunda contribución: indicar como trasladar la Actividad X del Proceso OP1, a la aplicación en el subproceso planteado en el primer paso. La tabla A, da un ejemplo de la forma en la cual se planteó.

A1. Planificación (O1)

Rol	Descripción
RGPY RAPE RDM	A1.1. Establecer los requerimientos para Descripción del Proyecto del subproceso programación de microcontroladores a diseñar con el Responsable de Gestión de Proyectos. Herramienta y Documentos usados para Descripción de Proyecto: [13] *Enfoque del Marco Lógico (EML)*, es una herramienta analítica para la planificación y gestión de proyectos orientada por objetivos. *La Ficha de Identificación*, es un instrumento de trabajo que sistematiza y condensa toda la información del proyecto disponible hasta ese momento y ofrece un marco de referencia homogéneo de la gran cantidad de aspectos a considerar manteniendo la idea de la posible recepción posterior de nuevos datos, es de carácter dinámico y es susceptible de tener sucesivas redacciones y cambios

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

	hasta su aprobación definitiva. Esto generará el *Documento de Formulación* que es la declaración formal de la Descripción del Proyecto.

Tabla A. Ejemplo de la Actividad A1. Planificación de creación propia.

Con este ejemplo se denota en el artículo como surgieron las adaptaciones de las actividades generales (A1.1 a A1.18) de Planificación propuesta por el modelo MoProSoft para la Categoría OP1 en Administración de Proyectos, al software embebido usando las siguientes características:

I. La descripción de la actividad va subrayado el producto que se desea obtener (ej. Descripción de Proyecto).
II. En el párrafo "Descripción" se hace énfasis en el producto que se desea revisar o generar en esta actividad (producto de entrada, salida o interno).
III. Se desglosa en ese mismo apartado las herramientas y/o documentos haciendo uso de un hipervínculo referenciado al material que se sugiere usar, denotándolo subrayado y en cursivas (Ej. Enfoque del Marco Lógico (EML)) incluyendo una breve descripción.
IV. También se hace énfasis en marca en el inicio la referencia bibliográfica correspondiente a la información encontrada para usar (ej. [13]).

La tercera sugerencia de contribución se enfoca en explicar cómo se usa para reportar la redistribución en forma de diagrama las salidas del proceso para un mejor manejo de la información, la figura 17 muestra un ejemplo.

En ese momento se pensó que trasladar el manejo de salidas, en cada actividad de cada categoría del Modelo MoProSoft, ejemplificaría de manera específica una forma de hacer más sencilla la vinculación entre las tres categorías del modelo al finalizar el mapa del proyecto, logrando hacer más visual el proceso y evitando omisiones de documentación en cualquier fase de desarrollo del proyecto. También se enfatiza el uso de los colores en el diagrama tal como lo indica el manual de MoProSoft para poder evaluar el nivel en el que se están cubriendo las actividades.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

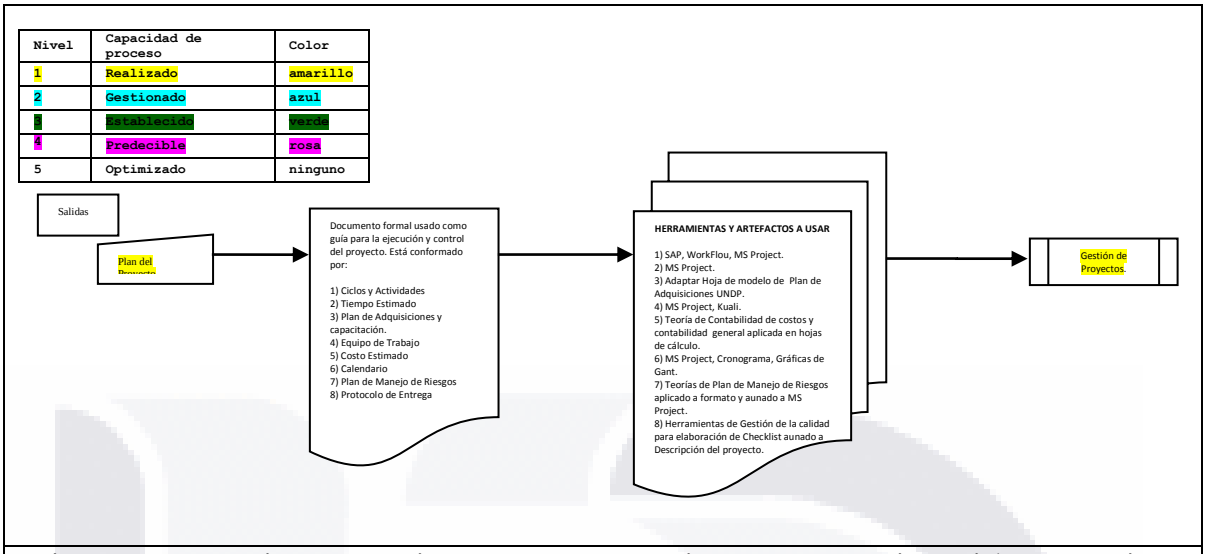


Figura 17. Ejemplo diagrama de salidas. Contribución propia.

Artículo 2: En este caso la aportación principal es la presentación del producto para la actividad A2. Especificación de Requerimientos del proceso OPE.2, el cual ya ha sido incluido en los anexos de la presente tesis. Dicho producto alberga la consistencia indicada en el manual de MoProSoft desde la perspectiva de los sistemas embebidos y la forma en la cual se lleva a cabo el desarrollo del software contenido en el mismo.

5.4 RECOMENDACIONES PARA INVESTIGACIONES SUBSECUENTES.

Artículo 1: El software embebido a pesar de ser tratado con metodologías de UML Statechars [2] y Metodología de Codiseño hardware y software [3] como se menciona en la Introducción del artículo, no ha logrado la completa aceptación de los diseñadores debido a su robustez, por lo cual se pretende hacer de esta adaptación de diagrama de entradas y salidas, así como las sugerencias de algunas herramientas o artefactos de registro, una forma de solucionar esta renuencia a documentar el proceso de generación de software embebido a la par de su desarrollo.

Adicionalmente, será recomendable una retroalimentación de los procesos para depurar o cambiar herramientas de uso general provenientes de otras áreas (teorías administrativas, contables, presupuestales, de riesgos, calidad, etc.) y dejar abierta la posibilidad de generación de nuevas herramientas para trabajos futuros de colaboradores y colegas interesados en el tema.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

Artículo 2: Al entregar el artículo se tenía en proceso de ajuste al producto OPE.2_DOC1_EDR_v1.0.0.2012, con la colaboración del equipo de MTE para modificarlo o restructurarlo. También se encontraba en desarrollo el producto OPE.2_DOC2_AYD_v1.0.0.2012 que cubre en la Categoría Operación del proceso Desarrollo y Mantenimiento de Software el producto de salida de la Actividad A3: "Análisis y Diseño". En ese momento se consideraba que su desarrollo implicaría un reto considerable ya que contiene la descripción textual y gráfica de la estructura de los componentes de software y para eso se requiere conocer las arquitecturas que los diferentes sistemas embebidos manejan y sugerir las más exitosas o dar una idea general para que el equipo desarrollador pueda proponer la suya propia siguiendo las características del modelo de MoProSoft.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

CAPÍTULO 6

DISCUSIÓN DE RESULTADOS

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

6. DISCUSIÓN DE RESULTADOS

Una vez llegado a este punto es necesario recordar que al inicio de la investigación se tenía el objetivo *O1*, indicado en el capítulo 2.2 junto con la pregunta de investigación.

Se concluyen que, en referencia al desarrollo de la investigación hecha, al haber estudiado la problemática de falta de modelos, métodos y/o metodologías a través de la investigación conceptual, lo cual fue mostrado y demostrado en los capítulos 3 y 4, se cumplió el objetivo al conjuntar el conocimiento para lograr la adaptación del modelo de MoProSoft en la Categoría Operación OPE.2 Desarrollo y Mantenimiento de software para las Salidas de "Especificación de Requerimientos" y "Análisis y Diseño" al Software Embebido usando como validación el caso de desarrollo de software en NI cRIO-9074 para el Proyecto Final Vision System.

En referencia a la comprobación de las proposiciones de investigación planteadas en el capítulo 2.3, que van en el mismo sentido del objetivo sólo cubren la parte del análisis de la Categoría Operación de MoProSoft, la búsqueda de técnicas en los sistemas embebidos enfocadas al software embebido y las herramientas en su desarrollo para llevarlo a la especificación como modelo general. Pero el verificar que la aplicación de este conocimiento no logrará o lograra garantizar la calidad del producto final (software embebido) quedó pendiente de sustentar, ya que la validación dué hecha con un producto ya terminado y liberado a su cliente, es decir ya contaba con el calificativo de "calidad" por así decirlo.

Pero en referencia a la parte que menciona la facilitación y delimitación de los requerimientos y la documentación del análisis y diseño del ingeniero desarrollador podemos decir que al ver los resultados de la tabla de semblanza comparativa si hay áreas de oportunidad para incluir mejoras en la documentación que se había llevado en el proyecto de Final Vision System, es decir con un esfuerzo enfocado al proceso de desarrollo de software específicamente, pues los documentos de MTE son bastante gráficos, consistentes y consideran otros detalles como los costos, riesgo, planeaciones, mucha referencia a funcionalidades del equipo en hardware, que son elementos indicados en otros procesos del modelo de MoProSoft o incluso no considerados, pero hay pequeñas porciones muy específicas de la documentación para el software que pueden ser incluidas y se han omitido. Esto

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

podría ayudarles a manejar en un futuro un repositorio de software basado en la documentación de la arquitectura de sistemas anteriores logrando un estándar común para miembros del equipo.

Algo que quedó pendiente es la transcripción al idioma inglés de ambos productos dada la talla internacional de Flextronics Manufacturing, que dadas cuestiones de tiempo resultó no posible pero que se espera completar y mejorar en un futuro.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

CONCLUSIONES

CONCLUSIONES DE APRENDIZAJE PERSONAL

Acerca del aprendizaje personal y como conclusión tomo el desarrollo de la investigación como un viaje bastante arduo y apasionante, en el cual al inicio pudo parecer que nada estaba muy claro, que el tema era muy amplio y difuso, pero conforme se avanzó las cosas poco a poco fueron tomando su lugar y para los últimos periodos de trabajo la idea era mucho más concreta y se divagaba menos en lo que era posible llegar a hacer y proponer.

Hubo momentos difíciles en los cuales la creatividad se veía mellada por cuestiones o factores ajenos al desarrollo de la investigación, sin embargo el verdadero amor al conocimiento, los buenos consejos de las personas involucradas de manera profesional o familiar, siempre trajo de vuelta al camino que llevó a la conclusión del trabajo.

CONCLUSIONES FINALES

Las aportaciones presentadas en esta tesis no pretenden ser un límite de aplicación del modelo de MoProSoft ni de otro modelo desarrollado en la Ingeniería de Software hacia los Sistemas Embebidos. Actualmente hay estándares internacionales que pueden ser retomados para nuevos planteamientos. En el camino de la presente investigación escuché a expertos en el área de pruebas y desarrollo de sistemas embebidos (Intel, TI, IBM) decir que hay mucho por hacer en esta área y que intentar desarrollar algo en ese ámbito puede producir buenos frutos.

Algo interesante a decir es que se aprendió que un modelo no es ni bueno ni malo, todo depende siempre de la aplicación donde se desee usar. Es decir, basados en las necesidades del producto de software a desarrollar y los recursos se requiere pensar el modelo que más ayude a lograr el objetivo del proyecto.

Esperando que el presente trabajo sea un aliciente a continuar en la búsqueda para las futuras generaciones o bien un principio de mejora, en él dejo un esfuerzo considerable y las mejores esperanzas de seguir avanzando.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

GLOSARIO

Artefactos:	También nombrados como productos son desarrollos para control de procesos de software bien identificados.
BigEndian/ LittleEndian:	El término inglés endianness ("extremidad") designa el formato en el que se almacenan los datos de más de un byte en un ordenador. Usando este criterio el sistema "big-endian" adoptado por Motorola entre otros, consiste en representar los bytes en el orden "natural": así el valor hexadecimal 0x4A3B2C1D se codificaría en memoria en la secuencia {4A, 3B, 2C, 1D}. En el sistema "little-endian" adoptado por Intel, entre otros, el mismo valor se codificaría como {1D, 2C, 3B, 4A}, de manera que de este modo se hace más intuitivo el acceso a datos, porque se efectúa fácilmente de manera incremental de menos relevante a más relevante (siempre se opera con incrementos de contador en la memoria), en un paralelismo "lo importante no es como empiezan las cosas, sino como acaban."
Debbugging:	Proceso de depuración, corrección de errores o bugs.
Ethernet:	Es un estándar de redes de área local para computadores con acceso al medio por detección de la onda portadora y con detección de colisiones (CSMA/CD). El estándar define las características de cableado y señalización de nivel físico y los formatos de tramas de datos del nivel de enlace de datos del modelo OSI.
EvalProSoft:	Técnica para la evaluación de la implementación de la metodología MoProsoft.
Hardware:	Conjunto de elementos físicos o materiales que constituyen una computadora o un sistema informático.
Ingeniería de Software:	Es el estudio de los principios y metodologías para el desarrollo y mantenimiento de sistemas software.
ISO 12207:	Information Technology / Software Life Cycle Processes, es el estándar para los procesos de ciclo de vida del software de la organización ISO.
ISO 15504:	También se le conoce como Software Process Improvement Capability Determination (SPICE), en español: Determinación de la Capacidad de Mejora del Proceso de Software. Es un modelo para la mejora, evaluación de los procesos de desarrollo, mantenimiento de sistemas de información y productos de software.
ISO 29110:	Es una nueva serie de estándares internacionales con el título de "Ingeniería de Software — Perfiles de ciclo de vida para pequeñas organizaciones (VSEs).
ISO 9001:2000:	Es la Norma que especifica los requisitos para un Sistema de Gestión de la Calidad (SGC) que pueden utilizarse para su aplicación interna por las organizaciones, sin importar si el producto o

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

	servicio lo brinda una organización pública o empresa privada, cualquiera que sea su tamaño, para su certificación o con fines contractuales.
ISO/IEC JTC1/SC7:	Ingeniería de software y sistemas es un subcomité de estandarización de la unión del comité técnico ISO/IEC JTC 1 de la Organización Internacional para la Estandarización (ISO) and la Comisión Internacional Electrotécnica (IEC), que desarrolla y facilita los estándares en el campo de la ingeniería de productos de software y sistemas. La secretaría internacional de ISO/IEC JTC1/SC7 es el Consejo de Estándares de Canadá (SCC) localizado en Canadá.
ISO/IEC 14598:	Norma de diferentes etapas que establece un marco de trabajo para evaluar la calidad de los productos de software proporcionando, métricas y requisitos para los procesos de evaluación de los mismos. En particular, es utilizada para aplicar los conceptos descritos en la norma ISO/IEC 9126. Se definen y describen las actividades necesarias para analizar los requisitos de evaluación, para especificar, diseñar y realizar acciones de evaluación y para concluir la evaluación de cualquier tipo de producto de software.
ISO/IEC 15939:	También nombrado ISO/IEC 15939:2007 define un proceso de mediciones aplicable a los sistemas e ingeniería de software y disciplinas de administración. El proceso es descrito a través de un modelo que define las actividades de los procesos de medición que son requeridos para adecuadamente especificar que información de la medición es requerida, como las mediciones y los resultados del análisis deben ser aplicados, y cómo determinar si los resultados del análisis son válidos. El proceso de medición es flexible, capaces de adaptarse a un propósito específico, y adaptables a las necesidades de diferentes usuarios.
ISO/IEC 90003: 2004:	Es la norma que proporciona una guía para las organizaciones en la aplicación de ISO 9001 en la adquisición, provisión, desarrollo, operación y mantenimiento de software y servicios de soporte relacionados.
ISO/IEC TR 9126:	Conocida como ISO/IEC TR 9126-4:2004 proporciona la calidad en el uso de las métricas para medir los atributos definidos en ISO/IEC 9126-1. ISO/IEC TR 9126-2 define las métricas externas e ISO/IEC TR 9126-3 define las métricas internas para la medición de las subcaracterísticas definidas en ISO/IEC 9126-1. La métricas Internas miden al software en sí mismo, las métricas externas miden la conducta del sistema de cómputo base que incluye al software, y la calidad en usar métricas de medición al efecto de usar al software en un específico contexto de uso.
Microcontroladores:	Circuito integrado programable, capaz de ejecutar las órdenes grabadas en su memoria. Está compuesto

