METODOLOGÍAS DE DESARROLLO DE SOFTWARE APLICADAS A PROYECTOS DE AUTOMATIZACIÓN DE PROCESOS

SOFTWARE DEVELOPMENT METHODOLOGIES APPLIED TO PROCESS AUTOMATION PROJECTS

Gabriela Melendrez Alaro
Universidad Católica del Norte, Chile
gabrielamelendrez@gmail.com

Recepción: 28/octubre/2020 **Aceptación:** 27/noviembre/2020

Resumen

El artículo presenta el estudio de los procesos de desarrollo de software y cómo ciertas características propias de las metodologías de desarrollo de software pueden ser aplicadas a proyectos de automatización. Las metodologías empleadas en procesos tecnológicos abarcan una línea básica de seguimiento, para un mejor control de los procesos que cada línea implique (Desarrollo de software - Automatización).

La identificación de características propias de las metodologías empleadas para el desarrollo de software permitió encontrar características que pueden ser empleadas en proyectos de automatización; pudiendo así acortar el tiempo de inmersión del talento humano al proyecto, contar con un historial de acontecimientos para futuras mejoras del proyecto de automatización. Las características propias de las metodologías ya consolidadas en la industria pueden juntas, alcanzar objetivos mejor trazados en la línea de tiempo del desarrollo del producto.

Palabras Clave: Ágil, automatización, ciclo, manifiesto, software.

Abstract

The article presents the study of software development processes and how certain characteristics of software development methodologies can be applied to automation projects. The methodologies used in technological processes cover a basic monitoring line, for better control of the processes that each line implies

(Software Development - Automation). The identification of characteristics of the methodologies used for software development, allowed to find characteristics that can be used in automation projects; thus being able to shorten the immersion time of human talent to the project, having a history of events for future improvements of the automation project. The characteristics of the methodologies already consolidated in the industry can together, achieve objectives better set in the product development timeline.

***Keywords**: Agile, automation, cycle, phase, manifesto, software.*

1. Introducción

A lo largo del desarrollo tecnológico, fueron emergiendo nuevas formas de organización, en el actuar de los equipos que trabajan en determinado proyecto, en procesos finitos que implican el desarrollo, tanto en proyectos de automatización y el proceso desarrollo de Software.

En el proceso, que los procesos en automatización y desarrollo de software tienen estrecha relación con la planificación a seguir. De acuerdo al proceso de desarrollo tecnológico, abarcan las siguientes fases: identificación, exploración, diseño, planificación, construcción, evaluación y divulgación. Mismo que nos muestra de manera general las fases que estas abarcan [Alvarez, 2014].

Los ciclos / etapas, comprenden actividades que permitirán llegar a los objetivos trazados, de acuerdo a las metodologías que el proyecto siga, el equipo genera documentos necesarios para la adquisición y montaje del proyecto, junto con la instrumentación e instalaciones necesarias, así como la definición de estrategias de control para cumplir con los objetivos en los proyectos industriales. Mismas que siguen un guión para controlar el avance que tiene cada proceso.

El estudio propone poder comparar los procesos contemplados actualmente en los procesos de automatización y las metodologías de desarrollo de software, de manera que estas puedan combinar sus características propias, para un futuro desarrollo basado en reutilización, donde podrán ser adoptadas entre sí. De acuerdo a un estudio publicado en Cielo de Colombia, indica que a nivel mundial el empleo de estas tecnologías en la industria se ha clasificado de acuerdo con el nivel de la

organización en donde se utilizan, bien sea en funciones administrativas, en el diseño integrado de producto y proceso, en la planeación de la producción, o en el control de esta; misma que nos da un panorama de la importancia de los procesos documentados, dentro del marco de automatización y desarrollo de software, ambos procesos tecnológicos.

