ESCUELA POLITÉCNICA DEL EJÉRCITO

EXTENSION LATACUNGA

MAESTRIA EN INGENIERIA DEL SW

I PROMOCION

"METODOLOGIA DE DISEÑO, DESARROLLO Y EVALUACION DE SOFTWARE PARA JUEGOS DE GUERRA"

ELABORADO POR:

ING. PABLO JACINTO CARRILLO ANDRADE
ING. LUIS EDUARDO ORTEGA GUTIERREZ

PROYECTO PREVIO A LA OBTENCION DEL TITULO DE:

MASTER EN INGENIERIA DEL SOFTWARE

Año 2012

CERTIFICACIÓN

Se certifica que el presente trabajo fue desarrollado por el Ing. Pablo Jacinto Carrillo Andrade y el Ing. Luis Eduardo Ortega Gutiérrez, bajo mi supervisión.

_____

Ing. Marco Quintana PHD

DIRECTOR DEL PROYECTO

ESCUELA POLITÉCNICA DEL EJÉRCITO
SUBDIRECCIÓN DE INVESTIGACIÓN Y VINCULACIÓN CON LA COLECTIVIDAD

CERTIFICADO

Ing. Marco Quintana PHD (DIRECTOR)

CERTIFICA:
APROBACIÓN DEL TUTOR

Que en el trabajo titulado **"METODOLOGIA DE DISEÑO, DESARROLLO Y EVALUACION DE SOFTWARE PARA JUEGOS DE GUERRA"**, realizado por el Ing. Pablo Jacinto Carrillo Andrade y el Ing. Luis Eduardo Ortega Gutiérrez, ha sido guiado y revidado periódicamente y cumple normas estatutarias establecidas por la ESPE, en el Reglamento de Estudiantes de la Escuela Politécnica del Ejército.

Debido a que constituye un trabajo de contenido científico y de raigambre militar que coadyuvará a la publicación de conocimientos en este campo y por ser de carácter reservado NO se recomienda su publicación.

El mencionado trabajo constara de TRES empastados y TRES Discos Compactos los cuales contienen los archivos en formato PDF. Autorizan a los señores Ing. Pablo Jacinto Carrillo Andrade y el Ing. Luis Eduardo Ortega Gutiérrez que lo entreguen al Sr. Tcrn. E.M. Robert Vargas Jefe de Investigación y Vinculación con la Colectividad de la ESPE Extensión Latacunga.

Latacunga, noviembre del 2012

_____
Ing. Marco Quintana PHD
DIRECTOR DEL PROYECTO

ESCUELA POLITÉCNICA DEL EJÉRCITO
SUBDIRECCIÓN DE INVESTIGACIÓN Y VINCULACIÓN CON LA COLECTIVIDAD

DECLARACIÓN DE RESPONSABILIDAD

Nosotros: Ing. Pablo Jacinto Carrillo Andrade
Ing Luis Eduardo ortega Gutiérrez

DECLARAMOS QUE:

El Proyecto de Grado denominado: **"METODOLOGIA DE DISEÑO, DESARROLLO Y EVALUACION DE SOFTWARE PARA JUEGOS DE GUERRA"**, ha sido desarrollado con base a una investigación exhaustiva, respetando derechos intelectuales de terceros, cuyas fuentes se incorporan en la bibliografía.

Consecuentemente este trabajo es de nuestra autoría.

En virtud de esta declaración, nos responsabilizamos del contenido, veracidad y alcance científico del proyecto de grado en mención.

Latacunga, noviembre del 2012.

_____	_____
Ing. Pablo Carrillo A.	Ing. Luis Ortega G.
C.I. 170715482-7	C.I. 170725445-2

AGRADECIMIENTO

El presente trabajo simboliza una etapa importante de nuestras vidas y materializa todo el esfuerzo realizado para mejorar la profesión por la que tanta vocación sentimos. Todo esto no hubiera sido posible sin la ayuda de las personas que nos rodean y a las cuales no quisiéramos dejar de agradecer.

A nuestras familias por inculcarnos la importancia de tener una profesión y generar el desafío personal por alcanzarla. A los profesores que me formaron durante esta etapa de nuestras vidas, a mis compañeros de estudio y al Sr. Tcrn. de E.M. Marco Quintana por el esfuerzo dedicado como director de la presente tesis.

ING. PABLO JACINTO CARRILLO ANDRADE
ING. LUIS EDUARDO ORTEGA GUTIÉRREZ

ÍNDICE

INDICE DE GRAFICOS

INDICE DE TABLAS

RESUMEN

La presente tesis realiza una contribución en cuanto al diseño, el desarrollo y la evaluación del software para juegos de guerra simulados. La construcción del software según los métodos y prácticas de la Ingeniería del Software (IS) trata de forma variada el proceso de desarrollo, la metodología que se propone deberá ayudar a los ingenieros de software a seleccionar las técnicas y metodologías que serán aplicables a los procesos de construcción del software para juegos de guerra en sus distintas etapas.

Este trabajo aporta, por primera vez, una metodología que facilita la evaluación de las técnicas utilizadas durante el proceso de desarrollo, dando respuesta a las posibles necesidades de un amplio abanico de organizaciones y procesos software.

Palabras clave: Juegos de guerra simulados, diseño, proceso de desarrollo, evaluación de software, construcción de software, evaluación de técnicas, procesos software.

SUMMARY

This thesis makes a contribution in the design, development and evaluation software for simulated war games. The construction of the software according to the methods and practices of Software Engineering (IS) is varied shape the development process, the methodology proposed should help software engineers to select the techniques and methodologies that apply to the processes of construction of war games software at different stages.

This work provides, for the first time, a methodology that facilitates the evaluation of the techniques used during the development process, responding to the possible needs of a wide range of organizations and software processes.

Keywords: simulated war games, design, development process, software construction, software evaluation, evaluation techniques, software process.

INTRODUCCIÓN

La era actual de la informática nos permite situarnos en escenarios virtuales simulando situaciones, sucesos y problemas reales; las herramientas actuales nos permiten desarrollar todos los requerimientos de simulación que requiere un entrenador táctico así como realizar las pruebas de evaluación de los modelos de conducta de las aplicaciones de Generación de Fuerzas por Computador.

Los juegos de guerra resultan vitales para enseñarle al liderazgo militar cómo planificar mejor, cómo formular las preguntas adecuadas, cómo prever y cómo adaptarse a múltiples situaciones; en la mayoría de los ejércitos constituye la herramienta más importante para la preparación de su personal. El juego de guerra fomenta el entendimiento del "arte operacional" de la guerra, provee experiencia en la toma de decisiones, le da vida al aprendizaje de los libros y al estudio en clase, robusteciendo las lecciones de la historia e iluminando las teorías detrás de una planificación y ejecución eficaz.

Sin embargo, estos grandes beneficios de un simulador de combate no suceden sin una inversión que comienza con el reconocimiento del valor que el juego de guerra representa para la capacitación profesional y el adiestramiento para las operaciones militares; por lo que se hace imprescindible que el desarrollo de estos sistemas de entrenamiento cumplan con las expectativas que de ellos se requiere.

Los sistemas de simulación son un conjunto de herramientas de software que se utilizan para el desarrollo de aplicaciones y la generación y ejecución de escenarios virtuales a través de interfaces gráficas (GUI) que permiten construir escenarios de manera fácil, posicionando fuerzas, creando rutas, capas, áreas y puntos de control, asignando tareas o planes a las unidades.

Con esta metodología se pretende que la evaluación del software para juegos de guerra sea una herramienta innovadora y que a futuro le permita al usuario lograr

estrategias de guerra exitosas. En este trabajo se muestra cómo un software bien desarrollado forma parte intrínseca de la misión de “organizar, capacitar y equipar” al servicio militar.

CAPITULO I

ESTADO DEL ARTE

Esta frase relativamente nueva en nuestro idioma viene de "State of the art", expresión del inglés, que denota el más alto nivel de desarrollo de un sistema o campo científico, alcanzado en un tiempo determinado.

En el ámbito del desarrollo de software, el estado del arte típicamente ha estado vinculado con la evolución de los lenguajes de computación. En sus inicios, el desarrollo de software fue considerado “programación automática” porque requería menos conocimientos técnicos que los lenguajes ensambladores.

Esta es una de las primeras etapas que se debe considerar en el desarrollo de un sistema puesto que su elaboración consiste en “ir tras las huellas” del tema que se pretende investigar, permite determinar cómo se va a tratar el tema, cómo se encuentra en el momento de realizar la propuesta de investigación y cuáles son las tendencias.

Dentro de un escrito académico técnico, se denomina *Estado del Arte* a la base teórica sobre la que se basa el escrito, o la cual se rebate en el desarrollo posterior en el escrito y que forma parte introductoria del mismo.

Quien desarrolla un sistema debe poseer conocimientos solidos sobre el estado del arte en los distintos aspectos de la construcción y evaluación de la calidad de los sistemas informáticos, de las técnicas modernas de la ingeniería de software para desarrollar sistemas informáticos, de los modelos formales subyacentes a estas técnicas; de los modelos de procesos y calidad más adecuados al tipo de producto a elaborarse, de las tendencias en tecnologías

informáticas tanto en lo referente a software y hardware como a comunicaciones.

Este trabajo, como muchos otros, no pretende establecer una nueva base teórica, sino que se apoya en una serie de conceptos abiertamente aceptados, a partir de los cuales se intenta avanzar hacia los objetivos propuestos. Es necesario hacer un repaso del estado actual de las tecnologías de las que se hace uso, así como de los conceptos sobre los que se apoya este trabajo.

1. LAS TEORÍAS DE LA GUERRA

1.1. Resumen

Para las gentes de otras épocas, fanatizadas y sin cultura, la comprensión del porqué de las guerras era fácil. Pero en estos tiempos la cuestión se vuelve complicada: la desinformación o la avalancha de información, hacen que la sociedad demande conocer las razones de tal o cual conflicto.

Como argumentaba Clausewitz, las guerras son un "arte complejo", en el que cada parte tiene sus razones, casi siempre encontradas e incomprensibles, no sólo para los contendientes, sino para el común de la gente.

Las relaciones humanas son conflictivas desde siempre, necesariamente chocamos unos y unas con otras y otros. Por tanto de manera realista podemos decir que estamos unidos para odiarnos, marginarnos, excluirnos y hacernos la guerra; pero también para querernos, integrarnos o crear instituciones de justicia, democracia o derecho internacional que regulen por medios pacíficos la

transformación pacífica de los conflictos que tenemos cuando chocamos.

En las culturas de las guerras, se consideran argumentos irrefutables, que «siempre ha habido guerras y las habrá» o que «siempre ha habido ricos y pobres y los habrá».

1.2. La teoría de la guerra y la paz

Las teorías sobre la guerra han estado siempre relacionadas por una interdependencia entre las configuraciones políticas que han tomado la humanidad y los medios bélicos utilizados para defender dichas configuraciones. Así, cuando dichos instrumentos no pasaban del arma blanca y del caballo como medio de transporte terrestre, las formas políticas habituales de organización eran el Estado-ciudad, la confederación de ciudades, los pequeños reinos medievales, etcétera.

Es verdad que hubo imperios en el mundo antiguo, pero su defensa no se depositaba en ninguna capacidad bélica adecuada a su enorme extensión, sino más bien en una cierta habilidad de organización política de determinados personajes y a la misma inabarcabilidad territorial de los mismos, lo que -unido a las dificultades de transporte de la época- permitía su supervivencia durante cierto tiempo.

El descubrimiento de la pólvora y su llegada a Europa permitió el desarrollo de una nueva forma de organización política: el Estado-nación, cuya configuración administrativa y extensión territorial se adecuaba al desarrollo armamentístico de la artillería, ciencia basada precisamente en la aplicación de la pólvora a las técnicas militares de defensa. Es entonces, cimentándose en el nuevo desarrollo de la tecnología bélica, cuando se pasa del concepto clásico de la legítima

defensa a una nueva elaboración ideológica: la teoría de la *guerra justa*. Según ésta, la guerra -por naturaleza, injusta- admite ciertos casos en que puede declararse: defensa de las fronteras nacionales, agresiones sangrientas a una población, rapiñas y apropiaciones de sus recursos, etcétera.

No es extraño que, al ser España el primer Estado europeo moderno, fuesen los teólogos-juristas españoles quienes elaboraran las primeras doctrinas sobre la *guerra justa*, íntimamente relacionadas con las teorías que desarrollaron los principios filosóficos de unas relaciones internacionales basadas en el derecho y en la justicia; es así como surge la fundamentación filosófico-jurídica del derecho internacional, magistralmente realizada por Francisco de Vitoria y, en cierto modo, por Bartolomé de las Casas.

A pesar de los nuevos avances de la tecnología militar del siglo XX, la situación ideológica sobre los fundamentos filosóficos de la guerra apenas han variado: "si quieres la paz, prepara la guerra" interpretación burda y directa para justificar la carrera armamentística, en la expresión de Von Clausewitz: "La guerra es la continuación de la política por otros medios".

Sin embargo, la aplicación de la tecnología nuclear a la guerra ha variado de tal modo la situación, que se impone de modo imperioso afrontarla de modo radicalmente nuevo, al introducir una posibilidad técnica inédita hasta la presente fecha: *el holocausto nuclear como fin de la especie humana*. Esta nueva situación, imprevisible hace 40 años, hoy es una realidad; al mismo tiempo nos encontramos con una situación política de enfrentamiento entre dos grandes bloques -el llamado socialista y el llamado capitalista- que afrontan la doble

realidad con un instrumento ideológico extremadamente pobre frente al inmenso reto: la conocida *teoría de la disuasión.*

Se supone que con un desarrollo suficientemente fuerte de la tecnología nuclear se disuade al enemigo de emplearla, sin tomar en cuenta de que ese desarrollo tecnológico inspira tal miedo al adversario que éste se ve obligado a fortalecerse e incrementar su desarrollo nuclear para disuadir a su vez al contrario de la tentación de emplearlo en algún momento.

El llamado *equilibrio del miedo* es una falacia, porque, en realidad, no existe tal equilibrio, sino una espiral creciente en la producción de armamento nuclear. La *teoría de la disuasión* resulta absolutamente disfuncional con la situación del mundo; en primer lugar, porque es falsa, ya que no produce tal equilibrio, sino un incremento cada vez mayor del armamento y, en consecuencia, un incremento también cada vez mayor de las posibilidades de guerra nuclear; en segundo lugar, porque la tal llamada *disuasión* no favorece la paz, como se dice, sino, en el mejor de los casos, la insatisfactoria supervivencia bajo un régimen de terror.

En cualquier caso, si nos situamos en una perspectiva que llamaremos optimista, la realidad que se nos impone es ésta: si la carrera nuclear no se detiene pronto, nos encontraremos dentro de 30, 50, 60, 100 años con impresionantes almacenes de artefactos nucleares cuya peligrosidad es por sí misma inconmensurable; incluso si se decidiese su eliminación pacífica y voluntaria, nos encontraríamos con un problema técnico de difícil solución y con repercusiones imprevisibles en el orden de la ecología y de la salud humana.

Es necesario, por tanto, afrontar esta situación mundial con una actitud radicalmente nueva, porque hay que ir a las raíces mismas de la situación que nos ha llevado al estado actual. Esas raíces no son otras que las de una filosofía basada en el afán de dominio y de poder característico del *homo faber* occidental, cuyo espíritu prometeico no concibe más relación con la naturaleza y el medio humano en que vive que la de la explotación, el sojuzgamiento, la subordinación y el control.

En la irracional y alienante carrera armamentística no percibimos sino las últimas consecuencias desesperadas de un afán de dominar el mundo, aunque ese afán se justifique bajo palabras altisonantes como libertad, democracia, paz, justicia, igualdad, reparto, lucha contra la explotación, etc.

En vista de este análisis, que conduce a la perspectiva más macabra que puede imaginar el hombre -su desaparición del planeta-, sólo cabe dar una respuesta: una ruptura radical, filosófica y ética con la citada situación, basada en una actitud de solidaridad con la naturaleza, alejada de todo afán de dominio de la misma, donde se vuelvan a expresar el amor a la vida, al hombre y al mundo en general. Sin duda, la inspiración de una filosofía oriental que nunca renegó de tales principios será esencial para recuperar el sentido de una cultura basada en la fe en el hombre, en la vida y en su capacidad de realización humana, liberando así un conjunto de energías paralizadas por el miedo, la angustia y la desconfianza.

En lo que respecta al tema de la guerra nuclear, esto supone afrontar el riesgo de un desarme nuclear y unilateral total. De momento, se supondría que toda una parte de la Humanidad se ha puesto del lado de la vida, del amor, de la fe en el hombre, y ésa sería la primera gran

victoria de los que tomasen la gran decisión: se habrían convertido, por ese solo hecho, en los grandes representantes del lado bueno de la Humanidad, la de aquellos que creen en el hombre y en la capacidad de progreso y de autorrealización de la especie.

En definitiva, frente a la teoría de la disuasión, existe la doctrina de la persuasión. Si la primera pone su énfasis en la capacidad de intimidar e infundir miedo, la segunda se basa en la confianza en la capacidad racional del hombre para discutir y dialogar, eligiendo lo que es más conveniente a sus intereses y a los de sus semejantes.

1.3. La Teoría de la guerra justa y la vida real

Acicateada por estos tiempos de invasiones y de evasiones, la discusión sobre una "guerra justa" ha tenido un renacimiento entre expertos e inclusive los encargados de formular una política.

Dejando de lado los conceptos, las acciones en el mundo real con frecuencia refuerzan la máxima de Tucidides de que "los poderosos hacen lo que pueden, en tanto los débiles sufren lo que deben". Y eso no sólo es injusto de manera indiscutible, sino que en la presente etapa de la civilización humana es una amenaza literal a la supervivencia de las especies.

En sus elogiadas reflexiones sobre la guerra justa, el periodista Michael Walzer describe la invasión de Afganistán como "un triunfo de la teoría de la guerra justa". Lamentablemente, como en otros, los argumentos se basan crucialmente en premisas tales como "me parece totalmente justificado" o "yo creo" o "sin duda alguna".

Usando la lógica de "la guerra justa", o de la lucha antiterrorista, Estados Unidos se exime de los principios fundamentales del orden

mundial en los cuales ha desempeñado un importante papel a la hora de formularlos y de hacerlos cumplir.

En Afganistán y en Irak las tropas han proferido malos tratos y torturas a detenidos enemigos en distinto grado y consideración: amenazas de muerte, apaleamientos, rotura de huesos, asesinatos, trabajo físico excesivo, toma de rehenes, privación de sueño y tratos degradantes y abusos sexuales como los fotografiados y filmados en la cárcel de Abu Ghraib en Irak.

Luego de la Segunda Guerra Mundial, fue instituido un nuevo régimen de leyes internacionales, las cuales están codificadas en la Carta de las Naciones Unidas, la Convención de Ginebra y los principios de Nüremberg, adoptados por la Asamblea General.

La Carta de la ONU prohíbe la amenaza o el uso de la fuerza a menos que sea autorizada por el Consejo de Seguridad o, bajo el artículo 51, sea en defensa o contra un ataque armado hasta que actúe el Consejo de Seguridad.

En un mundo repleto de amenazas potenciales, el riesgo al orden global y la norma de no intervención en la cual continúa basándose, es simplemente demasiado grande para que la legalidad de una acción preventiva unilateral como algo diferente a una acción respaldada colectivamente pueda ser aceptada. Si se permite a alguien que actúe de esa manera, se permite que todos hagan lo mismo.

De acuerdo con el Tribunal de Nüremberg, la agresión es "el supremo crimen internacional y sólo difiere de otros crímenes de guerra en que contiene en sí mismo el mal acumulado del resto". Por ejemplo, todo

el mal en la torturada tierra de Irak surgido de la invasión de Estados Unidos y el Reino Unido.

El concepto de agresión fue definido con meridiana claridad por el juez Robert Jackson, de la Corte Suprema estadunidense, quien fue el fiscal principal de Estados Unidos en Nüremberg, Alemania. El concepto fue repetido en una resolución de la Asamblea General. Un "agresor", señaló Jackson ante el tribunal, es un Estado que comete actos tales como "una invasión de sus fuerzas armadas, con o sin una declaración de guerra, del territorio de otro Estado".

También son relevantes las elocuentes palabras del juez Jackson en Núremberg: "Si ciertos actos de violación de tratados son crímenes, se trata de crímenes, sin importar que los cometan Estados Unidos o Alemania. No estamos preparados para estipular una norma de conducta criminal contra otros que no estemos dispuestos a invocar contra nosotros".

Para el liderazgo político, la amenaza de adhesión a esos principios, y al imperio de la ley en general, es realmente grave, urge seriamente a que se haga justicia con los hombres y mujeres de uniforme y se les dé criterios claros de conducta que reflejen los ideales por los que arriesgan sus vidas.

1.4. La teoría de la guerra moderna

La teoría de la **guerra moderna** se refiere a la **guerra de maniobras** y pretendió dar una solución práctica a la tendencia de las fuerzas terrestres estadounidenses de utilizar exagerada o inoportunamente el apoyo de fuego pesado como su principal “argumento” sobre el enemigo en los combates o en su preparación.

También buscaba consolidar en las operaciones terrestres el concepto de estrategia operativa, como nuevo y preciso nivel de actuación entre la estrategia militar y la táctica. A esta guerra moderna también se le conoce como **guerra de tercera generación.**

Las ideas fundamentales de la guerra de maniobras.

Es necesario definir la "Táctica" porque de ella se deriva casi todo su desarrollo y aplicación posterior: *Táctica es la combinación armónica y suficiente de conocimientos específicos, experiencias y técnicas militares, para producir una acción sorpresiva, suficiente, eficaz y poderosa sobre el enemigo, en condiciones dadas de tiempo, clima y oportunidad.*

Esa "*combinación*" debe ser matizada y modulada por el esfuerzo principal y las órdenes tipo misión del jefe y por las resistencias y los vacíos de la capacidad de combate que presente el enemigo, en su despliegue inmediato y según su intención.

En el combate se busca introducir al enemigo en ciclos sucesivos de "observación, situación, decisión y acción", que sean más rápidos que sus capacidades de actuación. De tal manera que las acciones enemigas sucesivas "elementales" resulten progresivas y cada vez más inadecuadas e ineficaces, para contrarrestar, superar u oponerse a las nuestras. Lo cual debe tender a destruir su cohesión de unidad y fomentar y extender el pánico entre sus hombres, ante la inutilidad manifiesta de las acciones que emprende.

Como se aprecia no se han tocado los aspectos básicos de la doctrina ni las técnicas de cómo disparar las armas, el adiestramiento físico, la lectura de cartas, la orientación o las técnicas avanzadas de cómo

realizar las distintas operaciones de marcha, avances a campo través, defensa, el empleo de un sistema de armas combinadas, algún tipo especial de ataque, etc. Aquéllos vendrán dados en su momento en las escuelas, institutos y en las unidades. Lo importante en esta teoría es la aceptación y la asunción de su enfoque nuevo, que resalta las cualidades de iniciativa y creatividad en los jefes para ahorrar medios físicos, militares y económicos, las vidas y el inapreciable tiempo, siendo mucho más eficaces.

El funcionamiento del nuevo estilo de actuación.

Si se quiere ser más rápido y eficaz que el enemigo, se tiene que tener una organización militar suficientemente descentralizada, actuando sobre él, de acuerdo al ciclo de actuación de las cuatro fases, es decir, la *observación* que realicen las subunidades en contacto deben ser comunicadas hacia arriba, siguiendo la cadena de mando; se debe definir la *situación* que se vive y la *decisión* se la debe tomar al más alto nivel, y, luego, la orden para la acción debe ser transmitida hacia abajo, a través de la cadena de mando, y, entonces, se procede a ejecutar la *acción*; lo que no se quiere es que las unidades en contacto con el enemigo deambulen sin sentido ni eficacia por el campo de combate, para guiarlas provechosamente en su actividad están las órdenes tipo misión; en ellas el comandante comparte y encarga una parte de su intención a las subunidades y le da libertad de acción (el cómo hacer), a cambio de que realice esa parte de su intención (el qué hacer), también el comandante establece un esfuerzo principal sobre el enemigo.

La subunidad que actúa en el esfuerzo principal recibe la cooperación de las acciones de sus unidades y la mayoría de los apoyos de la

unidad que el comandante recibe de su superior en subordinación táctica o de guerra.

Situaciones y medios tácticos preferidos.

En la guerra de maniobras es necesaria una gran actividad de los exploradores de combate, para generar la suficiente inteligencia para la aplicación de las técnicas apropiadas y la definición de la situación general y la del enemigo (la primera fase del ciclo de actuación). El despliegue de las subunidades en el contacto podría ser en forma de cuña invertida, cuando la situación enemiga no está nada aclarada. O, en el otro extremo, tomar la forma aproximada de la cuña, cuando se están buscando micro vacíos de su capacidad en su despliegue, para irrumpir limpiamente en la zona de defensa enemiga.

Los medios del comandante enfatizados en la guerra de maniobras son la reserva, el contrataque y los fuegos de apoyo. La reserva debe existir siempre y, al menos al principio, debe estar en manos de un subordinado capaz, experimentado y enérgico. La reserva es la apuesta del comandante para su triunfo táctico trascendente. No debe ser empleada para reforzar un esfuerzo más (para eso están los mayores fuegos pesados o los tanques e ingenieros o el estrechamiento de los sectores de avance o el escalonamiento en profundidad) o para realizar un ataque secundario o una diversión o para compensar, sin más, un error. Por tanto, no es una subunidad secundaria, ni una que está reorganizándose y tiene escasos medios, alistamiento de combate y moral.

El contrataque brinda iniciativa y movilidad a una defensa más o menos temporal y necesaria o buscada. Debe ser oportuno, potente y lanzado cuando el enemigo ha pasado el clímax de su ataque, ha

sufrido pérdidas y la situación puede ser recuperada por nosotros. Los fuegos de apoyo deben buscar impulsar la maniobra de la unidad. Sus tareas deben ser cegar, perturbar o neutralizar, más que destruir. Porque esto último resulta más costoso, muy difícil de conseguir y tarda más tiempo (un factor precioso). Por último, los sistemas de armas combinadas y los ingenieros son empleados continuamente en la guerra de maniobras y, los últimos, en el esfuerzo principal, aunque abastezcan de sus medios a todas las unidades.

1.5. La Teoría de la guerra de cuarta generación

La teoría de la Guerra de Cuarta Generación fue formulada por William S. Lind y cuatro oficiales del Ejército y del Cuerpo de Infantería de Marina de los Estados Unidos en 1989. Se le conoce como Fourth Generation Warfare (4GW).

Esta guerra, en realidad, ya se está librando en varios frentes del mundo y de nuestra América. Lo que pasa es que en muchos casos no vemos a los helicópteros enemigos, ni escuchamos los balazos, ni vemos cadáveres en los campos de batalla. Es la llamada Guerra de Cuarta Generación y nosotros estamos un poco atrasados.

En la Guerra de Cuarta Generación las operaciones psicológicas, la propaganda encubierta y los mensajes indirectos a través de los medios de comunicación complementan a los misiles, tanques y aviones. El objetivo es "ganar las mentes y corazones de los pueblos invadidos". Es una guerra cultural, no militar.

La guerra cultural no es táctica, sino estratégica. Se dirige "contra toda la nación enemiga y su objetivo es destruir todo elemento cultural que pueda generar en el futuro reacciones contra el imperio

anglosajón”. Abu Obeid Al Qurashi, lugarteniente de Osama Bin Laden y uno de los principales estrategas de Al Qaeda, define el tema de la siguiente manera: “En este tipo de guerras las informaciones aparecidas en los medios de comunicación serían un arma más poderosa que las divisiones militares”.

Draft Manual on 4G War no es un manual convencional, recurre a una pequeña narración para ejemplificar las tácticas que propone. En una guerra ficticia en un país ficticio (que sin duda es Irak), un oficial se niega a que se ejecute la Operación Goliath en su área de operaciones. La Operación Goliath implica entrar a sangre y fuego en una zona donde hay rebeldes. Lind recoge la experiencia de unidades de la Reserva y la Guardia Nacional en Irak, entre cuyos soldados hay policías y sheriffs acostumbrados a patrullar las calles de barrios peligrosos. Afirma que hay mucha menos violencia si el ejército procede como la policía de cualquier suburbio de Estados Unidos.

Del orden a la asimetría

La Guerra de Primera Generación corresponde a los enfrentamientos con tácticas de líneas y columnas. Duró aproximadamente desde 1648 hasta 1860, los combates eran formales y el campo de batalla era ordenado. El ordenamiento en el campo de batalla creó una cultura del orden militar; muchos de los aspectos que distinguen a los militares de los civiles –uniformes, saludos, la graduación minuciosa, los rangos– fueron producto de la primera generación y estaban diseñados para reforzar la cultura del orden.

El problema, es que, a mediados del siglo XIX el campo de batalla ordenado comenzó a desmoronarse. Ejércitos en masa, soldados que realmente querían luchar y el uso de ametralladoras, hicieron obsoletas las viejas tácticas de línea y columnas por demás suicidas.