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

	de varios bloques funcionales, los cuales cumplen una tarea específica. Un microcontrolador incluye en su interior las tres principales unidades funcionales de una computadora: unidad central de procesamiento, memoria y periféricos de entrada/salida.
Microprocesadores:	Circuito integrado central y complejo de un sistema informático; se le suele llamar por analogía el «cerebro» de un computador y está conformado por millones de componentes electrónicos.
Middleware:	Es un software que asiste a una aplicación para interactuar o comunicarse con otras aplicaciones, software, redes, hardware y/o sistemas operativos. Simplifica el trabajo de los programadores en la compleja tarea de generar las conexiones que son necesarias en los sistemas distribuidos. De esta forma se provee una solución que mejora la calidad de servicio, seguridad, envío de mensajes, directorio de servicio, etc.
Modelo de Proceso de Software:	Estrategia también conocida como paradigma de Ingeniería del Software y se usa para resolver los problemas reales de una industria incorporando una estrategia de desarrollo que acompañe al proceso con métodos y herramientas.
NI cRIO-9074:	Controlador Integrado en Tiempo Real de 400 MHz y FPGA de 2M de Compuertas.
NMX-I-006/02 -NYCE-2006:	Norma Mexicana para Tecnología de la Información - Evaluación de los procesos.
NMX-I-059/04 -NYCE-2005:	Norma Mexicana para Tecnología de la Información - Software - Modelos de procesos y evaluación para desarrollo y mantenimiento de software.
NYCE:	Organismo Nacional de Normalización y Certificación en materia de Tecnologías de Información, Electrónica y Telecomunicaciones.
Procesadores:	Componente electrónico donde se realizan los procesos lógicos.
Sistema Operativo:	También nombrado con el acrónimo SO u OS del inglés, es un programa o conjunto de programas que en un sistema informático gestiona los recursos de hardware y provee servicios a los programas de aplicación, ejecutándose en modo privilegiado respecto de los restantes y anteriores próximos y viceversa.
Sistemas Embebidos:	También conocido como sistemas empotrados o ciberfísicos son combinación de hardware computacional y software, quizá mecánica de manera adicional u otras partes, diseñadas para ejecutar una función dedicada.
Software Embebido:	Se refiere a los sistemas de cómputo que residen dentro de productos con algún elemento de procesamiento de datos o señales. Éste software forma parte de un sistema embebido.
Software:	Conjunto de programas y rutinas que permiten a la computadora realizar determinadas tareas.
Statecharts:	Diagramas de estado que han ganado un amplio uso desde que una variante ha llegado a ser parte de

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

	UML. El tipo de diagrama permite el modelado de superestados, regiones ortogonales y actividades como parte de un estado.
Testing:	En pruebas de software (software testing) son las investigaciones empíricas y técnicas cuyo objetivo es proporcionar información objetiva e independiente sobre la calidad del producto a la parte interesada (stakeholder). Es una actividad más en el proceso de control de calidad.
WG24:	Es un grupo de trabajo de ISO que se ha mantenido para dirigir las dificultades que enfrentan las empresas con menos de 25 empleados desarrollando estándares y guías más adecuados para estas empresas también conocidas como VSEs.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

BIBLIOGRAFÍA

DE LA METODOLOGÍA DE INVESTIGACIÓN

Eco, U. (1977). Cómo se hace una tesis. Barcelona, España: Editorial Gedisa.

Forward, A., & Lethbridge, T. C. (2008). A taxonomy of software types to facilitate search and evidence-based software engineering. In Proceedings of the 2008 conference of the center for advanced studies on collaborative research: meeting of minds (p. 14). ACM.

Instituto Internacional de Integración (S/F). La Investigación cualitativa, Bolivia. Recuperado el 31 de octubre de 2013 de http://www.iiicab.org.bo/Docs/doctorado/dip3version/M2-3raV-DrErichar/investigacion-cualitativa.pdf

Kerlinger, F. (1988). Investigación del Comportamiento. Capítulo 3: Constructos, Variables y Definiciones, Segunda Edición. México: McGraw-Hill.

Mora, T. M. Universidad Autónoma de Aguascalientes (2011). Material de curso Metodología de la Investigación. Sistemas de Información Centro de Ciencias Básicas. Artículo licenciado bajo Developing Nations 2.0.

Runeson, P., & Höst, M. (2009). Guidelines for conducting and reporting case study research in software engineering. Empirical Software Engineering, 14(2), 131-164.

DE LOS MODELOS DE PROCESOS

Alarcón Aldana, A. C., González Sanabria, J. S., & Rodríguez Torres, S. L. (2011). Guía para Pymes Desarrolladoras de Software, Basada en la Norma ISO/IEC 15504. Revista Virtual Universidad Católica del Norte, 1(34), 285-313.

Álvarez, R. F. J., etal. (2008). Interpretación del Modelo de Madurez de Capacidades (CMM) para Pequeñas Industrias de Software, Universidad Autónoma de Aguascalientes. 420 p.

Angeleri, P. M. (2009).Alternativas para mejorar la Calidad del Software. Revista UBit; Año 2, n°3. Recuperado el 31 de enero de 2013, de:

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

http://repositorio.ub.edu.ar:8080/xmlui/handle/123456789/898.

Barajas S. A., Álvarez R. F. J., Muñoz A. J., Muñoz L. J. (2009). RADIP: A Software Development Process for Mexican PyMEs. Advances in. Computer Science and Engineering , vol. 34, 311-322.

Barajas, S. A., Muñoz, A. J., Álvarez, R. F. J., & García, G. A. R. (2009). Developing Large Scale Learning Objects for Software Engineering Process Model through MIDOA Model. In 2009 *Mexican International Conference on Computer Science (ENC). México, Sept* (pp. 21-25).

Caja Costarricense de Seguro Social (2008). Procedimiento de Control de Versiones TIC-GAC-0005. Gerencia Infraestructura y Tecnologías, Subgerencia Tecnologías de Información y Comunicaciones: Área de Soporte Técnico [Versión 2.2.0].

Delgado, P. Y. R., Fuentes, M. L. Y. M., Javier, F., Rodríguez, Á., & Arteaga, J. M. Diseño de un Instrumento de Auto-evaluación para Diagnosticar el Estatus de las Organizaciones en México con Respecto al Modelo ProSoft: Proceso de Gestión de Procesos de la Categoría de Gestión., 78-86.

Delgado, P. Y. R., Fuentes, M. L. Y. M., Rodríguez, F. J. Á., & Arteaga, J. M. (2009). Aplicación de Instrumento Diagnóstico en Proceso "Gestión de Procesos" con base en MoProSoft. Investigación y Ciencia, (43), 30-37.

Díaz, N. H., & García, B. C. (2012). Métricas de Riesgos para los Proyectos Productivos de la Universidad de las Ciencias Informáticas. Revista Cubana de Ciencias Informáticas, 6(2).

Dybå, T. (2003, September). Factors of Software Process Improvement Success in Small and Large Organizations: An Empirical Study in the Scandinavian Context. In ACM SIGSOFT Software Engineering Notes (Vol. 28, No. 5, pp. 148-157). ACM.

Espinosa-Curiel, I. E., Rodríguez-Jacobo, J., & Fernández-Zepeda, J. A. (2010). Graphical Technique to Support the Teaching/Learning Process of Software Process Reference Models. Systems, Software and Services Process Improvement, 13-24.

Gallegos, V. A., & Ortíz, R., P. A. (2011). Elaboración del Estándar de Aplicación de la Norma ISO/IEC 12207, al

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

Desarrollo de Aplicaciones de Software para la UTIC de la ESPE.

Garcia, I., Pacheco, C., Cruz, D., & Calvo-Manzano, J. (2012). Implementing the Modeling-Based Approach for Supporting the Software Process Assessment in SPI Initiatives Inside a Small Software Company. Software Engineering Research, Management and Applications 2011, 1-13.

García, R. C. I. (2001). El Modelo de Capacidad de Madurez y su Aplicación en Empresas Mexicana de Software. "Capítulo 5: El modelo de Capacidad de Madurez y su Aplicación en Empresa Mexicana de Software", tesis presentada en la Escuela de Ingeniería, Universidad de las Américas, Puebla.

García-Mireles, G. A., & Rodríguez-Castillo, I. (2009, September). Software Engineering Area Curricular Evaluation Method Based in MoProSoft. In Computer Science (ENC), 2009 Mexican International Conference on (pp. 272-279). IEEE.

Garzón B. G. (2009). Presentación de: Introducción a la Norma Mexicana NMX-I-006/02-NYCE-2006. Recuperado el 31 de enero de 2013, de http://www.comunidadmoprosoft.org.mx/

Gasca, E. G., Gutiérrez, A. F., Rojas, A. P., & López, L. M. (2009). Acerca de la implementación de los modelos de calidad en la construcción de software en México. Tecnura, 13(25), 116-127.

Gil, R. A. C. (2006). Estructura básica del proceso unificado de desarrollo de software. Sistemas & Telemática, 2(3).

Guardati, S., & Ponce, A. (2011). Guía de pruebas de software para MoProSoft. REICIS: Revista Española de Innovación, Calidad e Ingeniería del Software, 7(2), 28-47.

Hanna Oktaba. "Tejiendo nuestra red". Revista Software Guru Ed. Año 01 No. 03, 05 de 2005; Año 02 No. 04 de 2006; Año 03 No. 1 de 2007. Febrero 2007.

Hauck, J. C. R. (2012). Um Método de aquisição de conhecimento para customização de modelos de capacidade/maturidade de processos de software.

Hauck, J. C. R., & von Wangenheim, C. G. Consolidação de um Método para Customização de Modelos de Capacidade/Maturidade de Processos de Software. Qualidade e Produtividade em

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

Software Projetos. Ciclo 2011, 55.

International Organization for Standarization (2004). ISO/IEC 90003:2004 Software engineering -- Guidelines for the application of ISO 9001:2000 to computer software. Consultado el 3 de septiembre de 2012 de http://www.iso.org/iso/catalogue_detail?csnumber=35867

Laporte, C. Y., Alexandre, S., & O'Connor, R. V. (2008). A Software Engineering Lifecycle Standard for Very Small Enterprises. Software Process Improvement, 129-141.

Mc Caffery, F., & Coleman, G. (2007). The Development of a Low-Overhead Assessment Method for Irish software SMEs. Recuperado el 31 de enero de 2013, de http://ulir.ul.ie/handle/10344/2342

Méndez Bazalar, Á. A. (2012). Mejora del Proceso Software de una Pequeña Empresa Desarrolladora de Software: Caso Competisoft-Perú-LIM. Omega, primer ciclo. Recuperado el 31 de enero de 2013, de http://tesis.pucp.edu.pe/repositorio/handle/123456789/1515.

Mishra, D., & Mishra, A. (2009). Software Process Improvement in SMEs: A Comparative View. Computer Science and Information Systems, 6(1), pp. 111-140.

Muñoz, M. D., & Oktaba, H. Especialización de MoProSoft Basada en el Método Ágil Scrum (MPS-Scrum). Recuperado el 31 de enero de 2013, de http://www.comunidadmoprosoft.org.mx/.

Ñaupac, V., Arisaca, R., & Dávila, A. (2012). Software Process Improvement and Certification of a Small Company Using the NTP 291 100 (MoProSoft). Product-Focused Software Process Improvement, 32-43.

O'Connor, R. V., & Laporte, C. Y. (2011). Deploying Lifecycle Profiles for Very Small Entities: An Early Stage Industry View. Software Process Improvement and Capability Determination, 227-230.

O'Connor, R. V., & Laporte, C. Y. (2010, June). Towards the Provision of Assistance for Very Small Entities in Deploying Software Lifecycle Standards. In Proceedings of the 11th International Conference on Product Focused Software (pp. 4-7). ACM.

Oktaba H., Alquicira E. C., Su R. A., et al.(2005). "Modelo

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

de Procesos para la Industria de Software, MoProSoft. Versión 1.3", pp. 1-153.

Oktaba, H. (2007). MoProSoft o Historia de una Norma. Comunidad MoProSoft. Recuperado el 31 de enero de 2013, de http://www.comunidadmoprosoft.org.mx/.

Oktaba, H. J., Trujillo, M. E. M., & Muñoz, M. D. (2012). KUALI-BEH: Software Project Common Concepts. Recuperado el 31 de enero de 2013, de http://pisis.unalmed.edu.co/~cmzapata/cursos/requisitos/kuali.pdf

Oktaba, H., & Ibargüengoitia G., G. (1998). Software Process Modeled with Objects: Static View. Computación y Sistemas, 1(004). pp. 228-238.

Oktaba, H., García, F., Piattini, M., Ruiz, F., Pino, F. J., & Alquicira, C. (2007). Software Process Improvement: The Competisoft Project. Computer, 40(10), 21-28.

Paternina P., K., & Ribón, D. (2011). Calidad de Software: "Aplicación en las industrias desarrolladoras de Colombia". INGENIATOR, 1(2).

Pesado, P., Bertone, R. A., Esponda, S., Pasini, A. C., Boracchia, M., Martorelli, S., & Rodriguez Eguren, S. (2012). Planes de Mejora, Mejora de Procesos de Gestión y Calidad en el Desarrollo de Sistemas de Software. In V Congreso de Tecnología en Educación y Educación en Tecnología

Pino, F. J., García, F., & Piattini, M. (2008). Software Process Improvement in Small and Medium Software Enterprises: a Systematic Review. Software Quality Journal, 16(2), 237-261

Pino, F. J., García, F., & Piattini, M. (2009). An Integrated Framework to Guide Software Process Improvement in Small Organizations. Software Process Improvement, 213-224.

Pino, F. J., García, F., Serrano, M., & Piattini, M. (2006). Medidas para Estimar el Rendimiento y Capacidad de los Procesos Software de Conformidad con el Estándar ISO/IEC 15504-5: 2006. Revista Española de Innovación, Calidad e Ingeniería del Software, 2(3).

Pino, F., García, F., Ruiz, F., & Piattini, M. (2006). Adaptación de las Normas ISO/IEC 12207: 2002 e ISO/IEC 15504: 2003 para la Evaluación de la Madurez de Procesos Software en

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

países en desarrollo. IEEE Latin America Transactions, 4(2), 17-24.

Pino, F., Vidal, J., García, F., & Piattini, M. (2007). Modelo para la Implementación de Mejora de Procesos en Pequeñas Organizaciones Software. XII Jornadas de Ingeniería del Software y Bases de Datos, JISBD, 2007, 326-335.

Ramírez, G. A. (2008). Taller de la Norma Mexicana NMX-I-059-NYCE-2005 (Moprosoft). Recuperado el 31 de enero de 2013, de http://www.comunidadmoprosoft.org.mx/.