La industria actual debe abordar un cambio tecnológico en los procesos de producción, para mantener el liderazgo en el entorno económico del tercer milenio. De acuerdo con el Grupo Asesor Industrial de la Comisión Europea [Mejia, 2017], las "fábricas del futuro" requieren sistemas automatizados más complejos, seguros y fiables. Para acometerlos, las claves de las tecnologías avanzadas de fabricación residen en la robótica, la automatización y la fabricación integrada por ordenador. En este contexto, el desarrollo de software para procesos de control, constituye uno de los retos en el campo de la automatización para garantizar la disponibilidad de una planta que proporcione una producción de alta calidad. El desarrollo de este tipo de software se enfrenta a la complejidad de integrar las tecnologías de la información en el entorno industrial, a la hora de transformar información en acciones sobre herramientas de un modo totalmente controlado. Para ello, es preciso generar, transmitir y procesar la información de modo rápido, exacto y fiable, en cantidad y calidad adecuada a las necesidades específicas, en el momento preciso y en el lugar idóneo. Este flujo distribuido de datos debe garantizar el correcto flujo de materiales de un modo automático.

Para satisfacer estos requisitos es preciso que los últimos avances en hardware, software y tecnologías de la información y las comunicaciones soporten el ciclo de vida de las aplicaciones. En este sentido, trabajos recientes realizan una inspección crítica sobre el estado del arte con relación a las prácticas de ingeniería de software, métodos y herramientas, que se utilizan en la etapa de desarrollo de procesos automatizados. Se coincide en la necesidad de nuevas metodologías y herramientas debido al gran esfuerzo que requiere la adaptación de las existentes para incorporarlas en el dominio de la automatización. Entre los temas pendientes para futuras investigaciones apuntan una nueva generación de herramientas de ingeniería que abarquen más de una fase de desarrollo; en particular, el análisis con

el fin de evitar malentendidos en el proceso de especificación de los requisitos. Por lo tanto, la adopción de los principios de ingeniería de software en la automatización industrial requiere la integración de métodos y herramientas, o al menos, que estas últimas proporcionen interfaces con otras herramientas de ingeniería. El punto métodos, hace referencia a los modelos y formas organizacionales que se conocen hoy en día.

2. Métodos

Los proyectos de automatización suelen presentar ciclos de vida lineales: cada fase se realiza una sola vez, detrás de la fase anterior y antes de la siguiente. Cada actividad del proyecto puede descomponerse de manera que, una fase no necesite resultados de las siguientes (realimentación), aunque pueden admitirse ciertos supuestos de realimentación correctiva. Desde el punto de vista de la gestión (para decisiones de planificación), es requerido que se prevea de antemano lo que va a ocurrir en cada fase antes de iniciarla.

Estos ciclos comprenden las actividades de diseño requeridas con el objeto de producir los documentos necesarios para la adquisición y montaje de sistemas de control, junto con la instrumentación e instalaciones necesarias, así como la definición de estrategias de control para cumplir los requerimientos de automatización, en los proyectos industriales.

Los proyectos de automatización, en plantas químicas, por ejemplo, suelen darse como parte integrante de proyectos industriales que engloban al resto de disciplinas (procesos, electricidad, civil y estructuras, mecánica, tuberías, dibujo, proyecto y otros), en los que las diferentes especialidades que participan interactúan dentro de los márgenes que les son propios en las diferentes fases del diseño. En la eficacia de este trabajo en equipo, influye la experiencia de los profesionales que participan y las decisiones más o menos acertadas que se adopten.

Es necesario fijar a priori, en el desarrollo de un proyecto industrial, el alcance de este, definiendo hasta qué fase llegará el proyecto. También es necesario definir las actividades a realizar en cada una de las fases que componen el ciclo de vida, así como las relaciones entre las diferentes actividades y la información que genera,

almacena e intercambia en cada actividad (proceso) con el resto (sistema de información del proyecto).

En este sentido, con el fin de hacer frente a la creciente complejidad del software, se introdujeron mayores niveles de abstracción durante el proceso de desarrollo en procesos de control, tal y como propugna MDE (Model Driven Engineering). Las tecnologías MDE combinan abstracciones específicas de dominio (descritas mediante meta-modelos que expresan la ontología del dominio) que permiten definir la estructura de la aplicación, su comportamiento y los requisitos para dominios particulares, y los motores de transformación y generadores que producen determinados aspectos de los modelos para sintetizar varios tipos de artefactos. Dado que en la bibliografía reciente existen numerosas propuestas de entornos de desarrollo integrados que adoptan los principios de la ingeniería del software conducida por modelos, MDE puede considerarse una tendencia en el área de la automatización industrial. La mayoría de los trabajos utilizan UML (Unified Modeling Language). Casi todos ellos soportan el desarrollo de aplicaciones de automatización, pero apenas cubren la fase inicial de análisis de requisitos.