Desde entonces, surgió una creciente contradicción entre la cultura militar y el desorden cada vez más presente en el campo de batalla.

La guerra de Segunda Generación de la guerra fue una respuesta a la contradicción entre la cultura del orden y el ambiente militar. Desarrollada por el ejército francés durante y después de la Primera Guerra Mundial (1914-1918), buscó una solución en la forma de potencia de fuego en masa, la mayoría de la cual era fuego de artillería indirecto. La doctrina fue descrita por los franceses como "la artillería conquista, la infantería ocupa".

La potencia de fuego controlada centralmente fue cuidadosamente sincronizada con órdenes específicas para la infantería, los tanques y la artillería. Esta segunda generación preservó la cultura del orden: el comandante era, en efecto, el conductor de una orquesta. La disciplina se imponía desde arriba hacia abajo. No se deseaba la iniciativa porque ponía en peligro la sincronización.

La Guerra de Tercera Generación es resultado de la Primera Guerra Mundial, fue desarrollada por el Ejército Alemán en el conflicto mundial de 1939-1945 y es comúnmente conocida como "guerra relámpago" (*Blitzkrieg*). No se basa en la potencia de fuego, sino en la velocidad y sorpresa. Tácticamente, se caracteriza por el ataque. Busca penetrar la retaguardia del enemigo y desde allí causar su derrumbamiento. En vez de "aproximarse y destruir", trata de sobrepasar y derrumbar. En la defensa, la intención es atraer el enemigo hacia las posiciones convenientes y luego cortar sus líneas.

Los ejércitos de la tercera generación se concentran en lo externo (cuál es la situación, qué hace el enemigo, cómo se resuelve la situación), y no en lo interno (proceso o método). Durante los juegos

de guerra del siglo XX, a los oficiales subalternos alemanes rutinariamente se les planteaban problemas que sólo podrían ser resueltos desobedeciendo las órdenes. Las órdenes especificaban el resultado deseado, pero nunca el método. La iniciativa fue más importante que la obediencia. Se toleraban errores, ya que provenían de demasiada iniciativa, en vez de falta de ella. Todo el concepto dependía de la autodisciplina, no de la disciplina forzada.

La Guerra de Cuarta Generación también llamada "guerra asimétrica", es todo lo contrario de las anteriores. Corresponde a la evolución de la tecnología, la cibernética y la información.

La evolución tecnológica, la aparición de enemigos que no responden a definiciones clásicas y la ofensiva terrorista, han llevado a los estrategas militares a concebir un nuevo modelo de guerra que se apoya, por primera vez, más en los elementos culturales que en el potencial bélico. Pretende la victoria a través de la movilización cultural contra un enemigo imperceptible y volátil como es el terrorismo. Oscila del aspecto armamentista al psicológico, pretende una movilización masiva de la población en un antagonismo integral contra el supuesto enemigo, que abarca los aspectos políticos, económicos, sociales y culturales de una nación con el objetivo de alcanzar el sistema mental y organizativo del adversario.

¿Por qué es una guerra asimétrica? Porque "opone a dos agentes que apenas tienen nada en común: por un lado potencias tecnológicamente muy desarrolladas, con capacidad para emplear armas inteligentes muy sofisticadas, y por otro agentes transnacionales o infranacionales, ya sean religiosos o étnicos, que se enfrentan a distintos símbolos como el mercado o el imperio, y con un nivel armamentista muy elemental".

En resumen, la teoría de la cuarta generación segmenta la historia occidental en cuatro períodos, a cada uno de los cuales le correspondió una generación de la guerra. Estos períodos son:

1. Clásico (entre tropas)
2. Medieval (tropas, dispositivos tecnológicos e inteligencia)
3. Moderno (tropas, tecnología, inteligencia, contrainteligencia y capacidad de fuego remoto)
4. Posmoderno (desinformación, comunicación borrosa, cibernética, formas de control de la población).

En la teoría de la cuarta generación la guerra regular es un patrimonio de Occidente y el combate irregular (terrorista) es visto como característico de lo no-occidental (oriental).

1.6. La teoría del caos

Durante los últimos 30 años, el estudio del caos ha preocupado a los investigadores, lo que ha llevado a muchos a ver un gran futuro en el estudio y la aplicación de la teoría del caos. En la ciencia y en la ingeniería la teoría del caos ha mejorado significativamente nuestro entendimiento de muchos fenómenos, desde la turbulencia atmosférica a la dinámica estructura.

La teoría del caos ha sido utilizada para mejorar significativamente el control sobre ciertos sistemas dinámicos. En las ciencias sociales ha existido un considerable interés respecto a sí ciertos fenómenos sociales, que antiguamente se pensaba ocurrían sólo por azar, no obedecerían a alguna forma oculta de orden caótico. Se han aplicado varios exámenes matemáticos de comportamiento caótico a datos históricos sobre el mercado accionario y el precio del algodón.

Estas pruebas indican que estos fenómenos económicos son caóticos y por ello tienen una base determinista (son gobernados por reglas) y no ocurren por azar. Naturalmente, esto ha despertado cierto interés en el mundo de los negocios, y actualmente muchas empresas están utilizando la teoría del caos para orientar su asesoría financiera.

Hay evidencia de que la guerra también puede ser caótica:

1. En primer lugar se ha advertido que los procesos de decisión estratégicos, son caóticos.
2. En segundo lugar, la no linealidad, que es un requerimiento del comportamiento caótico, parece ser un resultado natural de las fricciones Clausewitzianas.
3. En tercer lugar, algunos juegos de guerra computarizados y competencias armamentistas simuladas exhiben un comportamiento caótico.
4. En cuarto lugar, se han aplicado pruebas de caos a datos históricos relacionados con la guerra. Estas pruebas han demostrado que la guerra es caótica en los niveles estratégicos, operativos y tácticos.

A continuación se analizara algunas importantes implicancias de la teoría del caos en el contexto de la guerra y algunos aspectos relevantes de la teoría del caos.

Nolinealidad

En un sistema lineal, el resultado del sistema está relacionado linealmente con los antecedentes que le son incorporados. En otras palabras, si dichos antecedentes se multiplican por dos, el resultado igualmente se multiplicará por dos; si los antecedentes se triplican, el resultado igualmente se triplicará, y así sucesivamente. En los

sistemas no lineales, sin embargo, el resultado puede ser el cuadrado o el cubo de los antecedentes incorporados. Estos sistemas usualmente son muy sensibles a la incorporación de elementos. *Todos los sistemas caóticos son no lineales.*

La nolinealidad significa que esfuerzos pequeños pueden tener efectos desproporcionados. Dado que la guerra es caótica, la teoría del caos sugiere que es posible encontrar COGs (*Center Of Gravity*) donde existe un proceso nolineal en los sistemas enemigos. De hecho, la no linealidad está implícita en el concepto de COG. Debido a que uno no puede predecir el comportamiento futuro de un sistema caótico basado en las predicciones iniciales, la teoría del caos sugiere que los planificadores de la campaña deben concentrarse en procesar el sistema enemigo en vez de buscar información sobre su condición actual. También sugiere que la identificación de procesos nolineales es un ingrediente esencial para comprender la guerra y para ser capaz de manipular su resultado con el menor esfuerzo.

Los párrafos siguientes discutirán algunas de las muchas fuentes de nolinealidad en la guerra.

1. Una primera fuente de nolinealidad en la guerra son l***os ciclos de retroalimentación*** que constituyen un proceso que puede introducir efectos nolineales en muchos sistemas. Un ciclo de retroalimentación importante para la campaña aérea es la retroalimentación que la tasa de pérdidas le da a un comandante aéreo. Altas tasas de desgaste pueden forzar a un comandante a cambiar sus tácticas.

   Por ejemplo, la tasa de desgaste de un 16% sufrida por Estados Unidos en los bombardeos diurnos sobre Schweinfurt fue

suficiente para detener las operaciones de bombardeo por cuatro meses hasta la introducción de un avión de combate de largo alcance. Se utilizan éste y otros ejemplos históricos para argumentar que la razón máxima aceptable es de alrededor de un 10%. Sin embargo, se destaca que el efecto de una misión con una pérdida del 10% y nueve misiones con bajas sin importancia era mucho mayor que una pérdida permanente del 1% en 10 misiones.

En un sistema lineal no habría diferencia entre las dos - el efecto aditivo sería el mismo. El hecho de que existe una diferencia muestra que *la retroalimentación es nolineal*.

2. Una segunda fuente de nolinealidad en la guerra, es ***la psicología asociada con la interpretación de las acciones enemigas***. Esta nolinealidad llevó a Clausewitz a declarar, "Así, entonces, en estrategia todo es muy simple pero por eso mismo tampoco muy fácil." Posteriormente él amplió esto diciendo que mientras que algunas maniobras como los movimientos de flanco, son simples en concepto, son muy difíciles de ejecutar porque existe siempre el peligro de lo que el enemigo pueda estar haciendo. En este contexto, las acciones menores del enemigo usualmente tienen en la mente del comandante una significación mayor que la que merecen.

   Este efecto nolineal ocurrió en la I Guerra Mundial antes de la primera Batalla del Marne. Los alemanes, conscientes de una posible debilidad en su despliegue, tenían instrucciones de retirarse si el Ejército Británico avanzaba sobre el Marne. De hecho, una división inglesa envió una patrulla de reconocimiento. Los alemanes, interpretando esto como un

avance general, se retiraron en circunstancias que la vía estaba expedita para lograr la victoria.

3. Una tercera fuente de nolinealidad en la guerra es que ***hay una cantidad de procesos dentro de la guerra que parecen fundamentalmente nolineales***. El efecto de la masa es un ejemplo significativo. Esto se demuestra en las operaciones aéreas en las cuales las pérdidas varían desproporcionadamente con la razón de fuerzas empleadas. En 1944, por ejemplo, 287 aviones americanos atacaron un objetivo defendido por 207 aviones de combate alemanes. Los americanos perdieron 34 aviones. Un mes después, cuando 1641 aviones americanos fueron enfrentados por 250 cazas alemanes, los americanos perdieron 21 aviones - un porcentaje más bajo y un número absoluto, menor.

4. Una cuarta fuente de nolinealidad en la guerra la constituye la ***fricción Clausewitziana***. La idea básica es que en la guerra ocurrirán acontecimientos, posiblemente como resultado del acoso, que tendrán un efecto totalmente desproporcionado a su importancia aparente. Esta es una forma de nolinealidad extraordinariamente difícil de anticipar, pero de la cual se puede tomar ventaja una vez que ocurra. La doctrina alemana de *Aufragstaktik,* que permitía la iniciativa a los oficiales jóvenes, fue diseñada precisamente para esto.

5. Finalmente, ***el proceso mismo de decisión*** puede constituirse en una fuente de nolinealidad. A veces la decisión es clara, sin embargo, frecuentemente puede depender de circunstancias temporales relativamente menores. Una fuente sugiere que el motor de vapor perdió frente al motor de gasolina de

combustión interna fundamentalmente como resultado de una epidemia de una enfermedad parasitaria. Como resultado de esto, fueron retirados muchos abrevaderos que las máquinas de vapor utilizaban para llenar sus depósitos de agua. Una vez que la decisión está tomada, frecuentemente es irreversible debido a la tendencia a la estandarización.

Cualquier decisión importante, incluida aquellas tomadas durante la guerra, pueden estar fundadas en factores relativamente menores de una manera nolineal.

En conclusión la guerra es nolineal, esto implica una extrema sensibilidad a las condiciones iniciales, lo que significa que el planificador de la campaña debe concentrarse en los sistemas enemigos, el atacar los procesos nolineales promete el mayor efecto por el menor esfuerzo.

Predecibilidad de los sistemas caóticos

Los sistemas dinámicos se diferencian unos de otros en la forma cómo cambian con el tiempo. En los sistemas de azar, el comportamiento futuro es independiente del estado inicial del sistema y puede ser caracterizado solamente en términos de probabilidades. Por ejemplo, a menos que los dados estén cargados, la próxima jugada es totalmente independiente de la anterior.

Por otro lado, los sistemas periódicos regresan regularmente a la misma condición, de la manera como lo hacen los péndulos del reloj. Estos sistemas son totalmente predecibles, debido a que una vez que se conoce un período, todos los demás son idénticos. Los sistemas caóticos no se comportan al azar ni son periódicos. Ellos no se

comportan al azar debido a que el futuro del sistema caótico depende de la condición inicial. Ellos no son periódicos porque su comportamiento nunca se repite.

Los sistemas caóticos nunca se repiten exactamente porque su comportamiento futuro es extremadamente sensible a la condición inicial, de este modo, diferencias infinitesimales en la condición inicial eventualmente causan significativos cambios en el comportamiento del sistema. El tiempo atmosférico es un ejemplo frecuentemente utilizado de esta sensibilidad. El tiempo es tan sensitivo a la condición inicial que existe la creencia que el movimiento de las alas de una mariposa en América puede eventualmente causar un tifón en China. Es inconcebible que las condiciones en la tierra puedan duplicarse al punto que también se dupliquen todos los vuelos de mariposa. Por ello el tiempo atmosférico en la tierra nunca será periódico.

Dado que la guerra es caótica, no podemos hacer predicciones perfectas incluso si podemos reducir la guerra a un conjunto mecanístico de ecuaciones, la teoría del caos proporciona herramientas que pueden predecir patrones de comportamiento de los sistemas y pueden definir límites dentro de los cuales el comportamiento es impredecible, entonces esta teoría nos advierte *que pueden existir sistemas enemigos en etapas diferentes*. Las implicancias son que debemos estar alertas de estas etapas posibles y, de ser necesario, ser capaces de cambiar nuestros propios sistemas para contrarrestar la estrategia enemiga.

La simulación computarizada puede aumentar el entendimiento

Los modelos computarizados numéricos o la simulación han aumentado grandemente nuestro entendimiento de los sistemas

caóticos. Esto ocurre porque las ecuaciones que rigen los sistemas caóticos son no lineales y por lo tanto, en términos generales, no son solubles analíticamente. Sin embargo, la teoría del caos por sí misma no sirve para elucubrar una teoría de la guerra. Al igual que con cualquier otra teoría que describa un fenómeno, una teoría de la guerra debería estar basada en la observación, en hipótesis y en comprobaciones. Específicamente, el desarrollar un modelo de la guerra requeriría el desarrollo de la estructura del modelo, la determinación del número y tipo de variables y la determinación de la forma de la ecuación. Además, deberían identificarse los parámetros del sistema y los factores de control así como también las fuentes de ruido. Este es un trabajo muy complejo para un caso determinado y se complica por la posibilidad que diferentes modelos se apliquen a diferentes antagonistas.

La teoría del caos puede ayudarnos sugiriendo formas de desarrollar nuestro modelo y formas de utilizarlo una vez que esté desarrollado. Por ejemplo, el observar un sistema caótico puede servir para determinar la dimensión del sistema. El número de variables necesarias para describir el sistema debe ser al menos igual que la dimensión del sistema. Por ello, la teoría del caos puede ser utilizada para definir el número mínimo de variables necesarias en nuestro modelo computacional.

La teoría del caos también sugiere que los juegos de guerra computacionales deben contener algunas relaciones no lineales entre las variables del sistema de modo que el modelo computacional sea caótico y refleje así la naturaleza caótica de la guerra.

Esto puede ser ventajoso dado que la naturaleza fractal del sistema caótico puede permitir que juegos de guerra relativamente pequeños y

simples simulen la guerra con bastante precisión. Juegos de guerra realistas que puedan ser utilizados en un computador de escritorio tendrían ventajas operacionales y educacionales significativas. Finalmente, en un sistema caótico es posible calcular la razón de pérdida de información. Esta cantidad está relacionada con lo lejos que se puede llegar con las predicciones futuras.

La forma en que las computadoras han sido utilizadas para entender el comportamiento caótico en sistemas físicos, también sugiere vías para usar los computadores en modelos de guerra. Por ejemplo, aunque la teoría del caos explica algunos aspectos del tiempo atmosférico, la predicción del tiempo no es perfecta. Los sistemas caóticos son altamente dependientes de las condiciones iniciales pero no lo son de una manera siempre igual.

Si un sistema caótico está en la parte de su fase espacial donde las condiciones iniciales son críticas, entonces la incertidumbre en determinar las condiciones iniciales hacen posible un gran número de resultados. Si un sistema caótico está en su zona de espacio de fase en la cual las condiciones iniciales no son críticas, entonces es posible que ocurra un sólo resultado (predicción). En la práctica, los meteorólogos usan este comportamiento incorporando pequeños cambios en las condiciones iniciales de su modelo. Si los pequeños cambios producen variaciones pequeñas en la predicción, ellos ven que el sistema está en una región del espacio de fase donde las condiciones iniciales no son críticas y su predicción muy posiblemente sea verdadera. Si los cambios pequeños en las condiciones iniciales producen grandes desviaciones en el comportamiento futuro, los meteorólogos saben que muy posiblemente su predicción esté equivocada.

Se puede tomar el mismo camino para entender cuando las predicciones en la guerra serán precisas. Esto por sí solo sería una contribución importante a la simulación computacional para entender la guerra. Sin embargo, hay dos razones más de porqué esta aproximación sería más aplicable a la guerra que al tiempo:

1. En primer lugar, al contrario de lo que ocurre con los meteorólogos, nosotros tenemos cierta capacidad de cambiar las condiciones iniciales. Específicamente, si nos encontramos en una zona de gran incertidumbre, podemos determinar qué condiciones deberían ser cambiadas para poner al sistema en una posición en la cual el resultado sea predecible y deseable. La cantidad y tipo de fuerzas son ejemplos de la condición inicial que podemos cambiar.

2. En segundo lugar, podemos utilizar nuestro modelo para determinar qué condiciones iniciales y qué variables tienen el efecto más profundo en nuestra predicción. Esto puede ayudar a identificar los centros de gravedad COG (Center Of Gravity) y la información que necesitamos conocer con precisión. Ello nos diría dónde concentrar nuestro ataque y cuál es la información de inteligencia más fundamental.

2. LA SIMULACION

2.1 Historia de la Simulación

La Simulación no es un tema nuevo como muchos creen, por más de 40 años, este tema ha evolucionado considerablemente y últimamente debido al gran avance tecnológico de los sistemas y de las computadoras. Se puede afirmar que desde la prehistoria el hombre ha utilizado la simulación para resolver sus problemas, practicar o analizar una nueva estrategia para dar caza a un dinosaurio, se constituye ya en un asunto de la Simulación.

Se puede afirmar que la moderna Simulación nace en épocas de la II Guerra Mundial y la Guerra Fría, por una necesidad de resolver la problemática de la logística, las estrategias del combate y el uso efectivo de las armas; la Simulación en todos los sentidos demarcó una nueva técnica de análisis y estudio de las diferentes estratagemas de la guerra. Su proceso evolutivo fue abanderado por el **DoD (Departamento de Defensa de los Estados Unidos)** quienes se dieron a la tarea de desarrollar los diferentes métodos y niveles de esta ciencia.

EL DoD y la Simulación

En 1988 la Agencia de Proyectos de Investigación Avanzada para la Defensa (**Defense Advanced Research Projects Agency** – DARPA hoy **ARPA**) inició un programa llamado Simulator Networking – SIMNET, para crear los múltiples simuladores de tanques que se podrían enlazar a través de una red de tal forma que cada uno podría detectar, designar, y destruir a otros tanques. Este programa dio lugar al establecimiento de los principios importantes para la interacción de

la simulación y la creación de un protocolo de red para intercambiar datos esenciales. SIMNET era el precursor de los protocolos interactivos distribuidos de la simulación (Distributed Interactive Simulation - DIS).

Debido a su gran aceptación, este protocolo fue aprobado por el "Institute of Electrical and Electronics Engineers" (IEEE) como un estándar IEEE en 1992. DIS procuró generalizar la tecnología de SIMNET de modo que pudiera ser aplicado a una variedad más amplia de simuladores de vehículos de combate tales como carros, helicópteros, naves y soldados.

Al mismo tiempo, los miembros del grupo de entrenamiento constructivo desarrollaban los métodos para enlazar simulaciones con los eventos del combate a un nivel más alto. El sistema distribuido de Juegos de Guerra - DWS, enfocado a ciertos ejercicios, demostró la viabilidad de seguir unidades militares en otras simulaciones y designarlas con eficacia y exactitud. Este experimento conduce al desarrollo del Protocolo de Simulación del Nivel Agregado – ALSP, para demostrar el entrenamiento interoperable a nivel Staff.

ALSP ligó siete simulaciones existentes de cada servicio militar proporcionando los mensajes de red y los servicios de software para asegurar consistencia y causalidad entre las simulaciones. Otros modelos de este tipo corresponden al Sistema de Simulación Táctica - TACSIM y Simulación de Guerra Electrónica – EWS.

La **Oficina de Modelado y Simulación para la Defensa (DMSO)** desarrollo la Arquitectura de Alto Nivel (HLA) para sustituir DIS y ALSP. El estándar HLA basado es un estándar OTAN al cual la comunidad internacional ha adoptado como la arquitectura técnica de

interoperabilidad entre simuladores que se aplica tanto a los de realidad virtual, como a los de tipo constructivo e incluso como interfaz con sistemas reales. Esta arquitectura está basada en la definición de unos servicios de comunicaciones, un esquema de modelado de datos no persistentes (objetos) y una serie de reglas a la hora de utilizar los servicios y los modelos de objetos.

HLA proporciona al programador un mecanismo de intercambio de datos, definido sobre federaciones, mediante el cual crea una arquitectura basada en la distribución de servicios. A diferencia de DIS, HLA es una arquitectura que no se caracteriza especialmente por estar orientada hacia ámbitos militares, de hecho, sus elementos básicos, como son las *federaciones* y los *servicios*, se emplean asiduamente fuera de entornos militares y para la creación de grandes simulaciones civiles.

Aunque los sistemas militares nacen a partir de grandes presupuestos estatales son los equipos universitarios quienes han aportado mayores resultados en el campo de los sistemas distribuidos en red; es realmente a partir de los trabajos realizados por estos equipos durante la década pasada cuando se realizan las primeras referencias al término DVE (Distributed Virtual Environments),. Estos trabajos, muchos de ellos recogidos dentro del ámbito del estándar FIPA[1], se caracterizan por detallar las primeras estrategias básicas de simulación especialmente desarrolladas para el empleo en sistemas DVE. La figura 1.1 muestra el cronograma de la aparición de los diferentes tipos de sistemas DVE a lo largo de la historia.

[1] *Foundation for Intelligent Physical Agents* (FIPA) es un organismo para el desarrollo y establecimiento de estándares de software para agentes heterogéneos con el objetivo de definir un conjunto de normas para sistemas multiagente, infraestructura y aplicaciones.

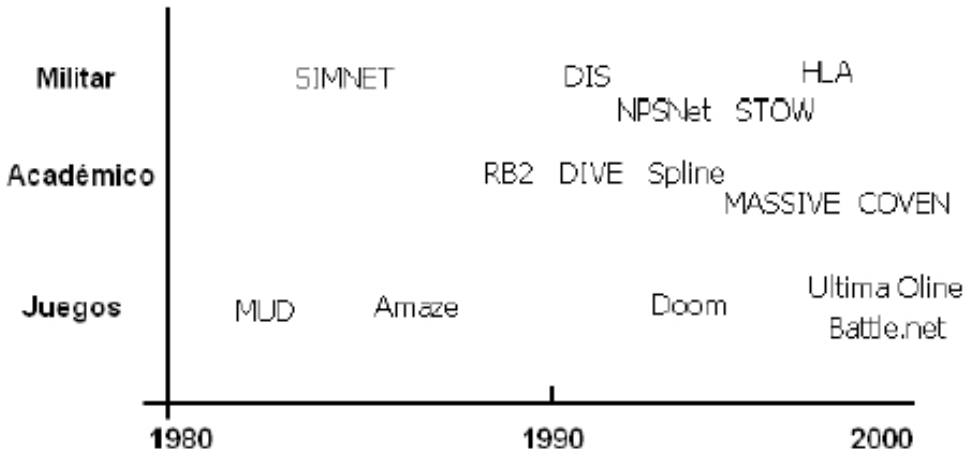


Figura 1: *Evolución histórica de los sistemas DVE según su naturaleza*

2.2 ¿Qué es la Simulación?

Conforme con el diccionario de la Real Academia, ***"simular es representar una cosa, fingiendo o imitando lo que no es".***

Esta definición lleva, en el fondo, a que simulación es un modelo cuyo interés se centra en un aspecto específico, real y observable. En consecuencia, y acotando el concepto de simulación, éste viene a constituir el empleo de un modelo de sistema, de la mayor realidad posible, con el propósito de colaborar en un proceso de investigación, de experimentación y/o de educación. Si la simulación es el concepto, emplearemos la expresión "simulador" para referirnos al equipo o sistema mismo.

Simulación implica crear un modelo que aproxima cierto aspecto de un sistema del mundo real y que puede ser usado para generar historias artificiales del sistema, de forma tal que nos permite predecir cierto aspecto del comportamiento del sistema. En particular, usaremos los computadores para imitar el comportamiento de los sistemas evaluando numéricamente un modelo del mismo. Estas evaluaciones numéricas son

las que nos permiten *generar las historias artificiales* que no son más que *experimentos*.

¿Para qué?: Simulamos para explicar, entender o mejorar el sistema.

Ejemplo: El diseño de un procesador involucra miles o millones de compuertas lógicas interconectadas. El proceso de crear el primer chip es sumamente costoso y no es posible darse el lujo de construir varios chips y luego verificar su funcionamiento. Lo que se hace es modelar el procesador y verificar su funcionamiento usando simulación.

¿Cuándo?

a) El sistema real no existe. Es costoso, peligroso, consume mucho tiempo, o imposible de construir y experimentar con prototipos (nuevo computador o procesador, reactor nuclear).
b) Experimentar con el sistema real es complicado, costoso, peligroso, o puede causar serios desajustes (sistema de transporte, sistema de manufactura, reactor nuclear).
c) Necesidad de estudiar el pasado, presente, o futuro del sistema en tiempo real, tiempo expandido, o tiempo comprimido (sistemas de control a tiempo real, estudios en cámara lenta, crecimiento poblacional).
d) El sistema es tan complejo que su evaluación analítica es prohibitiva, bien sea porque el modelado matemático es imposible, o porque el modelado matemático no tiene solución analítica o numérica simple y practica (colas de espera, ecuaciones diferenciales no lineales, problemas estocásticos).
e) Se puede validar satisfactoriamente el modelo de simulación.

Se podría decir “Simular cuando todo lo demás falla”, pero esto no es excusa para usar simulación inadecuadamente.

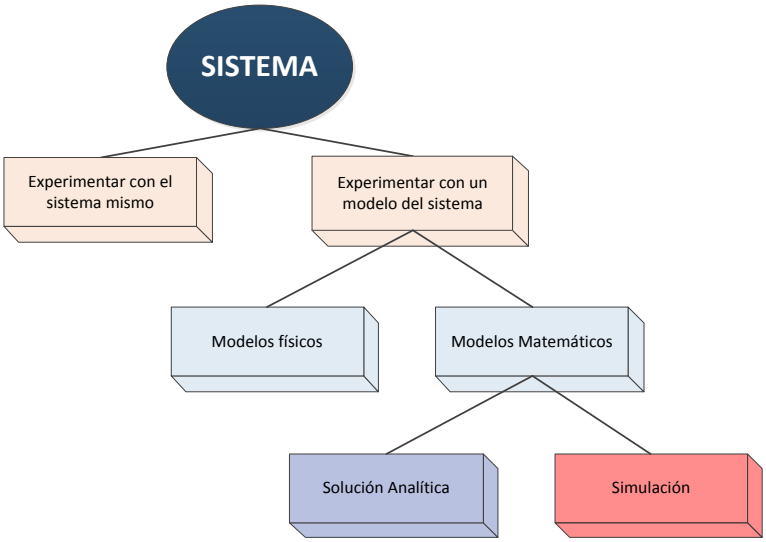


Figura 2. *La simulación en el estudio de sistemas.*

Las áreas de aplicación de la simulación son numerosas y entre ellas están:

- Diseño y análisis de sistemas de producción.
- Análisis de sistemas financieros o económicos.
- Evaluación de software y hardware.
- Evaluación de sistemas de armamento militar o sistemas tácticos.
- Determinación de políticas de inventario.
- Manejo de bosques.
- Diseño de sistemas de comunicación y protocolos
- Diseño de sistemas de transporte.
- Evaluación de diseños de organizaciones como hospitales, comedores, servicios de correo, etc.

2.3 Modelado

El término ***modelo*** también tiene su variada connotación dentro del contexto de la Simulación. Inicialmente aparecen los "modelos matemáticos", con lo que los sociólogos podían experimentar en un

laboratorio como lo hacen los físicos. Un modelo también es un "simulador" o una representación de un sistema. Por ejemplo, los modelos ***estocásticos*** dan lugar a la técnica del Análisis de Monte Carlo[2]. Los ***determinísticos***, definen la ***Simulación Caótica*** y los modelos ***dinámicos*** y ***estáticos*** tienen o no en cuenta el factor tiempo. El ***modelado*** es el proceso de construcción de un modelo.