Real Academia Española. (2001). Modelo. En *Diccionario de la lengua española* (22.a ed.). Consultado en http://lema.rae.es/drae/?val=modelo

Ribaud, V., Saliou, P., O'Connor, R. V., & Laporte, C. Y. (2010). Software Engineering Support Activities for Very Small Entities. Systems, Software and Services Process Improvement, 165-176.

Rios, B. L. F., Vargas, M. A. A., Espinoza, J. M. O., & Peralta, M. (2008). Experiences on the Implementation of MoProSoft and Assessment of Processes under the NMX-I-059/02-NYCE-2005 Standard in a Small Software Development Enterprise. In Computer Science, 2008. ENC'08. Mexican International Conference on (pp. 323-328). IEEE.

Ríos, C. A. (1998). Cuaderno de Calidad en Ingeniería de Software, *Administración de la Configuración del Software*, No. 1, México, D.F.

Rivera, M. G. (2004). Mapeo de CMMI con MoProSoft. Comunidad MoProSoft. Recuperado el 31 de enero de 2013, de http://www.comunidadmoprosoft.org.mx/.

Ruiz, L. V., Rios, B. L. F., & Espinoza, J. M. O. Arquitectura para la Coordinación de Flujos de Trabajo de MoProSoft por Niveles de Capacidad de Procesos. Recuperado el 31 de enero de 2013, de http://yaqui.mxl.uabc.mx/~publicaciones/art2006/Flores_Olguin_2006.pdf.

Sandoval, D. C. (2010). Herramienta de Soporte a la Valoración Rápida de Procesos Software Utilizando el Modelo Moprosoft Bajo un Enfoque RIA. Universidad Tecnológica de la Mixteca, Huajapan de León, OAX

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

Serna, H. F., & Lebrún, C. A. V. Planeación Estratégica y MoProSoft para obtener ventaja competitiva en PyMEs. Recuperado el 31 de enero de 2013, de http://cecip.upaep.mx/conacyt/memorias2012/Mesa%201%20F2023y4%20PEyDT/5.%20FABIOLA%20SERNA%20HERNANDEZ.pdf.

Tardío, M. A., Estrada, A. F., & Montalvan, D. P. (2011). Primeras Ideas de un Modelo Cubano de Referencia para el Desarrollo de Aplicaciones Informáticas. Revista Cubana de Ciencias Informáticas, 5(2).

Trujillo, M. M., Ventura, T., Oktaba, H., & Torres, R. From MoProSoft Level 2 to ISO/IEC 29110 Basic Profile: Bridging the Gap. Recuperado el 31 de enero de 2013, de http://cibse.inf.puc-rio.br/CIBSEPapers/artigos/artigos_CIBSE12/paper_7.pdf.

Valdés, G., Astudillo, H., Visconti, M., & López, C. (2010). The Tutelkan SPI Framework for Small Settings: A methodology Transfer Vehicle. Systems, Software and Services Process Improvement, 142-152.

Valdés, G., Visconti, M., & Astudillo, H. (2011). The Tutelkan Reference Process: A Reusable Process Model for Enabling SPI in Small Settings. Systems, Software and Service Process Improvement, 179-190.

Vanrell, J. A., Bertone, R., & García-Martínez, R. A Process Model for Data Mining Projects Un Modelo de Procesos para Proyectos de Explotación de Información. Recuperado el 31 de enero de 2013, de http://www.unla.edu.ar/sistemas/gisi/GISI/papers/LACREST-2012-ISBN-978-958-46-0577-1-pag-46-52.pdf

Vargas, E. C., Oktaba, H., Guardati, S., & Laureano, A. L. (2007, Julio). Agents, Case-Based Reasoning and Their Relation to the Mexican Software Process Model (MoProSoft). In Computer Software and Applications Conference, 2007. COMPSAC 2007. 31st Annual International (Vol. 2, pp. 326-334). IEEE.

DE LOS SISTEMAS EMBEBIDOS

Arilla, C., Arribas, L. (2009, septiembre). Tendencias y Aplicaciones de los Sistemas Embebidos en España. Depósito Legal: M-36.195-2009, Fundación OPTI y Fundación ASCAMM. Recuperado el 31 de enero de 2013, de

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

http://red.gnoss.com/comunidad/nbic/recurso/Tendencias-y-aplicaciones-de-los-Estudio-de-Prospe/3c17d894-ab8b-440b-bf93-37c66969733a

Axelsson, J. (2011). On How to Deal with Uncertainty when Architecting Embedded Software and Systems. In *Software Architecture* (pp. 199-202). Springer Berlin Heidelberg.

Azad, M. M., Amin, M. B., & Alauddin, M. (2012). Executive Information System. IJCSNS, 12(5), 106. Recuperado el 31 de enero de 2013, de http://paper.ijcsns.org/07_book/201205/20120517.pdf

Banks, S., Coleman, N., Papanagopoulos, G., Devito, M., & Lin, C. F. (1994, June). Architecture based approach to the control software design. In *American Control Conference, 1994* (Vol. 3, pp. 2731-2735). IEEE.

Berger, A. H. (2002). Embedded Systems Design: An Introduction to Processes, Tools, and Techniques. Taylor & Francis US.

Braun, C. L. (1980, May). Microprocessor Software Egineering Training: a Case Study. In *Proceedings of the May 19-22, 1980, national computer conference* (pp. 465-471). ACM.

Capilla, R. (2009, September). Embedded Design Rationale in Software Architecture. In *Software Architecture, 2009 & European Conference on Software Architecture. WICSA/ECSA 2009. Joint Working IEEE/IFIP Conference on* (pp. 305-308). IEEE.

Chandy, J. C. (2010). Challenges in the Design of Cyber-Physical Systems. Revista Ingenierías USBMed, 1(1), 6-14. Recuperado el 31 de enero de 2013, de http://dialnet.unirioja.es/servlet/articulo?codigo=3709971

Cisco (2010). Industrial Ethernet: A Control Engineer's Guide, White paper, pp. 1-19.

Daya, B. (2009). Rapid Prototyping Of Embedded Systems Using Field Programmable Gate Arrays. *Bachelor Of Science In Electrical Engineering, Spring*.

Dmitruk, A. E., Acosta, N., etal (2012). El software y lo sistemas embebidos o empotrados o insertados. Consultado en noviembre de 2012. http://ebookbrowse.com/el-software-y-los-

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

sistemas-embebidos-andres-dmitruk-doc-d180199719.

Duc, B. M. (2005, September). Uniform Object Modeling Methodology and Reuse of Real-time System Using UML. In *Proceedings of the 5th ACM international conference on Embedded software* (pp. 44-47). ACM.

Espino, E., & Torres, D. Estado del Arte de FPGAs para Control de Aplicaciones. Recuperado el 31 de enero de 2013, de http://scholar.google.com.mx/scholar?q=%22Estado+del+Arte+de+FPGAs+para+Control+de+Aplicaciones%22&hl=en&as_sdt=0%2C5

Gannod, G. C., Lutz, R. R., & Cantu, M. (2001, April). Embedded Software for a Space Interferometry System: Automated Analysis of a Software Product Line Architecture. In *Performance, Computing, and Communications, 2001. IEEE International Conference on.*(pp. 145-150). IEEE.

Gastaldi, G. G., Rapallini, J. A., & Pascual, H. O. (2002, April). Evaluación de Programas de Cálculo en Ingeniería como Herramienta de Desarrollo para Codiseño Hardware/Software. *In IWS2002 VIII Workshop IBERCHIP, Guadalajara, México*.

Ghosh, K., Mukherjee, B., & Schwan, K. (1994). A Survey of Real-Time Operating Systems.

González, P. L., & Ulrrego, G. G. (2008). Modelo de Requisitos para Sistemas Embebidos: Model of Requirements for Embedded Systems. *Revista Ingenierías Universidad de Medellín*, 7(13), 111-127.

Hamblen, J. O. (2008). Introduction to Embedded Systems Using Windows Embedded CE. Atlanta: Georgia Institute of Technology, 102.

Heath, S. (2003). Embedded systems design, EDN series for design engineers, (2 edición). Newnes. p. 8-15. Consultado en octubre de 2012 de http://books.google.com.mx/books?id=BjNZXwH7HlkC&pg=PA2&redir_esc=y#v=onepage&q&f=false

Hernández, V. J. I. (2010). El Software Embebido y los Retos que Implica su Desarrollo. *Conciencia Tecnológica*, (40), 42.

Juárez, B. Á., Vázquez, G. M., Aceves, F. M. A., Ramos, A. C. A., y Ramos, A. J. M. (2011). Metodología para el Diseño de Interfaces de Usuario para Sistemas con FPGA. *10º Congreso*

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

Nacional de Mecatrónica. Noviembre 3 y 4, 2011. Puerto Vallarta, Jalisco. Recuperado el 31 de enero de 2013, de http://www.mecamex.net/anterior/cong10/trabajos/art56.pdf

Kang, B., Kwon, Y. J., & Lee, R. Y. (2005, August). A Design and Test Technique for Embedded Software. In *Software Engineering Research, Management and Applications, 2005. Third ACIS International Conference on* (pp. 160-165). IEEE.

Kim, K. (2012). Embedded System Environment: Overview. *CECS Technical Report*, University of California Irvine.

Kim, S. S., Lee, D. G., & Park, J. H. (2011). Efficient Scheme of Verifying Integrity of Application Binaries in Embedded Operating Systems. Journal of Supercomputing-Heidelberg, 59(2), 676.

Lakhani, F., & Pont, M. J. (2012, June). Applying Design Patterns to Improve the Reliability of Embedded Systems through a Process of Architecture Migration. In *High Performance Computing and Communication & 2012 IEEE 9th International Conference on Embedded Software and Systems (HPCC-ICESS), 2012 IEEE 14th International Conference on* (pp. 1563-1570). IEEE.

Netrino, The Embedded Systems Experts (2011). Embedded Systems Glossary: E. Recuperado el 1 de septiembre de 2011 de http://www.netrino.com/Embedded-Systems/Glossary-E#embedded_system.

Núñez, G. O. R. N. (2011) Monitoreo Remoto de Variables Climáticas con NI Single-Board RIO. Universidad Autónoma de Ciudad Juárez, 2011. pp. 108

Padilla, Z. G., Villa, D. E., Montes de Oca, V. C. (2007). Ingeniería de la Confiabilidad de Software. Software Guru. Recuperado el 31 de enero de 2013, de http://sg.com.mx/content/view/271.

Pérez, A. David A. (2009). Sistemas Embebidos y Sistemas Operativos Embebidos. Centro de Investigación en Comunicación y Redes (CICORE), Escuela de Computación, Facultad de Ciencias, Universidad Central de Venezuela. Caracas, pp. 1-16.

Posadas, C. H. (2011). Estimación de Prestaciones para Exploración de Diseño en Sistemas Embebidos Complejos HW/SW. Recuperado el 31 de enero de 2013, de

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

http://www.tesisenred.net/handle/10803/32204

Prakash, V. C., Sastry, J. K. R., & Kamesh, M. D. B. K. (2011). Code Generation for Embedded Software for Modeling Clear Box Structures. *Int. J. Advanced Networking and Applications*, 3(02), 1080-1093. Recuperado el 31 de enero de 2013, de http://www.ijana.in/papers/V3I2-6.pdf

Rockwell Automation (2007, noviembre). Ethernet Design Considerations for Control System Networks: An Introduction. Publication ENET-SO001A-EN-E-Novembre, 2007, pp. 1-56.

Ruiz, M. B. (2010). Desarrollo de Software Basado en Modelos para Sistemas Embebidos. Recuperado el 31 de enero de 2013, de http://xa.yimg.com/kq/groups/22422604/1953961615/name/Barron_SAAEI_2010.pdf

Russell, J. T., & Jacome, M. F. (2003, May). Embedded Architect: a Tool for Early Performance Evaluation of Embedded Software. In *Proceedings of the 25th International Conference on Software Engineering* (pp. 824-825). IEEE Computer Society.

Scaife, N., Hammond, K., Jost, S., Loidl, H. W., Michaelson, G., & Sérot, J. (2008, April). Costing by Construction: Compositional Design and Verification of Embedded Applications using the Hume Software Development Methodology. *In 14th IEEE Real-Time and Embedded Technology and Applications Symposium, St. Louis, MO, United States*. Recuperado el 31 de enero de 2013, de http://www-fp.cs.st-andrews.ac.uk/embounded/pubs/Costing2008.pdf

Sun, C. (2012). Design and Application of Linux-Based Embedded Systems. In *Advances in Multimedia, Software Engineering and Computing Vol. 1* (pp. 641-645). Springer Berlin Heidelberg.

Ting, J. S. (2011). Arquitectura de Software para los Actuales Sistemas Ciberfísicos. *Revista Ingenierías USBMed*, 2(1), 29. Recuperado el 31 de enero de 2013, de http://dialnet.unirioja.es/servlet/articulo?codigo=3692798

Wang, C. L., Yao, B., Yang, Y., & Zhu, Z. (2001). A Survey of Embedded Operating System. *Techical Report, University of California, San Diego, USA*. Recuperado el 31 de enero de 2013, de http://ceit.aut.ac.ir/courses/file.php/35/Others/4-

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

Paper_Preparation.rar

Williams, B. C., & Ingham, M. D. (2006, January). Model-based programming: Controlling Embedded Systems by Reasoning About Hidden State. In *Principles and Practice of Constraint Programming-CP* 2002 (pp. 508-524). Springer Berlin Heidelberg.

Wong, S., & Cotofana, S. (2002, July). On Teaching Embedded Systems Design to Electrical Engineering Students. *In Proceedings of the International Conference on Information Communication Technologies in Education* (ICICTE2002).

DEL CASO DE ESTUDIO

National Instrument (2013). Recuperado el 15 de febrero de 2013, de http://mexico.ni.com/

National Instrument (2013,A). Company. Recuperado el 15 de febrero de 2013, de http://www.ni.com/company/

National Instrument (2013,B). NI cRIO-9074, Controlador Integrado en Tiempo Real de 400 MHz y FPGA de 2M de Compuertas. Recuperado el 15 de febrero de 2013, de http://sine.ni.com/nips/cds/view/p/lang/es/nid/203964

National Instrument (2013,C). NI cRIO-9074, Controlador Integrado en Tiempo Real de 400 MHz y FPGA de 2M de Compuertas. Recuperado el 15 de febrero de 2013, de http://sine.ni.com/ds/app/doc/p/id/ds-204/lang/es

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

ANEXOS

ANEXO A - OPE.2_DOC1_EDR_V1.0.0.2012.
ANEXO B - OPE.2_DOC2_AYD_V1.0.0.2012.
ANEXO C - ARTÍCULO CONISOFT 2012.
ANEXO D - ARTÍCULO CONISOFT 2013

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS TESIS

ANEXO A

- DOCUMENTO ESPECIFICACIÓN DE REQUERIMIENTOS -					
Nombre de la compañía:					**Logotipo de la Empresa:**
Nombre del proyecto:					
Categoría: Operativa.	**Proceso:** Desarrollo y Mantenimiento de Software				**Revisión:**
Fecha:				**Área o Departamento:**	
Elaborado por:					
Revisado por:					
Aprobado por:					

Instrucciones: Lea con atención cada apartado y llene la información solicitada escribiendo o seleccionando, según sea el caso.