Sin embargo, estos trabajos están lejos del dominio del problema, lo que conlleva una baja aceptación por parte de la industria, motivo por el cual no se ve una consolidación real sobre las herramientas usadas, pues nos siempre se tiene documentación recabada en cada proceso. La alternativa reside en lenguajes de modelado específicos de dominio y herramientas basadas en métodos y estándares ampliamente utilizados en el campo de la automatización. El lenguaje de modelado, GRAFCET (GRAphe Fonctionnel de Commande, Etapes, Transitions) y GEMMA constituye su extensión natural para definir los modos gráficos (representación de los procesos) de operación. En procesos de automatización, varios autores han desarrollado diferentes metodologías que combinan GEMMA y GRAFCET.

Por ejemplo, propone la implementación del ciclo de producción con GRAFCET como punto de partida para aplicar GEMMA con el fin de desarrollar sistemas de automatización, combina el diagrama de casos de uso de UML [Chiron, 2007], GEMMA y GRAFCET en una metodología orientada a objetos para sistematizar el modelado de sistemas secuenciales de eventos discretos, GEMMA traduce, el

asociado al comportamiento del sistema a un SFC (Sequential Function Chart) de alto nivel y cada estado GEMMA se traduce a SFCs de bajo nivel, logrando la sincronización a través de la coordinación vertical, como se muestra en la figura 1. Sin embargo, ninguna de estas metodologías se beneficia de las ventajas de MDE.

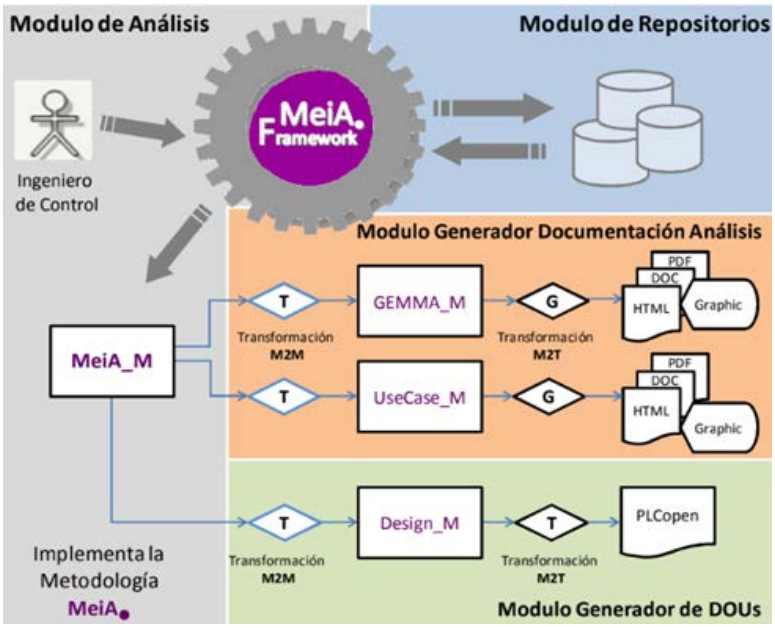


Figura 1 Arquitectura de la herramienta MeiA.

Con objeto de paliar esta problemática, este trabajo presenta una herramienta desarrollada bajo el enfoque MDE que implementa MeiA (Methodology for industrial Automation systems), concebida para desarrollar software de control en el ámbito de la ingeniería de automatización de procesos, MeiA proporciona directrices de diseño con el fin de guiar metodológicamente a los desarrolladores a través del proceso de desarrollo, desde el análisis y el diseño, hasta la implementación y operación. Como capa de abstracción para el usuario final, la herramienta proporciona una interfaz fácil de usar que permite un desarrollo sencillo y gráfico de las aplicaciones. Además, genera la documentación de análisis y diseño utilizando la terminología del dominio, y las unidades mínimas de diseño (denominadas DOUs - Design Organization Units) necesarias para generar los POUs (Program Organization Units). Estos últimos se generan en formato XML. MeiA satisface los siguientes requisitos:

- Una interfaz gráfica amigable que permita un proceder sencillo, intuitivo y gráfico.
- Un proceso de desarrollo incremental que permita trabajar a diferentes perfiles a través del análisis, el diseño y la implementación.
- La generación de documentación de análisis y diseño utilizando la terminología del dominio, es decir, la guía GEMMA, el diagrama de casos de uso con las descripciones de los casos de uso y los DOUs.
- La generación de los POUs en formato XML PLCopen a partir de los DOUs.
- El almacenamiento de los activos generados en repositorios distribuidos para su gestión.