Hoy en día reconocemos con el término "modelo" a aquellos diseños tridimensionales que tienen una conducta propia dentro del ambiente virtual. Un buque puede ser controlado por la intervención humana, luego se necesitan algunos "modelos matemáticos" para que se puedan manipular a gusto "modelos tridimensionales" o "modelos geométricos". Ahora, modelos como la niebla, nubes y el estado del mar pueden estar sincronizados con el sistema, dependiendo su ejecución en el tiempo.

Siendo así, el ***modelado 3D*** consiste en la realización de una representación visual de un objeto o conjunto de objetos mediante una computadora o cualquier otro dispositivo que permita observar el modelo final desde cualquier ángulo.

2.4 Evolución de la Simulación

La simulación ha evolucionado a través de siete categorías distintas, estas categorías nos proveen de una sólida experiencia para entender el dominio completo de la simulación.

[2] *El Método Monte Carlo* es una colección de técnicas que permiten obtener soluciones de problemas matemáticos o físicos por medio de pruebas aleatorias repetidas. Fue aplicado en aspectos de la bomba atómica.

Figura 3: *Categorías de la Simulación.*

1. ***Ensayo en vivo:*** Esto es, personas que se entrenan antes de formar parte del evento real.

   Se habla así de los "zafarranchos o simulacros". Un marino o soldado, se prepara a través de diferentes ejercicios de pensamiento, habilidad y destreza, para que llegado el día de un combate real, tenga la capacidad para enfrentar al enemigo, sin temor y con mayor decisión.

   Un bombero puede y debe cumplir esta tarea. Pero, ¿será que podemos incluir a un ciclista, un patinador, un nadador o cualquier deportista, que cumple con esta norma para poder obtener una medalla?

   Teniendo en cuenta el concepto militar de esta categoría, de ella se desprenden dos, las cuales hacen parte del ***Entrenamiento Asistido por Computador***, como son los ***Juegos de Guerra*** y la ***Realidad Virtual***.

2. ***Juegos de Guerra:*** Evento en el cual la destreza en la toma de decisiones y el uso de recursos, manejo de personal y control de la logística, son lo indispensable para el entrenamiento. Sólo las entidades militares hacen uso de este entrenamiento. Está basado en la Investigación de Operaciones.

3. ***Realidad Virtual:*** Es la inmersión de un grupo de trabajo, personas, equipos o individuos, en un mundo generado por computador para estimular sus conductas reactivas o para estimular su aptitud para hacer o desarrollar algo.

4. ***Análisis de Sistemas:*** Se utiliza para predecir el futuro o para realizar pruebas sobre las capacidades de un sistema disponible.

5. ***Simulación de Eventos Discretos:*** Técnica para estructurar el mundo como una serie de eventos secuenciales que determinen el estado de ese mundo.

6. ***Entretenimiento:*** Es la aplicación de métodos científicos en juegos de simulación, video–juegos y otros productos de entretenimiento masivo.

7. ***Interoperabilidad:*** Tecnología para enlazar múltiples simulaciones con el fin de que se pueda ***interactuar*** en un mundo virtual compartido. Un ***juego de guerra*** se puede enlazar con un ***simulador virtual local táctico*** proporcionando mayor realismo y capacidad de entrenamiento.

Las simulaciones han evolucionado en todos los frentes, como resultado, ha sido difícil establecer sus categorías o compararlas. Sin embargo, el ***Consejo para la Ciencia de la Defensa*** (***Defense Science***

Board), presentó la mejor categorización de la simulación para la defensa, facilitando la comunicación con la comunidad. Ninguna simulación encaja perfectamente en una categoría, pero cada simulación es dominada por las características de una de ellas. El entrenamiento simulado típicamente viene en cuatro categorías:

1. **Simulación "En Vivo" - de "Vida" (Instrumentadas):** es cuando las ***personas reales*** usan ***equipo simulado*** en el ***mundo real***, lo que permite que los alumnos practiquen las actividades físicas de la guerra con su equipo verdadero, el uso de ambientes de combate reales así como el empleo de fuerzas amigas y enemigas al mismo tiempo.

   "El objetivo es realizar simulacros de combate en un medio no letal".

   Ejemplo de este nivel de simulación es el Sistema MILES americano en el cual los soldados usan un sistema láser en el fusil para simular los disparos y un receptor en el cinturón, casco y suspender para conectar el equipo con el láser a fin de registrar el impacto del láser en el soldado.

2. **Simulación "Virtual" (Tripuladas):** es cuando las ***personas reales*** usan ***equipo simulado*** en ***mundos virtuales***. Son sistemas de simulación que proveen entrenamiento colectivo de personal, práctica de tiro y artillería, y entrenamiento para procedimientos especiales Se pueden reconfigurar para la simulación de varios sistemas y armamentos (T-72, T-62, T-55, BVP, BMP, M1, helicópteros, vehículos con ruedas, etc.). Son compatibles con los protocolos de comunicación DIS (simulación interactiva

distribuida), y HLA (arquitectura de alto nivel) que se utilizan en simulaciones de combate y ejercicios combinados.

Son de alta fidelidad y de bajo costo lo que permite lograr las metas de entrenamiento ya fijadas, combinan el uso de equipos reales con escenarios proyectados en el computador o por medio de un casco. Se utilizan los gráficos de computador para estimular a los soldados que operan equipos de combate, sumergiéndolos en un mundo virtual donde las acciones físicas tales como conducir o disparar un arma, tienen una directriz visible en el mundo sintético en el que se encuentra.
"***Su objetivo es el entrenamiento del soldado a través de su inmersión en un mundo virtua***l".

Ejemplo de este nivel es el simulador de movimiento completo, denominado *full motion simulator flight*, el cual duplica todos los aspectos de una aeronave y de su entorno, incluyendo movimientos básicos de la aeronave. Este tipo de simuladores pueden generar movimientos de modo que los ocupantes sientan un nivel de realismo tal como pasaría en una aeronave real, engañando a las tripulaciones y haciéndoles creer que estos se encuentran volando. Para poder realizar esto se combina una serie de aspectos tecnológicos que estimulan el sistema visual y vestibular de los pilotos. Lo que convierte a la simulación de vuelo en un área de conocimientos intensivos.

3. **Simulación "Constructiva"**, es cuando ***personas simuladas***, usan ***equipo simulado***, en ***ambientes simulados***, se conocen extensamente como "Juegos de Guerra" (Wargames). Las decisiones tácticas y estratégicas se reflejan en el movimiento de iconos militares (NTDS – Naval Tactical Display System) en un

mapa, probando la capacidad del comandante y del personal para utilizar sus fuerzas con eficacia.

Se utiliza en los niveles de Batallón, Brigada y División, y permite que los comandantes y estados mayores "observen" lo planificado mediante modelos que simulan la conducta de unidades en el terreno. Opera con bases de datos de terrenos y armamentos que son específicos de cada país. Permite la integración de simuladores tripulados y de campos de entrenamiento instrumentados.

.

Se atiene a los estándares de la OTAN para simulación incluyendo los protocolos de comunicación DIS (simulación interactiva distribuida), y HLA (arquitectura de alto nivel).

"Su objetivo es el entrenamiento para mejorar la habilidad en la toma de decisiones".

Ejemplo de este nivel de simulación es el Sistema de entrenamiento táctico SETAC del Ejército Chileno, el sistema táctico de adiestramiento tanto a nivel Brigada (SISTAB) como a nivel Batallón (SABRE) del Ejército de Brasil, el SETAC del Ejército de El Salvador

9. **Simulación "Analítica"** se utiliza para estudiar problemas como la composición de la fuerza, la eficacia de las armas, y logística. Las simulaciones analíticas se diferencian generalmente en que no se centran en intercambios interactivos con la gente durante una simulación. Es capaz de producir simulaciones parecidas a la constructiva y permitir la evaluación y análisis inmediato de los eventos.

La ciencia permite que estos niveles sean aplicados a cualquier ambiente diferente al entorno militar como el entrenamiento en Defensa Civil y en asistencia de desastres. El entrenamiento y toma de decisiones por parte de civiles ha logrado un estatus de importancia en el desarrollo de nuevos modelos orientados a la educación de cualquier persona.

2.5 Modelado & Simulación

Por lo general, un ***modelo*** es una representación física, matemática o lógica de un sistema, de una entidad, de un fenómeno o de un proceso. El anterior término definido por el DoD, nos obliga a establecer una definición para los términos ***sistema, entidad, fenómeno*** y ***proceso***.

El término ***sistema*** para nuestro interés lo definiremos como un conjunto de elementos interrelacionados entre los que existe cierta cohesión y unidad de propósito. En él se puede considerar un sistema de cómputo, el sistema digestivo, la embotelladora de líquidos, el aeropuerto, el sistema solar, por ejemplo. Cuando hablamos de ***entidad*** nos referimos a su connotación de colectividad, es decir aquello que denota un conjunto de personas o cosas, por ejemplo, el ejército o un avión.

Por su parte, ***fenómeno***, podemos definirlo en dos formas: Como una cosa o hecho que puede percibirse por los sentidos y como una manifestación de una actividad que se produce en la naturaleza. Ejemplo de esto es el toque virtual, los fenómenos atmosféricos y el sonido. Por último, el término ***proceso***, el cual consiste en el desarrollo o las fases sucesivas de un fenómeno. También se define como un método o sistema adoptado para llegar a un determinado fin.

Un modelo entonces puede ser representado físicamente, matemáticamente o en una forma lógica a la que se puede llamar representación ***virtual***. Lo cierto es que no todo modelo que se represente virtualmente, se puede representar matemáticamente y viceversa. Prácticamente uno complementa al otro.

El término ***virtual*** se aplica a aquello que tiene existencia aparente o potencial pero no real o efectiva. Nuestra imagen en el espejo, por ejemplo, los físicos la denominaron ***imagen virtual***. Esto quiere decir, que si bien representa una entidad o un sistema, este carece de realismo y por esto se dice que pertenece al mundo virtual o sintético.

También se refiere en su abstracción más representativa, al ambiente tridimensional. Así que en este documento nos referimos al término ***virtual*** como la representación tridimensional con movimiento propio en el tiempo, de un sistema, una entidad, un fenómeno o un proceso, incluso un dispositivo que represente uno real. Sólo si se presenta este caso podemos incluir aquí los aspectos bidimensionales o 2D que interactúan con el sistema virtual.

Un ejemplo claro es el de un simulador de vuelo, el cual se puede constituir en un sistema que representa una entidad como el avión, viajando a través de ciertos fenómenos naturales (clima) y procesos como el despegue y aterrizaje y es controlado por un sistema de navegación bidimensional. El término ***virtual*** se aplica a los dispositivos o ambientes simulados no al entrenamiento.

El término ***sintético*** se refiere al hecho de resumir en un ambiente los aspectos de la realidad (*método de sintetizar*) aunque en la práctica se mantiene el sinónimo de ***artificial***. Un ejemplo es cuando decimos que la simulación nos proporcionar una percepción de ciertos problemas

donde la evaluación matemática de un modelo no es posible, pero lo podemos representar de forma tridimensional. Entonces podemos hablar de ***ambiente sintético***. Semánticamente, *síntesis* es la operación inversa del *análisis*.

Otro término que se ha venido utilizando muy reiteradamente es ***ambiente***, el cual lo podemos definir como el conjunto de circunstancias físicas que rodean a un ser vivo y que influyen en su desarrollo. Entonces, dentro de éste ambiente sintético se encuentran resumidos los modelos que representan un sistema, entidades, fenómenos y procesos. Hablamos entonces de ***ambiente 3D*** o ***Virtual environment VE***.

La ***Simulación*** es la técnica de imitar la conducta de ciertas situaciones o sistemas (económicas, mecánicas...) por medio de un modelo análogo, situación o aparato, bien sea para ganar información convenientemente o para entrenar personal. En palabras más sencillas, Simulación es ejecutar un modelo, representado por un software y que nos entrega información.

La ***Simulación Virtual*** es por lo tanto, un método para poner modelos en ejecución con el fin de entrenar de manera efectiva a un grupo de personas en un aspecto común. Tal es el caso de un simulador de navegación, para entrenar al personal conformado por un comandante y una tripulación.

Otro término que enfatiza y acompaña a la definición de la Simulación Virtual se refiere al concepto **interactivo**. No todos los entrenamientos son interactivos, es decir, existen modelos y simulaciones con los cuales no podemos extraer y/o compartir sensaciones o comunicarnos en el tiempo.

La interactividad significa “con acción”, “por efecto”, “acción recíproca entre objetos, fuerzas o funciones”. El toque virtual, consiste precisamente en poder sentir de cierta forma un objeto que pertenece al mundo virtual sintético. En una cirugía de corazón, se podría sentir el palpitar del mismo, con un objeto que simule la sensación recibida – retroalimentación táctil –. Tal procedimiento se dice que es interactivo. Pero en otras palabras, queremos compartir sensaciones y comunicarnos en el ambiente virtual a lo largo del tiempo. Nuestros sentidos son excitados o estimulados por efectos que proceden del ambiente virtual, sea cual sea su naturaleza. Un juego en la red entre varios jugadores se dice que es interactivo.

Estos son ejemplos de la **Simulación Virtual Interactiva**. Por lo tanto, la **interactividad** es la capacidad del usuario para cambiar el resultado de un evento o un proceso. La Simulación Interactiva es la representación de un evento, cuyo resultado es modificable por el usuario. Pero la tendencia va más allá. Con la ayuda de la electrónica, podemos crear sistemas que nos permitan interactuar y navegar en el ambiente virtual.

La Realidad Virtual, por lo tanto, nos proporciona los conocimientos para el desarrollo de sistemas inmersivos y no inmersivos. Son inmersivos porque nos hacen estar directamente relacionados o interactuar a través de nuestros sentidos con un mundo virtual. Un guante o un casco, e incluso un sistema virtual para el entrenamiento, son sistemas inmersitos, ya que permiten que el usuario pueda sentirse “sumergido” en el interior del mundo virtual**.**

Dentro de la Realidad Virtual, además de los sistemas de visualización y el toque virtual, un aspecto importante que incluiremos en este tratado de la Simulación es el ***sonido***. El sonido es otro efecto de

sensación producido por los sistemas inmersivos. Existen bibliotecas que proporcionan ciertas funciones para producir salida de audio de alta calidad – ***high–quality audio***, en especial salida multicanal de arreglos 3D de ***fuentes*** (***sources***) de sonido alrededor de un oyente (***listener***).

En resumen, la Realidad Virtual está enfocada a la creación de interfaces de navegación (en el espacio 3D), visuales, táctiles y auditivas (incluso olfativas). La Simulación y la Realidad Virtual se desarrollan en un medio de soportes recíprocos de combinación natural la cual en ciertos aspectos una es complemento de la otra y viceversa. Por último podemos agregar que la ***Simulación Virtual Interactiva*** es para ***propósitos de entrenamiento***.

2.6 Aspectos esenciales del desarrollo de la Simulación

Existen ciertos aspectos que se deben tener en cuenta a la hora de desarrollar un producto de simulación para cualquier categoría existente.

- Por lo general ningún modelo es una representación perfecta o exacta del sistema real. Un modelo perfecto debería ser una copia o instancia del sistema mismo. Las imperfecciones son completamente normales, razonables y aceptables, pero se debe siempre buscar o tender al mejoramiento continuo. La experimentación con los modelos en busca de fallas y errores que podrían afectar el entorno global son difíciles de detectar conforme pasa el tiempo.
- El valor inherente de un modelo estará representado en el grado en el cual soluciona un problema o permite un entrenamiento aceptable.

- Por lo general, diferentes problemas requieren diferentes modelos. No existe un modelo universal que atienda a su vez diferentes problemas. Reutilizar códigos no aporta al verdadero desarrollo de la simulación.
- La mayoría de los modelos valiosos se ajustan a los requerimientos del cliente y no a las preferencias del programador.
- Un gran modelo para una solución puede ser un terrible problema para otras soluciones. Los modelos que se apliquen al desarrollo de un sistema mecánico de cierta marca de vehículo, pueden ser un desastre para otro vehículo.

2.7 Ventajas y Desventajas de la Simulación

La primera y gran ventaja de la Simulación y desde el punto de vista de la ciencia e ingeniería, es permitir responder a las preguntas ***What if...? ¿qué pasa sí...?***, lo que se define en la ciencia como, ***problema directo*** y ***¿qué debo hacer para?,*** lo que en la Ingeniería se conoce como ***problema inverso***. Estas dos preguntas encierran y responden al dilema de la simulación, para la evaluación y desarrollo de un proyecto permitiendo focalizar de antemano la finalidad de para qué estamos desarrollando un producto de simulación.

El ***problema directo*** nos permite entrenarnos de la forma tradicional, en que nos sometemos a un proceso para alcanzar un resultado u objetivo, pero con la posibilidad de extender las capacidades del sistema real. También nos permite experimentar cuando hacemos el cambio de ciertos patrones o variables, que no son comunes tratar en un hecho real. Una de las finalidades y ventajas de la simulación es la posibilidad de experimentar sobre todo cuando el modelo real es

peligroso. Un ejemplo puede ser la simulación de ¿qué pasaría si se derrama combustible en un río?

Cuando hablamos del ***problema inverso***, la simulación nos permite entrenarnos, conociendo el fin pero no los medios. Es decir, ¿qué debo hacer para lograr un objetivo que ya se conoce? Conocemos el objetivo, pero vamos descubriendo cuál es la mejor forma para llegar a esos objetivos. No todos los procedimientos son iguales. Un ejemplo puede ser la tarea de extraer el vehículo que ha caído al río.

Otras ventajas son:

- ***Desarrollo de la cognición*** – Efecto de conocer. Permite comprender el porqué de las cosas. La simulación reduce la incertidumbre y estimula la percepción.
- ***Entrenamiento*** – Posibilita el estudio de sistemas reales y la capacitación de los operarios.
- ***Toma de decisiones*** – Se refiere a la destreza adquirida para el análisis inmediato de un procedimiento con el fin de emitir órdenes.
- ***Detección de problemas*** – Permite detectar y diagnosticar problemas con relación al aspecto simulado.
- ***Evita desacuerdos*** – Permite el acuerdo y conformidad de aspectos puntuales en la solución de un problema.
- ***Ahorro monetario*** – Las simulaciones reducen los costos que implica el entrenamiento en el modelo real.

Pero también la simulación tiene sus desventajas que podríamos llamar más bien desventajas externas de la simulación, las cuales podemos catalogar en tres aspectos relevantes: costos, resultados y competitividad. Veamos cada uno de ellos:

- ***Resultados inapropiados*** – Quizá lo más peligroso de un producto de simulación inadecuado son los resultados inapropiados que arroja, sobre todo cuando se aprende a tomar decisiones sobre estos.
- ***Competitividad desleal*** – Esto es, además de los altos costos, crear productos funcionales con modelos matemáticos inadecuados. Los programas de simulación también son susceptibles a la corrupción.
- ***Altos costos*** – Las simulaciones mal planificadas, podrían demandar altas inversiones y gastos de diseño e implementación. Saber invertir en un producto de simulación es tanto como saber obtener beneficios de la adquisición.

2.8 La interoperabilidad de simuladores

Es un elemento clave en el desarrollo de sistemas de simulación puesto que aporta un valor añadido a dichos sistemas frente a aquellos que operan aisladamente. Las ventajas que introduce el concepto de interoperabilidad son:

- Una mayor reusabilidad
- Aumento de capacidad operativa de los simuladores
- Disminución de costes
- Mejora en el entrenamiento y capacitación de las unidades.

La capacidad de interoperabilidad ha afectado la manera en que los sistemas se definen, diseñan, desarrollan, prueban y manejan. Para mejorar la productividad de dichos procesos ante la nueva problemática surgida del concepto de interoperabilidad, se han tenido que desarrollar nuevos estándares que ayuden a la construcción de sistemas de simuladores interoperables.

Los simuladores de tiempo real incluyen aquellos que requieren de interacción humana con sistemas físicos reales, mientras que los de tipo constructivo representan juegos de guerra o simulan las acciones de uno o varios humanos. Los simuladores que interoperan con sistemas reales se encargan de interactuar y alimentar datos a sistemas de mando y control o sistemas de armas.

2.9 Impacto de la simulación en el aprendizaje

El Instituto de Ciencias del Comportamiento (NTL) Fundación de Salamanca, España, que dedica parte de sus recursos a investigaciones sobre el uso de diferentes métodos de aprendizaje, después de realizar un estudio entre distintas experiencias de aprendizaje y analizando posteriormente su impacto en la organización, comprobó cómo las simulaciones digitales se situaban en primer lugar para mejorar la tasa media de retención en el aprendizaje.

ACTIVIDADES DE APRENDIZAJE	TASA MEDIA DE RETENCION DE APRENDIZAJE
Escuchar	5 %
Leer	10 %
Ver y escuchar con elementos multimedia	20 %
Practicar haciendo tareas	50 %
Formación usando simulación	Hasta el 80 %

Tabla 1: *Tasa media de retención en el aprendizaje*

2.10 Errores comunes en la Simulación

1. **Nivel de detalle inapropiado** Un modelo analítico es menos detallado que un modelo de simulación. El análisis requiere de muchos supuestos y simplificaciones. El detalle en un modelo de simulación está limitado por el tiempo disponible para desarrollarlo.

Más detalle ⇒ más tiempo

⇒ incrementa la posibilidad de errores y es más difícil detectarlos

⇒ incrementa el tiempo de corrida del modelo

Más detalle no necesariamente es mejor, requiere más conocimiento de los parámetros de entrada, que si no están disponibles pueden hacer el modelo más inexacto.

Ejemplo:

Si en la simulación de un sistema de tiempo compartido (timesharing) debemos simular el tiempo requerido para satisfacer accesos a disco. Una opción es generarlos usando una distribución exponencial. Una alternativa más detallada seria simular el movimiento de los cabezales y la rotación del disco.

En la segunda alternativa se pueden tener mejores resultados solo si conocemos las referencias a sectores y pistas. Sin embargo, si esta información no está disponible a la hora de la entrada de datos, hay que terminar generándolos exponencialmente y hubiese sido menos costoso irse por la primera alternativa.

Es mejor partir de un modelo sencillo, obtener resultados, estudiar la sensibilidad, e introducir más detalles en las áreas que impactan más los resultados.

2. Lenguaje inapropiado

Lenguajes de simulación de propósito especial requieren menos tiempo para implementar el modelo y facilitan actividades como verificación (mediante el uso de opciones de trazado) y de análisis estadístico. Lenguajes de propósito general son más portables y proveen mejor control sobre la eficiencia y el tiempo de corrida de la simulación.

3. Modelos no verificados

Los modelos de simulación son generalmente programas grandes, que si no se tienen las precauciones respectivas, es posible tener errores de programación que hagan las conclusiones sin sentido.

4. Modelos inválidos

Aun cuando no haya errores de programación, puede que el modelo no represente al sistema real adecuadamente por supuestos incorrectos en su formulación. Es esencial que el modelo sea validado para asegurar que las conclusiones a las que se pueda llegar sean las mismas que se obtendrían del sistema real. Todo modelo de simulación debe estar bajo sospecha hasta que se pruebe lo contrario por modelos analíticos, mediciones, o intuición.

5. Tratamiento incorrecto de las condiciones iniciales

Generalmente la parte inicial de una corrida de simulación no es representativa del comportamiento de un sistema en estado estable, por lo tanto debe ser descartada.

6. Simulaciones muy cortas

Por tratar de ahorrar tiempo de análisis y de computación, las corridas de simulación pueden ser muy cortas. Los resultados en estos casos dependen fuertemente de las condiciones iniciales y pueden no representar al sistema real. El tiempo de corrida adecuado depende de la exactitud deseada (intervalos de confianza) y de la varianza de las cantidades observadas.

7. Generadores de números aleatorios inadecuados

Las simulaciones requieren de cantidades aleatorias que son producidas por procedimientos llamados generadores de números aleatorios. Es mejor usar generadores que han sido bien analizados a usar los de uno mismo. Aun buenos generadores presentan problemas.

8. Selección de semillas inadecuadas

Los generadores de números aleatorios son procedimientos que dado un numero aleatorio generan otro. El primer número aleatorio de la secuencia es llamado la semilla y debe ser proporcionada por el analista. Las semillas para diferentes secuencias deben ser cuidadosamente seleccionadas para mantener independencia entre las secuencias. Los analistas usualmente usan una misma secuencia para diferentes procesos o usan la misma semilla para todas las secuencias. Esto introduce correlación entre los procesos y puede llevar a conclusiones erróneas.

9. **Estimación inadecuada del tiempo para desarrollar el proyecto**

Es común subestimar el tiempo y el esfuerzo requerido para desarrollar modelos de simulación. Si la simulación es exitosa y produce información útil, sus usuarios quieren incorporar más funciones, parámetros y detalles. Por el contrario, si no provee de información útil, usualmente se espera que al añadir elementos la puedan hacer útil.

En ambos casos el proyecto se extiende más allá de las proyecciones iniciales. Para proyectos grandes se deben hacer previsiones para incorporar cambios que son inevitables sobre largos periodos de tiempo.

10. **Metas inalcanzables**

La simulación es un proceso largo y complejo y se debe tener claramente definido un conjunto de metas que sean específicas, minuciosas, medibles, y alcanzables. Un ejemplo común de una meta inalcanzable es "modelemos X." Es posible modelar muchas características diferentes de X a muchos niveles de detalle.

11. **Mezcla incompleta de habilidades**

Un proyecto de simulación requiere por lo menos:

a. **Liderazgo:** Habilidad para motivar, guiar y manejar a los miembros del equipo de simulación.

b. **Modelaje y estadísticas:** Habilidad para identificar las características claves del sistema y modelarlas al nivel de detalle requerido.

c. **Programación:** Habilidad para escribir código entendible y verificable que implemente el modelo correctamente.

d. **Conocimiento del sistema modelado:** Habilidad para entender el sistema, explicarlo al equipo de modelaje, e interpretar los resultados del modelo en términos de su impacto en el diseño del sistema.

12. Nivel inadecuado de participación de los usuarios

Es esencial que el equipo de simulación y los usuarios de la organización estén en constante contacto para intercambiar y discutir ideas. La mayoría de los sistemas evolucionan y cambian con el tiempo y un modelo desarrollado sin la participación de los usuarios raramente resulta exitoso.

13. Documentación inexistente u obsoleta

La mayoría de los modelos de simulación se desarrollan en largos periodos de tiempo y continuamente son modificados a medida que el sistema cambia o es mejor comprendido. Su documentación muchas veces es desatendida y rápidamente se vuelve obsoleta. Es recomendable documentar los programas y usar lenguajes que sean fáciles de leer.

14. Inhabilidad para gerenciar el desarrollo de programas de computación grandes

Hay muchas herramientas de ingeniería de la programación que permiten vigilar los objetivos del diseño, los requerimientos funcionales, las estructuras de datos y los estimados de progreso. También hay un conjunto de principios de diseño, como diseño de arriba abajo y programación estructurada, para desarrollar grandes proyectos en forma ordenada . Sin el uso de estas herramientas y técnicas es imposible desarrollar exitosamente un modelo de simulación grande.

15. Resultados misteriosos

Resultados misteriosos generalmente son debido a errores de programación, supuestos incorrectos en el modelo, o falta de entendimiento del sistema real. Nunca deben ser obviados.

3. EL DESARROLLO DE LOS JUEGOS DE GUERRA

3.1 Resumen

Los Juegos de Guerra son de raigambre militar y se han usado para preparar a los líderes militares a enfrentarse a circunstancias imprevistas en el combate. Los usaron los antiguos griegos y en 1811 los *prusianos* introdujeron los tableros de juego tridimensionales para añadirle realismo al juego.

En la Segunda Guerra Mundial el Almirante Nimitz, con los juegos de guerra, previó todas las batallas navales que se dieron en el Pacífico excepto la táctica japonesa de los kamikazes. En años recientes, con la ayuda de computadores y de programas inteligentes, los juegos de

guerra previeron la caída de la Unión Soviética, determinaron las opciones para el uso de la fuerza militar en la campaña "Tormenta del Desierto" en Irak y el desembarco en Haití.

El vertiginoso adelanto científico y tecnológico que el mundo entero está inmerso y experimentando, obliga a sus Fuerzas Armadas a que informaticen sus actividades, para mejorar sus procesos, ganando tiempo y economizando recursos.

Para que esto ocurra es necesario mejorar y adaptar la infraestructura para la simulación de los juegos de guerra y así cumplir de una mejor manera con las necesidades actuales y futuras de las instituciones militares. Histórica y pragmáticamente hablando, hay razones sólidas para volver a enfocar y volver a definir el uso de esta herramienta de valor incalculable para poder planificar y ejecutar mejor la guerra.

Uno no debe argumentar si el juego de guerra representa la capacitación o el adiestramiento o si es operacional o analítico. Todas las doctrinas son útiles para producir temas que preparan a los soldados y a los planificadores a tomar buenas decisiones (figura 4).