1.- Introducción: Esta parte se usará para garantizar la descripción general del sistema, el software y su uso final.

a) Marque en el paréntesis la opción que indica cómo se considera al sistema final que contiene el software embebido a diseñar.
( ) Sistema de riesgo bajo, no hay vidas humanas ni otros subsistemas críticos que dependan del software embebido a diseñar.
( ) Sistema de riesgo medio, no hay vidas humanas pero si hay otros subsistemas críticos que dependen del software embebido a diseñar.
( ) Sistema de riesgo alto, hay vidas humanas y puede haber otros subsistemas críticos que dependen del software embebido a diseñar.

b) Indique con una "X" los componentes que tendrá el sistema que alberga el dispositivo con el software embebido:

DESCRIPCIÓN	
a) Módulo de CPU o unidad que aporta capacidad de cómputo al sistema. Marca, Modelo, Fabricante:_____	
b) Módulo de comunicación mediante interfaces estándar de cable o inalámbricas (puertos de comunicación).	
c) Módulo subsistema de presentación (salida visible).	
d) Módulo de actuadores (elementos electrónicos o eléctricos que el sistema se encarga de controlar).	
e) Módulo de entradas/salidas analógicas.	
f) Módulo de entradas/salidas digitales.	
g) Módulo externo para digitalizar señales análogas.	
h) Módulo de reloj encargado de generar las diferentes señales de reloj a partir de un único oscilador principal (considerar frecuencia, estabilidad y consumo de corriente requerido).	
i) Módulo de energía (convertidores dc/ac, dc/dc).	
j) Módulo de consumo de energía para alimentación	
k) Otros módulos (filtros, circuitos integrados de uso específico, supervisores de energía, etc.)	

Extraída, resumida y adaptada de[Dmitruk, A. E., Acosta, N., etal, 2012; Heath, S., 2003) y experiencia personal.

TESIS TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

2.- Funcionalidad: El apartado se usará para establecer las necesidades que debe satisfacer el software cuando es usado en condiciones específicas. Las funcionalidades deben de ser adecuadas, exactas y seguras.

a) Describa brevemente y en forma general lo que debe hacer el sistema embebido.

_____
_____
_____
_____
_____

b) Describa brevemente lo que debe hacer, monitorear, medir o controlar el software embebido.

_____
_____
_____
_____
_____

3.- Interfaz con usuario: Este apartado dará la definición inicial solicitada de aquellas características de la interfaz de usuario que permiten que el software sea fácil de entender, aprender, que genere satisfacción y con el cuál el usuario pueda desempeñar su tarea eficientemente. Incluyendo la descripción del prototipo de la interfaz.

a) Marque los elementos que considere necesarios para que el usuario interaccione con el sistema. En caso de seleccionar la opción 1, salte a la parte de Interfaces externas.

CONCEPTO	MARCA	CONCEPTO	MARCA	CONCEPTO	MARCA
1.-No usa interface.		*5.- Led/foco.*		*9.- Palanca.*	
2.- Pantalla LCD.		*6.- Teclado Alfanumérico..*		*10.- Pedal.*	
3.- Display.		*7.- Interruptor deslizable*		*11.- Teclado numérico.*	
4.- Pantalla Touch		*8.- Botón.*		*12.- Otro (puerto, infrarrojo, RF, etc.).*	

Generada y adaptada de comentarios en conceptos de [Azad, M. M., etal, 2012] para Human-Machine Interaction Engineeringy, Human–Computer Interface (HCI) and Man–Machine Interface (MMI) y experiencia personal.

Si marcó la casilla 12, especifique que otro(s) dispositivo de interfaz con usuario se planea usar: _____
_____
_____
_____

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

b) Indique si actualmente se cuenta con el archivo o documento que contiene los datos para cada interface seleccionada (fabricante, ficha técnica, especificaciones físicas, características)._____

c) Elabore un pequeño bosquejo en bloques o diagrama del prototipo de la interfaz.

Prototipo de la interfaz de usuario.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS TESIS

4.- Interfaces externas: En este apartado listarán las interfaces con otro software o con hardware solicitadas en forma inicial para el software embebido que se elaborará para el sistema embebido. Tome en cuenta la decisión de partición entre hardware y software, es decir ¿qué problema se resolverá en hardware y qué problema se resolverá en software? Piense en las consecuencias en costos que puede implicar.

A) EL SISTEMA EMBEBIDO Y/O SU SOFTWARE TIENE INTERFACES CON LOS SIGUIENTES MÓDULOS DE SOFTWARE		**B) EL SISTEMA EMBEBIDO Y/O SU SOFTWARE TIENE INTERFACES CON LOS SIGUIENTES MÓDULOS DE HARDWARE**	
ENTRADA A:	**SALIDA A:**	**ENTRADA A:**	**SALIDA A:**

Adaptación basada en [Berger, A. H, 2002].

5.- Confiabilidad: En este apartado se hará la especificación del nivel de desempeño del software con respecto a la madurez, tolerancia a fallas y recuperación.

a) Genere un conjunto completo para el perfil de operación del software embebido (ej. tareas o funcionalidades lógicas principales del sistema) con su correspondiente probabilidad de ocurrencia o uso esperado considerando que los recursos toman un nivel cuantitativo basado en la importancia de cada operación del sistema, la suma al final de la probabilidad debe ser 1. Considere para ello la respuesta dada en el inciso 1a.

OPERACIÓN EN EL SOFTWARE (tareas o funcionalidades lógicas principales del sistema)	**FRECUENCIA (probabilidad de uso, Valores entre 0.1 y 0.9)**
TOTAL	1.00

Adaptación basada en [Padilla, Z. G., etal, agosto, 2007].

TESIS TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS TESIS

b) Genere una lista de severidades de falla basándose en la importancia de frecuencia del perfil de operación. Incluya tantas líneas como sean necesarias.

NIVEL SEVERIDAD DE FALLA	DESCRIPCIÓN	IMPACTO EN LA CAPACIDAD DEL SISTEMA.
1		
2		
3		
4		
5		

Adaptación basada en [Padilla, Z. G., etal, agosto, 2007].

c) Defina y anote si usará una herramienta de simulación para probar el software y/o una herramienta estadística que valide los resultados de confiabilidad del sistema.
_____
_____
_____
_____
_____

6.- Eficiencia: Especificación del nivel de desempeño del software con respecto al tiempo y a la utilización de recursos.

a) ¿La forma en la cuál el software en el sistema realiza sus tareas y/o funciones en tiempo real será evaluado con alguna herramienta estadística, herramientas de simulación para el procesador elegido o parámetro de algún estándar?, si es así escriba cuál es:
_____
_____
_____
_____
_____
_____

b) ¿Para medir los recursos consumidos por el procesador que correrá al software embebido se usará …? Marque con una X:

i. LDAP (Link Access Protocol – D) Benchmark _____
ii. Protocolo D-Channel de ISDN _____
iii. Medición en base al manejo de interrupciones y switches _____
iv. Medición en base a tareas (Salvado de registros, variables en stack (pila), espacio de memoria, etc) _____
v. EDN o EEMBC (Embedded Microprocesor Benchmark Consortium). _____
vi. Tarjetas de evaluación para sus microprocesadores embebidos. _____
vii. Herramientas de ejecución del RTOs. _____
viii. Otra (especifique)_____ _____

Adaptación basada en [Berger, A. H., 2002].

TESIS TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS TESIS

7.- Mantenimiento: Descripción de los elementos que facilitarán la compresión y la realización de las modificaciones futuras del software

ELEMENTO	SE USARÁ
Aplicar herramienta de Ingeniería Inversa.	
Herramientas para permitir observar el código (vital).	
Optimización basada en: selección, partición e integración del sistema.	
Otro (especificar):_____.	

Adaptación basada en [Berger, A. H, 2002].

8.- Portabilidad: En este apartado debe dar la descripción de las características del software que permitan su transferencia de un ambiente a otro. Anote si se considera que el software embebido a diseñar podrá se usado en una plataforma diferente o componente CPU diferente en un uso futuro._____
_____
_____
_____
_____

9.- Restricciones de diseño y construcción: Necesidades impuestas por el cliente. Para definir las restricciones de diseño y construcción marque cuales de las siguientes características deben cumplirse en el diseño del sistema embebido y su software:

CONCEPTO	OBSERVACIONES	CONTEMPLADO
El software del sistema embebido estará dedicado a una tarea específica.		
El software del sistema debe considerar soportar un amplio arreglo de procesadores y arquitecturas de procesadores.		
El dispositivo que alberga al software embebido es sensible al costo.		
Se tiene listo el presupuesto de inversión para el desarrollo del sistema.		
El sistema que maneja al dispositivo con el software embebido tiene restricciones de tiempo real ya sean restricciones sensibles o restricciones críticas.		
El software del sistema contiene un sistema operativo en tiempo real (RTO's).		
Se contemplará el uso de watchdog timer para controlar las fallas de software.		
El ambiente en el cuál se usará el dispositivo del sistema tienen restricciones de alimentación, calor, ventilación, CPU, RAM, ROM, I/O.		

TESIS TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS TESIS

La operación del sistema es en extremas condiciones ambientales.		
Los recursos del sistema en referencia a su CPU son limitados.		
El dispositivo almacenará todo su código objeto en ROM.		
Se contemplará el uso de herramientas especializadas y métodos para diseñarlo eficientemente.		
Se considera el uso de un circuito debbuging dedicado.		
El cliente posee y puede proveer los documentos del desarrollo del proceso y para realizar la prueba del sistema.		
El sistema debe de cumplir con la norma RosH.		
El cliente puede proveer las dimensiones del espacio donde el sistema se instalará.		

Adaptación basada en [Berger, A. H., 2002] y experiencia del equipo de MTE.

10.- Legales y Reglamentarios: En algunas ocasiones se requiere contemplar algunas necesidades impuestas por leyes, reglamentos, estándares, entre otros. En este apartado debe definir si el diseño de su software estará sujeto a estas necesidades. Llene la información referente:

a) ¿Las herramientas para el desarrollo y diseño del sistema y/o software usarán licencias para programas, compiladores, pruebas o mantenimientos? Escríbalos: ___

_____
_____
_____
_____
_____
_____
_____

b) ¿El sistema y/o software depende de alguna patente vigente o norma? Escríbala: ___

_____
_____
_____
_____
_____

c) ¿El sistema y/o software debe hacer uso de algún estándar o protocolo (eléctrico, electrónico, comunicaciones, restricción de frecuencias, seguridad, mecánico)? Escríbalo:

_____
_____
_____
_____

TESIS TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS TESIS

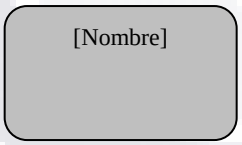
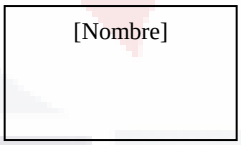
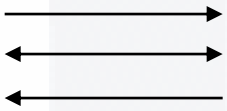
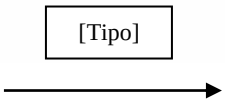
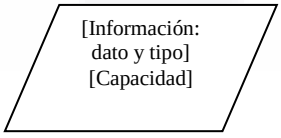
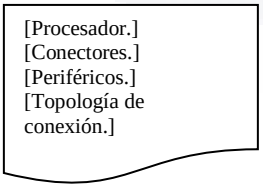
ANEXO B

- DOCUMENTO ANÁLISIS Y DISEÑO -					
Nombre de la compañía:					**Logotipo de la Empresa:**
Nombre del proyecto:					
Categoría: Operativa.	**Proceso:** Desarrollo y Mantenimiento de Software				**Revisión:**
Fecha:				**Área o Departamento:**	
Elaborado por:					
Revisado por:					
Aprobado por:					

Instrucciones: Llene por favor cada uno de los reactivos con la información exacta involucrada en el desarrollo de software que se integrará en el desarrollo del sistema. Al finalizar podrá tener una descripción textual y gráfica de la arquitectura y los componentes para su construcción.

1.- Arquitectónica: A continuación deberá realizar un diagrama a bloques para documentar la arquitectura del sistema embebido y su software. Los componentes de la arquitectura son: Elemento, Conexión, Tipo de conexión, Mensaje, Información, Capacidad y Restricciones de plataforma. Tenga en consideración iniciar por el sistema general y partir de él para describir subsistemas si los hay.

Guía de símbolos para la arquitectura

		
a1) Elemento hardware	**a2) Elemento software**	**b) Conexión**
		
c) Tipo de conexión	**d, e, f) Mensaje: Información: dato y tipo; Capacidad.**	**g) Restricciones de plataforma (Tomado de los requerimientos).**

Banks, S., etal. (1994, June); Gannod, G. C., etal. (2001, April)

TESIS TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS TESIS

h) Recuadro de arquitectura:

Diagrama [A..Z].[0..N].ver[a..z]; fecha elaboración [SS/MM/AAAA]; responsable [].

2.- Detallada: A partir de los diagramas de arquitectura del software embebido, diseñados en el apartado anterior, detalle a continuación los componentes necesarios para llevar a cabo la construcción del código y las pruebas en el ambiente de programación. Use una tabla para cada diagrama elaborado conservando la nomenclatura del diagrama.

Diagrama [A..Z].[0..N].ver[a..z]; fecha elaboración [SS/MM/AAAA]; responsable [].	
COMPONENTES	**DESCRIPCIÓN**
a)Modulo (diseño, lista requerimientos y operaciones).	
b)Operación (Descripción formal de función).	
c)Dato(s) (descripción, formato, contenido de acuerdo al tipo de dato).	
d)Tipo de dato (definición del dato).	
e)Archivo (formato físico del dato en un disco físico).	
f)Comunicación (tomar de referencia las conexiones en la arquitectura para definir la topología del diseño, enunciar cuales operaciones se comunican mutuamente y describir el mecanismo por el cual el dato obtiene desde una operación, con sus propiedades generales de comunicación).	
g)Tipos de comunicación (definir la clase de comunicación a ser usada en el diseño, referirse a los tipos de conexión de la arquitectura y determinar el tipo de modelo: Cooperación/Comunicación.	

Banks, S., etal. (1994, June); Kang, B., etal. (2005, August)

TESIS TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

ANEXO C

Esfuerzo de Implementación de la Categoría de Operación - Administración de Proyectos Específicos – del Modelo MoProSoft, para Desarrollo de Software Embebido

Vianney Sotelo Mauries
Ingeniero en Electrónica
ITA, México
vianney.sotelo@hotmail.com

Mónica Brizuela Sandoval
Ingeniero en Sistemas
UAA, México
mbrizuel@correo.uaa.mx

Francisco J. Álvarez Rodríguez,
Catedrático Núcleo Académico UAA
UAA, México
fjalvar@correo.uaa.mx

Abstract

Actualmente existe una problemática respecto a la escasa información y práctica de cómo usar ingeniería de software -modelos y metodologías- en el desarrollo de software. Tendencia que se extiende al software embebido.

El presente trabajo, esfuerzo de contribución para la materia Ingeniería de Software, dio una orientación de cómo implementar el modelo MoProSoft en el desarrollo de software embebido en microcontroladores, delimitando el problema para darle continuidad en una tesis.

Se sugirió cómo integrar algunas herramientas en la Actividad de Planeación del proceso Administración de Proyectos en la Categoría de Operación, donde reportamos una redistribución en diagrama de sus salidas.

Trasladar este manejo de salidas, en cada actividad de cada categoría del Modelo MoProSoft, ejemplifica de manera específica una forma de hacer de manera más sencilla la vinculación entre las tres categorías del modelo y ser más visual para evitar omisiones de documentación a lo largo del desarrollo del proyecto.

1. Introducción

MoProSoft [1] es un Modelo de Procesos para la Industria de Software definido por niveles de capacidad de procesos. MoProSoft se divide en 3 categorías principales: Categoría de Alta Dirección (Proceso: Gestión de negocios), Categoría de Gerencia (Procesos: Gestión de Procesos, Gestión de Proyectos, Gestión de Recursos –Humanos y Ambiente de trabajo, Bienes, Servicios e Infraestructura, Conocimiento de la Organización-) y Categoría de Operación (Procesos: Administración de Proyectos Específicos, Desarrollo y Mantenimiento de Software).