La arquitectura de esta herramienta consta de 4 módulos (figura 1):

- El Módulo de Análisis guía al usuario a través de las fases y pasos de MeiA por medio de menús y perspectivas. Como salida se obtiene el modelo MeiA_M que contiene toda la información de análisis y diseño del sistema de control a implementar.
- El Módulo Generador de Documentación de Análisis genera el modelo GEMMA_M y el modelo del diagrama de casos de uso (UseCase_M). A partir de estos modelos se genera la documentación de análisis.
- El Módulo Generador de DOUs genera el modelo de diseño Design_M compuesto por los DOUs, a partir de los cuales se generan los POUs automáticamente en el formato XML de PLCopen.
- El Módulo de Repositorios consta de repositorios distribuidos que almacenan los modelos y los POUs generados para el sistema de control, de forma que se pueda realizar una gestión efectiva de los mismos.

Los modelos citados son conformes a sus respectivos metamodelos. Cada metamodelo describe los conceptos del dominio, las relaciones entre ellos y las reglas de composición que definen las limitaciones del dominio semántico. La generación de los modelos de análisis (GEMMA_M, UseCase_M) y el modelo de diseño (Design_M) se realiza mediante transformaciones modelo a modelo (M2M -

Model To Model) a partir del modelo MeiA_M. Para la definición de las transformaciones se han establecido un conjunto de reglas que describen cómo elementos de un metamodelo son transformados en uno o más elementos del metamodelo destino. Por otra parte, se utilizan generadores que implementan transformaciones M2T (Model To Text) para generar la documentación de análisis y diseño (tanto en representación gráfica como textual) y la generación de los POUs en formato PLCopen. Conociendo el modelo enfocado a procesos de automatización, brinda una estructura organizacional al momento de desarrollar un proyecto bajo esta línea, pero además de conocer las herramientas que existen debemos conocer el proceso general que siguen los proyectos de automatización.

Es importante definir al inicio del proyecto, un procedimiento estricto para la inclusión de cambios de alcance o de pequeñas modificaciones que puedan ser requeridas durante el ciclo de vida del proyecto. Las fases que componen el ciclo de vida de un proyecto industrial las observamos en la figura 2.

Etapas	Fases	Realización por (depende de la magnitud del proyecto)
	Definición y Análisis de Objetivos	Cliente
	Requisición del Proyecto	Cliente
	Preparación de Ofertas	Empresa de Ingeniería
Planificación	Adjudicación: Comparación de Ofertas, Adjudicación del Proyecto y Realización del Pedido	Cliente
	Ingeniería Conceptual	Empresa de Ingeniería
	Ingeniería Básica	Empresa de Ingeniería
	Ingeniería de Detalle	Empresa de Ingeniería
Inplementación	Ejecución del Proyecto	Empresa de Ingeniería (con contratistas)
Entrega	Entrega Proyecto: Pre-comisionado, Comisionado y Puesta en Marcha	Empresa Ingeniería - Cliente (con contratistas)
Mantenimiento	Mantenimiento y Modificaciones	A definir por Cliente
Desmantelamiento	Decomisionado	A definir por Cliente