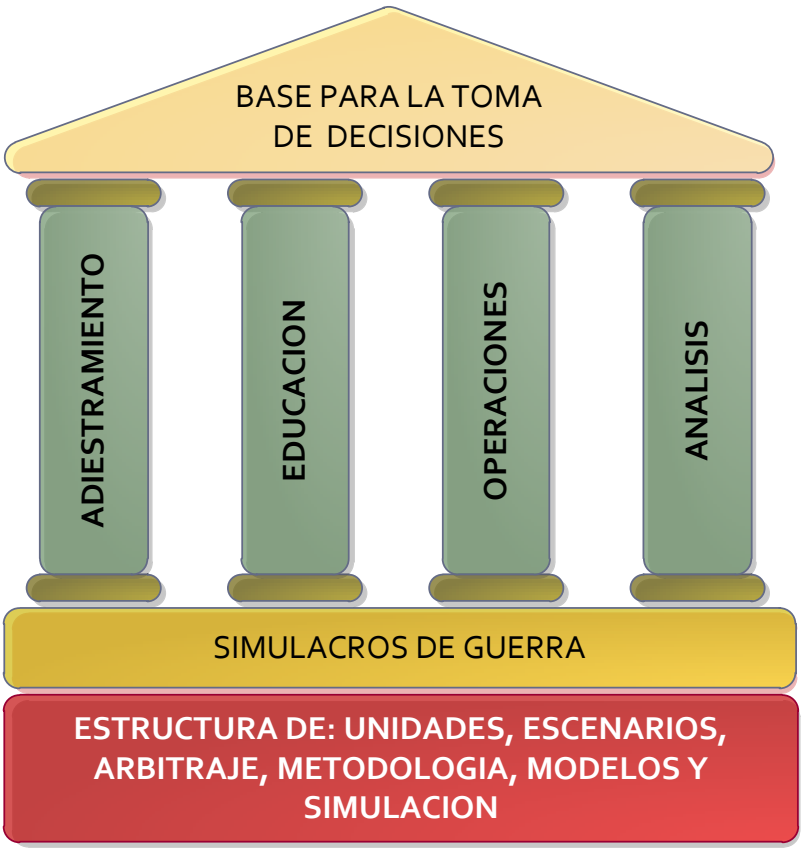


Figura 4: *La Base de la Toma de Decisiones*

Junto con sus herramientas de apoyo de diseño y simulación, el juego de guerra le enseña al personal a procesar los problemas de manera más eficaz al tomar buenas decisiones. En ese sentido, el uso profesional del juego de guerra puede abarcar un amplio espectro de tiempos, marcos hipotéticos y circunstancias.

Nuestro personal militar y nuestras organizaciones tienen que estar bien preparadas, y el juego de guerra puede hacer la diferencia, en términos de adiestramiento decisivo y capacitación, en la aptitud (o falta de la misma) de los líderes en el campo de batalla.

Podemos encontrar muchos ejemplos históricos de los aportes de los juegos de guerra a las estrategias, operaciones o tácticas de éxito, y es

tentador intentar comprobar el valor del juego de guerra señalando la causalidad directa entre los juegos de guerra y el éxito en la guerra.

Los historiadores sólo pueden adivinar cuan frecuente y substancialmente las experiencias de juegos de guerra anteriores han influenciado las decisiones de los comandantes en tiempo de guerra. La continuidad no registrada entre el juego de guerra y los pensamientos de los comandantes es inmensa. No obstante, encontramos mérito al reconocer que el juego de guerra ha gozado de una influencia históricamente importante en el arte operacional de la guerra.

3.2 Clasificación de los juegos de guerra

En la actualidad se utilizan diversos tipos de simuladores para el entrenamiento de las unidades militares, algunos asistidos por computador, otros totalmente computarizados y aquellos que emplean sistemas de simulación.

1. Los *juegos de guerra asistidos por computador*, son juegos de mesa que simulan combates, batallas o enfrentamientos, ya sean: terrestres, navales, aéreos o de submarinos y se juegan sobre cartografía o sobre un tablero y los cálculos se los realiza mediante el computador en forma externa.

   Se crea una situación bélica con su respectiva cadena de mando para permitir a los comandantes manipular modelos a través de posibles escenarios en su planificación militar, simulando situaciones habituales de incertidumbre militar, unos mapas predeterminados para simular el escenario, se usan iconos para marcar las posiciones en la que se enfrentan dos equipos, gráficos con calcos para representar la operación y finalmente

son dirigidos o arbitrados por un árbitro lo que genera confusiones a la hora de ejecutar las órdenes e imposibilidad de cumplirlas al cambiar las situaciones tácticas por la falta coherente de resultados.

2. Los *juegos de guerra computarizados* son aquellos que se juegan directamente sobre la pantalla del computador, mediante un modelo digital del terreno, modelos matemáticos y un software comercial diseñado y construido especialmente para el efecto.

   Es un sistema construido mediante una combinación de medios humanos (un equipo multidisciplinario), elementos técnicos (servidores, computadoras) un software comercial para desarrollo de la interfaz del usuario, un software para la definición y tratamiento del escenario y una base de datos para almacenar la información, lo cual permite hacer viable el desarrollo de los juegos de guerra.

3. Los *juegos de guerra simulados* son los que tienen en esencia un motor de ejecución, contiene parámetros estructurados (Base de Datos Documental) fácil manipulación de ambientes 3D, desarrollado en base a los protocolos DIS y HLA, permite la reusabilidad de componentes de simulación

   Son un campo de la tecnología muy relacionado con la inteligencia artificial, su arquitectura es dirigida por eventos, la información enviada está basada en los cambios de estado de las entidades y en su interacción, las aplicaciones se comunican a través de un mecanismo de distribución de datos llamado el

RTI[3] y sus atributos, asociaciones, e interacciones son soportadas por una federación[4]

3.3 Los desarrollos actuales de software para juegos de guerra

Un juego de guerra es aquel que recrea un enfrentamiento armado de cualquier nivel (táctico, operacional o estratégico) con reglas que implementan cierta simulación de la tecnología, estrategia y organización militar usada en cualquier entorno histórico, hipotético o fantástico y que no implica en algún momento el uso de violencia física.

Es una simulación de combate o acción bélica, ya sea como un juego de mesa, como un videojuego por consola o por computadora, o como una recreación real. Los juegos de guerra se han desarrollado en muchos países para desempeñar un papel importante en el entrenamiento del personal militar, enfocándose en la capacitación y adiestramiento en el ámbito táctico de la guerra.

La tecnología de la Simulación en los centros de entrenamiento a nivel mundial, ha permitido el desarrollo de Simuladores de Juegos de Guerra cuya ventaja significativa es la disminución relativa de costos por entrenado, costos equipos y disminución de riesgos posibles que se presentan en una situación de entrenamiento real. A continuación se detallan algunos ejemplos del desarrollo alcanzado por algunos países:

[3] ***RTI:*** "*Runtime Infrastructure*" Proporciona un conjunto de servicios común a los federados, su función primaria es la distribución de datos, bajo HLA los federados envían la información al RTI, el cuál transmite los datos a los federados apropiados

[4] ***Federación:*** Colección de simulaciones de las que consta el ejercicio.

EJERCITO DE CHILE

Figura 5: *Escudo del Ejercito de Chile*

Desde la década del '70 el Ejército Chileno comienza a incorporar distintos tipos de ayudas de instrucción que se encuadran en el primer nivel de simulación, sobre la necesidad básica de reducir gastos por el uso de munición real, disminuir el desgaste y deterioro de armas de diferentes calibres y mejorar las destrezas de apuntadores y sirvientes de armas individuales y colectivas. La gran mayoría son de tipo mecánico para efectuar una tarea parcial de la instrucción.

En un comienzo las Armas más beneficiadas fueron, en aquel entonces, la Infantería, la Artillería, la Caballería y los Blindados y, posteriormente, se amplió a otros segmentos de la Institución. A modo de ejemplo se mencionan los siguientes tipos:

- Dispositivos de subcalibres de 25 mm. para el tiro de morteros en los calibres 60, 81 y 120 mm. y 4.2 pulgadas fabricados por Singapur y EE.UU. respectivamente.
- Granadas inertes de morteros 81 mm. para tiro en espacios reducidos.

- Simulador de gabinete para tiro de morteros por medio de un haz de luz.
- Dispositivo de subcalibre de 14,5 mm. para adiestramiento de sirvientes en el tiro de obús de 105 mm. Otomelara.
- Dispositivo de subcalibre de 14,5 mm. para adiestramiento de comandante y artillero del tanque M-41.
- Simulador de tiro para apuntador de misil antitanque Mamba.
- Simulador de tiro virtual para tanque AMX-30.

A fines de la década de los '80 ya existía una base de las ventajas de contar con equipos de simulación, antecedente que, sumado a las experiencias de ejércitos modernos que hacían uso de simuladores con tecnología computacional y de láser, influyó en la decisión de adquirir y experimentar este tipo de ingenios, específicamente el año 1988, mediante cuatro unidades de simuladores de entrenamiento táctico técnico para tanques M-51 con la finalidad de medir sus bondades en el combate entre tanques.

Los resultados fueron evidentes y ese mismo año el Ejército formaliza un programa de adquisición y de cursos de operación y de mantenimiento de sistemas Simfire, de origen inglés, para dotar a las Unidades Tácticas que contaban con material blindado, actividad que concluye a fines del año 1990. Junto a la compra de este material se consideran simuladores Simgun para tiro de fusil, de similar propósito que para los tanques, pero para el enfrentamiento entre tropas de infantería.

Desde el término de este programa se adoptó como política institucional que las futuras adquisiciones de sistemas de armas deberían prever un simulador para su operación, aspecto que se refleja, por ejemplo, en la incorporación de material antitanque Mapat.

El año 1992 nace el "sistema de entrenamiento táctico computacional" (SETAC) y es puesto en marcha blanca hasta el año 1993. Junto a ello, se edifica una infraestructura en el mismo recinto de la Academia de Guerra destinada exclusivamente a la administración de este proyecto, denominándosele Centro de Entrenamiento Operativo Táctico.

SETAC - Sistema de Entrenamiento Táctico Computarizado

Figura 6: *Logotipo del CEOTAC*

Es un *software* computacional de tecnología abierta que permite la realización de Juegos de Guerra en la modalidad denominada doble acción, con dos elementos adversarios que se enfrentan, este fue diseñado el año 1993 en la Academia de Guerra del Ejército de Chile, a través del trabajo integrado de Oficiales de Estado Mayor, Ingenieros Politécnicos Militares e Ingenieros Civiles provenientes de las principales Universidades del país.

Este *software* desarrollado íntegramente en Chile permite que se enfrenten en tiempo real, dos hipotéticos adversarios sobre un mismo escenario geográfico. Los resultados del enfrentamiento de las Unidades participantes son entregados por los modelos matemáticos en que se sustenta el sistema, fallos que están en directa relación, con

las capacidades de los sistemas de armas que las unidades poseen y con las variables de terreno y tiempo atmosférico existentes al momento de producirse los enfrentamientos.

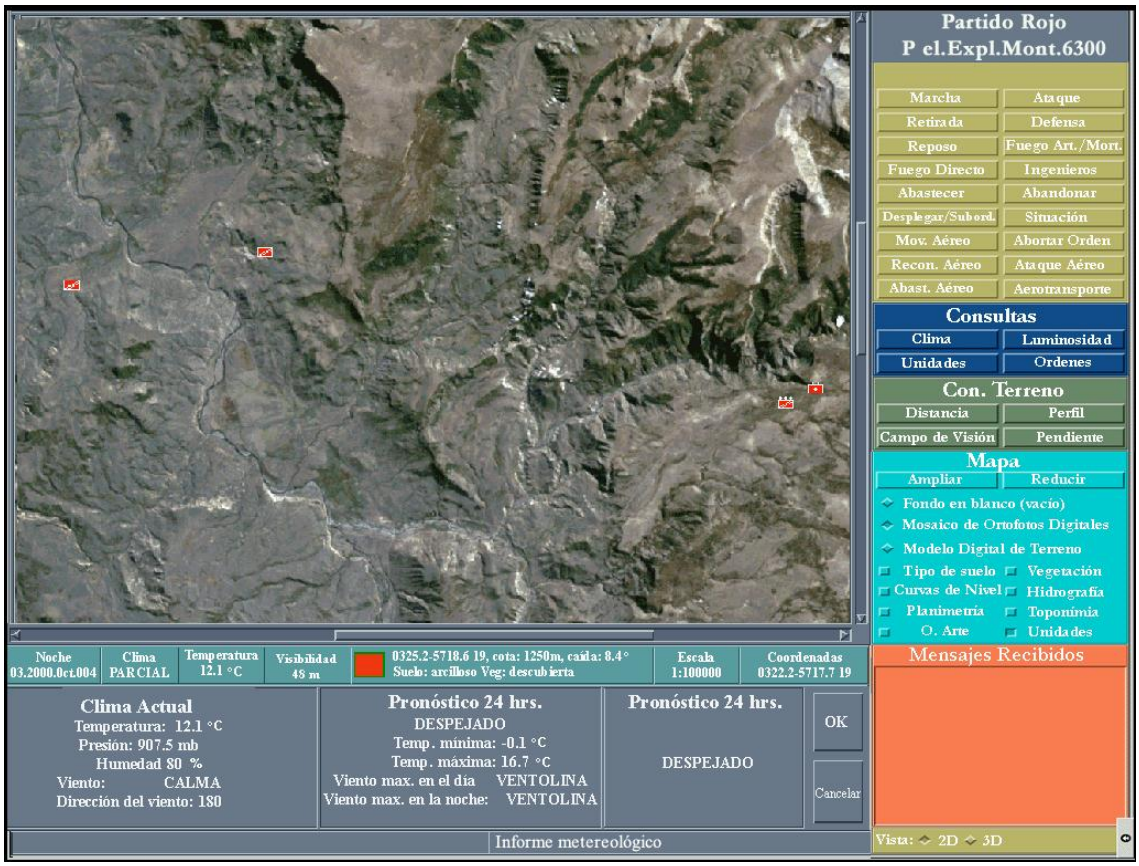


Figura 7: *Interfaz gráfica del Comandante*

El objetivo de este *software*, es entrenar a través de simulación a los Comandantes y Estados Mayores de nivel Brigada y División (en la versión utilizada por la Academia de Guerra y las Unidades Operativas del Ejército) y a los Comandantes de Batallón y sus Planas Mayores (en la versión en uso en las Escuelas de Infantería y de Caballería Blindada), en el proceso de toma de decisiones militares, planificación, ejecución y control de las acciones tácticas, bajo un ambiente de presión y apremio.

SETAC representa en forma dinámica las variables que componen el campo de batalla, en el que los acontecimientos que ocurren, son producto de las acciones que realizan en tiempo real, cada uno de los Comandantes, todos con su propia voluntad de combate, pero con la influencia siempre presente de los factores asociados al espacio geográfico donde transcurren las acciones militares. Tres componentes se consideraron para el desarrollo de SETAC.

1 *El primero de ellos es el tecnológico*. Hoy para nadie es desconocido que la tecnología ha revolucionado todos los campos del quehacer humano, incluido, lógicamente el militar y dentro de este, al campo de batalla moderno. La tecnología por tanto, condiciona en gran medida la efectividad de una unidad en el combate, donde el manejo rápido de la información, la precisión en la acción, la coordinación de los medios y la oportunidad de su empleo, constituyen los pilares básicos para el logro del éxito.

   Lo anterior significa en términos prácticos que, un Comandante cuya unidad está empeñada en combate, debe lograr con su gestión, que su unidad llegue al lugar indicado, en el momento preciso y con la potencia de combate necesaria para alcanzar su objetivo.

2 *El segundo componente tiene relación con lo económico y ambiental.* El crecimiento de las ciudades, la expansión de los sectores agrícolas, la explotación del mar, la preocupación por la contaminación acústica y la protección del medio ambiente, dentro de otros aspectos, representan las dificultades existentes hoy, para realizar entrenamientos militares con las fuerzas en presencia.

   A lo anterior, hay que agregar el alto costo asociado al empleo de armamento con efectos cada vez más devastadores, en actividades

de entrenamiento, lo que además de resultar de alto costo, altera significativamente el escenario, contraponiéndose en forma directa, a las políticas de protección del medio ambiente, realizadas por el Ejército de Chile.

3 *El tercer y último componente, tiene relación con la modernización de los sistemas de entrenamiento de uso en el Ejército*, de forma tal, de buscar alternativas que permitieran optimizar el entrenamiento individual, especializado y colectivo de sus hombres y unidades, con el objeto de mantener la eficiencia operacional de la fuerza.

Los Juegos de Guerra tradicionales, pese a todas sus bondades, presentaban algunas falencias relacionadas, principalmente, con la objetividad en los fallos, la abstracción de la realidad por parte de los entrenados, la ejecución de los procesos asociados al mando en tiempo real, y la aplicación práctica de conceptos tales como comando y control o ritmo de combate. Surgía así, entonces, la necesidad de contar con una herramienta que permitiera ejecutar Juegos de Guerra, idealmente de doble acción, que entregara resultados objetivos de las resoluciones y enfrentamientos ocurridos, que permitiera una dinámica en tiempo real y que posibilitara la aplicación práctica y la comprobación de los conceptos enseñados en aula.

La estructura y funcionalidad de SETAC generada para la materialización de un Juego de Guerra de doble acción, está conformada por 15 diferentes modelos, cuyos parámetros, nivel de detalle, integración y efectos deseados, permiten generar una abstracción de la realidad, que trata de representar en tiempo real, las vivencias del frente de combate de las fuerzas de dos adversarios que se enfrentan, con el objeto que los Comandantes y sus Asesor es,

tengan que resolver, idealmente, problemas propios de la ejecución del combate.

Figura 8: *Exposición de un ejercicio táctico*

El sistema opera en forma distribuida sobre una red variable de estaciones de trabajo bajo ambiente LINUX, usando un sistema de base relacional (ORACLE) y protocolo de comunicaciones entre procesos remotos basados en sockets. La arquitectura de *software* es abierta y está diseñada para ser instalada en plataforma de computadores personales, permitiendo la incorporación de nuevas funcionalidades. Las 300.000 líneas de códigos están escritas orientadas al objeto, con paralelismo y gráfica de última generación, ocupando lenguaje C++ y Open/GL.

Posee un sistema de información geográfico propio, que permite manejar imágenes satelitales geo referenciadas en un mapa digital, aspecto que lo destaca dentro de otros sistemas de simulación de este

tipo que existen en el mundo. Lo anterior, debido a que se realizó con éxito la incorporación de imágenes raster en su modalidad de ortofotos digitales e imágenes de satélites, permitiendo contar con un nivel adicional de información geográfica y actualizada del territorio nacional, posibilitando que los niveles de información vectorial, se aproximen a la realidad temporal, pudiendo incorporar los elementos más críticos del espacio geográfico.

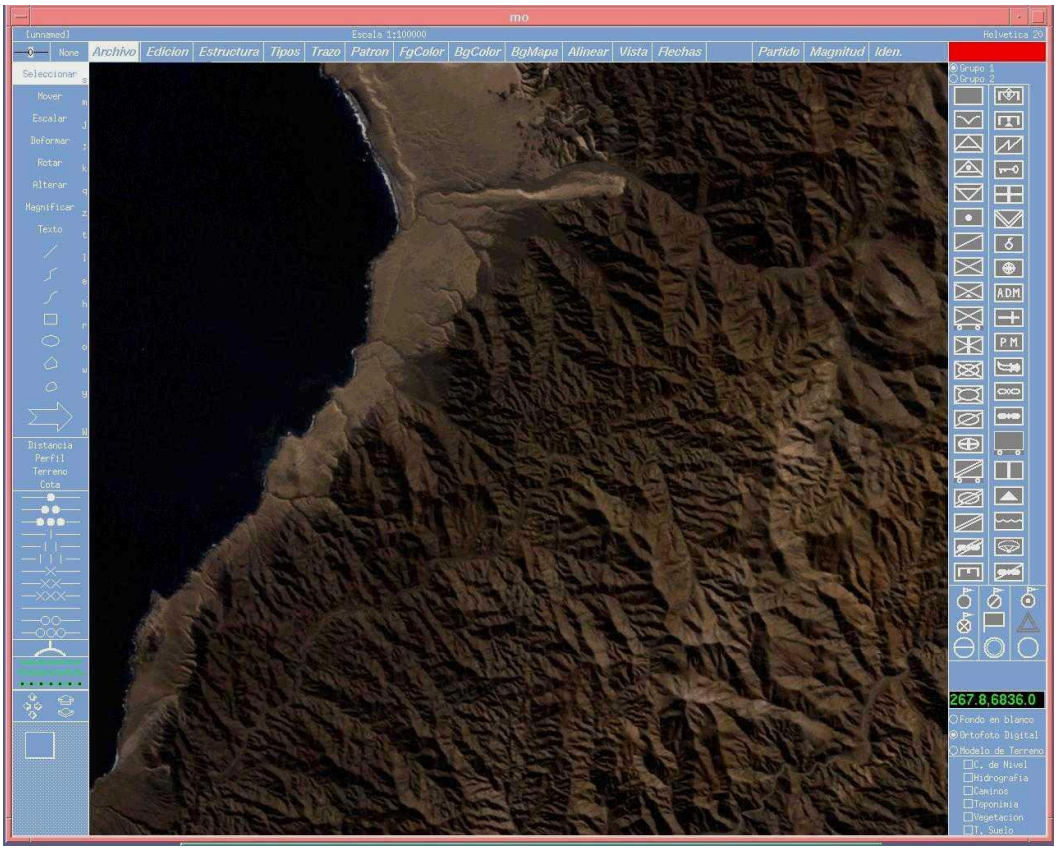


Figura 9: *Interfaz gráfica del usuario*

Con el transcurso del tiempo y debido a los procesos de investigación y desarrollo y a la propia explotación del sistema en la Academia de Guerra del Ejército, el Centro de Entrenamiento Operativo Táctico Computacional (CEOTAC), organismo a cargo del sistema SETAC ha realizado diversas modificaciones en su estructura. Estas han tenido por objeto mejorar sus herramientas, aumentar su confiabilidad y

satisfacer las crecientes demandas tácticas y técnicas de los usuarios del sistema. SETAC en su versión 5.0 actualmente en uso, es muy diferente en calidad, rendimiento, prestaciones y confiabilidad, con respecto a las versiones anteriores, situación que lo ubica en una posición de privilegio entre los simuladores tácticos del campo de combate existentes en el mundo.

SETAC ha incorporado en la simulación a los radares de vigilancia terrestre y los visores nocturnos a través del modelo de detección, de forma tal, que el Comandante cuente con un medio más de búsqueda de información y en la vigilancia de espacios no cubiertos por tropas.
Los efectos de estas capacidades están debidamente modelados de acuerdo a las características técnicas de cada tipo de material.

Otro importante avance es el Graficador Militar, herramienta que permite la graficación de situaciones, órdenes y resoluciones, la cual fue modificada para que funcionara en computadoras personales y en ambiente Windows. Esto permite que los profesores y alumnos de la Academia de Guerra y el personal de todas las Unidades de la Institución, tengan acceso a esta aplicación para el desarrollo de sus propios ejercicios de entrenamiento, en sus lugares de trabajo, disminuyendo los tiempos destinados a la preparación de las exposiciones, usando una moderna tecnología diseñada completamente para fines institucionales.

Lo anterior, facilita la discusión académica y el intercambio de experiencias sobre temas o situaciones tácticas específicas, de tal forma que, visualmente, todos los involucrados en la ejecución del Juego, pueden conocer “que sucedió”, “porqué sucedió” y lo más importante, extraer sus propias conclusiones para mejorar aquellos

aspectos señalados como deficientes por el personal a cargo del entrenamiento.

El impacto del uso de la simulación ha sido determinante para el entrenamiento de los Comandantes y sus Cuarteles Generales y para la utilización de los sistemas y procedimientos de mando y control, especialmente en lo relacionado con los procesos de planificación e integración de las funciones de Personal, Inteligencia, Operaciones y Logística.

STEEL BEASTS PROFESSIONAL

Durante el año 2002, se desarrolló en el Regimiento Reforzado Nº 2 "Cazadores" un proyecto tendiente a incrementar la eficiencia de las tripulaciones de tanques Leopard 1-V, específicamente para los puestos de artillero y comandante, mediante la utilización de un juego computacional denominado "Steel Beasts", recurso didáctico que hasta la fecha se utiliza con buenos resultados.

Si bien el objetivo de este "simulador" era preparar a los tripulantes de los citados vehículos blindados, pronto se evidenció que era posible ampliar su espectro de empleo hacia el área de la "Táctica", entrenando comandantes de unidades pequeñas, específicamente de los niveles de Pelotón y Escuadrón, ya que cada acción dentro del "mundo virtual" debía ser planificada y ejecutada tanto técnica, como tácticamente.

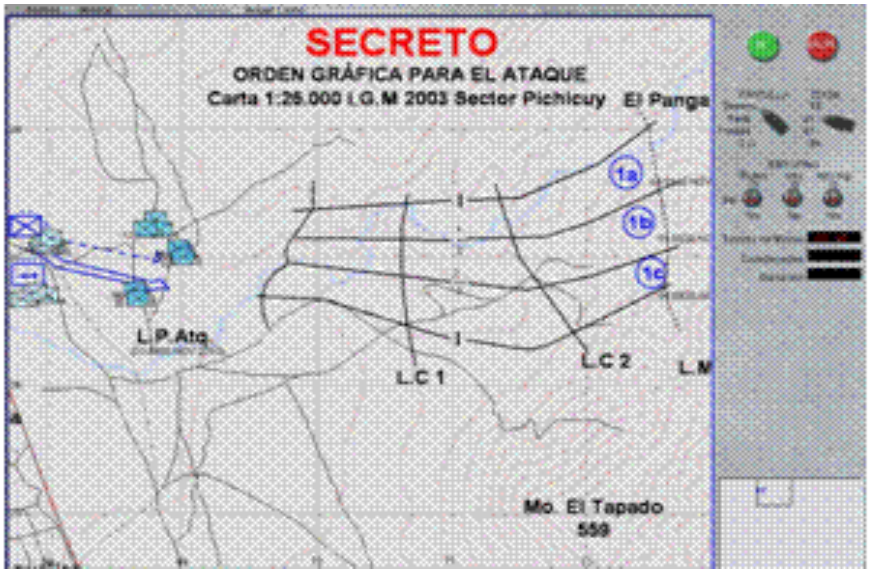


Figura 10: *Orden grafica para el ataque (compañía)*

Dentro de este contexto, la Escuela Militar, incluyo este software para la ejecución de una de sus principales asignaturas, denominada "Mando y Conducción", la que busca capacitar a los alumnos para gestar una resolución y planificar acciones militares ante situaciones propias de la guerra regular.

Figura 11: *Alumno exponiendo su Orden táctica.*

Las características más destacables de este software son:

- Permite la ejecución de ejercicios demostrativos, capaces de ilustrar a los alumnos el desarrollo de acciones tácticas fundamentales.
- Tiene la capacidad para generar ejercicios de doble acción, fundamentales en el entrenamiento de comandantes, ya que se enfrentan ante adversarios con inteligencia y voluntad.
- Utiliza el sistema de graficación y la nomenclatura OTAN
- Incluye menús y textos en español, lo que facilita la comprensión por parte de todos los usuarios.
- Es posible importar terrenos reales y transformarlos en virtuales, los que son desplegados en dos y tres dimensiones.
- Tiene la capacidad para otorgar distintos niveles de operacionalidad a cada unidad, diferenciando calibres y cantidades y tipos de munición.
- Una característica destacable es el sistema de intercambio de información entre los usuarios registrados. Lo anterior, se materializa mediante un sistema de "foro en línea", lo que permite conocer avances y generar contactos con otros ejércitos y academias. Esta agrupación es denominada "comunidad".
- Por la condición de software "licenciado", no es posible acceder a los códigos de programación; sin embargo, es relevante el sistema de "actualizaciones colaborativas", en donde cada nuevo desarrollo del programa, financiado por un usuario específico, es puesto a disposición de la "comunidad".

SETAC 2 WEB

La Academia de Guerra, a través del CEOTAC (Centro de Entrenamiento Operativo Táctico del Ejército de Chile) actualmente está utilizando un sistema de simulación de guerra desarrollado por la empresa InterSystems en colaboración con ingenieros de la Universidad de Chile, ingenieros politécnicos militares y oficiales de Estado Mayor. Basado en tecnología de punta, este sistema -denominado Setac 2 web (S2W) permite entrenar a distancia -vía Internet- a los involucrados en un 'juego de guerra'. Es decir, comandantes desde distintos lugares del País pueden participar de la misma simulación, en tiempo real, accediendo a través de Internet Explorer a su cuenta, como si fuera su e-mail.

Este programa permite recrear la complejidad del campo de batalla moderno y comprobar una planificación y conducción de unidades, además de entrenar comandantes y asesores de las distintas funciones al mando, en un escenario determinado, bajo condiciones ficticias lo más parecidas a la realidad posibles.