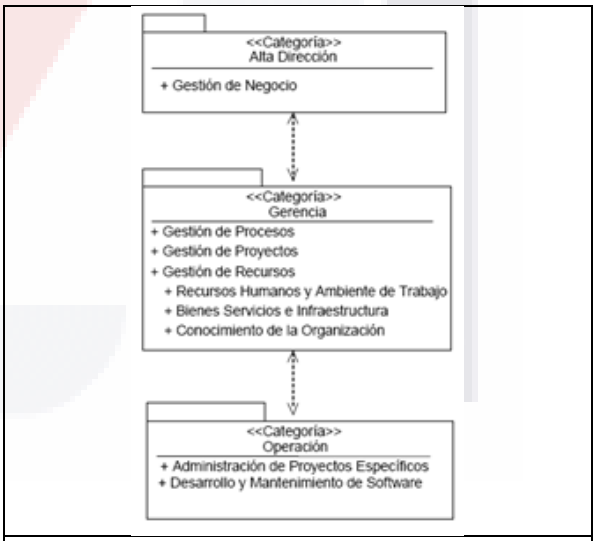


Figura 1. Categorías de modelo de MoProSoft [1]

MoProSoft es un modelo de uso general, es decir, aplicable en cualquier PyME enfocada al desarrollo de software. Su objetivo es la estandarización de los procesos, procedimientos y documentación en el ciclo de desarrollo del software, logrando así generar una base del

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

conocimiento que se pueda someter a un análisis para lograr el mejoramiento continuo en todos los procesos involucrados para minimizar esfuerzos en diseños futuros.

Según literatura reciente [2], la práctica del uso de modelos relacionados con el desarrollo de software no está muy extendida, por lo cual resulta interesante impulsar algunas metodologías.

Por otra parte, existe dentro de las categorías de diseño de software, el diseño de software embebido contenido en el desarrollo de los sistemas embebidos. Éste puede tener la siguiente clasificación [3]:

- De propósito general: Puede ser programado por un usuario para ser utilizado en diferentes aplicaciones y ha sido diseñado para aplicaciones generales.
- De propósito especial (o sistemas dedicados): Para cumplir una tarea específica fija. La mayoría es configurado una sola vez y luego funciona independientemente.

Algunas de sus características mencionadas en [3]:

1. Están embebidos en otros productos.
2. Funcionan por cuenta propia.
3. Usualmente se programan una sola vez.
4. Funcionan frecuentemente de modo reactivo, respondiendo a estímulos (entradas) externos.
5. Son implementados por numerosos procesos concurrentes requiriendo una comunicación entre procesadores.
6. Tienen severos requerimientos de tiempo.
7. Son utilizados en forma intensiva respecto a las entradas – salidas (E-S).
8. Son extremadamente sensibles respecto a criterios de costo, potencia y rendimiento.
9. Tiene fuertes restricciones de fiabilidad y exactitud.

De la misma manera, se mencionan de forma resumida y concisa, como atributos en [4], la confiabilidad, limitaciones en recursos de hardware y su respuesta en tiempo real.

Hernández [4] también hace un contraste sobre la visión del software empresarial, el cual se enfoca en abstracciones como entidades de información y proceso de negocios, mientras que el software embebido está enfocado en la interacción con el mundo físico, ya que su aplicación está mayormente ligada a productos tales como automóviles, reproductores, cámaras, reproductores de música, dispositivos para medición de tiempo o que sean capaces de detectar y responder a eventos del ambiente, restricciones físicas y respuestas en tiempo real. Por ende, menciona que al tratar de cubrir todos estos atributos se presentan dificultades de ingeniería de software. Textualmente, menciona los cuatro grandes retos considerados en el proceso de desarrollo:

1. Planeación de desarrollo.
2. Establecimiento de una metodología.
3. Aseguramiento de la calidad.
4. Herramientas de diseño.

Enfocándose en que no hay una metodología o un modelo específico para guiar en general a la construcción de software, se reportan esfuerzos realizados para tratar de adaptar algunas al software embebido, tales como:

- El caso específico de uso de herramientas UML Statechars (Creados por David Harel) dedicadas al diseño de sistemas embebidos [2].
- Uso de la "Metodología de Codiseño hardware y software" bajo un análisis de herramientas y conceptos de codiseño de hardware y software [3].

Cabe mencionar que al estar el software embebido muy ligado al hardware, se deben contemplar otras consideraciones al momento de adoptar una metodología o seguimiento de un modelo.

Un punto de partida que puede orientar la adaptación de un modelo, es el llamado gráfico Y, mostrado en la Figura 2. Esta es una representación de los conceptos de dominio de descripción y nivel de abstracción del software embebido.

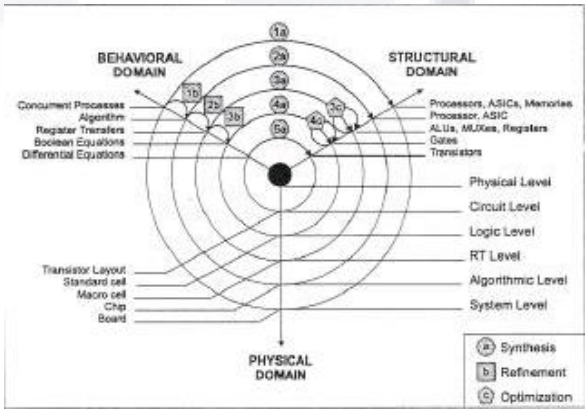


Figura 2. Gráfico Y, tomado de [3]

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

El gráfico Y de la Figura 2 describe las tareas a realizarse, dependiendo del nivel y dominio en el cual se desee incursionar para el desarrollo.

Por otro lado hablando del modelo, en la Figura 3 se muestra el diagrama de clases del proceso de software propuesto por MoProSoft [1].

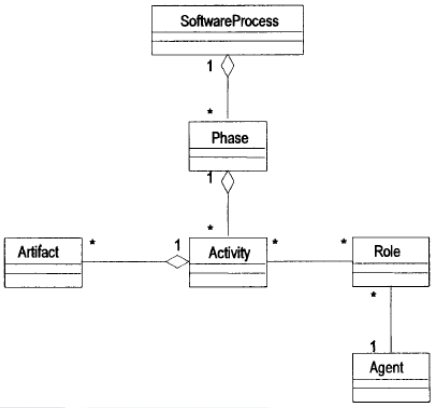


Figura 3. Diagrama de clases del proceso de software, tomado de [1]

Considerando la clase "Actividades" en el diagrama de la figura 3, se puede hacer un esfuerzo para relacionar el modelo de MoProSoft con el Gráfico Y en determinado momento, quizá integrando sus actividades en la clase.

2. Propuesta.

El esfuerzo del presente trabajo se centró en proponer una descripción de cómo organizar algunas herramientas para el Proceso Administración de Proyectos en Específico de la Categoría de Operación del Modelo de MoProSoft para el desarrollo de software embebido, a través de replantear sus salidas en un diagrama mucho mas comprensible para evitar omitir pasos o documentos y poderlos ligar en forma mas sencilla a las otras categorías a través de sus respectivas actividades.

Así mismo consideró redactar cada actividad con un enfoque hacia el software embebido en cada enunciado.

3. Desarrollo

Como se comentó en la introducción, el software embebido es una pequeña parte de los sistemas embebidos, los cuales tienen un amplio campo de acción. Por ello se consideró delimitar el campo de acción en un subproceso delimitado de la siguiente manera.

Proyecto: Sistemas embebidos.
Proceso: Diseño y desarrollo de software embebido, módulo de CPU.
Subproceso: Programación de microcontrolador.

El proceso OP1 Administración de Proyectos Específicos de la Categoría de Operación del Modelo de MoProSoft tiene como propósito establecer y llevar a cabo sistemáticamente las actividades que permitan cumplir con los objetivos de un proyecto en tiempo y costos esperados [1].

Adicionalmente, el modelo coloreado de MoProSoft propone hacer uso de conocimientos, habilidades, técnicas y herramientas para las siguientes actividades del proyecto:

- Planificación (18 actividades).
- Realización (11 actividades).
- Evaluación y control (3 actividades).
- Cierre (4 actividades).

Para poder cubrir estas 4 actividades principales el modelo menciona la generación de entradas (provenientes de otro proceso o actividad), salidas (destinadas a otro proceso o actividad) y productos internos (monitoreo interno

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

del proceso). A continuación, se describen de manera general:

Entradas:

a) Configuración de Software.
b) Reporte de Actividades.
c) Reporte de Mediciones y Sugerencias de Mejora.
d) Plan de Comunicación e Implantación.
e) Documentación de Procesos.
   • Desarrollo y Mantenimiento de Software.
f) Plan de Procesos:
   • Plan de Mediciones de Procesos.
g) Descripción del Proyecto.
   • Descripción del Producto.
   • Alcance.
   • Objetivos.
   • Entregables.
h) Responsable de Administración del Proyecto Específico.
i) Acciones Correctivas o Preventivas.
j) Metas Cuantitativas para el Proyecto.
k) Asignación de Recursos.
l) Solicitud de Cambios.

Salidas:

a) Reporte de Mediciones y Sugerencias de Mejora.
b) Plan del Proyecto.
c) Reporte de Seguimiento.
d) Documento de Aceptación.
e) Plan del Proyecto:
   • Plan de Adquisiciones y Capacitación.
f) Lecciones Aprendidas.
g) Plan de Desarrollo.

Productos internos:

a) Acciones Correctivas.
b) Minuta (s).
c) Reporte de Verificación.
d) Reporte de Validación.

El inicio de nuestra actividad fue enfocar las actividades de Planificación que menciona el modelo de MoProSoft de la manera más precisa y cercana al desarrollo de software embebido. En la Figura 4 se muestra un ejemplo de una de las actividades que se reportó en los avances del trabajo:

A1. Planificación (O1)	
Rol	Descripción
RGPY RAPE	A1.1. Establecer los requerimientos para Descripción del Proyecto del subproceso programación de microcontroladores a diseñar con el Responsable de Gestión de Proyectos.
RDM	Herramienta y Documentos usados para Descripción de Proyecto: [13] *Enfoque del Marco Lógico (EML),* es una herramienta analítica para la planificación y gestión de proyectos orientada por objetivos. *La Ficha de Identificación,* es un instrumento de trabajo que sistematiza y condensa toda la información del proyecto disponible hasta ese momento y ofrece un marco de referencia homogéneo de la gran cantidad de aspectos a considerar manteniendo la idea de la posible recepción posterior de nuevos datos, es de carácter dinámico y es susceptible de tener sucesivas redacciones y cambios hasta su aprobación definitiva. Esto generará el *Documento de Formulación* que es la declaración formal de la Descripción del Proyecto.

Figura 4. Ejemplo de la Actividad A1. Planificación

Los puntos que proponemos al momento de realizar las adaptaciones de las actividades generales (A1.1 a A1.18) de Planificación propuesta por el modelo MoProSoft para la Categoría OP1 en Administración de Proyectos, al software embebido son:

I. En nuestra descripción de actividad subrayamos el producto que se desea obtener (ej. Descripción de Proyecto).
II. En el párrafo subsecuente hacemos énfasis en el producto que se desea revisar o generar en esta actividad, ya sea producto de entrada, salida o interno.
III. Desglosado en ese apartado de herramientas y/o documentos usamos un hipervínculo que hace referencia al material que se sugiere usar. Asimismo, se le denota subrayado y en cursivas (Ej. *Enfoque del Marco Lógico (EML)*) y se le incluye una breve descripción.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

IV. A este apartado también se le marca en el inicio con la referencia bibliográfica correspondiente a la información encontrada para usar ([13]).

Otro punto en el cual hemos centrado la atención es en llevar a cabo una propuesta de modelación entradas/salidas, basándonos en los detalles de MoProSoft. Esta consiste en tomar las salidas de la actividad para mostrar las herramientas y/o técnicas que se usan en la generación de los documentos que serán legados a otras categorías, tal como se muestra en la Figura 5.

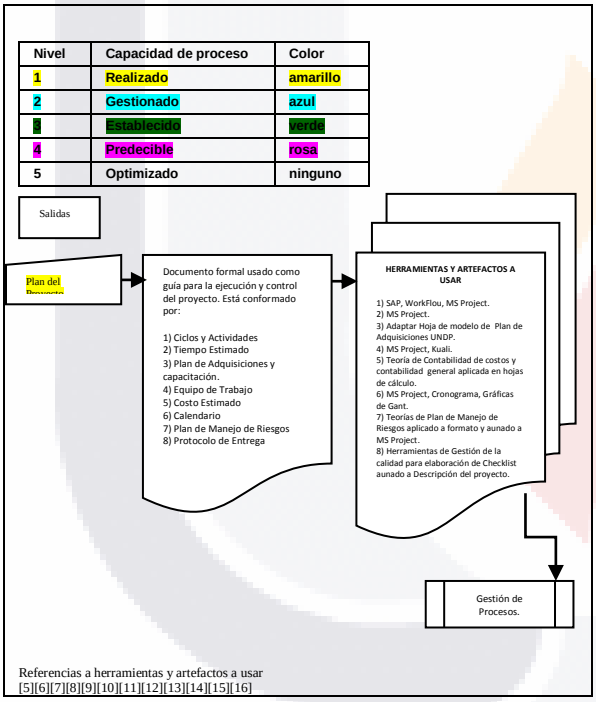


Figura 5. Modelo propuesto

La finalidad de trasladar la información a este tipo de diagrama propuesto es poder vincular el final las tres categorías principales del modelo de MoProSoft de una forma mucho más rápida al ir cubriendo de manera paralela los procesos y descubrir de manera oportuna los apartados que carecen de información o de control.

Asimismo, queremos enfatizar que a lo largo del diagrama se usan los colores indicados en el manual de MoProSoft para poder evaluar el nivel en el que se están cubriendo las actividades.

4. Resultados

Se desarrolló en el trabajo a la par la Categoría de Gerencia, siguiendo la misma propuesta. Es decir, enfocándonos en adaptar las actividades generales en enunciados específicos a desarrollo de software embebido y también las salidas en el diagrama para ligarla a la Categoría de Operación y la Categoría de Alta Dirección.

A lo largo del desarrollo de nuestro trabajo hemos notado que se han llevado a cabo diversas colaboraciones que dan inicio al desarrollo de herramientas basadas en el modelo de MoProSoft. Sin embargo, dejan huecos o trabajos inconclusos que dan pie a nuevas implementaciones, tal como lo menciona las referencias [17], [18], [19] y [20].

5. Conclusiones

El esfuerzo llevado a cabo para implementar una metodología como apoyo al desarrollo de software embebido, adaptando el modelo de MoProSoft, contiene la visión de que al estar éste enfocado a industrias PyME, tiene la ventaja de poder adaptarse de manera rápida a procesos no muy grandes.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

Como hemos descrito, el software embebido, a pesar de su complejidad, está destinado a ser desarrollado en equipos pequeños, por lo cual consideramos positivo el esfuerzo de adaptar este modelo.

6. Trabajo Futuro

Como trabajo futuro se tiene la propuesta de llevar el tema como parte integral o marco de referencia en la investigación de tesis que se encuentra en proceso de generación: "Software Embebido: Análisis de Categoría Operativa de Modelo MoProSoft, Caso Tarjeta de Adquisición de Datos Ethernet".

Para el presente trabajo se avanzó hacia la completa documentación y referencia de herramientas para las actividades de Administración de Proyectos y Gestión de Procesos, pero aún quedó pendiente recopilar material para cubrir todas las actividades de las tres categorías, hasta llegar al punto de vincularlas y elaborar métodos de evaluación de éstas para el software embebido en los microcontroladores debido a la corta duración del semestre en que se imparte la materia.