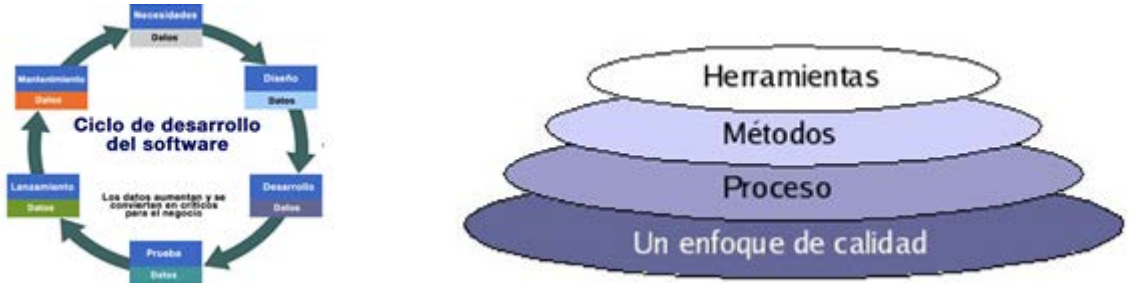
Figura 2 Etapas y fases de proceso de automatización.

Mediante la definición de objetivos, quien solicita el proyecto (cliente) determina cual es el objeto de este y realiza una estimación de costes (para cada fase), así como la estimación del tiempo de realización (por fases). También se determinan cuáles son los requerimientos iniciales y se realiza el estudio de la factibilidad. Durante esta fase se toma la decisión de acometer o no el proyecto. Si se decide acometer el

proyecto, una vez definidos los objetivos, se describen las actividades principales del mismo, se planifican, y se identifican los recursos (presupuesto), identifican los recursos (presupuesto). La requisición del proyecto de automatización es el documento (especificación técnica) que recoge los requerimientos del cliente para la realización del proyecto de automatización (diseño, fabricación, programación del software, configuración, pruebas y entrega). Este documento incluye, además, los estándares requeridos por el cliente para la realización de este. La requisición es utilizada por las empresas de ingeniería para confeccionar ofertas que incluyan los requerimientos del cliente, el coste económico y los plazos de entrega.

Un proceso de desarrollo de software tiene como propósito la producción eficaz y eficiente de un producto software que reúna los requisitos del cliente. Un producto software en sí es complejo, es prácticamente inviable conseguir un 100% de confiabilidad de un programa por pequeño que sea. Existe una inmensa combinación de factores que impiden una verificación exhaustiva de las todas posibles situaciones de ejecución que se puedan presentar (entradas, valores de variables, datos almacenados, software del sistema, otras aplicaciones que intervienen, el hardware sobre el cual se ejecuta, etc.).

Por otro lado, un producto software es intangible y por lo general muy abstracto, esto dificulta la definición del producto y sus requisitos, sobre todo cuando no se tiene precedentes en productos software similares. De manera que los requisitos sean difíciles de consolidar tempranamente. Así, los cambios en los requisitos son inevitables, no sólo después de entregar el producto sino también durante el proceso de desarrollo. El proceso de desarrollo de software no es único. No existe un proceso de software universal que sea efectivo para todos los contextos de proyectos de desarrollo de software. Debido a esta diversidad, es difícil seguir la misma línea metodológica del proceso de desarrollo de software. Pero estas siguen un proceso similar, como lo muestra la figura 3. Al igual que en proyectos de automatización. Ambas al ser proyectos tecnológicos, presentan un punto en común; la documentación de los procesos de desarrollo. La figura 3 muestra la interrelación de herramientas, métodos, procesos todo para un enfoque de calidad, mismas que también son reflejadas en pasos para el diseño del sistema de control.



a) Ciclo de desarrollo de software. *b) Software como tecnología multicapa.*

Figura 3 Ciclo de desarrollo de software y Software como tecnología multicapa.

Para conocer el proceso que siguen las metodologías de desarrollo de software, citaremos algunas:

- El modelo de Cascada es considerado como el método tradicional de explicar el proceso de desarrollo de software, por lo que actualmente es visto como anticuado. Sin embargo, aún sigue siendo aplicado a proyectos con metas y requisitos claros, sobre todo considerando que este enfoque permite a los negocios deshacerse del papeleo innecesario, reuniones regulares que consumen mucho tiempo y retrasos en sus procesos de negocio, pero la documentación no es dejada de lado, ya que es presentado un informe al final del proyecto, es conocida además como un modelo bastante inflexible.
- El modelo de Espiral es la última evolución del modelo cascada, aprovechando el hecho de que los proyectos de desarrollo funcionan mejor cuando son incrementales e iterativos. La metodología espiral refleja la relación de tareas con prototipos rápidos, mayor paralelismo y concurrencia en las actividades de diseño y construcción. El método en espiral debe todavía ser planificado metódicamente, con las tareas y entregables identificados, debidamente documentadas, para cada paso en la espiral.
- Metodología de Prototipo, es un procedimiento de desarrollo especializado que permite a los desarrolladores la posibilidad de poder solo hacer la muestra de la resolución para poder validar su esencia funcional ante los clientes, y hacer los cambios que sean fundamentales antes de crear la solución final auténtica. De hecho, la mejor parte de esta metodología es que tiende a resolver un conjunto de problemas de diversificación que ocurren