Para el desarrollo de este moderno sistema táctico de entrenamiento, se utilizó la base de datos "Caché" de InterSystems, una base de datos de alto rendimiento, que se ejecuta cinco veces más rápido que las bases de datos relacionales. Caché permitió al programa de simulación, acceder a un desarrollo rápido de aplicaciones web, además de lograr una extraordinaria velocidad en el procesamiento de transacciones y escalabilidad masiva.

Además trabajan con la plataforma .Net de Microsoft y en las computadoras clientes usan Windows XP. Estos factores marcan la diferencia con los sistemas que anteriormente utilizaba el CEOTAC,

permitiéndoles realizar consultas en tiempo real de datos transaccionales, con un mínimo mantenimiento y requerimiento de hardware. Para la cartografía usan Adobe SVG Viewer componente que permite visualizar documentos gráficos vectoriales en formato SVG (Scalable vector Graphics).

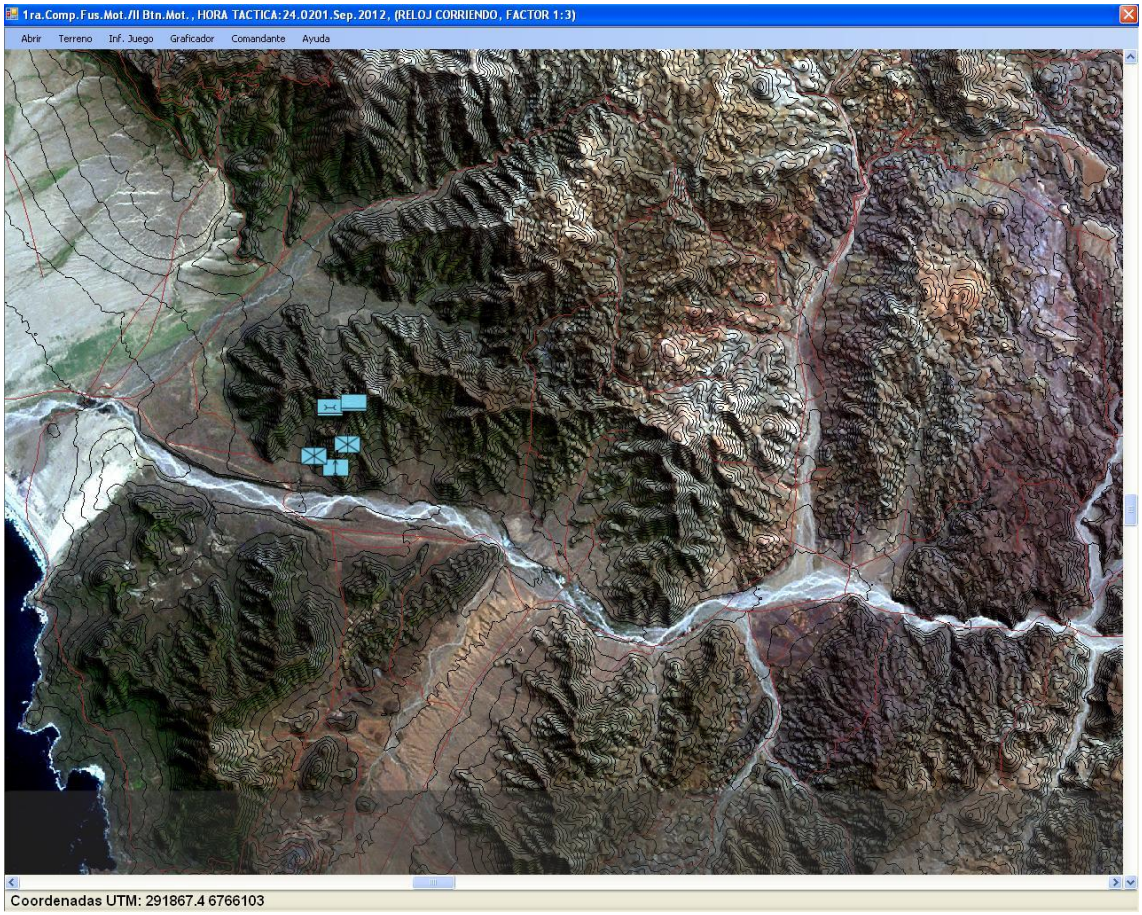


Figura 12: *Interfaz gráfica del usuario.*

El sistema actúa básicamente como un juego en que los participantes trabajan sobre eventos de magnitud real. La simulación consiste en recrear un terreno digitalizado, basado en un determinado conflicto, en que se movilizan diferentes unidades, según las órdenes que da el usuario del sistema. En esta recreación se representan factores como el clima, el terreno, armas y efectos utilizados por las distintas unidades, como también el tiempo y las fuerzas que se usarán en las actividades que se desarrollan.

El programa se enfoca en el comandante y en el proceso de toma de decisiones del personal que entrena, como un modo de preparar a oficiales en la planificación táctica de guerra, el mando operacional y la conducción de fuerzas militares en situaciones de conflicto.

EJERCITO DE EL SALVADOR

Figura 13: *Escudo del Ejercito de El Salvador*.

Durante el año de 1994 en vista de la necesidad de contar con una herramienta que sirviera para entrenar a Comandantes y Estados Mayores en la planificación y conducción operativa y táctica y Apoyar el sistema enseñanza - aprendizaje en las Escuelas del CODEM, surge la idea de la creación del Centro de Entrenamiento Táctico Computarizado (CETAC).

Como una primera investigación sobre el Sistema de Entrenamiento Táctico Computarizado (SETAC), se envió en Febrero de 1994 a Chile, una Comisión de Oficiales, para observar los aspectos doctrinarios de Estado Mayor y el segundo los aspectos técnicos del SETAC.

Este fue el primer acercamiento investigativo sobre el SETAC, que dejó como resultado una oferta de FAMAE para la adquisición del Sistema. El 25 de agosto de 1995, se formalizó el contrato de adquisición del SETAC; y en octubre de ese mismo año se conformó una comisión para administrar el proyecto.

De febrero a Octubre de 1996, se realizó en Chile la Investigación de Campo para la adecuación del Software del SETAC Versión 2.5 a las necesidades reales de la Fuerza Armada de El Salvador. En lo relacionado a aspectos técnicos se realizaron mediciones las cuales se orientarían específicamente a los pesos en libras y volúmenes en metros cúbicos del armamento, equipo individual y munición, capacidad de carga en metros cúbicos y libras, de los diferentes medios de transporte.

El Centro de Entrenamiento Táctico Computarizado, fue inaugurado el 23 de Octubre de 1996.

Figura 14: *Logotipo del CETAC.*

Configuración General

El SETAC fue construido a base de la más alta tecnología computacional, transportable, flexible y modular, que mediante una combinación de medios humanos, elementos técnicos (computadoras) y una base de datos, permitió hacer viable el desarrollo del juego de guerra

El sistema cuenta con una base para efectuar la simulación de las diferentes situaciones de combate y de apoyo logístico y administrativo, con una serie de modelos computacionales (programas) que, como resultado de la interacción producida, entrega los resultados que sirven para estructurar las situaciones que afectarán a cada unidad.

Los modelos empleados son:

- Terreno
- Estructura de Unidades
- Movimiento o Desplazamiento
- Tiempo Atmosférico
- Luminosidad
- Búsqueda de Informaciones
- Reconocimiento o Exploración Aérea
- Potencia de Combate de Armas y Unidades
- Logística
- Personal
- Apoyo Administrativo
- Trabajos de Ingenieros.

Estos modelos permiten que los participantes se entrenen realizando íntegramente y de acuerdo a su doctrina, las dos etapas de la conducción militar (preparación y ejecución); permite además la simulación de diversas situaciones y en diversos escenarios; esta capacidad de programación se logra mediante la elección de la información adecuada, desde una base de datos acumulada o ingresada durante la etapa de preparación previa al inicio del juego.

Los comandantes y asesores utilizan el computador para el proceso de apreciación de la situación, de las distintas funciones primarias del mando en las siguientes actividades:

- Cálculos o previsiones administrativas y logísticas
- Estudio del terreno
- Cómputo de potenciales
- Información sobre tiempo atmosférico
- Situación, características, capacidades y limitaciones de las unidades en juego

Los datos incorporados se emplean en nuevos y posteriores ejercicios, por la capacidad del sistema de ir almacenando la información que se ingrese. Lo anterior permite reeditar algunas situaciones o resoluciones que se necesita resaltar durante la crítica o clases posteriores, a fin de permitir retroalimentar el proceso de enseñanza-aprendizaje y mejorar el desarrollo de futuros juegos de guerra, como producto de las experiencias obtenidas.

Características Específicas

Las características técnicas de los medios computacionales le ofrecen al usuario las siguientes facilidades que le permiten desarrollar sus actividades con mayor realismo y eficiencia:

- Visión de todo o parte del terreno digitalizado, en dos o tres dimensiones
- Impresión de lo mostrado en pantalla
- "Zoom" sobre cualquier parte del terreno en dos o tres dimensiones
- Visiones distintas del mismo sector del terreno (de acuerdo al punto geográfico en el que se esté observando)
- Introducción de gráficos adicionales, unidades y otros
- Campo de vista o mapa de visibilidad
- Perfiles
- Cálculos de distancia
- Representación gráfica de las unidades y de los movimientos de éstas
- Determinación de pendientes entre dos puntos en particular y de sectores en Términos gráficos y numéricos
- Colores de vegetación
- Generación de efectos climatológicos
- Simulación de trayectorias y alcances balísticos
- Transitabilidad de caminos
- Introducción de gráficos o características puntuales digitalizadas desde fotografías o mapas
- Edita el contenido de la base de datos.

Existe además un enlace de comunicación entre los diferentes niveles del juego, a través del correo electrónico propio de la red local, complementados, por supuesto, por los medios de telecomunicaciones dispuestos para estos fines, entre los comandantes y los cuarteles generales, para simular la red y la comunicación mediante documentos escritos.

En conjunto con todas las características enunciadas, los comandantes y asesores pueden acceder a ejecutar y obtener computacionalmente una amplia gama de actividades relacionadas a cada función primaria del mando, desde cálculo de bajas, reemplazos, prisioneros de guerra, pasando por aspectos de M.E.T.T., capacidades de apoyo de las unidades logísticas, procedimientos de abastecimientos, etc., hasta las acciones tácticas fundamentales y complementarias, incluyendo efectos de enfrentamiento entre unidades.

De este modo, en el transcurso del juego, se van planteando situaciones tácticas que los participantes deben afrontar para alcanzar el objetivo que les ha sido encomendado, esto es, básicamente, reconquistar el territorio ocupado por el enemigo.

En el área técnica se realizaron modificaciones al SETAC entre ellos:

1. Cambios de parámetros para:

    a. Tiempos de ejecución en el trabajo de ingenieros (sembrado y levantamiento de campos minados, alambradas, construcción/relleno de fosas antitanque y cráteres, etc.)
    b. Tonelaje máximo de soporte para los Puentes Baileys y Normales.
    c. Área destructiva para la munición de artillería, Fuerza Aérea.
    d. Velocidad de marcha de las unidades mecanizadas al entrar en combate.
    e. Velocidades de marcha motorizada en los diferentes tipos de caminos.
    f. Distancia para poder abastecer unidades (5 kms.) sin requerir marchas de aproximación.

g. Distancia para evacuar/remplazar personal (5 kms) sin requerir marchas de aproximación.
h. Consumos de munición para todo tipo de armamento

2. Posibilidad de apertura de brechas a los obstáculos con artillería
3. Posibilidad para hacer fuego directo e indirecto sobre unidades propias.
4. Posibilidad de que unidades propias sean afectadas con campos minados propios.
5. Formatos de reportes de situación de las unidades.
6. Capacidad de generar reportes con unidades no operacionales.
7. Eliminación del proceso automático del PAS (evacuación de HEA´s).
8. Implementación de un procedimiento general y detallado para las evacuaciones y Reemplazos.
9. Implementación del proceso de sepultura dentro del modelo de ingenieros.
10. Capacidad de reorganizar unidades al poder sustraer abastecimientos y personal de unidades no operacionales.
11. Implementación del Track de vuelo para el transporte aéreo.
12. Capacidad para lanzar desde el aire a unidades de paracaidistas.
13. Poder colocar unidades en Defensa Fortificada mediante la Terminal de la Dirección.
14. Descontar personal, bastimentos, munición, equipo, armamento, vehículos, etc.; de acuerdo a un porcentaje.

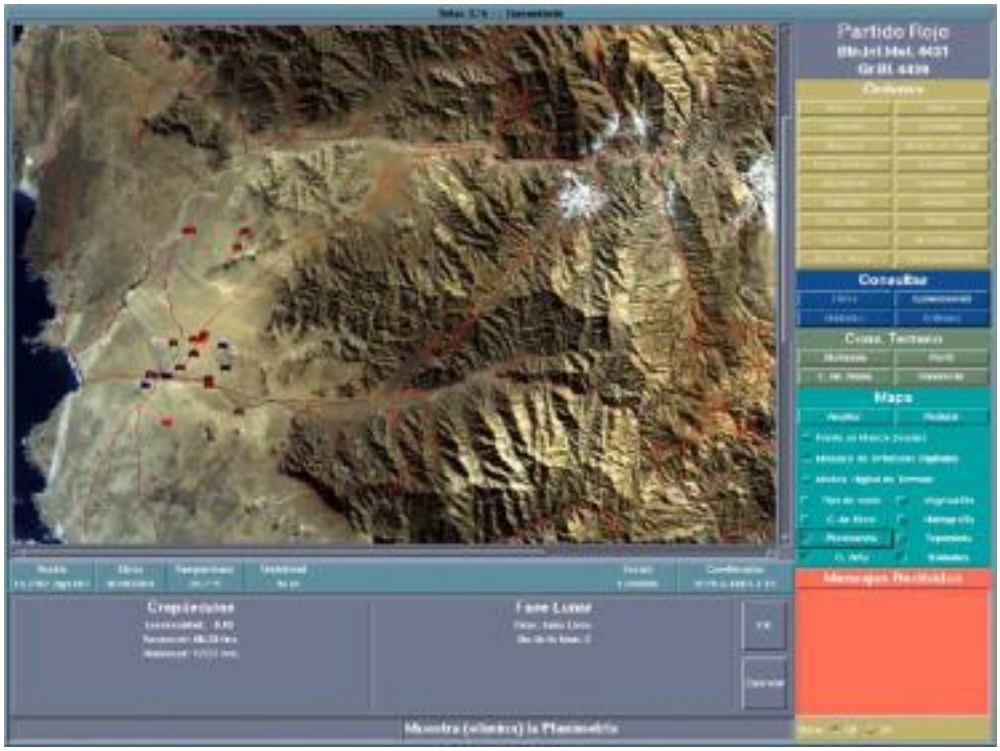


Figura 15: *Interfaz del Comandante del SETAC.*

En el año de 1999 se desarrolló el sistema HERMES 1.0 el cual permite que durante el desarrollo de una simulación presente datos estadísticos de gastos de municiones por armas y abastecimientos por periodos de una hora; asimismo al final de la simulación presentar un dato estadístico general de todos los gastos que se hayan tenido y permitir al comandante de la unidad tener un estimado de los gastos que en la realidad podría tener.

Durante el año 2000 se desarrolla el sistema de control de simulaciones el cual permite llevar un registro estadístico del personal que participa en las simulaciones especificando su grado, nombre, unidad y cargo que desempeño en el ejercicio; también fue establecido el sistema de Evaluación en el Campo de Batalla el cual permite por medio de lista de chequeo evaluar las decisiones tomadas por los comandantes y miembros de los Estados Mayores que participan en el juego.

Asimismo se entregó un Graficador militar que puede ser manejado en un ambiente de Windows con el propósito de que las unidades tengan un previo adiestramiento antes de realizar una simulación.

En ese año se adquieren algunos programas informáticos entre ellos el programa ARCVIEW 3.2 y se inicia el proceso de investigación en el uso de dicho programa, creando mapas temáticos. Asimismo en ese mismo año se obtiene el terreno digitalizado de El Salvador.

En el año 2001 se inicia el desarrollo del nuevo SETAC V. 2000 el cual trabaja en la plataforma de LINUX y adaptado a redes de computadoras personales; a la vez se incorpora un nuevo modelo dentro del sistema "modelo de Emergencia Nacional" el cual permite al usuario desarrollar todas aquellas actividades después de un evento (un desastre nacional).

Ese año a pesar de los terremotos de los meses de enero y febrero y con el propósito de ir actualizando y modernizando el CETAC, se desarrolló el Graficador Militar, dentro de la plataforma de LINUX, iniciando con la conversión de todo el terreno digital.

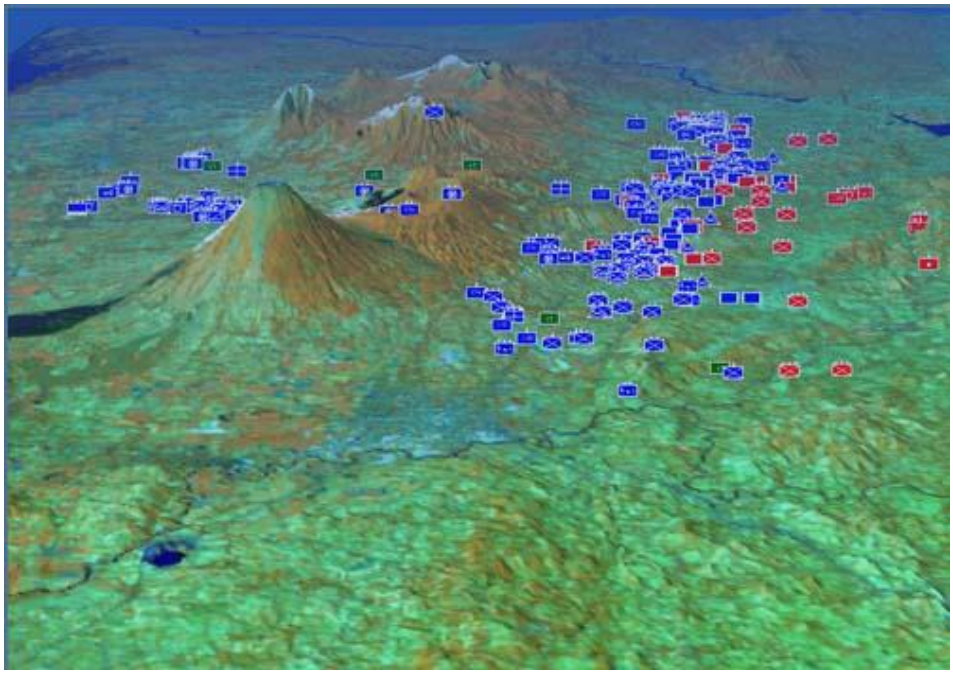


Figura 16: *Edición de cartografía temática para las unidades militares*

Se establece una visión más clara del CETAC la cual consiste en hacer del CETAC un Centro con un nivel de desarrollo constante, siempre capacitado y listo para cumplir con éxito su misión, empleando eficaz y eficientemente sus medios, asimismo el CETAC es considerado una herramienta que genera las condiciones de prueba cercanas a la realidad, empleando recursos humanos y materiales en un ambiente virtual, con el objeto de mejorar las capacidades de toma de decisiones bajo presión (Manejo de crisis).

A partir del año 2000, el Ejército de El Salvador realizó su primera experiencia en la utilización de recursos informáticos aplicados a la simulación de situaciones de guerra. El 18 de julio de 2005, el Ejército de El Salvador realizó una nueva experiencia con el sistema, en la que participaron oficiales de las fuerzas armadas de Bolivia, Guatemala, Honduras, México, Nicaragua, Perú, República Dominicana y El Salvador.

Figura 17: *Ejercicio de empleo del Sistema SETAC 2000*

EJERCITO DE ARGENTINA

Figura 18: *Escudo del Ejército Argentino*

El Ejército Argentino introdujo cambios organizacionales, doctrinarios, metodológicos y tecnológicos, necesarios y suficientes, para producir el cambio educacional y cultural que les permita incorporar la simulación como metodología aplicada al proceso de capacitación de alumnos en ámbitos académicos y al adiestramiento de Estados Mayores de Comandos de Grandes Unidades (GGUU),

Planas Mayores de Jefaturas de Unidades, Conjuntos y Fracciones, así como desarrollar el conocimiento para transferir a la sociedad los beneficios de las nuevas tecnologías aplicados al entrenamiento de organizaciones complejas y / o multidisciplinarias.

El resultado de este esfuerzo iniciado en el año 1999, concretado en el año 2001 e impulsado a partir del año 2003 ha permitido producir el desarrollo del conocimiento necesario para concretar las Metas fijadas mediante un Plan de Proyecto denominado "Batalla Virtual", en el cual además de desarrollar la visión del primer sistema, definió el alcance de la línea de investigación y del proyecto completo, estableciendo 2 objetivos:

- Construir un **centro de desarrollo de software** como *pilar tecnológico informático del Ejército en materia de simulación*, fundamental para investigar, incrementar y preservar el conocimiento que se desarrollara en el marco de trabajo definido por el proyecto.
- Construir un **centro de adiestramiento** como *sustento metodológico del Ejército en materia de adiestramiento operacional con simulación*, necesario para desarrollar nuevas metodologías que contemplaran las propias características, doctrina y necesidades de la Fuerza.

El primer objetivo se concretó en el año 2001, inaugurándose el CENTRO DE INVESTIGACIÓN Y DESARROLLO DE SOFTWARE (CIDESO), oportunidad en la cual se presentó el primer PROTOTIPO del SISTEMA BATALLA VIRTUAL con el cual el Proyecto homónimo fue aprobado por el Jefe del Estado Mayor General del Ejército.

En el ámbito ACADEMICO: Se doto a los Institutos de Formación, Capacitación y Perfeccionamiento, de los sistemas de simulación necesarios para la capacitación del personal de alumnos y docentes en la conducción de las operaciones con soporte digital, por lo que se crearon los siguientes proyectos:

- Batalla Virtual Académico (BVA)
- Batalla Virtual Logístico (BVL)
- Batalla Virtual Estratégico (BVE).

Figura 19: *Ejercicio de empleo del Sistema BVA*

En el ámbito OPERACIONAL: Se proporcionó a los Comandos de GGUU y Unidades dependientes, de sistemas de adiestramiento modernos que les permitan adiestrarse tanto en forma reunida como a distancia, explotando la infraestructura tecnológica existente, por lo que se crearon los siguientes proyectos:

- Batalla Virtual Brigada (BVB)

- Batalla Virtual Unidad (BVU)
- Portal Comunitario de Adiestramiento. (*Actualmente en desarrollo).*

A la vez sentar las bases para emprender el proyecto de un Comando de Brigada que pueda operar con soporte digital, por lo que se están creando los siguientes proyectos:

- Herramientas de Estado Mayor / Utilitarios de Adiestramiento (*Actualmente en desarrollo).*
- Vincular los sistemas de adiestramiento con los sistemas de comando y control.
- Herramientas de Estado Mayor / Utilitarios de Adiestramiento (*Actualmente en desarrollo).*

Figura 20: *Ejercicio de empleo del Sistema BVB*

APOYO A LA COMUNIDAD: extender los sistemas para la ejercitación en casos de desastres naturales y catástrofes, por lo que se crearon los siguientes proyectos:

- **EMERCAT y Gestión y Control de Crisis.** (*Actualmente en desarrollo).*
- **OPERACIONES DE PAZ:** permitir el entrenamiento de fracciones y planas mayores que vayan a desempeñar Misiones de Paz.
- **SIMUPAZ** (*Actualmente en desarrollo).*
- **ACCION CONJUNTA:** desarrollar sistemas compatibles con los existentes en las otras FFAA.
- **Estándares Militares para la Interoperabilidad de Sistemas Basados en Computadoras.** *Presentado, se encuentra en etapa de análisis.*

En mayo del año 2007 se alcanzó el segundo objetivo, inaugurándose el CENTRO DE ADIESTRAMIENTO PARA EL EJERCICO DE LA CONDUCCION Y LA EXPERIMENTACION DE DOCTRINA (CAECED), dependiente de la Dirección de Educación Operacional y Doctrina / Comando de Educación y Doctrina.

3.4 La función del software para juegos de guerra

El juego de guerra permite desarrollar el arte operacional de la guerra, proveyendo las herramientas esenciales a los estados mayores, planas mayores y a los comandantes para que comiencen a explorar su entendimiento de las opciones de los adversarios al igual que las

ramificaciones y secuelas, modernizando algunos de los procesos de preparación del juego de guerra.

Bajo la estructura de los diseños actuales de los juegos de guerra, los participantes a menudo invierten la mayor parte de su tiempo decidiendo cómo desplazar, emplear y sostener a las tropas, estos no son los objetivos principales del juego de guerra, pero definitivamente forman parte de su asignación de tareas.

Los requisitos para elaborar un plan de campaña podría ser el propósito principal de los juegos de guerra que se llevan a cabo en el ambiente operacional, donde la experiencia del aprendizaje es superior. En cambio, las sesiones informativas de los equipos que controlan el juego con relación a algunos de los asuntos más fundamentales del empleo de fuerzas y de la planificación de la campaña, con base en el aporte limitado de los participantes principales, *podrían liberar a los participantes para poder concentrarse principalmente en los asuntos claves identificados por el anfitrión del juego sin que la colaboración del participante en el juego se vea afectada.*

Los juegos de guerra se llevan a cabo para desarrollar un mejor entendimiento de unos cuantos asuntos claves. Librar a los participantes en los Juegos de temas que tienen que ver con el desplazamiento y el empleo de las fuerzas lo que les permite concentrarse en temas de política decisivos.

La función principal del juego de guerra *radica en la exploración del arte operacional para elaborar ideas en cuanto a cuestiones y conceptos de las operaciones actuales y futuras*, ***no radica en intentar***

predecir o probar resultados, como para justificaciones presupuestarias o de estructura de fuerza.

La experiencia muestra que los juegos pueden revelar campos fructíferos para más tarde analizarlos a fondo. Los problemas que surgen de los juegos de guerra pueden servir como un mapa para enfocar las iniciativas de los laboratorios de batalla, los centros de guerra, los proyectos de experimentaciones conjuntas y las iniciativas de las fuerzas expedicionarias. Este mapa fomentaría una iniciativa de juego de guerra coherente, eficaz y económica. De la misma manera que los aviadores han aprendido las maneras más eficaces de aplicar el poder aeroespacial, nosotros también debemos explorar los usos más productivos del juego de guerra.

4. LA INGENIERÍA DE SOFTWARE

4.1. Resumen

El software es un sistema informático compuesto por un conjunto de instrucciones que, cuando se ejecutan en un dispositivo físico (hardware) produce resultados de acuerdo con los objetivos y función principal predeterminada. Dicho conjunto de instrucciones está organizado en estructuras de datos[5] que permiten la manipulación de la información. Según PRESSMAN el ingenio del "desarrollador" (o del equipo de desarrollo) es el único aspecto que limita el diseño de una estructura de datos.

La estructura de datos asociada al tipo de hardware usado influye decisivamente en el desempeño del software. Así pues, durante el diseño y programación de un software es importante seleccionar el tipo de estructura de datos más adecuado respecto al tipo de procesamiento previsto.

En el desarrollo de sistemas de software es de suma importancia el seguir alguna especificación que permita a los desarrolladores el tener una disciplina que haga que todas las etapas del desarrollo del sistema, desde el levantamiento de requerimientos hasta las pruebas finales del sistema, sean no solo más coherentes sino también más formales.

[5] *La estructura de datos* es una representación de la relación lógica entre elementos de datos individuales, determina la organización, los métodos de acceso, el grado de asociatividad y las alternativas de procesamiento de informaciones.

Usando el concepto abstracto del software (parte lógica del sistema "computacional"[6]) como punto de partida, se presentan algunas características que distinguen el software del hardware y ofrecen una buena comprensión de su significado:

1. Su desarrollo o proyecto no se basa en manufactura clásica sino que en la Ingeniería.

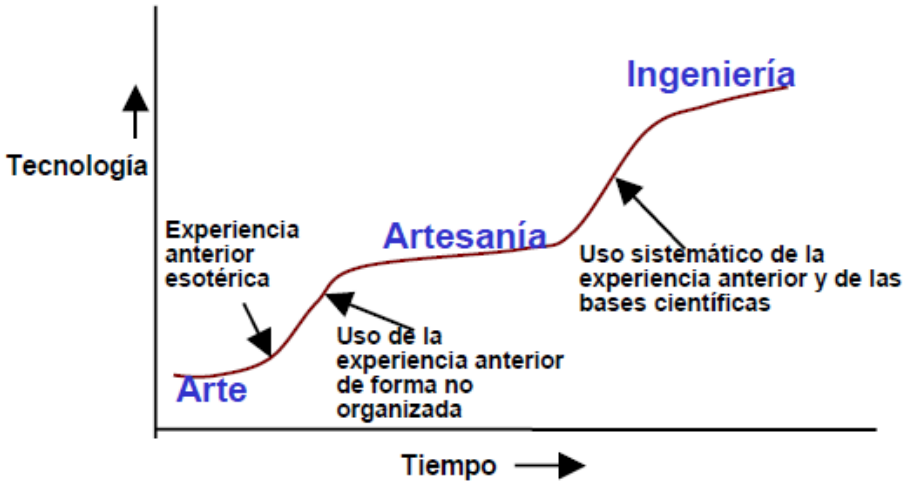


Figura 21: *Modelo de desarrollo de la Ingeniería*

2. El software no se deteriora con el tiempo, la suciedad, las temperaturas altas y la vibración, como es el caso del hardware.