El software embebido a pesar de ser tratado con metodologías de *UML Statechars [2]* y *Metodología de Codiseño hardware y software [3]* como se menciona en la Introducción, no ha logrado la completa aceptación de los diseñadores debido a su robustez, por lo cual se pretende hacer de esta adaptación de diagrama de entradas y salidas, así como las sugerencias de algunas herramientas o artefactos de registro, una forma de solucionar esta renuencia a documentar el proceso de generación de software embebido a la par de su desarrollo.

Adicionalmente, se pretende llevar a cabo, al final, una retroalimentación de los procesos para depurar o cambiar herramientas de uso general provenientes de otras áreas (teorías administrativas, contables, presupuestales, de riesgos, calidad, etc.) y dejar abierta la posibilidad de generación de nuevas herramientas para trabajos futuros de colaboradores y colegas interesados en el tema.

7. Referencias

[1] Hanna Oktaba, Claudia Alquicira Esquivel, Angélica Su Ramos, et al., "Modelo de Procesos para la Industria de Software, MoProSoft. Versión 1.3", 2005, pp. 1-153.

[2] Mariano Barrón Ruíz. "Desarrollo de software basado en modelos para sistemas embebidos", SAAEI, 2010, pp. 1-6

[3] Guillermo G. Gastaldi , José A. Rapallini, Héctor O. Pascual, "Evaluación de programas de cálculo en ingeniería como herramienta de desarrollo para Codiseño Hardware / Software". IberChip Guadalajara VIII Workshop International, 2002, pp. 1-6.

[4] M.C. José Isidro Hernández Vega, "El software embebido y los retos que implica su desarrollo", Conciencia Tecnológica No. 40, Julio-Diciembre 2010, pp. 42-45

[5] SAP México y Centroamérica, Soluciones para responder a las necesidades de PYMES www.sap.com/mexico/

[6] WORKFLOU Methodology and Software by Juan Jorge Oetling Ladron de Guevara is licensed under a Creative Commons Attribution-Non Commercial - ShareAlike 3.0 Unported License. Based on a work at www.workflou.com. Permissions beyond the scope of this license may be available at www.workflou.com.

[7] Microsoft México, S. de R.L. de C.V., Av. Vasco de Quiroga No. 1700 Piso 7, Col. Centro de Ciudad Santa Fe, México D.F. C.P. 01210 México www.microsoft.com/project/en-us/try-buy.aspx

[8] Copyright © PNUD Argentina 2008 - Esmeralda 130, piso 13, CPA C1035ABD - Ciudad Autónoma de

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

Buenos Aires, Argentina - Fax: (54 11) 4320 8754 - Tel.: (54 11) 4320 8700 http://www.undp.org.ar/docs/Manual/04CG.pdf

[9] Gustavo A. Figueroa M. "La Metodología de Elaboración de Proyectos como una Herramienta para el Desarrollo Cultural". Serie Bibliotecología y Gestión de Información No. 7 septiembre, 2005.

[10] Universidad Nacional Autónoma de México, Facultad de Economía, División de Estudios Profesionales, Plan de Estudios 1994, Línea de Estudio: Instrumentales, Programa de la asignatura: Contabilidad General y de Costos.

[11] Excel 2010, © 2012 Microsoft Corporation. Reservados todos los derechos

[12] Gustavo A. Figueroa M. Ibid.

[13] SmartSheet, Administración de proyectos con diagrama de Gantt en línea interactivo www.smartsheer.es

[14] Practical Project Management, Wrike Inc., 2411 Lascar Pl, San Jose, CA 95124, EE.UU 1-877-77-WRIKE http://www.wrike.es/

[15] Administración del Riesgo, Plan de Manejo de los Riesgos Institucionales, elaboración y compilación Participantes Angélica María Alaix Martínez, Damaris Blanco Barragán, Omar Vivas Cortés BOGOTA, D.C. AGOSTO 2006 www.procuraduria.gov.com

[16] Sistemas de Gestión de la Calidad, Implantación de un Sistema de Gestión de la Calidad según la UNE-EN-ISO 9001:2008, Fases de la implantación de un sistema de Gestión de la Calidad. Copyright 2009. All Rights Reserved.Getion-calidad.com

[17] Diana Angélica Caballero De La Villa "Manejador de Documentos de MoProSoft". Universidad de las Américas Puebla, Escuela de Ingeniería, Departamento de Ingeniería en Sistemas Computacionales . Tesis profesional. Cholula, Puebla, México a 11 de mayo de 2005, Derechos Reservados ©2005.

[18] Paola Yuritzy Reyes Delgado, Ma. Lourdes Y. Margain Fuentes, Francisco Javier Álvarez Rodríguez, Jaime Muñoz Arteaga "Aplicación de instrumento diagnóstico en proceso "gestión de procesos" con base en MoProSoft". Investigación y Ciencia de la Universidad Autónoma de Aguascalientes, No. 43 enero-abril 2009, pp. 30-37.

[19] Leonel Valenzuela Ruíz, Brenda Leticia Flores Ríos, José Martín Olguín Espinoza "Arquitectura para la Coordinación de Flujos de Trabajo de MoProSoft por niveles de Capacidad de Procesos". A. L. Morán y Solares, E. Martínez Martínez (eds): Memorias del CiComp'06, Ensenada B.C., México, 6 al 8 de Noviembre de 2006, pp. 7-12

[20] Dagoberto Cruz Sandoval "Herramienta de Soporte a la Valoración Rápida de Procesos Software Utilizando el Modelo Moprosoft Bajo un Enfoque Ria", Tesis Universidad Tecnológica de la Mixteca. Huajuapan De León, Oax.; 18 de Marzo de 2010

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

ANEXO D

Modelo MoProSoft, una Adaptación de la Categoría Operación, Proceso Desarrollo y Mantenimiento de Software en "Especificación de Requerimientos" al Proceso de Desarrollo de Software Embebido, Caso de Estudio: NI cRIO-9074 de National Instrument para Proyecto Final Vision System de Flextronics Manufacturing

Vianney Sotelo Mauries[1], Francisco J. Álvarez Rodríguez[2], Juan M. Mora Tavarez[3], Guillermo Ramírez Prado[4]
Universidad Autónoma de Aguascalientes
vianney.sotelo@hotmail.com[1], {fjalvar[2],mmora[3],grprado[4]}@correo.uaa.mx

Pablo Espinosa Lepe[5]
Flextronics Manufacturing
pablo.espinosa@mx.flextronics.com[5]

Abstract

En la actualidad, existe una cantidad importante de trabajos que mencionan modelos de uso internacional para la estandarización de procesos de desarrollo de software, sin embargo la implementación de éstos implica la inversión a largo plazo en tiempo, recursos y esfuerzo, resultado costeables únicamente para las organizaciones grandes, es decir con más de 50 empleados o más de 20 proyectos.

Si ha sido difícil el uso de modelos en la industria para aplicaciones generales de desarrollo de software, en los Sistemas Embebidos, los cuáles implican desarrollo de Software Embebido, se tienen menos prácticas reportadas debido a la amplia variedad de aplicaciones que manejan y al nivel complejo que en ocasiones llega a tener su construcción.

Partiendo de estas dos premisas se presenta en este artículo el resultado del desarrollo de la adaptación del modelo de procesos de software MoProSoft en su categoría OPE.2, para el producto de la salida en la actividad A2 "Especificación de Requerimientos" para desarrollo del software embebido. El presente artículo expone como se ha dado la generación del producto de salida para A2, el cual está en pruebas de ajuste para su validación con el proyecto Final Vision System de Flextronics Manufacturing a través de la colaboración del equipo de trabajo en el departamento Manufacturing Tool Engineering (MTE), el cuál dadas sus características se considera un grupo pequeño enfocado al desarrollo de herramientas automatizadas que incluyen software embebido.

1. Introducción

1.1. Modelos para el Desarrollo de Software.

Como punto de partida para la presente introducción se cita una definición dada por un marco de referencia cubano [1] definiendo que el término italiano "modelo" se usa como concepto con diferentes aplicaciones y significados. Por ejemplo, según la Real Academia de la Lengua Española (RAE) un modelo es un arquetipo o punto de referencia para imitarlo o reproducirlo. En las obras de ingenio y acciones morales es un ejemplo que se debe seguir o imitar por su perfección. Igualmente, un modelo es una estructura conceptual que sugiere un marco de ideas para un conjunto de descripciones que de otra manera no podrían ser sistematizadas. El artículo hace uso de esta definición para denotar que hay varios modelos y estándares en los que se tratan temas para la generación de productos y/o servicios de software,

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

tales como ISO 9001:2000; ISO/IEC 12207:2002, CMMI, MoProSoft o MPS.BR, entre otros [1].

Por otra parte citado de [4] "… el proceso de software ... es un ambiente de capacidades de recursos interrelacionados, administrando una secuencia de actividades usando métodos apropiados y prácticas para desarrollar un producto de software que está conformado para cumplir los requerimientos del cliente."

Con esta breve definición de un modelo y el proceso de software, se describirán ahora los Modelos de Calidad, definiéndolos en forma específica bajo la óptica de la referencia [2] que indica a los Modelos de Calidad como herramientas que guían a las organizaciones a la mejora continua y la competitividad, dándoles especificaciones de qué tipo de requisitos deben de implementar para poder brindar productos y servicios de alto nivel. En referencia a la definición dada, las autoras alertan de que un Modelo de Calidad no es una metodología que resuelva la vida de forma sencilla y clara, haciendo énfasis en que éste más bien dirá "Qué hacer" y no "Cómo hacerlo", ya que este último cuestionamiento depende de la metodología a usar y de los objetivos de cada negocio.

Pero hablando de calidad en general, encontramos que en la industria se denotan un sin número de productos físicos, los cuales son creados bajo ciertos criterios o normas que garantizan dicha calidad del producto. Quizá es sencillo visualizar la calidad en productos tales como los alimentos, que deben pasar por normas estrictas de organismos de salubridad; productos electrodomésticos que cumplan las normas de seguridad eléctrica; incluso en el caso de la educación o los servicios se puede escuchar hablar de calidad. Visualizando la calidad desde la perspectiva de un ciudadano(a) consumidor(a), se puede percibir cómo cada uno de ellos prefiere adquirir diferentes clases de productos, que de un individuo a otro varía la elección dependiendo del gusto, costumbre, región o necesidad específica. Del mismo modo el software, a pesar de no ser un producto físico palpable, para poder satisfacer las necesidades de su usuario debe cumplir con cierta "calidad". La referencia [2] menciona que la calidad en el proceso de generación de los productos de software se tiene una exigencia creciente, consecuencia del amplio uso del software en procesos críticos para las organizaciones o maquinaria.

La respuesta que se reporta para lograr software de calidad en forma consciente es la de usar Modelos de Madurez de Procesos. Partiendo de esta premisa se especifica en la cita de la referencia [2] que "el modelo de madurez de procesos más popular es CMMI (Modelo de Capacidad y Madurez Integrado). Sin embargo, este modelo es complejo para implementar en empresas pequeñas". La referencia [4] acerca de la calidad del software dice que depende en gran medida del proceso que se usa para crearlo.

Siguiendo la misma línea de explicación de modelos, se tiene que las referencias [3] y [4] hacen énfasis en compartir que las propuestas de mejora del SEI e ISO (como CMMI, IDEAL, SCAMPI, ISO 12207, ISO 15504) han sido creadas y están estructuradas para ser utilizadas por organizaciones grandes y difícilmente pueden ser aplicadas a organizaciones pequeñas. Esto es porque un proyecto de mejora supone gran inversión en dinero, tiempo y recursos, así como la alta complejidad de las recomendaciones y también porque el retorno de la inversión se produce a largo plazo y tal como es de esperarse, las pequeñas empresas desean la obtención de una alta calidad a un bajo costo como se cita en la referencia [4].

Surgido de estas desventajas de los modelos, la referencia [3] comenta que a partir de finales de los años noventa ha tomado gran fuerza en la comunidad de Ingeniería de Software (industria e investigadores) la Mejora de Procesos de Software (SPI, Software Process Improvement) en pequeñas organizaciones, dedicadas a la creación de software, para resolver los inconvenientes de los modelos ya mencionados. Esta opinión se refuerza en la referencia [4] bajo la observación de que SPI ha sido reconocido como una forma efectiva en las compañías para la mejora de la calidad de software.

Para la adecuación de los modelos a las pequeñas organizaciones en la referencia [3] se menciona que se requirió de una razón de impulso en esta área, la cual se especifica bajo la opinión de muchos autores acerca de que las características especiales de las pequeñas organizaciones de software hacen que los programas de mejora de procesos deban aplicarse de un modo particular y visiblemente diferente a como se hace en las grandes organizaciones y que esto no es tan sencillo como el hecho de considerar a dichos programas de mejora como versiones a escala de los usados en las grandes compañías. Esto se puede traducir en que muchas de las pequeñas organizaciones no poseen un gran número de recursos materiales o humanos para realizar su proceso de desarrollo de software, tal como lo hacen las grandes organizaciones, y en la mayoría de las ocasiones su estructura organizacional difiere de los grandes corporativos. Quizá es conveniente en este momento aclarar que al referirse a una pequeña empresa se habla de una SMEs, bajo el concepto inicial de la referencia [4], cuyo promedio de empleados es de 1 a 50 y se caracteriza por su insuficiencia de recursos

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

humanos, la falta de desarrollo y medio ambiente de apoyo, la falta de presupuesto y su dependencia de las grandes organizaciones, por lo cual estas SMEs encuentran a la mejora de proceso de software como un reto mayor. Del mismo modo la referencia [4] cita a Johnson and Brodman para definir que para ellos una pequeña organización, o haciendo la analogía una SMEs, está compuesta por menos de 50 desarrolladores de software y con proyectos pequeños menores a 20 proyectos de desarrollo.

Pero no es el único reto a superar, también la referencia [1] habla la importancia del problema cultural cuando se quieren importar y adoptar modelos definidos en otros países. Citando a Zahran (1998): "Si el proceso no casa con la cultura de la organización será rechazado por el cuerpo organizacional como sucede en los trasplantes de órganos" y la referencia [4] enfatiza que "... las organizaciones ... no tienen una cultural de procesos y en una cultura de procesos también las costumbres de la gente y su conducta son influidas por la orientación del pensamiento y la administración de los principios del proceso".

Pero no hay que perder de vista que, tal como lo expresa la referencia [4], "hay evidencia de que la mayoría de las pequeñas organizaciones desarrolladoras de software no están adoptando los estándares existentes, esto derivado de que las percepciones negativas de las pequeñas firmas son principalmente conducidas por una vista negativa de los costos, la documentación y la burocracia". Empero también estas organizaciones pequeñas y medianas han reconocido la necesidad de mejorar sus productos de software y evaluarlo parece insuficiente ya que se sabe que su calidad depende del proceso usado para crearlo, por lo cual buscan la forma de evaluar sus procesos y productos.

En conclusión el análisis de estas investigaciones lleva a resumir que al día de hoy la industria que desarrolla software, que es generalmente una pequeña entidad, está interesada en mejorar sus procesos, especialmente considerando a SPI. También es destacable mencionar que este tipo de compañías son las que más aportan a la economía de cada país. Y quizá es posible aventurarse a pensar que, debido a su definición, son las que mayormente tienden a emerger como sector. La pregunta, consecuencia de estas investigaciones, es: ¿se puede trabajar en algún modelo que pueda cubrir las expectativas para la evaluación de sus procesos y productos como pequeñas empresas volviéndose en consecuencia competitivas? Ya sea en un contexto nacional o internacional. En respuesta a esta pregunta, aunque quizá con otras palabras otorgadas por la comunidad investigadora en conjunto con la industria, se expone en la referencia [5] el resumen de los resultados del interés que algunos países tienen en el correcto desarrollo y crecimiento de la industria del software, apareciendo en consecuencia estándares como "Impact, Processus, Adept, Rapid, Agile SPI, Mares, MesoPyME, SIMEC-SW, MR-MPS, Competisoft" por mencionar algunos.