con el método de la cascada. Además de esto, la gran ventaja de optar por este enfoque es que da una idea clara sobre el proceso funcional del software, reduce el riesgo de falla en una funcionalidad de software y asiste bien en la recolección de requisitos y en el análisis general, el prototipo es parte de la documentación.

- Desarrollo Rápido de Aplicaciones (RAD), con el objetivo de otorgar resultados rápidos, se trata de un enfoque que está destinado a proporcionar un excelente proceso de desarrollo con la ayuda de otros enfoques, pero, además, está diseñado para aumentar la viabilidad de todo el procedimiento de desarrollo de software para resaltar la participación de un usuario activo. Dicho esto, algunas de las ventajas a destacar de este tipo de desarrollo son las siguientes: Hace todo el proceso de desarrollo sin esfuerzo; asiste al cliente en la realización de revisiones rápidas y alienta la retroalimentación de los clientes para su mejora; cada una de estas es debidamente documentada.
- Metodología de Programación Extrema (XP), metodología ágil (eXtreme Programming) que se utiliza principalmente para evitar el desarrollo de funciones que actualmente no se necesitan, pero sobre todo para atender proyectos complicados. Sin embargo, sus métodos pueden tomar más tiempo, así como recursos humanos en comparación con otros enfoques.
- Scrum, esta metodología, contiene un marco de trabajo de procesos ágiles que trabaja con el ciclo de vida iterativo e incremental, donde se va liberando el producto por pares de forma periódica, aplicando las buenas prácticas de trabajo colaborativo (en equipo), facilitando el hallazgo de soluciones óptimas a los problemas que pueden ir surgiendo en el proceso de desarrollo del proyecto. Con Scrum se realizan entregas regulares y parciales (sprint) del producto final, todas ellas con una prioridad previamente establecida que nace según el beneficio que aportan al cliente, minimizando los riesgos que pueden surgir de desarrollos extremadamente largos. Es por tal motivo, que Scrum está especialmente indicado para proyectos en entornos complejos, donde se necesitan obtener resultados de manera inmediata y donde son

fundamentales los siguientes aspectos: la innovación, la productividad, la flexibilidad y la competitividad.

Estas son solo algunas de las metodologías de Desarrollo de Software que existen, pero lo importante es que se tenga en cuenta que al estar familiarizado con estos ayudará a poder detectar, qué metodología como herramienta, pueda ayudar a la organización los proyectos, utilizando un enfoque puro o combinando algunos de ellos.

3. Resultados

Se conoce que, en la actualidad, se realizan proyectos de automatización, junto a software a medida, y además se conoce que la tecnología va en constante cambio a mejoras, tanto software como hardware. Por tal motivo es necesario seguir una línea de desarrollo que permita planificar actividades incluso para la documentación de cada proceso. Si bien las herramientas organizacionales y de control, ayudan en el proceso, estas no hacen énfasis a documentación que pueda estar relacionada tanto como en sistemas de control con procesos del software que los controla. Es posible contar con un control estricto de los procesos realizados en ambas líneas de desarrollo, generando documentación de cada periodo de tiempo, pudiendo así aplicar el desarrollo basado en reutilización.

De acuerdo con las metodologías de desarrollo de software planteadas, podemos observar que cada una de ellas presentan particularidades, propias de la metodología. La tabla 1 muestra que los modelos formales pueden ayudarnos a realizar el proceso de documentación.

Tabla 1 Visibilidad de proceso de acuerdo con el tipo de modelo.