3. La evolución del hardware impone cambios de paradigmas en la filosofía de diseño del software, de manera que se identifica la implementación de nuevas versiones con el propósito de adaptarlas a las nuevas tecnologías.

[6] *Sistema computacional:* Entorno de trabajo, estudio, entretenimiento u otra actividad, el cual está compuesto por tres elementos interrelacionados: el software (parte lógica), el hardware (parte física) y el peopleware (parte humana).

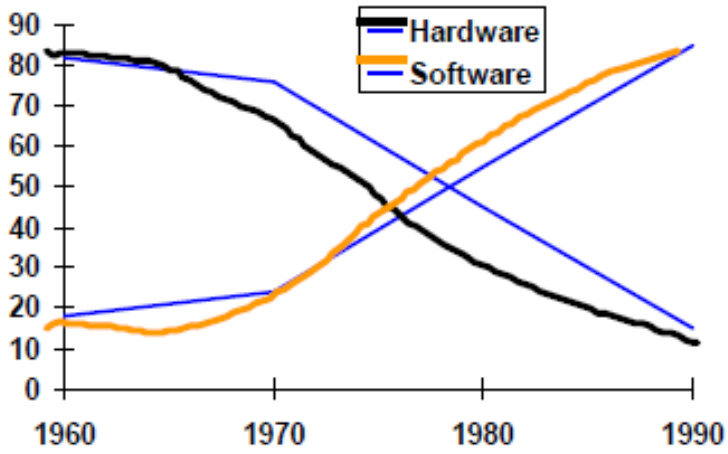


Figura 22: *Costos de la evolución del Software y del Hardware*

4. El software se desarrolla a la medida, sin embargo en la actualidad se observa una tendencia de desarrollo que consiste en diseñar e implementar componentes de software que puedan ser usados según la necesidad.

El desarrollo del software es un continuo desafío no solo debido a los problemas conceptuales y de diseño sino también por los problemas técnicos y económicos que se generan durante el proyecto de software.

Las propuestas académicas y de la industria para reducción de estos problemas como por ejemplo las herramientas CASE[7] y las metodologías de desarrollo, han permitido que el software evolucionara provocando, a partir de las necesidades creadas, la aparición de una nueva disciplina: *La Ingeniería del software.*

[7] *La sigla CASE* es acrónimo de *Computer Aided Software Engineering*. Es un sistema compuesto por software, hardware y una base de datos que ofrece soporte a la ingeniería del software, permitiendo el análisis estructurado, la implementación y test de sistemas informáticos.

La ingeniería del software abarca un conjunto de tres elementos clave: *métodos, herramientas y procedimientos*, que faciliten al gestor el control el proceso de desarrollo y suministren a los implementadores bases para construir de forma productiva software de alta calidad.

Los métodos indican cómo construir técnicamente el software, y abarcan una amplia serie de tareas que incluyen la planificación y estimación de proyectos, el análisis de requisitos, el diseño de estructuras de datos, programas y procedimientos, la codificación, las pruebas y el mantenimiento. Los métodos introducen frecuentemente una notación específica para la tarea en cuestión y una serie de criterios de calidad.

Las herramientas proporcionan un soporte automático o semiautomático para utilizar los métodos. Existen herramientas automatizadas para cada una de las fases vistas anteriormente, y sistemas que integran las herramientas de cada fase de forma que sirven para todo el proceso de desarrollo. Estas herramientas se denominan CASE.

Los procedimientos definen la secuencia en que se aplican los métodos, los documentos que se requieren, los controles que permiten asegurar la calidad y las directrices que permiten a los gestores evaluar los progresos.

4.2. Necesidad de una metodología para desarrollo de software

Cuando surgió la necesidad de adaptar los sistemas informáticos a las exigencias del mercado, el programador realizaba un relevamiento de las solicitudes de quien necesitaba cierto programa o producto

software y con aquellos requerimientos bajo el brazo comenzaba la dura tarea de codificar.

Esta tarea no estaba administrada, supervisada o gestionada de ningún modo, por lo que se iba corrigiendo a medida que surgían los errores, tanto los lógicos provenientes de la codificación, como los de requerimientos solicitados por el cliente o usuario final.

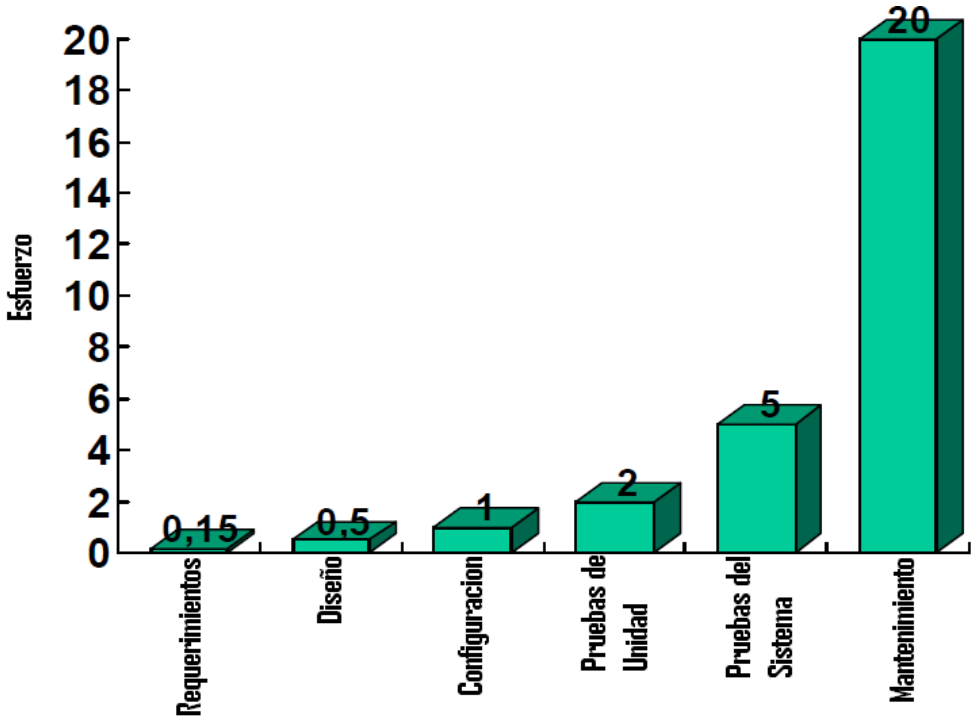


Figura 23: *Esfuerzo para reparar el Software*
(Según la etapa en la que se detecta el defecto)

En la década de 1970 los programas fueron creciendo en complejidad, por lo que la técnica de ***code & fix***[8] que se utilizaba termino quedando obsoleta. Al no seguir normas para el proyecto, el cliente o usuario solo impartían especificaciones muy generales del producto final; se programaba, se corregía y se volvía a programar sobre la misma marcha del proyecto.

[8] ***code & fix:*** Modelo básico utilizado en los inicios del desarrollo de software, contiene dos pasos: escribir código y corregir problemas en el código.

El ciclo de vida de este tipo de proyectos finalizaba cuando se satisfacían las especificaciones, no solo las primeras por las cuales nació la necesidad del programa, sino también todas aquellas que fueron surgiendo sobre la marcha.

Esta técnica tenía la ventaja de no gastar recursos en análisis, planificación, gestión de recursos, documentación, etc., además era muy cómoda y muchas veces recomendable cuando el proyecto es muy pequeño y era llevado adelante por uno o dos programadores. Por otro lado, cuando el sistema no era pequeño o era más complejo de lo previsto, traía desventajas en lo que se refiere a costos de recursos, que siempre eran mayores de lo previsto; aumentaba el tiempo de desarrollo y la calidad del código era bastante dudosa.

La construcción del software y sus resultados han sido históricamente cuestionados debido a los problemas asociados, entre ellos podemos destacar los siguientes:

- Los sistemas no responden a las expectativas de los usuarios.
- Los programas “fallan” con cierta frecuencia.
- Los costos del software son difíciles de prever y normalmente superan las estimaciones.
- La modificación del software es una tarea difícil y costosa.
- El software se suele presentar fuera del plazo establecido y con menos prestaciones de las consideradas inicialmente.
- Normalmente, es difícil cambiar de entorno hardware usando el mismo software.
- El aprovechamiento óptimo de los recursos (personas, tiempo, dinero, herramientas, etc.) no suele cumplirse.

Según el Centro Experimental de Ingeniería de Software (CEIS)[9], el estudio de mercado *The Chaos Report* realizado por Standish Group Internactional[10], concluyó que sólo un 16% de los proyectos de software son exitosos (terminan dentro de plazos y costos y cumplen los requerimientos acordados). Otro 53% sobrepasa costos y plazos y cumple parcialmente los requerimientos. El resto ni siquiera llega al término. Algunas deficiencias comunes en el desarrollo de software son:

- Escasa o tardía validación con el cliente.
- Inadecuada gestión de los requisitos.
- No existe medición del proceso ni registro de datos históricos.
- Estimaciones imprevistas de plazos y costos.
- Excesiva e irracional presión en los plazos.
- Escaso o deficiente control en el progreso del proceso de desarrollo.
- No se hace gestión de riesgos formalmente.
- No se realiza un proceso formal de pruebas.
- No se realizan revisiones técnicas formales e inspecciones de código.

El desarrollar software sin el uso de metodologías representa un enfoque "artesanal" de desarrollo que hoy en día sigue siendo una práctica muy común. Las metodologías nos indican cómo hacer más eficiente el desarrollo de sistemas de información.

4.2.1 Definición de metodología

La metodología para el desarrollo de software es un modo sistemático de realizar, gestionar y administrar un proyecto

[9] http://www.ceis.cl/Gestacion/Gestacion.htm
[10] http://standishgroup.com/

para llevarlo a cabo con ***altas posibilidades de éxito***. Esta sistematización nos indica como dividiremos un gran proyecto en módulos más pequeños llamados etapas y las acciones que corresponden a cada una de ellas, nos ayuda a definir entradas y salidas para cada una de las etapas y sobre todo normaliza el modo en que administraremos el proyecto.

Entonces: "*una metodología para el desarrollo de software son los procesos a seguir sistemáticamente para idear, implementar y mantener un producto software desde que surge la necesidad del producto hasta que cumplimos el objetivo para el cual fue creado*".

Una **metodología** puede seguir uno o varios modelos del Ciclo de Vida y nos indica ***cómo*** hay que obtener los distintos productos parciales y finales. Un **Ciclo de Vida nos** indica qué obtener a lo largo del desarrollo, pero no ***cómo*** hacerlo

4.2.2 Características y clasificación de las metodologías

La comparación y/o clasificación de metodologías no es una tarea sencilla debido a la diversidad de propuestas y diferencias en el grado de detalle, información disponible y alcance de cada una de ellas. Las metodologías de desarrollo de sistemas deben definir: objetivos, fases, tareas, productos y responsables, necesarios para la correcta realización del proceso y su seguimiento. Los principales objetivos de una metodología de desarrollo son:

- Asegurar la uniformidad y calidad tanto del desarrollo como del sistema en sí.

- Satisfacer las necesidades de los usuarios del sistema.
- Conseguir un mayor nivel de rendimiento y eficiencia del personal asignado al desarrollo.
- Ajustarse a los plazos y costes previstos en la planificación.
- Generar de forma adecuada la documentación asociada a los sistemas.
- Facilitar el mantenimiento posterior de los sistemas.

A grandes rasgos, si tomamos como criterio las notaciones utilizadas para especificar artefactos producidos en actividades de análisis y diseño, podemos clasificar las metodologías en dos grupos: ***Metodologías Estructuradas*** y ***Metodologías Orientadas a Objetos***.

Otras metodologías, denominadas ***Metodologías Ágiles***, están más orientadas a la generación de código con ciclos muy cortos de desarrollo, se dirigen a equipos de desarrollo pequeños, hacen especial hincapié en aspectos humanos asociados al trabajo en equipo e involucran activamente al cliente en el proceso.

4.2.3 Metodologías estructuradas

Muchos especialistas en sistemas de información reconocen la dificultad de comprender de manera completa sistemas grandes y complejos. El método de desarrollo del análisis estructurado tiene como finalidad superar ésa dificultad por medio de la *división del sistema en componentes* y la *construcción de un modelo del sistema*.

El método incorpora elementos tanto de ***análisis*** como de ***diseño***.

¿Qué es el análisis estructurado?

El análisis estructurado especifica lo que se requiere que haga el sistema o la aplicación. No se establece cómo se cumplirán los requerimientos o la forma en que se implantará la aplicación. Más bien permite que las personas observen los elementos lógicos (lo que hará el sistema) separados de los componentes físicos (computadoras, terminales, sistemas de almacenamiento, etc.) Después de esto se puede desarrollar un diseño físico eficiente para la situación donde será utilizado.

Elementos del análisis estructurado:

1. ***Símbolos gráficos:*** El análisis estructurado utiliza símbolos, o íconos, para crear un modelo gráfico del sistema. Los modelos de este tipo muestran los detalles del sistema. Si se seleccionan los símbolos y notación correctos entonces casi cualquier persona puede seguir la forma en que los componentes se acomodarán entre sí para formar el sistema.

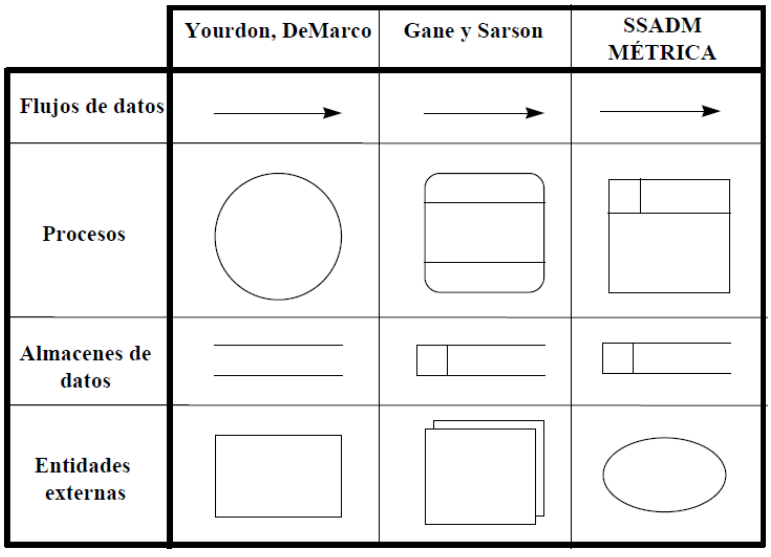


Figura 24: *Símbolos del análisis estructurado*

2. ***Diagramas de flujo de datos (DFD):*** Muestra las fuentes y destinos de los datos, identifica y da nombre a los procesos que se llevan a cabo, identifica y da nombre a los grupos de datos que relacionan una función con otra y señala los almacenes de datos a los que se tiene acceso.

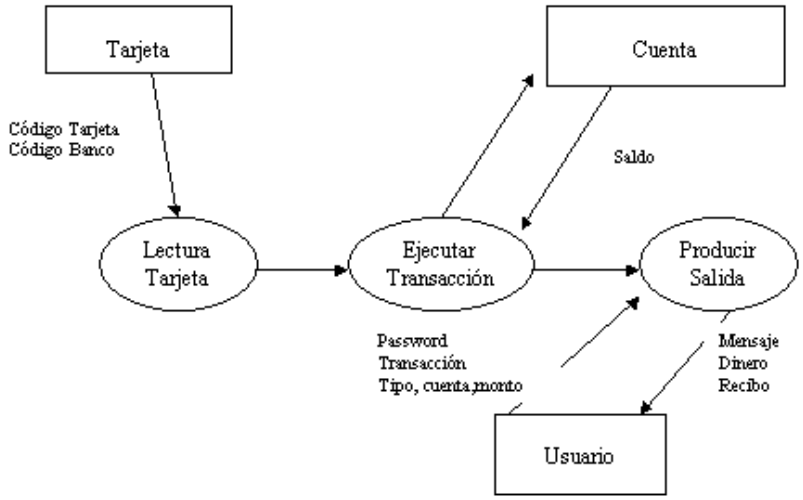


Figura 25: *DFD de un cajero automático*

3. ***Diccionario centralizado de datos:*** Todas las definiciones de los elementos en el sistema (flujo de datos, procesos y almacenes de datos) se describen en forma detallada en el diccionario de datos.

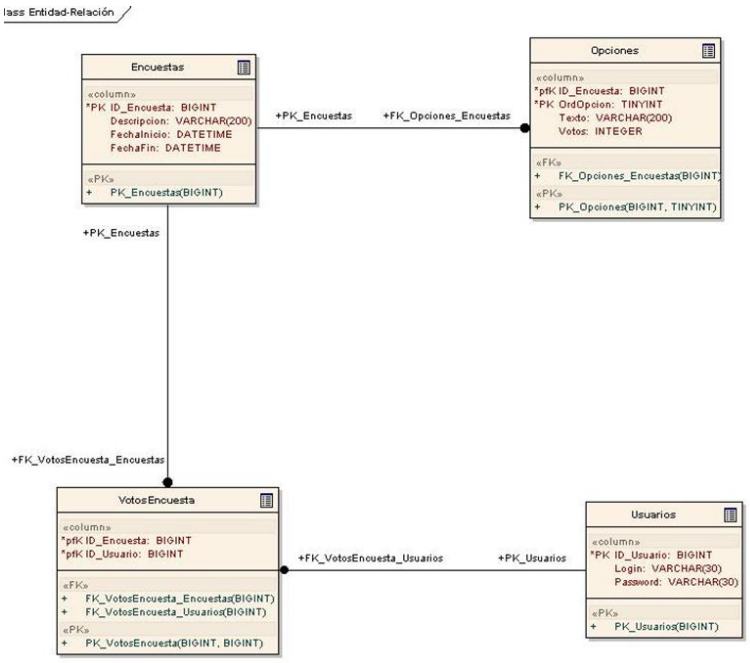


Figura 26: *Diccionario de Datos*

¿Qué es el diseño estructurado?

Es una técnica orientada a procesos utilizada para fragmentar un programa grande en un conjunto jerarquizado de módulos y obtener un programa informático más fácil de implantar y mantener (cambiar).

La meta del diseño estructurado es crear programas formados por módulos independientes unos de otros desde el punto de vista funcional. *Se enfoca en el desarrollo de especificaciones del software*.

El diseño estructurado *es una técnica específica para el diseño de programas* y no un método de diseño de comprensión. Esta técnica conduce a la especificación de módulos de programa que son funcionalmente independientes.

La herramienta fundamental del diseño estructurado es el ***diagrama estructurado***, los cuales son de naturaleza gráfica y evitan cualquier referencia relacionada con el hardware o detalles físicos. Su finalidad no es mostrar la lógica de los programas.

Los diagramas estructurados describen la interacción entre módulos independientes junto con los datos que un módulo pasa a otro cuando interacciona con él. Estas especificaciones funcionales para los módulos se proporcionan a los programadores antes que dé comienzo la fase de escritura de código.

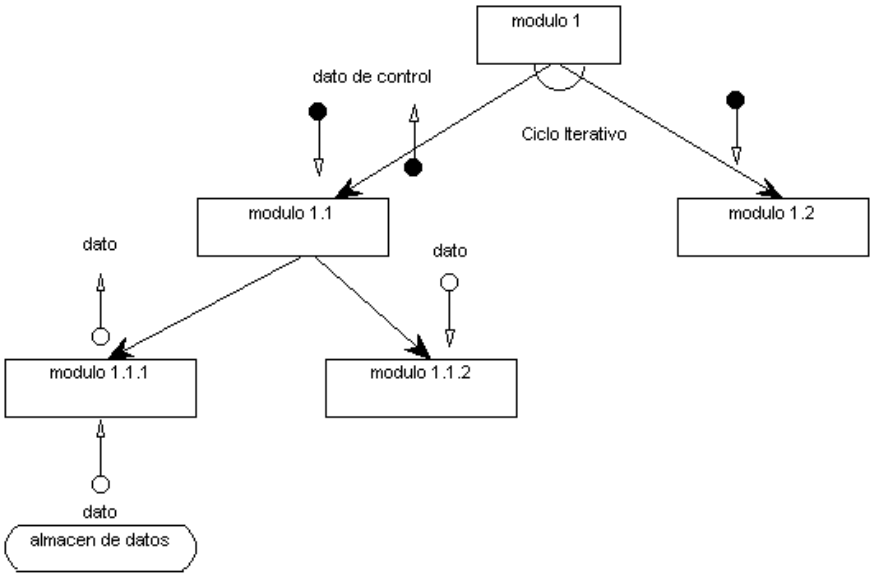


Figura 27: *Diagrama estructurado*

Entre las ventajas que ofrece el diseño estructurado se pueden mencionar:

1. ***Que los programas que se desglosan conforme al diseño estructurado son más fáciles de leer y probar por equipos de programadores múltiples***, ya que como las interfaces entre los módulos también estarán bien definidas y limitadas por las reglas, los módulos cuya prueba sea correcta deberían funcionar igual de bien cuando se integren en el sistema global.

2. ***Los sistemas y los programas desarrollados con diseño estructurado son más fáciles de mantene***r. Además que este busca disminuir intencionalmente el efecto rizo al reducir al mínimo las conexiones y la dependencia entre los módulos.

Estas metodologías funcionaban muy bien con los lenguajes de programación estructurados, como por ejemplo el COBOL, C, Pascal, FORTRAN y Modula 2. Son particularmente apropiadas en proyectos que utilizan para la implementación lenguajes de 3ra y 4ta generación.

Ejemplos de metodologías estructuradas de ámbito gubernamental: MERISE[11] (Francia), MÉTRICA[12] (España), SSADM[13] (Reino Unido). Ejemplos de propuestas de métodos estructurados en el ámbito académico: Gane & Sarson[14], Ward & Mellor[15], Yourdon & DeMarco[16] e Information Engineering[17].

[11] http://perso.club-internet.fr/brouardf/SGBDRmerise.htm (7.5.2002)
[12] http://www.map.es/csi/metrica3/ (7.5.2003)
[13] http://www.comp.glam.ac.uk/pages/staff/tdhutchings/chapter4.html (7.5.2003)
[14] http://portal.newman.wa.edu.au/technology/12infsys/html/dfdnotes.doc (29.8.2003)
[15] http://www.yourdon.com/books/coolbooks/notes/wardmellor.html (29.8.2003)
[16] http://wombat.doc.ic.ac.uk/foldoc/foldoc.cgi?Yourdon%2FDemarco (29.8.2003)
[17] http://gantthead.com/Gantthead/process/processMain/1,1289,2-12009-2,00.html

4.2.4 Metodologías orientadas a objetos

Las metodologías orientadas a objetos han derivado de las metodologías anteriores a éste, han evolucionado para ayudar a los desarrolladores a explotar el poder de los lenguajes de programación basados en objetos y orientados a objetos, utilizando las clases y objetos como bloques de construcción básicos.

Actualmente el modelo de objetos ha sido influenciado por un número de factores no sólo de la *Programación Orientada a Objetos*, *POO* (*Object Oriented Programming*, *OOP* por sus siglas en inglés). Además, el modelo de objetos ha probado ser un concepto uniforme en las ciencias de la computación, aplicable no sólo a los lenguajes de programación sino también al diseño de interfaces de usuario, bases de datos y arquitectura de computadoras por completo.

La razón de ello es, simplemente, que una orientación a objetos nos ayuda a hacer frente a la inherente complejidad de muchos tipos de sistemas.

Objeto: Se define a un objeto como "*una entidad tangible que muestra alguna conducta bien definida*". Un objeto "*es cualquier cosa, real o abstracta, acerca de la cual almacenamos datos y los métodos que controlan dichos datos*".

(29.8.2003)

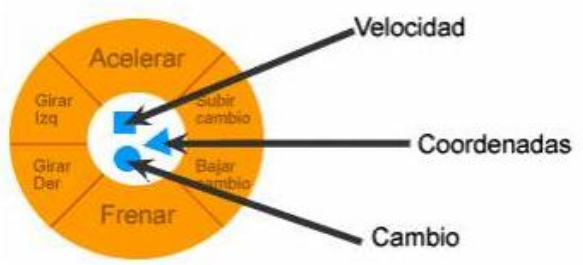


Figura 28: *Objeto auto*

Los objetos tienen una cierta "integridad" la cual no deberá ser violada. En particular, un objeto puede solamente cambiar estado, conducta, ser manipulado o estar en relación con otros objetos de manera apropiada a este objeto.

Clase: Una *clase* es una plantilla para objetos múltiples con características similares. Las clases comprenden todas esas características de un conjunto particular de objetos. Cuando se escribe un programa en lenguaje orientado a objetos, no se definen objetos verdaderos sino se definen clases de objetos.

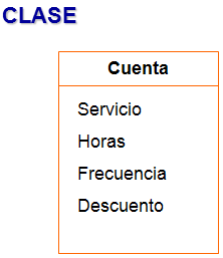


Figura 29: *Ejemplo de clase*

Instancia: Una *instancia* de una clase es otro término para un objeto real. Si la clase es la representación general de un objeto, una instancia es su representación concreta. A menudo se utiliza indistintamente la palabra objeto o instancia para referirse, precisamente, a un objeto.

- *Una instancia es un objeto creado a partir de una clase.*
- *La clase describe la estructura de la instancia (información y comportamiento), mientras que el estado actual de la instancia es definido por las operaciones ejecutadas.*

En los lenguajes orientados a objetos, cada clase está compuesta de dos cualidades: *atributos* (estado) y *métodos* (comportamiento o conducta).

- ***Los atributos*** son las características individuales que diferencian a un objeto de otro (ambos de la misma clase) y determinan la apariencia, estado u otras cualidades de ese objeto. Los atributos de un objeto incluyen información sobre su estado.

- ***Los métodos*** de una clase determinan el comportamiento o conducta que requiere esa clase para que sus instancias puedan cambiar su estado interno o cuando dichas instancias son llamadas para realizar algo por otra clase o instancia. El comportamiento es la única manera en que las instancias pueden hacerse algo a sí mismas o tener que hacerles algo. Los atributos se encuentran en la parte interna mientras que los métodos se encuentran en la parte externa del objeto.

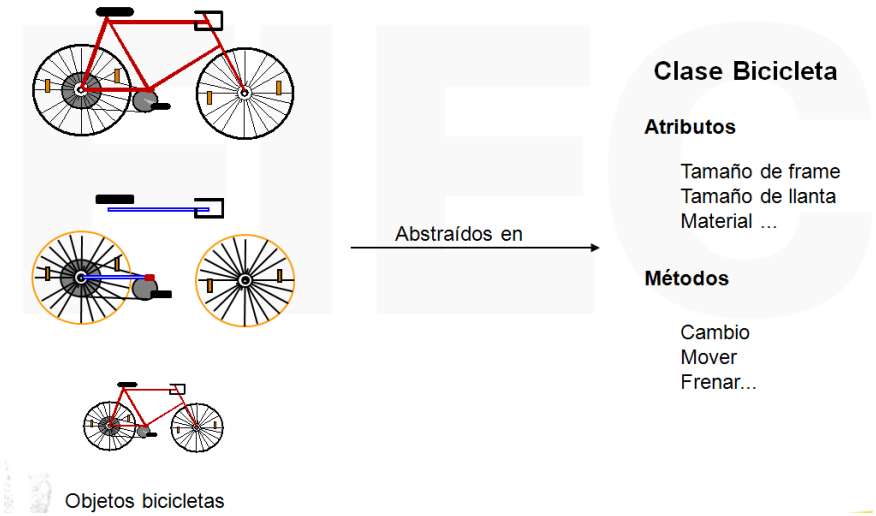


Figura 30: *Ejemplo de Instancia*

Para definir el comportamiento de un objeto, se crean métodos, los cuales tienen una apariencia y un comportamiento igual al de las funciones en otros lenguajes de programación. Los métodos no siempre afectan a un solo objeto; los objetos también se comunican entre sí mediante el uso de métodos. Una clase u objeto puede llamar métodos en otra clase u objeto para avisar sobre los cambios en el ambiente o para solicitarle a ese objeto que cambie su estado. Cualquier cosa que un objeto no sabe, o no puede hacer, es excluida del objeto.

Encapsulamiento: Al empaquetamiento de las variables de un objeto con la protección de sus métodos se le llama *encapsulamiento*. Típicamente, el encapsulamiento es utilizado para esconder detalles de la puesta en práctica no importantes de otros objetos. Entonces, los detalles de la puesta en práctica pueden cambiar en cualquier tiempo sin afectar otras partes del programa.