En el contexto nacional, el cuál es de especial interés en esta investigación, se extrae el resumen de la referencia [5] que para atender la necesidad interna de apoyo a la industria desarrolladora de software el trabajo conjunto de la Secretaría de Economía y la UNAM para los Estados Unidos Mexicanos generaron el Modelo de Procesos: "MoProSoft", y su forma de evaluarlo: "EvalProSoft". MoProSoft está basado en CMMI y el estándar internacional ISO/IEC 9001:2000 e ISO/IEC 12207, mientras que EvalProSoft está basado en las recomendaciones de ISO/IEC 15504-2. Ambos están enfocados a la pequeña y mediana empresa (o SMEs o VSEs como en algunos casos se les denota) o bien a pequeños grupos de desarrollo de software en una organización grande. Su objetivo es promover la estandarización de procesos en la industria del software.

La Figura 1 muestra la estructura básica del Modelo MoProSoft. Para mayor detalle y definiciones de cada categoría se puede consultar el documento [6] en la página de la comunidad de MoProSoft.

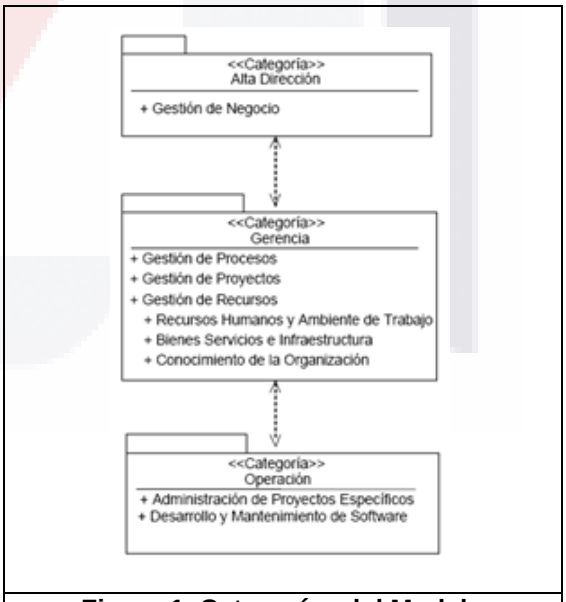


Figura 1. Categorías del Modelo MoProSoft, extraído de [6]

Desde la aprobación de MoProSoft y EvalProSoft a la actualidad se tiene como consecuencia la norma estándar mexicana: NMX-I-

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

059-NYCE-2005 y por ende el interés de las SMEs en los Estados Unidos Mexicanos de adquirir la certificación ha crecido. La norma NMX-I-059-NYCE-2005 correspondiente a Tecnologías de la Información-Procesos de Software y Modelo de Procesos y Evaluaciones para el desarrollo y mantenimiento de software, consiste en 4 partes descritas en la Tabla 1 extraída de la referencia [5].

Hasta aquí se tiene la ruta o guía que explica el surgimiento de los Modelos y lleva a la justificación para elegir MoProSoft como objetivo para aplicar al desarrollo de software embebido. Ahora se presenta el apartado que explica al sistema embebido y su software.

Tabla I.
Estructura de NMX-I-059-NYCE-2005

Parte 01 Descripción	NMX-I-059/01-NYCE-2005 Definición de Conceptos y Productos.
Parte 02 Descripción	NMX-I-059/02-NYCE-2005 Proceso de Requerimientos (MoProSoft).
Parte 03 Descripción	NMX-I-059/03-NYCE-2005 Guía de Implementación de los Procesos.
Parte 04 Descripción	NMX-I-059/04-NYCE-2005 Guía de Evaluación de Procesos (EvalProSoft).

1.2. Sistemas Embebidos, enfocándonos en el módulo: Software Embebido.

Este apartado es iniciado citando la referencia [7] en la cual se enmarcar el surgimiento de los sistemas embebidos:

"… (el inicio de) los sistemas embebidos se remonta a comienzos de los años 60, cuando dispositivos basados en microprocesadores y microcontroladores comenzaron a emplearse en el control de tareas aeronáuticas y espaciales. Las limitaciones de alto costo y diseño de estos primeros dispositivos provocaron una espera hasta 1992 en el que se creó el consorcio PC/104, formado por Ampro, RTD y otros fabricantes".

Pero para poder hablar de un sistema embebido, se debe dar una definición, basado en la cita incluida en la referencia [8]: "un sistema embebido, es un sistema de procesamiento de información de uso específico, integrado en otro sistema de mayor tamaño y conformado por componentes hardware y software". Otra forma en la cual se les conoce es como sistemas ciberfísicos, que definido sería "proceso que integra la computación con los procesos físicos" como lo enuncia la referencia [9]. Complementariamente, del texto en la referencia [10], existen dos clases diferentes de sistemas embebidos: los de propósito general (computadoras que van desde las estaciones de trabajo a las computadoras personales de mano) y los de propósito especial o sistemas dedicados ("automóviles, aviones, teléfonos, equipo del audio, robots, aparatos, juguetes, los sistemas de seguridad, armas, menús de televisiones , copiadoras, escáneres, clima, control de sistemas, sistemas industriales, entre otros" [7]). Estos últimos quizá son los más complejos en diseño.

Denotados en la definición formal están los dos componentes básicos que destacan a un sistema embebido: *hardware* y *software*. Del resumen en la referencia [7] se define a hardware como los componentes físicos del sistema que controlan, o son controlados, en el proceso gobernado por el sistema embebido (partes mecánicas, eléctricas y/o electromecánicas) y software es el sistema de cómputo residente en muchos casos, sin que el usuario se entere, dentro de un subsistema electrónico de procesamiento, programado para realizar una o pocas funciones que cumplen un objetivo específico. A éste software se le denota como: "Software Embebido o Empotrado".

Definido en la referencia [7] "software embebido es el procesamiento de información integrado con procesos físicos o bien software que se ejecuta en dispositivos distintos de una computadora personal o un servidor de cómputo."

Hasta este punto queda definido el objeto de interés de estudio: el software embebido y tratando de explicar lo que le rodea en referencia a metodologías y estándares de desarrollo se resumen los siguientes datos.

La referencia [8] menciona que en sistemas embebidos, los procesos de ingeniería de requisitos se complican, que en las áreas de aplicación típica más del 50% de los problemas se producen porque el sistema no cumple con las expectativas del usuario, debido a la captura errónea de los requisitos y es así como surgen nuevas tareas de rediseño que conllevan a un aumento en los costos y en los tiempos de entrega.

Tocante al tema de las limitantes del sistema embebido, la referencia [9] indica que históricamente los sistemas embebidos fueron en gran medida un problema industrial (recursos y tamaños limitados que exigen mejorar el rendimiento o funcionalidad). Por lo tanto el software embebido difiere de otro tipo de software (administrativo, comercial, financiero u otro) y desde este punto de vista se resume en una sola frase que el problema del software embebido: es un problema de optimización de los recursos. Esto incluye al equipo de trabajo que lo desarrolla, pues

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

la mayoría de los proyectos de desarrollo de software embebido se llevan a cabo por un grupo de trabajo pequeño. De aquí se resume que las limitantes que rodean a este tipo de sistemas están basadas en falta de control de los requerimientos y en considerar los recursos de los cuáles se dispone para que cumplan su función.

Del mismo modo las referencias [8], [9] y [10] citan algunas metodologías que han intentado usarse para el desarrollo de los sistemas embebidos (metodologías orientadas al diseño, metodología basada en el análisis de estados, metodología CARA, metodología ECSAM, transformación de requisitos a un grafo conceptual, codiseño HW/SW) como un esfuerzo para dar solución al problema de optimización de los recursos y garantizar la calidad del producto. Pero en resumen la documentación consultada no reporta el uso de una metodología específica para manejar el desarrollo del "software embebido" contenido en el sistema a través del uso o la perspectiva de la ingeniería de software. Los casos reportados se enfocan al sistema embebido en general, lo muestra un hueco de oportunidad para proponer una metodología más específica en lo que a requerimientos respecta.

Por otra parte dentro de los estándares de IEEE consultados aparecen en la lista los referentes a Sistemas e Ingeniería de Software que nos hablan de como adoptar: ISO/IEC 26512 (Requisitos para compradores y proveedores de documentación de usuario), ISO/IEC 26513 (Requisitos para verificadores y revisores de la documentación de usuario) e ISO/IEC 26514 (Requisitos para verificadores y revisores de usuario), que como podrá notarse están enfocados a la documentación que se debe de generar para usuarios. Y en el caso del ISO, existe

ISO/IEC 90003:2004 (Ingeniería de Software - Directrices para la aplicación de la norma ISO 9001:2000 al Software Computacional) una guía de cómo aplicar ISO 9001:2000 para la adquisición, suministro, desarrollo, operación y mantenimiento de software computacional. Ésta nos cita poder enfocarse al software embebido en un producto, sin embargo aún sigue siendo muy general y como fue mencionado en el apartado de los Modelos de Procesos pues costoso y burocrático para la pequeña y mediana empresa, consecuentemente lo sería aún más para un grupo pequeño de desarrolladores.

Hasta este punto, en resumen, existe evidencia de que se tiene la inquietud de encontrar un punto en común donde a través de un Modelo de Procesos más específico sea viable lograr la calidad del producto Software Embebido analizando sus características limitadas y que sea de fácil adopción por la pequeña y mediana empresa o bien por un pequeño departamento encargado de su desarrollo en una organización. Esto lleva a generar una propuesta de aportación a través de la presente investigación.

2. Propuesta

Como se vio en la introducción, uno de los Modelos de Procesos en el contexto nacional que se ha introducido para empresas desarrolladoras de software es el modelo MoProSoft que reúne las mejores prácticas en estándares internacionales, ajustándolos a la pequeña y mediana industria (PyMe) o pequeños grupos de trabajo. Del mismo modo no se tienen Modelos de Procesos específicos para el desarrollo del Software Embebido. Estas dos premisas llevan a proponer el uso del modelo MoProSoft para auxiliar al desarrollo del Software Embebido.

La propuesta ha venido trabajándose desde la publicación del artículo de la referencia [11], en el cuál el objetivo fue usar la Categoría Operación del modelo MoProSoft, centrándose en el proceso "Administración de Proyectos Específicos" aplicado a Software Embebido. En dicho artículo la primera propuesta era enfocar las actividades de Planificación de la manera más precisa y cercana al desarrollo de Software Embebido y la segunda llevar a cabo una propuesta de modelación entradas/salidas de dicho proceso.

A partir de este trabajo se avanzó llevando a la práctica una primera propuesta en la misma categoría Operación pero ahora para el proceso "Desarrollo y Mantenimiento de Software" que es el objetivo reportar en el presente artículo. Para ello se toman en consideración dos cosas:

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

Primeramente el diagrama de flujo de trabajo propuesto por MoProSoft para el proceso de Desarrollo y Mantenimiento de Software, que se puede consultar en la referencia [6], el cual consta de 6 actividades (numeradas de A1 a A6). Cada actividad genera productos de salida (16 productos) y productos internos (2 productos). Se ha tomado el producto de salida, "Especificación de Requerimientos" de A2, para ajustarlo al desarrollo del Software Embebido. La elección no se hizo al azar, fue basada en la referencia [12] que indica a la fase de elección del procesador como la parte crítica del sistema embebido y ésta se encuentra en la generación de los requerimientos del sistema.

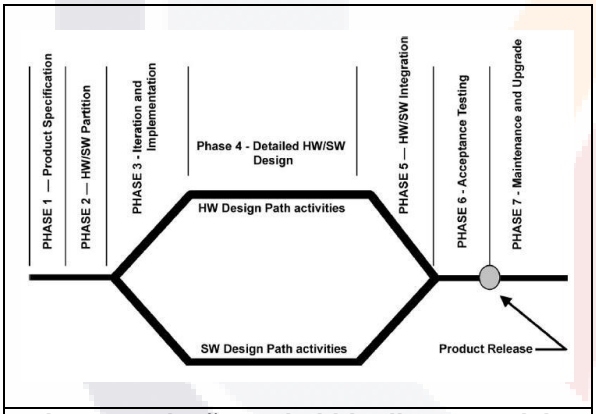


Figura 2. Diseño Embebido diagrama ciclo de vida, extraído de [12]

Como segunda consideración para la propuesta se ha elegido un modelo de ciclo de vida del desarrollo de sistemas embebidos reportado en la referencia [12] y que mostrado en la figura 2. Explicado con palabras, el modelo muestra las 7 fases requeridas para el desarrollo de un sistema embebido, la propuesta presentada se enfoca en la rama que lleva a las actividades de dirección del diseño del software tomando así las fases 1 y 2 llevándola al producto de la actividad A2, indicada anteriormente.

Hasta aquí se dan por sentadas las bases del bosquejo para la propuesta de adaptación del modelo de MoProSoft al Software Embebido para reportar en el presente artículo.

3. Desarrollo

La primera parte del trabajo propuesto en el presente artículo, consiste en la construcción del producto "Especificación de Requerimientos" de A2 (OPE.2_DOC1_EDR_v1.0.0.2012, nombre basado en la referencia [17]). Se inicia acotando el tema, al tomar del diagrama de flujo de trabajo para el proceso "Desarrollo y Mantenimiento de Software" de la categoría Operación del modelo de MoProSoft el producto generado "Especificación de Requerimientos" para A2. La figura 3 muestra el extracto de interés tomado del diagrama original mostrado en la referencia [6] y muestra en rojo el producto.

En segundo lugar se consideran a las fases 1 y 2 del ciclo de vida para el desarrollo de un sistema embebido bajo las premisas que la referencia [12] da para los sistemas embebidos. Estas dos fases quedan resumidas en los siguientes tres grandes apartados:

1.- En la selección del procesador (implícita al inicio de la fase 2) se deben considerar los benchmarks[1] del dispositivo a usar, la experiencia en proyectos pasados, diseños similares desarrollados, aplicaciones de soporte

[1] Benchmark: Su traducción al castellano es "punto de referencia", pero en los sistemas embebidos es el término usado para los programas que prueban en forma parcial o total el rendimiento del procesador.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

que tenga el lenguaje a usar para la programación del dispositivo electrónico, el simulador que usa y sus instrucciones y por último el conjunto de herramientas que dan soporte a la programación del dispositivo electrónico que albergará al software embebido.

2.- En la fase 1 del ciclo, "Especificación del Producto" se considera un método de requerimientos (no hay modelo formal, sólo recomendaciones), considerar un grupo de trabajo pequeño especialista para requerimientos, repartir los roles, elaborar un debriefing[2] y llevar a cabo un resumen de resultados. Aquí también se lleva a cabo el análisis del diseño, para lo cual es necesario convertir el concepto en producto, comprender las necesidades, hacer preguntas correctas al cliente, determinar los mejores desempeños de proyectos pasados, considerar los costos, integrar un buen equipo de trabajo, especificar las herramientas que serán necesarias para llevar a cabo el diseño del producto y hacer juntas para que el equipo se mantenga en común acuerdo y para verificar que todos entienden la meta y no avanzar en el desarrollo hasta llegar a un consenso.

3.- En la fase 2 se tiene la partición del hardware y software que básicamente consiste en definir los problemas que el sistema resolverá por software y los que hará por hardware para garantizar la integración. Se deben considerar bien los recursos y si es posible hacer diagramas a bloques para ayudar a definir la partición. Es recomendable llevar a cabo una administración de los riesgos y considerar la optimización en el producto. Básicamente todo esto dependerá de lo indicado en el punto 1: la selección del procesador que albergará al Software Embebido.