Modelo de proceso	Visibilidad del proceso
Modelo de cascada	Buena visibilidad, cada actividad produce un documento o resultado.
Desarrollo evolutivo	Visibilidad pobre, muy caro al producir documentos en cada iteración.
Modelos formales	Buena visibilidad, en cada fase deben producirse documentos.
Desarrollo orientado a la reutilización	Visibilidad moderada. Importante contar con documentación de componentes reutilizables.
Modelo espiral	Buena visibilidad, cada segmento y cada anillo de la espiral debe producir un documento.

En figuras 4 y 5 encontramos la relación hallada entre las etapas del desarrollo de software y los pasos para el diseño de un sistema de control, donde se marca con los colores respectivos la afinidad de cada etapa/paso. Donde la elaboración de la documentación puede ser apoyada en cada fase.

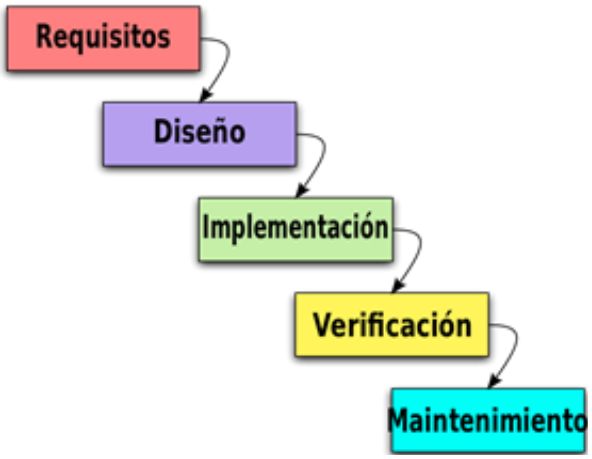


Figura 4 Etapas del desarrollo de software.

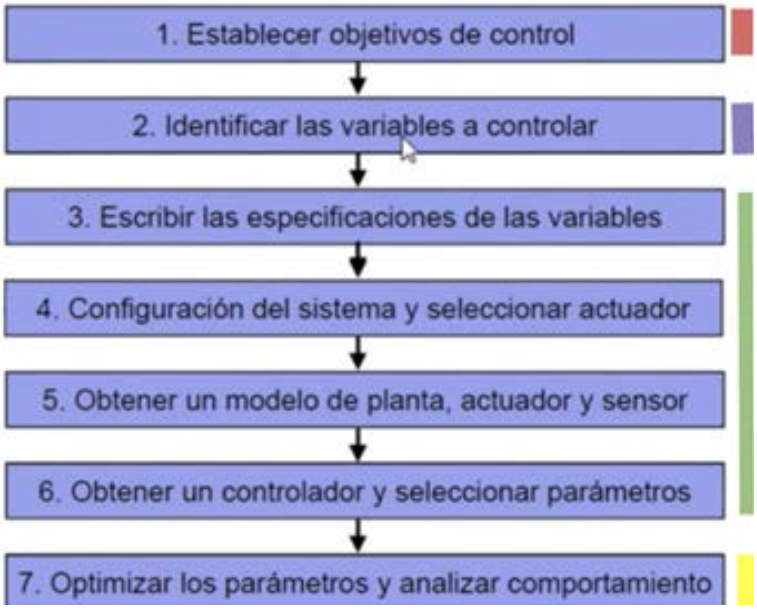


Figura 5 Pasos para el diseño de un sistema de control.

En la figura 6 observamos que los conceptos de la terminología mostrada tienen estrecha relación con las etapas de software.

Figura 6 Ciclo lineal para un proyecto de automatización.

4. Discusión

En la tabla 2 de manera conjunta se aprecia la relación fáctica de cada proceso. Donde los procesos del proyecto de automatización y desarrollo de software encuentran 'sintaxis', en cuanto a proceso se refiere, y no 'semántica'. Pero dicha 'sintaxis' tiene gran aporte en dichos procesos, puesto que de manera paralela mas no junta, tiene la fortaleza de contemplar el mundo del sistema como tal, traducida en manuales integrales, de apoyo para el proyecto.

Tabla 2 Relación fáctica de cada proceso.