Esta imagen conceptual de un objeto —un núcleo de variables empaquetadas en una membrana protectora de métodos— es una representación ideal de un objeto y es el ideal por el que los diseñadores de sistemas orientados a objetos luchan. Sin embargo, no lo es todo: a menudo, por razones de eficiencia o la puesta en práctica, un objeto puede querer exponer algunas de sus variables o esconder algunos de sus métodos.

El encapsulamiento de variables y métodos en un componente de software es una idea poderosa que provee dos principales beneficios a los desarrolladores de software:

- ***Modularidad***, esto es, el código fuente de un objeto puede ser escrito, así como darle mantenimiento, independientemente del código fuente de otros objetos. Así mismo, un objeto puede ser transferido alrededor del sistema sin alterar su estado y conducta.

- ***Ocultamiento de la información***, es decir, un objeto tiene una "interfaz publica" que otros objetos pueden utilizar para comunicarse con él. Pero el objeto puede mantener información y métodos privados que pueden ser cambiados en cualquier tiempo sin afectar a los otros objetos que dependan de ello.

Los objetos proveen el beneficio de la modularidad y el ocultamiento de la información. Las clases proveen el beneficio de la reutilización. Los programadores de software utilizan la misma clase, y por lo tanto el mismo código, una y otra vez para crear muchos objetos.

Herencia: La herencia es un mecanismo poderoso con el cual se puede definir una clase en términos de otra clase; lo que significa que cuando se escribe una clase, sólo se tiene que especificar la diferencia de esa clase con otra, con lo cual, la herencia dará acceso automático a la información contenida en esa otra clase. Con la herencia, todas las clases están arregladas dentro de una jerarquía estricta.

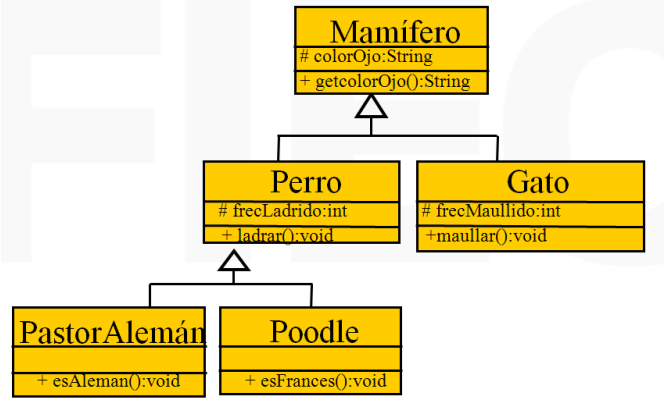


Figura 31: *Ejemplo de herencia*

De esta manera, se puede pensar en una jerarquía de clase como la definición de conceptos demasiado abstractos en lo alto de la jerarquía y esas ideas se convierten en algo más concreto conforme se desciende por la cadena de la superclase.

La herencia presenta los siguientes beneficios:

- Las subclases proveen conductas especializadas sobre la base de elementos comunes provistos por la superclase. A través del uso de herencia, los programadores pueden reutilizar el código de la superclase muchas veces.

- Los programadores pueden implementar superclases llamadas clases abstractas que definen conductas

"genéricas". Las superclases abstractas definen, y pueden implementar parcialmente, la conducta pero gran parte de la clase no está definida ni implementada. Otros programadores concluirán esos detalles con subclases especializadas.

Dentro de la orientación a objetos durante muchos años se plasmaron diferentes formas de realizar su ***análisis***, **diseño** y ***programación***, y aún esa variedad sigue presente.

- ***Análisis orientado a objetos***

  El Análisis Orientado a Objetos (AOO) es un método de análisis para examinar los requerimientos desde una perspectiva de clases y objetos, buscados en el vocabulario del dominio del problema, para poder representar la experiencia del usuario en la esquematización del requerimiento.

  Actualmente, el *Análisis Orientado a Objetos* (*AOO*) va progresando como método de análisis de requisitos por derecho propio y como complemento de otros métodos de análisis. En lugar de examinar un problema mediante el modelo clásico de entrada-proceso-salida (flujo de información) o mediante un modelo derivado exclusivamente de estructuras jerárquicas de información, el AOO introduce varios conceptos nuevos, estos conceptos nuevos le parecen inusuales a mucha gente, pero son bastante naturales.

- ***Diseño orientado a objetos***

  El Diseño Orientado a Objetos (DOO) es un método de diseño para comprender el proceso de descomposición y notación orientado a objetos, obteniendo el modelo lógico (estructuras de clases y objetos) y físico (arquitectura de módulos y procesos), así como los modelos estáticos y dinámicos.

- ***Programación orientada a objetos***

  La Programación Orientada a Objetos (POO) es la implementación del diseño, en donde los programas son colecciones de objetos cooperantes. Cada objeto representa una instancia de alguna clase y las clases pertenecen a una jerarquía de clases relacionadas por la herencia.

4.2.5 Ventajas de la metodología orientada a objetos:

- *Reutilización*. Las clases están diseñadas para que se reutilicen en muchos sistemas, para maximizar la reutilización, las clases se construyen de manera que se puedan adaptar a los otros sistemas. El objetivo fundamental es lograr la reutilización masiva al construir el software.
- *Estabilidad*. Las clases diseñadas para una reutilización repetida se vuelven estables, de la misma manera que los microprocesadores y otros chips se hacen estables.
- *Se construyen clases cada vez más complejas*. Se construyen clases a partir de otras clases, las cuales a su vez se integran mediante clases. Esto permite construir

componentes de software complejos, que a su vez se convierten en bloques de construcción de software más complejo.

- *Calidad*. Los diseños suelen tener mayor calidad, puesto que se integran a partir de componentes probados, que han sido verificados y pulidos varias veces.
- *Un diseño más rápido*. Las aplicaciones se crean a partir de componentes ya existentes, muchos de estos componentes están construidos de modo que se pueden adaptar para un diseño particular.
- *Integridad*. Las estructuras de datos (los objetos) sólo se pueden utilizar con métodos específicos. Esto tiene particular importancia en los sistemas cliente-servidor y los sistemas distribuidos, en los que usuarios desconocidos podrían intentar el acceso al sistema.
- *Mantenimiento más sencillo*. El programador encargado del mantenimiento cambia un método de clase a la vez. Cada clase efectúa sus funciones independientemente de las demás.
- *Una interfaz de pantalla sugestiva para el usuario*. Hay que utilizar una interfaz de usuario gráfica de modo que el usuario apunte a iconos o elementos de un menú desplegado, relacionados con los objetos.
- *Independencia del diseño*. Las clases están diseñadas para ser independientes del ambiente de plataformas, hardware y software. Utilizan solicitudes y respuestas con formato estándar. Esto les permite ser utilizadas en múltiples sistemas operativos, controladores de bases de datos, controladores de red, interfaces de usuario gráficas, etc. El creador del software no tiene que preocuparse por el ambiente o esperar a que éste se especifique.

- *Interacción*. El software de varios proveedores puede funcionar como conjunto. Un proveedor utiliza clases de otros. Existe una forma estándar de localizar clases e interactuar con ellas. El software desarrollado de manera independiente en lugares ajenos debe poder funcionar en forma conjunta y aparecer como una sola unidad ante el usuario.
- *Migración*. Las aplicaciones ya existentes, sean orientadas a objetos o no, pueden preservarse si se ajustan a un contenedor orientado a objetos, de modo que la comunicación con ella sea a través de mensajes estándar orientados a objetos.
- *Mejores herramientas CASE*. Las herramientas CASE (Computer Aided Software Engineering, Ingeniería de Software Asistida por Computadora) utilizarán las técnicas gráficas para el diseño de las clases y de la interacción entre ellas, para el uso de los objetos existentes adaptados a nuevas aplicaciones. Las herramientas deben facilitar el modelado en términos de eventos, formas de activación, estados de objetos, etc. Las herramientas OO del CASE deben generar un código tan pronto se definan las clases y permitir al diseñador utilizar y probar los métodos recién creados. Las herramientas se deben diseñar de manera que apoyen el máximo de creatividad y una continua afinación del diseño durante la construcción.

La historia de esta metodología va unida a la evolución de los lenguajes de programación orientada al objeto, los más representativos: a fines de los 60’s SIMULA, a fines de los 70’s Smalltalk-80, la primera versión de C++ por Bjarne

Stroustrup en 1981 y actualmente Java[18] o C# de Microsoft. A fines de los 80's comenzaron a consolidarse algunos métodos Orientadas a Objeto.

En 1995 Booch y Rumbaugh proponen el Método Unificado con la ambiciosa idea de conseguir una unificación de sus métodos y notaciones, que posteriormente se reorienta a un objetivo más modesto, para dar lugar al Unified Modeling Language (UML)[19], la notación OO más popular en la actualidad.

Algunos métodos OO con notaciones predecesoras de UML son: OOAD (Booch), OOSE (Jacobson), Coad & Yourdon, Shaler & Mellor y OMT (Rumbaugh). Algunas metodologías orientadas a objetos que utilizan la notación UML son: Rational Unified Process (RUP)[20], OPEN[21], MÉTRICA (que también soporta la notación estructurada).

4.2.6 Metodologías Agiles

Las Metodologías Ágiles o "ligeras" constituyen un nuevo enfoque en el desarrollo de software, mejor aceptado por los desarrolladores de e-projects que las metodologías convencionales (ISO-9000, CMM, etc) debido a la simplicidad de sus reglas y prácticas, su orientación a equipos de desarrollo de pequeño tamaño, su flexibilidad ante los cambios y su ideología de colaboración.

[18] http://java.sun.com/ (7.5.2003)
[19] http://www.uml.org/ (7.5.2003)
[20] http://www.rational.com/products/rup/index.jsp (7.5.2003)
[21] http://www.open.org.au/ (17.9.2003)

Estas metodologías surgen como una alternativa a las metodologías tradicionales y principalmente a su burocracia. Los diseñadores de software centran su interés en metodologías lo suficientemente documentadas, que faciliten la obtención de información y que dispongan de algún tipo de certificación y training.

Un proceso es ágil cuando el desarrollo de software es **incremental** (entregas pequeñas de software, con ciclos rápidos), **cooperativo** (cliente y desarrolladores trabajan juntos constantemente con una cercana comunicación), **sencillo** (el método en sí mismo es fácil de aprender y modificar, bien documentado), y **adaptable** (permite realizar cambios de último momento). Entre las metodologías ágiles identificadas que han adquirido reconocimiento tenemos:

- Extreme Programming.
- Scrum.
- Cristal Methods
- Feature Driven Development
- Proceso Unificado Rational (RUP) una configuración ágil.
- Dynamic Systems Development Method (DSDM).
- Adaptive Software Development

Existen otras Metodologías Agiles reconocidas por los especialistas, pero aquí solo se analizan aquellas que han alcanzado a figurar y cuentan en su haber con casos sustanciales documentados. La idea no es hacer un censo de todo lo existente, sino tratar las metodologías de acuerdo a su importancia.

Manifiesto de las Metodologías Ágiles

En un taller de dos días en Snowbird, Utah, en febrero de 2001 se reunieron los representantes de cada una de las metodologías ágiles. Había mucho en común, y este reconocimiento era mucho mayor que las diferencias entre los procesos.

Además de un contacto útil entre los líderes de procesos, existía también la idea de emitir una declaración conjunta en favor de procesos de desarrollo de software ágiles.

El resultado es un Manifiesto para el Desarrollo de Software Ágil, una declaración de los principios y valores comunes de los procesos ágiles. Hay también un deseo de colaborar más en el futuro, para animar más tanto a tecnólogos como a gente de negocios para usar y requerir acercamientos ágiles al desarrollo de software.

El manifiesto fue sólo eso, una publicación que actuó como un punto de partida para aquellos que compartían estas ideas básicas. Uno de los frutos del esfuerzo fue la creación de un cuerpo más longevo, la *Alianza Ágil*, que es una organización sin fines de lucro que busca promover el conocimiento y la discusión de todos los métodos ágiles. Muchos de los líderes de metodologías de desarrollo de software ágiles son miembros y líderes de la Alianza Ágil.

Los principios que se establecieron en el manifiesto son:

1. La prioridad principal es satisfacer al cliente mediante tempranas y continuas entregas de software que le reporte un valor.
2. Dar la bienvenida a los cambios. Los MA capturan los cambios para que el cliente tenga una ventaja competitiva.
3. Entregar frecuentemente software que funcione, desde un par de semanas a un par de meses, con el menor intervalo de tiempo posible entre una entrega y la siguiente.
4. La gente del negocio y los desarrolladores deben trabajar juntos a lo largo del proyecto.
5. Construir el proyecto entorno a individuos motivados. Darles el entorno y el apoyo que necesitan y confiar en ellos para conseguir el trabajo.
6. El diálogo cara a cara es el método más eficiente y efectivo para comunicar información dentro de un equipo de desarrollo.
7. El software que funciona es la medida principal de progreso.
8. Los procesos ágiles promueven un desarrollo sostenible. Los promotores, desarrolladores y usuarios deberían ser capaces de mantener una paz constante.
9. La atención continua a la calidad técnica y al buen diseño mejora la agilidad.
10. La simplicidad es esencial.
11. Las mejores arquitecturas, requisitos y diseños surgen de los equipos organizados por sí mismos.
12. En intervalos regulares, el equipo reflexiona respecto de cómo llegar a ser más efectivo, y según esto ajusta su comportamiento.

La tabla siguiente permite apreciar las convergencias y divergencias de las Metodologías Agiles, así como resume sus características claves y sus fechas de aparición.

Metodología	Acrónimo	Creación	Tipo de modelo	Características
Extreme Programming	XP	Beck 1999	Disciplina en prácticas de Ingeniería	Método Ágil Radical
Scrum	Scrum	Sutherland 1994 Schwaber 1995	Proceso (framework de management)	Complemento de otros métodos, agiles o no
Cristal Methods	CM	Cockburn 1998	Familia de Metodologías	Énfasis en modelo de ciclos
Feature Driven Development	FDD	De Luca & Coad 1998	Metodología	Método ágil de diseño y construcción
Rational Unified Process	RUP	Kruchten 1996	Proceso Unificado	Método ágil con modelado
Dynamic Systems Development Method	DSDM	Stapleton 1997	Framework / Modelo de ciclo de vida	Creado por 16 expertos en RAD
Adaptive Software Development	ASD	Higsmith 2000	Practicas + Ciclo de vida	Inspirado en sistemas adaptativos complejos.

Tabla 2: *Características Metodologías Agiles*

4.3. Modelos del Ciclo de Vida

Los proyectos de ingeniería de software tienen fines ligados a la obtención del producto, proceso o servicio los cuales deben ser generados a través de diversas actividades. Algunas de estas actividades pueden agruparse en fases porque globalmente contribuyen a obtener un producto intermedio, necesario para continuar hacia el producto final y facilitar la gestión del proyecto. Al conjunto de estas fases empleadas se le denomina **"ciclo de vida"**.

La selección de un ciclo de vida facilita el **control sobre los tiempos** en que es necesario aplicar recursos de todo tipo (personal, equipos, suministros, etc.) al proyecto. Si el proyecto incluye subcontratación

de partes a otras organizaciones, el **control del trabajo subcontratado** se facilita en la medida en que esas partes encajen bien en la estructura de las fases. El **control de calidad** también se ve facilitado si la separación entre fases se hace corresponder con puntos en los que ésta deba verificarse (mediante comprobaciones sobre los productos parciales obtenidos).

Las principales diferencias entre los distintos modelos de ciclo de vida están en:

- El **alcance** del ciclo dependiendo de hasta dónde llegue el proyecto correspondiente. Un proyecto puede comprender un simple estudio de viabilidad del desarrollo de un producto, o su desarrollo completo o, llevando la cosa al extremo, toda la historia del producto con su desarrollo, fabricación, y modificaciones posteriores hasta su retirada del mercado.

- Las **características** (contenidos) de las fases en que dividen el ciclo. Esto puede depender del propio tema al que se refiere el proyecto (no son lo mismo las tareas que deben realizarse para proyectar un avión que un puente), o de la organización (interés de reflejar en la división en fases aspectos de la división interna o externa del trabajo).

- La **estructura** de la sucesión de las fases que puede ser lineal, con prototipado, o en espiral. Veámoslo con más detalle:

4.3.1 Elementos del ciclo de vida

El ciclo de vida de un proyecto se compone de **fases sucesivas** compuestas por tareas planificables. Según el modelo de ciclo

de vida, la sucesión de fases puede ampliarse con **bucles de realimentación**, de manera que lo que conceptualmente se considera una misma fase se pueda ejecutar más de una vez a lo largo de un proyecto, recibiendo en cada pasada de ejecución aportaciones de los resultados intermedios que se van produciendo (realimentación).

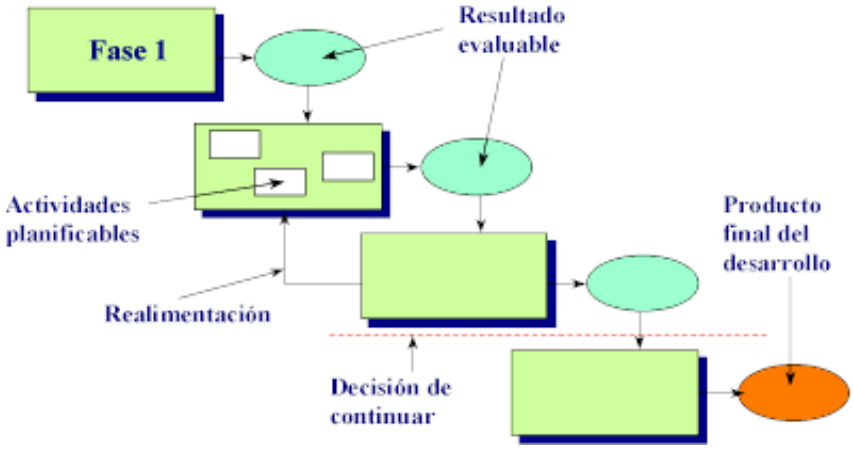


Figura 32: *Elementos del ciclo de vida*

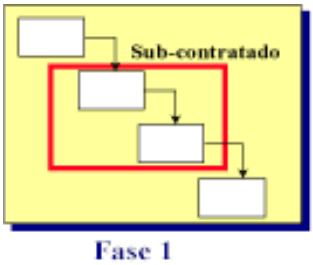
Para un adecuado control de la progresión de las fases de un proyecto se hace necesario especificar con suficiente precisión los resultados evaluables, o sea, productos intermedios que deben resultar de las tareas incluidas en cada fase. Normalmente estos productos marcan los hitos entre fases.

A continuación se presentan los distintos elementos que integran un ciclo de vida:

- **Fases**: Una fase es un conjunto de actividades relacionadas con un objetivo en el desarrollo del proyecto. Se construye agrupando tareas (actividades elementales) que pueden compartir un tramo determinado del tiempo de vida de un proyecto. La agrupación temporal de tareas impone

requisitos temporales correspondientes a la asignación de recursos (humanos, financieros o materiales).

Cuanto más grande y complejo sea un proyecto, mayor detalle se necesitará en la definición de las fases para que el contenido de cada una siga siendo manejable. De esta forma, cada fase de un proyecto puede considerarse un "*micro-proyecto*" en sí mismo, compuesto por un conjunto de micro-fases.



Otro motivo para descomponer una fase en subfases menores es el interés de separar partes temporales del proyecto que se subcontraten a otras organizaciones, requiriendo distintos procesos de gestión.

Cada fase viene definida por un conjunto de elementos observables externamente, como son las **actividades** con las que se relaciona, los **datos de entrada** (resultados de la fase anterior, documentos o productos requeridos para la fase, experiencias de proyectos anteriores), los **datos de salida** (resultados a utilizar por la fase posterior, experiencia acumulada, pruebas o resultados efectuados) y la **estructura interna** de la fase.

Figura 33: *Esquema general de operación de una fase*

- **Entregables**: Son los productos intermedios que generan las fases. Pueden ser materiales (componentes, equipos) o inmateriales (documentos, software). Los entregables permiten evaluar la marcha del proyecto mediante comprobaciones de su adecuación o no a los requisitos funcionales y de condiciones de realización previamente establecidos. Cada una de estas evaluaciones puede servir, además, para la toma de decisiones a lo largo del desarrollo del proyecto.

4.3.2 Objetivos de cada fase

Dentro de cada fase general de un modelo de ciclo de vida, se pueden establecer una serie de objetivos y tareas que lo caracterizan.

- **Fase de definición (¿qué hacer?)**
  - Estudio de **viabilidad.**
  - **Conocer los requisitos** que debe satisfacer el sistema (funciones y limitaciones de contexto).
  - Asegurar que los **requisitos son alcanzables.**

- Formalizar el **acuerdo** con los usuarios.
- Realizar una **planificación** detallada.

- **Fase de diseño (¿cómo hacerlo? Soluciones en coste, tiempo y calidad)**

  - Identificar **soluciones tecnológicas** para cada una de las funciones del sistema.
  - Asignar **recursos** materiales para cada una de las funciones.
  - Proponer (identificar y seleccionar) **subcontratas.**
  - Establecer métodos de **validación** del diseño.
  - **Ajustar las especificaciones** del producto.

- **Fase de construcción**

  - Generar el producto o servicio pretendido con el proyecto.
  - Integrar los elementos subcontratados o adquiridos externamente.
  - Validar que el producto obtenido satisface los requisitos de diseño previamente definidos y realizar, si es necesario, los ajustes necesarios en dicho diseño para corregir posibles lagunas, errores o inconsistencias

- **Fase de mantenimiento y operación**

  - **Operación**: asegurar que el uso del proyecto es el pretendido.
  - **Mantenimiento**: Aquel que no se limita a reparar averías o desgastes habituales.

4.3.3 Tipos de modelos del ciclo de vida

Las principales diferencias entre distintos modelos de ciclo de vida están en:

- El **alcance** del ciclo dependiendo de hasta dónde llegue el proyecto correspondiente. Un proyecto puede comprender un simple estudio de viabilidad del desarrollo de un producto, o su desarrollo completo o, llevando la cosa al extremo, toda la historia del producto con su desarrollo, fabricación, y modificaciones posteriores hasta su retirada del mercado.

- Las **características** (contenidos) de las fases en que dividen el ciclo. Esto puede depender del propio tema al que se refiere el proyecto (no son lo mismo las tareas que deben realizarse para proyectar un avión que un puente), o de la organización (interés de reflejar en la división en fases aspectos de la división interna o externa del trabajo).

- La **estructura** de la sucesión de las fases que puede ser lineal, con prototipado, o en espiral.

4.3.3.1 Ciclo de vida Incremental

Los riesgos asociados con el desarrollo de sistemas largos y complejos son enormes, una forma de reducir los riesgos es construir sólo una parte del sistema, reservando otros aspectos para niveles posteriores.

El desarrollo incremental es el proceso de construcción que periódicamente va incrementando subconjuntos de requerimientos al sistema, típicamente, un documento de requerimientos es escrito al capturar todos los requerimientos para el sistema completo.

El desarrollo incremental es 100% compatible con el modelo en cascada, no demanda una forma específica de observar el desarrollo de algún otro incremento. El modelo de desarrollo incremental provee algunos beneficios significativos para los proyectos:

- Construir un sistema pequeño es siempre menos riesgoso que construir un sistema grande.
- Al ir desarrollando parte de las funcionalidades, es más fácil determinar si los requerimientos planeados para los niveles subsiguientes son correctos.
- Si un error importante es realizado, sólo la última iteración necesita ser descartada.
- Reduciendo el tiempo de desarrollo de un sistema (en este caso en incremento del sistema) decrecen las probabilidades que esos requerimientos de usuarios puedan cambiar durante el desarrollo.
- Si un error importante es realizado, el incremento previo puede ser usado.
- Los errores de desarrollo realizados en un incremento, pueden ser arreglados antes del comienzo del próximo incremento.

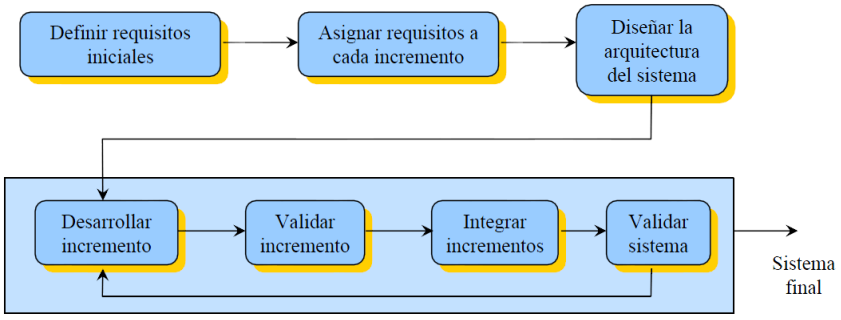


Figura 34: *Ciclo de vida Incremental*

4.3.3.2 Ciclo de vida con prototipado

A menudo ocurre en desarrollos de productos con innovaciones importantes, o cuando se prevé la utilización de tecnologías nuevas o poco probadas, en las que las incertidumbres sobre los resultados realmente alcanzables, o las ignorancias sobre el comportamiento de las tecnologías, impiden iniciar un proyecto lineal con especificaciones cerradas.

Si no se conoce exactamente lo que se quiere o no se comprende cómo desarrollar un determinado producto o cuáles son las especificaciones de forma precisa, suele recurrirse a definir especificaciones iniciales para hacer un **prototipo**, o sea, un producto parcial (no hace falta que contenga funciones que se consideren triviales o suficientemente probadas) y provisional (no se va a fabricar realmente para clientes, por lo que tiene menos restricciones de coste y/o prestaciones). Este tipo de procedimiento es muy utilizado en desarrollo avanzado.

Figura 35: *Ciclo de vida con Prototipado*

4.3.3.3 Ciclo de vida en espiral

El ciclo de vida en espiral puede considerarse como una generalización del anterior para los casos en que **no basta con una sola evaluación de un prototipo** para asegurar la desaparición de incertidumbres y/o ignorancias. El propio producto a lo largo de su desarrollo puede así considerarse como una sucesión de prototipos que progresan hasta llegar a alcanzar el estado deseado. En cada ciclo (espirales) las especificaciones del producto se van resolviendo paulatinamente.

A menudo la **fuente de incertidumbres es el propio cliente**, que aunque sepa en términos generales lo que quiere, no es capaz de definirlo en todos sus aspectos sin ver como unos influyen en otros. En estos casos la evaluación de los resultados por el cliente no puede esperar a la entrega final y puede ser necesaria repetidas veces.

El esquema del ciclo de vida para estos casos puede representarse por un bucle en espiral, donde los cuadrantes son, habitualmente, fases de **especificación, diseño, realización y evaluación** (o conceptos y términos análogos).

En cada vuelta el producto gana en "*madurez*" (aproximación al final deseado) hasta que en una vuelta la evaluación lo apruebe y el bucle pueda abandonarse.

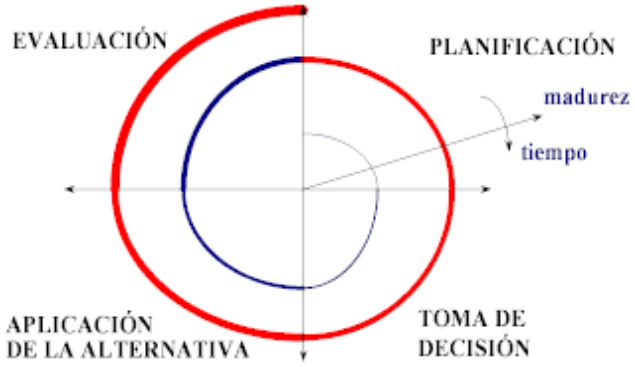


Figura 36: *Ciclo de vida en espiral*

El modelo espiral captura algunos principios básicos:

- Decidir qué problema se quiere resolver antes de viajar a resolverlo.
- Examinar tus múltiples alternativas de acción y elegir una de las más convenientes.
- Evaluar qué tienes hecho y qué tienes que haber aprendido después de hacer algo.
- No ser tan ingenuo para pensar que el sistema que estás construyendo será "EL" sistema que el cliente necesita, y
- Conocer (comprender) los niveles de riesgo, que tendrás que tolerar.

4.3.3.4 Ciclo de vida en Cascada

En este modelo, un proyecto progresa a través de una secuencia ordenada de pasos partiendo del concepto inicial del producto hasta la prueba para evaluar la efectividad y características esperadas.

Cuando la revisión determina que el producto no está listo para pasar a la fase de desarrollo, permanece en la etapa de construcción hasta que cumpla con lo prioritariamente establecido.

No proporciona resultados tangibles (software o hardware completo) hasta el final del ciclo de vida, pero la documentación que se genera proporciona indicaciones significativas del progreso a lo largo del ciclo de vida.