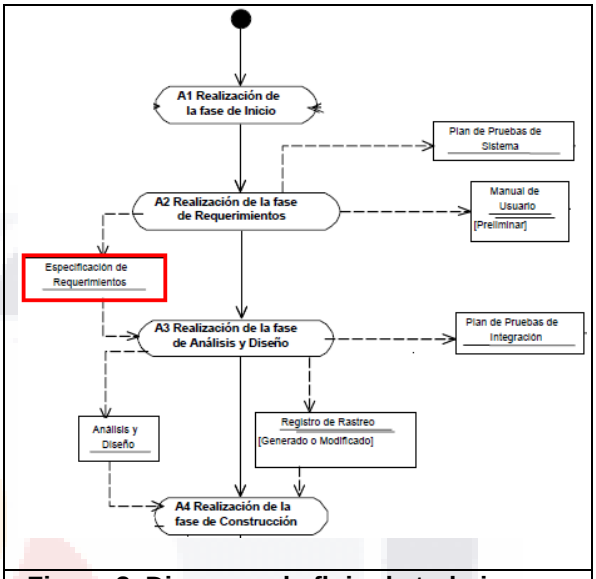


Figura 3. Diagrama de flujo de trabajo para Desarrollo y Mantenimiento de Software, extraído de [6]

Aunado a estos tres apartados se considera importante tomar características peculiares en los análisis y resúmenes de las referencias [13], [14], [15] y [16], las cuales aportan información de carácter considerable acerca de diferentes tipos de sistemas embebidos.

4. Resultados

El proceso de análisis y desarrollo anterior llevó a la construcción de OPE.2_DOC1_EDR_v1.0.0.2012, el cuál es el producto de la salida de la actividad A2 del modelo MoProSoft para la Categoría Operación, proceso "Desarrollo y Mantenimiento de Software", mostrado en el extracto del

[2] Debriefing: Es el nombre que se le da a la elaboración de una lista de interrogantes que están relacionadas a la descripción de lo que se desea construir como sistema embebido.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

diagrama de flujo, figura 3. Basado en las características que se mencionan obligatorias en el modelo MoProSoft, se han incluido los apartados en el producto, redactando instrucciones para su correcto llenado, tal como se puede observar en el extracto tomado para la figura 4. El documento consta de siete páginas para los requerimientos y una para la bibliografía.

El producto de salida "Especificación de requerimientos" indicado en MoProSoft, lista nueve apartados a considerar: Introducción, Funcionales, Interfaz con Usuario, Interfaces Externas, Confiabilidad, Eficiencia, Mantenimiento, Portabilidad, Restricciones de diseño y construcción y Legales y reglamentarios. Para la generación de OPE.2_DOC1_EDR_v1.0.0.2012, se ha adaptado cada apartado basado en el análisis de la literatura que describe a los sistemas embebidos y su software embebido. A continuación se resumen estos apartados adaptados al producto:

1.- Introducción: Esta parte es usada para garantizar la descripción general del sistema embebido, el software embebido y su uso final. Está compuesto por dos incisos entre los que se encuentra una tabla basada en literatura del área.

2.- Funcionalidad: Apartado usado para establecer las necesidades que debe satisfacer el software embebido cuando es usado en condiciones específicas. Se compone de dos incisos que abarcan el detalle del sistema embebido y software embebido.

3.- Interfaz con usuario: Este apartado sirve para obtener la definición inicial de características para la interfaz de usuario permitiendo que el software embebido sea fácil de entender y cumpla con su tarea eficientemente. El apartado se subdivide en tres incisos y se solicita incluir la descripción del prototipo de la interfaz con un diagrama a bloques o de flujo.

4.- Interfaces externas: Aquí se listarán interfaces con otro software o con hardware solicitadas de manera inicial para el software embebido que se elaborará e incluirá en sistema embebido, se considera ¿qué problema se resolverá en hardware y qué problema se resolverá en software? Pensando en evitar las consecuencias en costos que pueda implicar la decisión.

5.- Confiabilidad: Sirve para la especificación del nivel de desempeño del software con respecto a la madurez, tolerancia a fallas y recuperación. Compuesta por dos incisos hace uso de una tabla de perfil de operación, una lista de severidades de falla y preguntas de las herramientas a usar en pruebas.

6.- Eficiencia: Aquí va la especificación del nivel de desempeño del software con respecto al tiempo y a la utilización de recursos. Compuesta de dos incisos que indagan la forma de medir las funciones en tiempo del sistema y los recursos del procesador.

7.- Mantenimiento: A través de una tabla sugerida se solicita la descripción de elementos que facilitarán la compresión y la realización de las modificaciones futuras del software embebido.

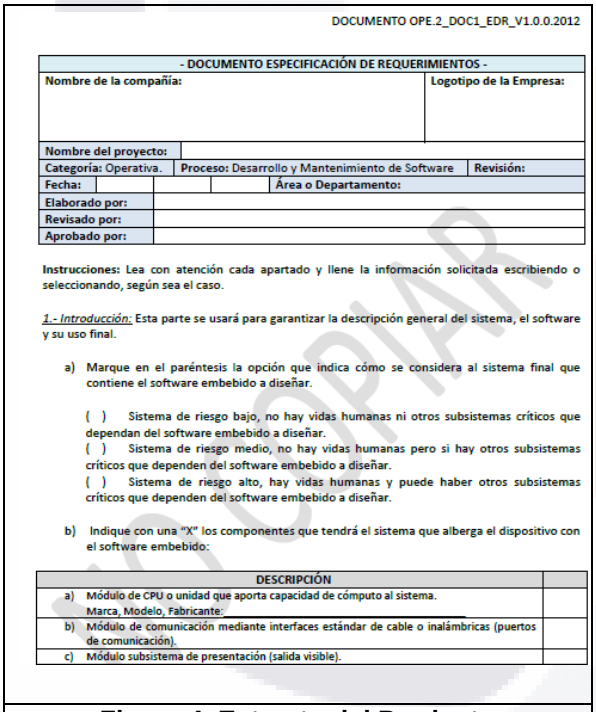
8.- Portabilidad: Este apartado procura visualizar la posibilidad de que, dadas las características del software embebido y el sistema embebido, sea posible la transferencia a una plataforma diferente o componente CPU diferente en un uso futuro.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

9.- Restricciones de diseño y construcción: En este apartado se solicita definir restricciones de diseño y construcción del sistema embebido y su software embebido usando una tabla de 12 consideraciones sugeridas y basada en literatura del área.

10.- Legales y Reglamentarios: Usamos este apartado para contemplar necesidades impuestas por leyes, reglamentos, estándares, entre otros. A través de tres incisos se solicita información de lo aplicable al sistema y su software.



DOCUMENTO OPE.2_DOC1_EDR_V1.0.0.2012

- DOCUMENTO ESPECIFICACIÓN DE REQUERIMIENTOS -

Nombre de la compañía:	**Logotipo de la Empresa:**
Nombre del proyecto:	
Categoría: Operativa. **Proceso:** Desarrollo y Mantenimiento de Software	**Revisión:**
Fecha:	**Área o Departamento:**
Elaborado por:	
Revisado por:	
Aprobado por:	

Instrucciones: Lea con atención cada apartado y llene la información solicitada escribiendo o seleccionando, según sea el caso.

1.- Introducción: Esta parte se usará para garantizar la descripción general del sistema, el software y su uso final.

a) Marque en el paréntesis la opción que indica cómo se considera al sistema final que contiene el software embebido a diseñar.

( ) Sistema de riesgo bajo, no hay vidas humanas ni otros subsistemas críticos que dependan del software embebido a diseñar.
( ) Sistema de riesgo medio, no hay vidas humanas pero si hay otros subsistemas críticos que dependen del software embebido a diseñar.
( ) Sistema de riesgo alto, hay vidas humanas y puede haber otros subsistemas críticos que dependen del software embebido a diseñar.

b) Indique con una "X" los componentes que tendrá el sistema que alberga el dispositivo con el software embebido:

DESCRIPCIÓN	
a) Módulo de CPU o unidad que aporta capacidad de cómputo al sistema. Marca, Modelo, Fabricante:	
b) Módulo de comunicación mediante interfaces estándar de cable o inalámbricas (puertos de comunicación).	
c) Módulo subsistema de presentación (salida visible).	

Figura 4. Extracto del Producto OPE.2_DOC1_EDR_v1.0.0.2012, sugerencia de autor.

De manera extraordinaria y aunque no lo menciona explícitamente el modelo MoProSoft para esta salida del proceso, se propone incluir un recuadro al inicio del producto que permita llevar a cabo el control de las versiones y siempre deje en claro a los involucrados los cambios o correcciones de los requerimientos que vayan surgiendo.

Actualmente el producto se encuentra en fase de validación con la colaboración del personal del departamento de MTE en Flextronics Manufacturing a través de la aplicación al proyecto Final Vision System que usa un NI cRIO-9074 de National Instrument para alberga al software embebido. Cabe mencionar que el proyecto ya ha sido concluido y la herramienta entregada a su usuario final, lo cual proporciona la facilidad de hacer una comparativa y verificar si el proceso de documentación de requerimientos durante el proyecto mejora con la aplicación del producto de nuestra propuesta del modelo.

5. Conclusiones

Los modelos para la calidad de Procesos de Software y SPI son tópicos que han tomado relevancia en el sector PyMe. Investigadores y especialistas de la Ingeniería de Software y en áreas afines han notado el interés de este sector por adoptar estándares bajo los limitados recursos que las caracterizan y sus formas de trabajo, con la finalidad de poder competir en el mercado internacional logrando la calidad de sus productos de software. A través de esos esfuerzos a nivel nacional se ha consolidado y se les ha facilitado el modelo de MoProSoft para certificarse en la norma NMX-I-059-NYCE-2005 y lograr sus objetivos.

Por otra parte los sistemas embebidos son un campo de acción que está teniendo un crecimiento considerable en la industria a través de la automatización de tareas y procesos y todos ellos dependen de la fiabilidad de su software embebido. Sin

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

embargo hay pocos esfuerzos reportados en la literatura que documenten de manera formal la fase de requerimientos y documentación del desarrollo del software embebido.

Es por ello que la presente propuesta pretende incursionar en el área del Software Embebido a través de la óptica de la Ingeniería de Software, facilitando la implementación de un modelo más formal y que se adapte a pequeños grupos desarrolladores de este tipo de software dadas sus características que lo diferencian de sistemas de software administrativo o de negocio. La creación del producto OPE.2_DOC1_EDR_v1.0.0.2012 es el inicio de un proyecto que se vislumbra con un futuro prometedor, en cuanto a simplicidad del proceso, a pesar de la complejidad que conlleva el desarrollo de los sistemas embebidos y sobre todo lleno de trabajos aún por hacer o proponer.

6. Trabajo Futuro

Actualmente se tiene en un proceso de ajuste al producto OPE.2_DOC1_EDR_v1.0.0.2012, lo cual implica que a través de la colaboración del equipo de MTE se hagan algunas modificaciones o restructuraciones para dejarlo en un punto acorde para su aplicación en otro tipo de proyectos.

A la par se encuentra el desarrollo del producto OPE.2_DOC2_AYD_v1.0.0.2012 que cubrirá en la Categoría Operación del proceso Desarrollo y Mantenimiento de Software el producto de salida de la Actividad A3: “Análisis y Diseño”. Su creación implica un reto considerable ya que debe contener la descripción textual y gráfica de la estructura de los componentes de software, pero esto conlleva a conocer también las arquitecturas que los diferentes sistemas embebidos manejan y sugerir las más exitosas o dar una idea general para que el equipo desarrollador pueda proponer la suya propia siguiendo las características del modelo de MoProSoft. Aunado a su conclusión se encontrará también la etapa de validación del producto con la colaboración del equipo de MTE.

7. Referencias.

[1] Tardío, María Antonia, Ailyn Febles Estrada, y Deborat Pérez Montalvan. "Primeras ideas de un modelo cubano de referencia para el desarrollo de aplicaciones informáticas." Revista Cubana de Ciencias Informáticas 5, no. 2 (2011), pp. 1-12.

[2] Paternina Palacio, Katia, y Deisy Ribón. "Calidad de software: Aplicación en las industrias desarrolladoras de Colombia”. Ingeniator 1, no. 2 (2011), pp. 65-74.

[3] Pino, Francisco, Juan Vidal, Félix García, y Mario Piattini. "Modelo para la implementación de mejora de procesos en pequeñas organizaciones software." XII Jornadas de Ingeniería del Software y Bases de Datos, JISBD 2007 (2007), pp. 326-335.

[4] Mishra, Deepti, y Alok Mishra. "Software process improvement in SMEs: A comparative view." Computer Science and Information Systems 6, no. 1 (2009): 111-140.

[5] Rios, Brenda Leticia Flores, María Angélica Astorga Vargas, José Martín Olguín Espinoza, and MdC Peralta. "Experiences on the Implementation of MoProSoft and Assessment of Processes under the NMX-I-059/02-NYCE-2005 Standard in a Small Software Development Enterprise." In Computer Science, 2008. ENC'08. Mexican International Conference on, pp. 323-328. IEEE, 2008.

[6] Hanna Oktaba, Claudia Alquicira Esquivel, Angélica Su Ramos, et al., “Modelo de Procesos para la Industria de Software, MoProSoft. Versión 1.3”, 2005, pp. 1-135.

[7] Hernández Vega, José Isidro. "El Software Embebido y los Retos que Implica su Desarrollo." ConCiencia Tecnológica 40 (2010): 42.

[8] González Palacio, Liliana, and Germán Urrego Giraldo. "Modelo de requisitos para sistemas embebidos." Revista Ingenierías Universidad de Medellín 7, no. 13 (2011): 111-127.

[9] Chandy, John C. "Challenges in the Design of Cyber-Physical Systems / Desafíos en el Diseño de Sistemas Cyber-Físicos." Facultad de Ingenierías 15 (2010): 5.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

[10] Gastaldi, Guillermo G., José A. Rapallini, and Héctor O. Pascual. "Evaluación de programas de cálculo en ingeniería como herramienta de desarrollo para Codiseño Hardware/Software." In IWS2002 VIII Workshop IBERCHIP, Guadalajara, México, Abril 2002. 2002.

[11] Sotelo Mauries, Vianney, Brizuela Sandoval, Mónica, y Álvarez Rodríguez, Francisco J. "Esfuerzo de Implementación de la Categoría de Operación - Administración de Proyectos Específicos – del Modelo MoProSoft, para Desarrollo de Software Embebido". Conisoft 2012, Guadalajara, México, Marzo 2012.

[12] Berger, Arnold, and Arnold H. Berger. Embedded systems design: An introduction to processes, tools, and techniques. Cmp, 2002.

[13] Dmitruk, Andrés E. UNLaM; Acosta Nelson; etal. "El software y lo sistemas embebidos o empotrados o insertados". Consultado en noviembre de 2012. http://ebookbrowse.com/el-software-y-los-sistemas-embebidos-andres-dmitruk-doc-d180199719.

[14] Heath, Steve "Embedded systems design". EDN series for design engineers, (2 edición). Newnes. p. 8-15. 2003.

[15] Mir Mohammad Azad and Mohammad Bin Amin, Alauddin. Shanto "Executive Information System".. Mariam University of Creative Technology, Uttara, Dhaka, Bangladesh. IJCSNS International Journal of Computer Science and Network Security, VOL.12 No.5, May 2012.

[16] "Ingeniería de la Confiabilidad de Software". Software Guru, 2005. Consultada en noviembre 2012. http://www.sg.com.mx/content/view/271.

[17] Caja Costarricense de Seguro Social (2008). Procedimiento de Control de Versiones TIC-GAC-0005. Gerencia Infraestructura y Tecnologías, Subgerencia Tecnologías de Información y Comunicaciones: Área de Soporte Técnico

TESIS TESIS TESIS TESIS TESIS