Actividades fundamentales del proceso del desarrollo de un proyecto de automatización y proceso de desarrollo de software.	
El proceso de desarrollo del software	Proceso de desarrollo de proyecto de automatización
Especificación de software	Definición de requisitos
Se debe definir la funcionalidad y restricciones operacionales que debe cumplir el software.	
Diseño e implementación	Diseño del producto
Se diseña y construye el software de acuerdo con la especificación.	
Validación	Construcción, instalación
El software debe validar, para asegurar que cumpla con lo que quiere el cliente.	
Evolución	Aceptación por el cliente
El software debe evolucionar, para adaptarse a las necesidades del cliente.	

5. Conclusiones

Se concluye que:

- El conocer la esencia de las metodologías, muestran herramientas que permiten una mejor organización, en cuanto al proyecto como tal y a los tiempos que estos otorgan para una mejor documentación.

- El contar con documentación elaborada de manera integral, para proyectos que desarrollen ambos procesos (automatización y desarrollo de software), muestra un panorama más amplio y preciso, mismos que son de gran ayuda al momento de tomar en cuenta nuevos requerimientos o adaptabilidad a nuevas tecnologías y reducir costos de adaptabilidad, cuando nuevos miembros se incluyen en los equipos.
- La metodología aplicada a los procesos de automatización muestra al proyecto de manera conjunta, conociendo de manera integral los objetivos y métricas de tiempo más específicas para cada requerimiento.

6. Bibliografía y Referencias

[1] Álvarez, M., L.; Departamento de Ingeniería de Sistemas y Automática, EUITI de Bilbao, UPV/EHU, Desarrollo Metodológico de Sistemas de Control Aplicando Ingeniería Conducida Por Modelos, Actas de las XXXV Jornadas de Automática, ISBN-13: 978-84-697-0589-6, Comité Español de Automática de la IFAC (CEA-IFAC), Valencia, España, 3-5 de septiembre de 2014.

[2] Alvarez, M., L., Sarachaga, I., Burgos, A., Estévez, E., Marcos, M., (2012), A methodological support for designing industrial control systems, 17th IEEE International Conference on Emerging Technologies and Factory Automation, Poland.

[3] Aschhoff, V., Crass, B., Cremers, D., Grimpe K., Rammer, C., Brandes, C., Diaz Lopez, F., Klein Woolthuis, F., Mayer, R., M., Montalvo, C., (2010) European Competitiveness in Key Enabling Technologies, ZEW & TNO.

[4] Chiron, F., Kouiss, K., (2007), Design of IEC 61131-3 Function Blocks Using SysML, 15th Mediterranean Conference on Control & Automation, Athens, pp. 915-919.

[5] EFFRA & ManuFuture, (2010), Factories of the Future PPP. Strategic Multi-Annual Roadmap, Ad-hoc Industrial Advisory Group of the Factories of the Future Public- Private Partnership.

[6] Jacobson, I., (1992) Object Oriented Software Engineering. A Use Case Driven Approach, Addison Wesley.

[7] Maida, Esteban Gabriel, Pacienzia, Julián, METODOLOGÍAS DE DESARROLLO DE SOFTWARE, Tesis Final, Cátedra: Seminario de Sistemas Carrera: Licenciatura en Sistemas y Computación, Universidad Católica Argentina, 2015.

[8] Mejia, Reporte proyecto industrial, Santiago De Queretaro, México, 30 de agosto de 2017.

[9] Rozo Nader Janeth, Metodología de Desarrollo de Software: MBM (Metodologia Basada en Modelos), INGENIARE, Universidad Libre-Barranquilla, Año 9, no. 16, pp. 111-125, ISSN: 1909-2458, 2014.

[10] Sáez Mateo Julián, Metodología para el Desarrollo de Proyectos de Automatización Industrial (Instrumentación y Sistemas de Control) Aplicaciones Prácticas, Cartagena, 18 de mayo de 2011.

[11] Thramboulidis, K., Perdikis, D., Kantas, S., (2007), Model driven development of distributed control applications, International Journal of Advanced Manufacturing Technology, vol. 33, no. 3-4, pp. 233-242.

[12] Vallejo y Vallejo, Aspectos generales de la automatización industrial del sector farmacéutico. Rev. Col. Cienc. Quím. Farm. Vol. 35 (1), 47-63.