Las características principales de este modelo son:

- Requiere establecimiento explícito de requerimientos desde el comienzo
- Exige al cliente/usuario gran paciencia
- Dificulta la incorporación de modificaciones durante el desarrollo

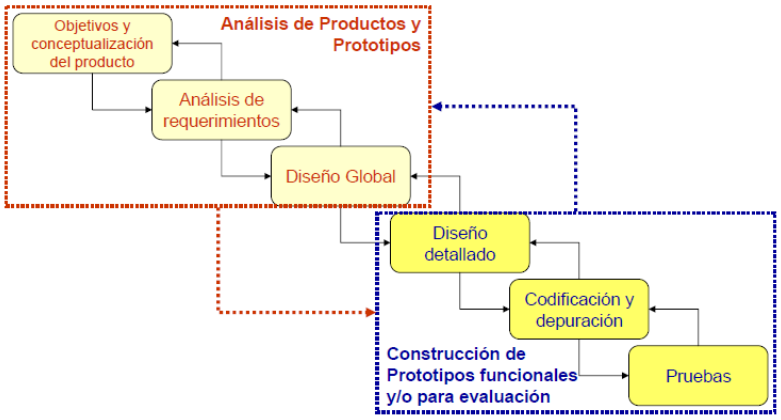


Figura 37: *Ciclo de vida en cascada*

El modelo de ciclo de vida cascada, captura algunos principios básicos:

- Planear un proyecto antes de embarcarse en él.

- Definir el comportamiento externo deseado del sistema antes de diseñar su arquitectura interna.
- Documentar los resultados de cada actividad.
- Diseñar un sistema antes de codificarlo.
- Testear un sistema después de construirlo.

Cuadro Comparativo Modelos Del Ciclo de Vida

MODELO	ENFOQUE	VENTAJAS /DESVENTAJAS	APLICABILIDAD
C A S C A D A	El inicio de cada etapa debe esperar a la finalización de la inmediatamente anterior. Cualquier error de diseño detectado en la etapa de prueba conduce necesariamente al rediseño y nueva programación del código afectado, aumentando los costes del desarrollo.	No todos los requisitos son expuestos al principio, de forma explícita como requiere este modelo. El cliente debe tener paciencia, ya que la aplicación sólo estará disponible en un estado muy avanzado del proyecto. Ampliamente criticado desde el ámbito académico y la industria.	Utilizado cuando existen especificaciones amplias de los requerimientos del cliente.
P R O T O T I P A O	No posee la funcionalidad total del sistema pero si condensa la idea principal del mismo. Paso a Paso crece su funcionalidad. Alto grado de participación del usuario.	El cliente puede pensar que el prototipo es una versión acabada. Pueden llegar a pasarse por alto la calidad del software global o el mantenimiento a largo plazo. Las herramientas elegidas pueden ser inadecuadas. La clave del éxito de este modelo consiste en definir bien, desde el principio, las reglas del juego. Alto grado de participación del usuario	Se utiliza si en el mercado no se encuentra el producto pero el cliente desea resultados inmediatos. Conveniente en caso de ser necesario desarrollar módulos Para sistemas interactivos pequeños o de tamaño pequeño. Para partes de sistemas grandes Para sistemas con vida corta.

I N C R E M E N T A L	El sistema no se entrega de una vez, sino que se divide y se entregan incrementos. Con cada incremento se entrega la parte de la funcionalidad que se ha establecido. Los requisitos son priorizados. Los requisitos con una más alta prioridad se incluyen en los incrementos más tempranos. Los requisitos de un incremento son inamovibles. Sin embargo estos puede verse modificados en incrementos posteriores. Este proceso se repite hasta la obtención de un producto completo. Sin embargo el modelo incremental se centra en la entrega de un producto operativo en cada incremento.	Los clientes no tienen que esperar hasta tener el sistema completo. El primer incremento satisface los requisitos más críticos. Los primeros incrementos sirven como prototipo y ayudan en la tarea de detectar los posteriores requisitos. Existe un riesgo bajo de fallar en el proyecto total. Los servicios del sistema con la prioridad más alta tienden a ser los más probados. Puede ser difícil ajustar los requisitos a los incrementos.	Reemplazar el antiguo desarrollo con uno nuevo que satisfaga las nuevas necesidades según las redefiniciones del problema. Manejo de Versiones
E S P I R A L	Es una mejora del Modelo Basado en prototipos. Cada vuelta en la espiral representa una fase del proceso. No hay fases fijas, cada vuelta en la espiral determina las actividades a realizar. La dimensión radial representa el coste acumulado en la	Requiere comunicación permanente con el cliente por lo tanto si se cambia el contacto con le cual se realiza desarrollo es necesario que esté al tanto de lo realizado y lo pendiente, el cliente debe ser gran conocedor del sistema.	Utilizado para el desarrollo de aplicaciones complejas y/o específicas. (Ej. Investigación Genética)

	financiación de las fases. La dimensión angular representa el progreso hecho en completar cada ciclo de la espiral. Un ciclo a través de la espiral es simular un paso a través de un modelo en cascada		

Tabla 3: *Cuadro Comparativo de Modelos Del Ciclo de Vida*

4.4 El concepto de la calidad

4.4.1 La crisis del software

La crisis del software aparece en la segunda era de la evolución de los sistemas informáticos, alrededor de 1968 en la primera conferencia organizada por la OTAN sobre desarrollo de software y con él se etiquetaron los problemas que surgían en el desarrollo de sistemas de software. Uno de los principales problemas en el desarrollo de software, es que muchos proyectos empiezan la programación tan pronto se definen, y concentran mucho de su esfuerzo en la escritura del código. Posteriormente las actividades de mantenimiento del software (corrección de fallas, modificación por cambios de requerimientos de usuarios, y adaptación a nuevos dispositivos) y el esfuerzo empleado en dicho mantenimiento comenzaran a absorber recursos en una medida alarmante.

Básicamente, la crisis del software se refiere a la dificultad en escribir programas libres de defectos, fácilmente comprensibles, y que sean verificables. Las causas son, entre otras, la complejidad que supone la tarea de programar, y los

cambios a los que se tiene que ver sometido un programa para ser continuamente adaptado a las necesidades de los usuarios. Seguin el Reporte SOAR (State Of the Art Report):

- Las compañías desarrolladoras de software están liberando productos a sus clientes con 15% de defectos en el producto.
- Muchas compañías de desarrollo se gastan entre 30% y 40% de su tiempo y dinero en correcciones y ajustes a los productos.
- Sólo un50% de las compañías emplean cronogramas.
- Alrededor del 25% de los proyectos de software son cancelados.
- El costo de obtener y mantener el software en los 80´s fue el doble de lo que costó su desarrollo.
- Durante los 90´s el costo de licenciamiento y mantenimiento se incrementó en un 30% más que en los 80´s.
- La mitad de los proyectos de software se pasaron del cronograma definido.

Según el reporte GAO los problemas relacionados con el desarrollo del software son:

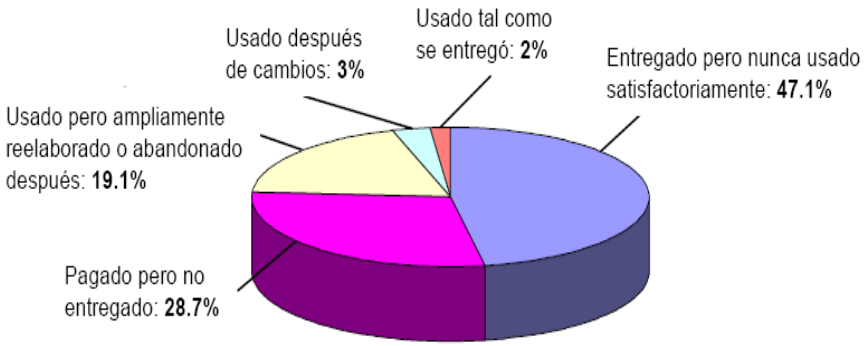


Figura 38: *El reporte GAO*

Según el reporte CHAOS los problemas relacionados con el desarrollo del software son:

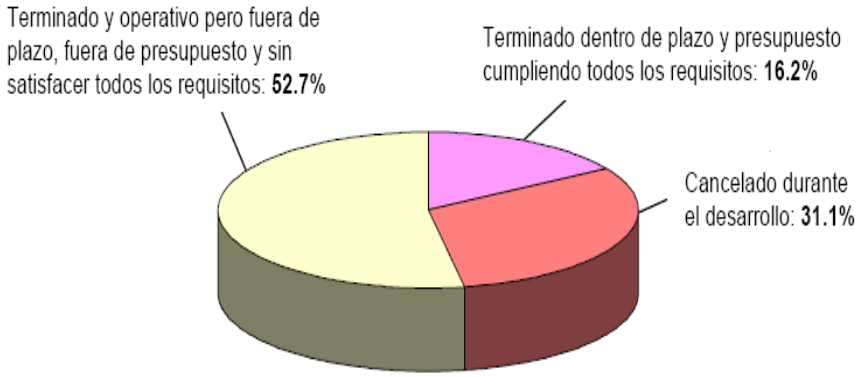


Figura 39: *El reporte CHAOS*

Es difícil establecer cuál es la cantidad media de defectos que un sistema software "normal" contiene. De hecho, hasta el software más depurado y considerado de alta fiabilidad contiene defectos remanentes según se avanza en el proceso de búsqueda de defectos, el coste de detección de fallos y eliminación de las faltas que los provocan empieza a rebasar con mucho las mejoras conseguidas en la fiabilidad del sistema, un reporte reciente del Software Engineering Institute, hace referencia a causas probables por las cuales existe fracaso:

Figura 40: *Porque existe fracaso*

4.4.2 ¿Qué es la calidad del software?

La calidad del *software* es el conjunto de cualidades que lo caracterizan y que determinan su utilidad y existencia. La calidad es sinónimo de eficiencia, flexibilidad, corrección, confiabilidad, mantenibilidad, portabilidad, usabilidad, seguridad e integridad.

La calidad del *software* es medible y varía de un sistema a otro o de un programa a otro. Un *software* elaborado para el control de naves espaciales debe ser confiable al nivel de "cero fallas"; un *software* hecho para ejecutarse una sola vez no requiere el mismo nivel de calidad; mientras que un producto de *software* para ser explotado durante un largo período (10 años o más), necesita ser confiable, mantenible y flexible para disminuir los costos de mantenimiento y perfeccionamiento durante el tiempo de explotación.

La calidad del software puede medirse después de elaborado el producto. Pero esto puede resultar muy costoso si se

detectan problemas derivados de imperfecciones en el diseño, por lo que es imprescindible tener en cuenta tanto la obtención de la calidad como su control durante todas las etapas del ciclo de vida del *software*.

4.4.3 ¿Cómo obtener un software de calidad?

La obtención de un *software* con calidad implica la utilización de metodologías o procedimientos estándares para el análisis, diseño, programación y prueba del *software* que permitan uniformar la filosofía de trabajo, en aras de lograr una mayor confiabilidad, mantenibilidad y facilidad de prueba, a la vez que eleven la productividad, tanto para la labor de desarrollo como para el control de la calidad del *software*.

La política establecida debe estar sustentada sobre tres principios básicos: tecnológico, administrativo y ergonómico.

- El principio tecnológico define las técnicas a utilizar en el proceso de desarrollo del *software.*

- El principio administrativo contempla las funciones de planificación y control del desarrollo del *software*, así como la organización del ambiente o centro de ingeniería de *software*.

- El principio ergonómico define la interfaz entre el usuario y el ambiente automatizado.

La adopción de una buena política contribuye en gran medida a lograr la calidad del *software*, pero no la asegura. Para el aseguramiento de la calidad es necesario su control o evaluación.

4.4.4 ¿Cómo controlar la calidad del Software?

Para controlar la calidad del *software* es necesario, ante todo, definir los parámetros, indicadores o criterios de medición, ya que, como bien plantea Tom De Marco, "usted no puede controlar lo que no se puede medir".

Las cualidades para medir la calidad del *software* son definidas por innumerables autores, los cuales las denominan y agrupan de formas diferentes. Por ejemplo, John Wiley define métricas de calidad y criterios, donde cada métrica se obtiene a partir de combinaciones de los diferentes criterios.

La *Metodología para la evaluación de la calidad de los medios de programas* de la CIC, de Rusia, define indicadores de calidad estructurados en cuatro niveles jerárquicos: factor, criterio, métrica, elemento de evaluación, donde cada nivel inferior contiene los indicadores que conforman el nivel precedente. Otros autores identifican la calidad con el nivel de complejidad del *software* y definen dos categorías de métricas: de complejidad de programa o código, y de complejidad de sistema o estructura.

Todos los autores coinciden en que el *software* posee determinados índices medibles que son las bases para la calidad, el control y el perfeccionamiento de la

productividad. Una vez seleccionados los índices de calidad, se debe establecer el proceso de control, que requiere los siguientes pasos:

- Definir el *software* que va a ser controlado: clasificación por tipo, esfera de aplicación, complejidad, etc., de acuerdo con los estándares establecidos para el desarrollo del *software*.

- Seleccionar una medida que pueda ser aplicada al objeto de control. Para cada clase de *software* es necesario definir los indicadores y sus magnitudes.

- Crear o determinar los métodos de valoración de los indicadores: métodos manuales como cuestionarios o encuestas estándares para la medición de criterios periciales y herramientas automatizadas para medir los criterios de cálculo.

- Definir las regulaciones organizativas para realizar el control: quiénes participan en el control de la calidad, cuándo se realiza, qué documentos deben ser revisados y elaborados, etc.

A partir del análisis de todo lo anterior, nuestro Centro se encuentra enfrascado en un proyecto para el Aseguramiento de la Calidad del *Software* (ACS), válido para cualquier entidad que se dedique a la investigación, producción y comercialización del *software*, el cual incluye la elaboración de un Sistema de Indicadores de la Calidad del *Software*, la confección de una *Metodología para el Aseguramiento de la*

Calidad del Software y el desarrollo de herramientas manuales y automatizadas de apoyo para la aplicación de las técnicas y procedimientos del ACS, de forma tal que se conforme un Sistema de Aseguramiento de la Calidad del *Software*.

Lograr el éxito en la producción de *software* es hacerlo con calidad y demostrar su buena calidad. Esto sólo es posible con la implantación de un Sistema para el Aseguramiento de la Calidad del *Software* directamente relacionado con la política establecida para su elaboración y que esté en correspondencia con la definición internacional ISO de calidad, ampliamente aceptada, y por los estándares del grupo ISO 9000.

4.4.5 ¿Cómo verificar la calidad del software?

A través de técnicas y actividades ligadas al control de calidad del software para tratar de comprobar si los productos construidos en una fase del ciclo de vida satisfacen los requisitos establecidos en una fase anterior y/o si el software construido satisface los requisitos del usuario, es decir si el producto de software funciona como el usuario quiere y realiza las funciones que se habían solicitado.

4.4.6 Factores que determinan la calidad del software

- **Operaciones del producto**: características operativas

  – *Corrección* (¿Hace lo que se le pide?)

El grado en que una aplicación satisface sus especificaciones y consigue los objetivos encomendados por el cliente

- *Fiabilidad* (¿Lo hace de forma fiable todo el tiempo?)
  El grado que se puede esperar de una aplicación lleve a cabo las operaciones especificadas y con la precisión requerida.

- *Eficiencia* (¿Qué recursos hardware y software necesito?)
  La cantidad de recursos hardware y software que necesita una aplicación para realizar las operaciones con los tiempos de respuesta adecuados.

- *Integridad* (¿Puedo controlar su uso?)
  El grado con que puede controlarse el acceso al software o a los datos a personal no autorizado.

- *Facilidad de uso* (¿Es fácil y cómodo de manejar?)
  El esfuerzo requerido para aprender el manejo de una aplicación, trabajar con ella, introducir datos y conseguir resultados.

• **Revisión del producto**: capacidad para soportar cambios

- *Facilidad de mantenimiento* (¿Puedo localizar los fallos?)
  El esfuerzo requerido para localizar y reparar errores

- *Flexibilidad* (¿Puedo añadir nuevas opciones?)

El esfuerzo requerido para modificar una aplicación en funcionamiento.

– *Facilidad de prueba* (¿Puedo probar todas las opciones?)
  El esfuerzo requerido para probar una aplicación de forma que cumpla con lo especificado en los requisitos.

• **Transición del producto**: adaptabilidad a nuevos entornos

– *Portabilidad* (¿Podré usarlo en otra máquina?)
  El esfuerzo requerido para transferir la aplicación a otro hardware o sistema operativo.

– *Reusabilidad* (¿Podré utilizar alguna parte del software en otra aplicación?)
  Grado en que partes de una aplicación pueden utilizarse en otras aplicaciones.

– *Interoperabilidad* (¿Podrá comunicarse con otras aplicaciones o sistemas informáticos?
  El esfuerzo necesario para comunicar la aplicación con otras aplicaciones o sistemas informáticos

4.4.7 Métricas de calidad del software

Es difícil, y en ciertos casos es imposible, desarrollar medidas directas de los factores de calidad del software, cada factor de

calidad **Fc** se puede obtener como combinación de una o varias métricas:

$F_c = c_1 * m_1 + c_2 * m_2 + \ldots + c_n * m_n$

- **c_1** factor de ponderación de la métrica i, que dependerá de cada aplicación específica.
- **m_i** métrica i
- Habitualmente se puntúan de 0 a 10 en las métricas y en los factores de calidad.

Métricas para determinar los factores de calidad

– Facilidad de auditoria
– Exactitud
– Normalización de las comunicaciones
– Completitud
– Concisión
– Consistencia
– Estandarización de los datos
– Tolerancia de errores
– Eficiencia de la ejecución
– Facilidad de expansión
– Generalidad
– Independencia del hardware
– Instrumentación
– Modularidad
– Facilidad de operación
– Seguridad
– Auto documentación
– Simplicidad

– Independencia del sistema software

– Facilidad de traza

– Formación

5. LA EVALUACIÓN DEL SOFTWARE PARA JUEGOS DE GUERRA

5.1 Resumen

Todos, alguna vez, hemos sufrido algún error informático, o la destrucción de todo un día de trabajo por culpa de un fallo misterioso en el software; tales problemas nacen de la complejidad del software. La extrema dificultad para construir sistemas software multiplica la probabilidad de que persistan errores aún después de haberse finalizado y entregado el sistema, manifestándose cuando éste es utilizado por el cliente.

La construcción de un sistema software tiene como objetivo satisfacer una necesidad planteada por un cliente. ¿Cómo puede saberse si el producto construido corresponde exactamente con lo que el cliente deseaba? y ¿Cómo se puede estar seguro de que el producto que se ha construido va a funcionar correctamente?

Desgraciadamente, nuestra capacidad para medir la fiabilidad del software es muy inferior a lo que sería necesario. Sería deseable que los informáticos pudiéramos demostrar matemáticamente la corrección de los programas, al estilo de los otros ingenieros. Los otros ingenieros recurren a los análisis matemáticos para predecir cuál será el comportamiento de sus creaciones en el mundo real. Esa predicción permite descubrir defectos antes de que el producto esté operativo. Por desdicha, las matemáticas tradicionales, aptas para la descripción de sistemas físicos (los tipos de sistemas tratados por las otras ingenierías), no son aplicables al universo sintético binario de un programa de ordenador.

Dada la imposibilidad de aplicar métodos matemáticos rigurosos, el modo que tenemos los informáticos para respaldar la confianza de los

programas es mediante la evaluación del software. La fiabilidad de los programas irá creciendo a lo largo de este proceso.

5.2 La evaluación

De forma general, podemos decir que el término evaluación es una palabra elástica que tiene usos diferentes y que puede aplicarse a una gama muy variada de actividades humanas. Considerada la evaluación en su acepción más amplia, nos encontramos con definiciones como la de la Real Academia Española: evaluar es "*señalar el valor de una cosa*". Para el D*iccionario del Español Actual,* evaluar significa "*valorar (determinar el valor de alguien o de algo)*".

Como una primera aproximación a la precisión conceptual del término, podemos decir que la palabra evaluación designa el conjunto de actividades que sirven para dar un juicio, hacer una valoración, medir "algo" (objeto, situación, proceso) de acuerdo con determinados criterios de valor con que se emite dicho juicio. En la vida cotidiana permanentemente estamos valorando sobre todo cuando ponderamos las acciones y decisiones que tomamos. Son formas de **evaluación informal**, las que no necesariamente se basan en una información suficiente y adecuada, ni pretenden ser objetivas y válidas. Pero cuando queremos evaluar servicios o actividades profesionales no basta la evaluación informal. Debemos recurrir a formas de **evaluación sistemática** que, utilizando un procedimiento científico, tienen garantía de validez y fiabilidad.

Existen una serie de términos que se emplean con frecuencia y que en ocasiones se utilizan de forma similar al de evaluación, siendo necesario precisar y diferenciar su alcance. Dicho en otras palabras, cuando hablamos de evaluación debemos diferenciarla de:

- ***Medición:*** que se refiere a la extensión y/o cuantificación de algo, pero sin determinar su valor.

- ***Estimación:*** que tiene un carácter aproximado y una carga subjetiva, ya que no implica exigencia metódica y formal como la evaluación sistemática.

- ***Seguimiento:*** que es el proceso analítico para registrar, recopilar, medir y procesar una serie de informaciones que revelan la marcha o desarrollo de un programa y que asegura una retroalimentación constante para una mejor ejecución del mismo.

- ***Control:*** que consiste en una verificación de resultados, no de su valoración (lo que constituiría una evaluación).

La evaluación se entiende como una *actividad sistemática y funcional*, ya que se trata de comprobar el grado de consecución de unos objetivos para tomar decisiones mediante una serie de procedimientos que permitirán identificar los factores que han influido en los resultados y formular recomendaciones con el fin de introducir correcciones.

Resulta necesario especificar, además, por qué *es preciso evaluar*, ya que este proceso debe tener una finalidad y una funcionalidad. En primer lugar, la evaluación sirve para dar cuenta de la gestión; sin embargo, evaluar va más allá de explicar si el gasto ha sido adecuado y conforme a la ley, pues, en última instancia, debe servir para proveer de información que facilite la toma de decisiones.

De este modo, la evaluación no se limita a medir resultados para esa toma de decisiones, sino que contribuye al conocimiento que fundamenta la intervención aportando el aprendizaje y los datos que los propios equipos, a cargo de los proyectos, tienen del problema que abordan. Este aprendizaje es el que permite ampliar y enriquecer las perspectivas conceptuales facilitando un mejor control sobre los factores que influyen en los resultados y los cambios que se pretende conseguir.

5.3 Enfoques teóricos

En el momento de realizar cualquier evaluación lo primero que debemos tomar en cuenta es su valoración, ya que la debemos fundamentar sobre una serie de ideas o posturas que determinen su sentido y alcance definitivo.

El origen de estas diferentes posiciones teóricas radica, en última instancia, en la interpretación que se haga acerca de la naturaleza de nuestro conocimiento. ¿Cómo conocemos la realidad? ¿De qué manera nos es posible estudiarla y valorarla? En tal virtud podemos diferenciar dos grandes enfoques:

El **enfoque cuantitativo** intenta acercarse a la realidad desde una perspectiva *experimental y analítica* acotando al máximo la realidad que se evalúa y controlando todas las variables que intervengan en ella, con el fin de encontrar relaciones causales entre ellas.

El **enfoque cualitativo** pretende estudiar la realidad con una pretensión descriptiva y comprensiva sin aspirar a explicarla como en el anterior posicionamiento. Opta por la comprensión global y abierta de los fenómenos, sin asilar ni manipular las variables que intervienen

en ellos, para no modificar o alterar el contexto y las circunstancias naturales en las que se dan. Los métodos utilizados desde esta perspectiva son los basados en la observación y descripción de los fenómenos (etnografía, estudio de casos, observación sistemática, triangulación). Junto a estos dos enfoques extremos, en la práctica se encuentran posturas intermedias que abogan por una utilización conjunta de los enfoques descritos, según las circunstancias y características de cada evaluación.

5.4 Tipos de evaluación

Los distintos tipos de evaluación que se pueden utilizar, pueden clasificarse de la siguiente manera:

Según el momento en que se evalúa	- Ex ante - Durante - Final, ex post o de impacto
Según las funciones que cumple	- Formativa - Sumativa - De impacto
Según la procedencia de los evaluadores	- Externa - Interna - Mixta - Autoevaluación
Según el aspecto objeto de evaluación o contenidos	- Las necesidades o contexto - El diseño o planificación - El proceso y desarrollo del programa - Resultados o productos

Tabla 4: *Tipos de evaluación*

5.4.1 *Según el momento en que se evalúa*

- **Ex ante:** Esta evaluación se realiza antes de la ejecución del programa y tiene como objeto de recogida de datos acerca de la situación inicial del programa.

- **Durante:** Es aquella que se realiza a lo largo del proceso de ejecución y que recoge de modo continuo y sistemático datos acerca del funcionamiento del programa.

- **Final, ex post o de impacto:** Es realizada cuando el programa ha concluido. Nos permite recoger datos acerca de la ejecución, funcionamiento, efectos o resultados de un programa, cualesquiera que éstos hayan sido, tanto los esperados como los no esperados, y valorar en qué medida las necesidades que originó el programa han sido satisfechas o no o se han generado otras nuevas.

5.4.2 *Según las funciones que cumple*

- **Formativa:** La función que cumple es la de ir suministrando información a medida que avanza el programa de tal modo que puedan tomarse decisiones pertinentes para cambiar las acciones en curso.

- **Sumativa:** Se realiza una vez ha concluido el programa, y pretende determinar los resultados obtenidos a partir de la implementación de sus actividades, indicando si ha

sido capaz de dar respuesta a las necesidades que lo generaron.

- **De impacto:** Cuando lo que se trata es de comprobar y valorar los efectos o la repercusión que un determinado hecho ha tenido sobre el medio en el que aconteció.

5.4.3 *Según la procedencia de los evaluadores*

- **Externa:** Quienes toman la iniciativa de efectuar la evaluación deben ser externos a ella, con esto se pretende lograr máxima objetividad; de ahí que la tarea de evaluar la emprendan expertos que no han participado en el proceso de planificación ni de ejecución, pese a que ello suponga un desconocimiento de los acontecimientos.

- **Interna:** Es la que efectúan profesionales pertenecientes a la propia institución pero que no intervienen en el desarrollo del programa, de modo que pueden valorar objetivamente tanto el trabajo realizado o el proceso seguido como los resultados obtenidos a fin de facilitar las decisiones pertinentes.

- **Mixta:** También denominada coevaluación, pretende ser una combinación entre los dos tipos antes descritos. Se trataría de efectuar tanto la evaluación interna como la externa para posteriormente contrastar los datos procedentes de ambas y dar cuenta de las divergencias o concordancias.

- **Autoevaluación:** En este caso son los responsables de la ejecución del proyecto quienes llevan a cabo la actividad

evaluativa. Pretenden reflexionar acerca del trabajo realizado o los resultados obtenidos. Presenta ventajas y desventajas similares a las indicadas para la evaluación interna, y, de hecho, a veces se la considera como tal.

5.4.4 *Según el aspecto objeto de evaluación o contenidos*

- **Las necesidades o contexto:** Esta evaluación se hace después de haber estudiado la realidad y es previa a la formulación del proyecto. Se evalúa el contexto y la realidad sobre la que se quiere intervenir y se realiza un diagnóstico de las necesidades de y con el grupo de desarrollo, para dirigir nuestra acción.

- **El diseño o planificación:** Esta es una evaluación del diseño del programa, de su *coherencia y su aplicabilidad.* Puede cumplir una función de ayuda, a la hora de dar forma a propuestas y proyectos y formar decisiones sobre la estructuración de los mismos.

- **El proceso y desarrollo del programa:** Este tipo de evaluación sirve para *guiar el proceso de ejecución del programa*, de manera que se obtenga una información útil para realizar los ajustes convenientes mientras el programa se está llevando a cabo. Busca, pues, explicaciones de lo que pasa, de los *fallos* y lo *cambios* que se producen.

- **Resultados o productos:** Describe y juzga los resultados de un programa, relacionándolos con los objetivos y las necesidades, para evaluar el mérito y valía del programa en su conjunto.

5.5 Evaluación del entorno informático

Desde la perspectiva Informática en general, un software debe tener ventajas que permitan descubrir a partir de conceptos y análisis técnicos, si es óptimo, si tiene proyección futura, si tiene un buen respaldo o si simplemente es un producto de software más sin estándares en su diseño.

Este análisis no considera las fortalezas que pueda tener un software para realizar las tareas para las cuales fue creado. Considera de modo general pero preciso la importancia del *entorno informático* que se recomienda tener en cuenta previo a iniciar la selección de alguna herramienta de software para cualquier objetivo. Los siguientes, son los tópicos básicos que se deben considerar para validar lo señalado.

Diseño estándar: El hardware y el software han evolucionado en forma dinámica y permanente, esto implica estar acorde con los nuevos desarrollos tecnológicos y los nuevos retos asociados con la necesidad de optimizar la eficiencia y eficacia en la producción de bienes y/o en la prestación de los servicios y el mejoramiento de la calidad.

Creado por especialistas: Una de las variables que se debe tener en cuenta es que haya sido creado por especialistas en soluciones del software, lo que garantiza que el nivel de conocimiento de los diseñadores debió obtenerse a partir de las muchas experiencias compartidas con los usuarios.

Lenguaje estándar: La tendencia es generar aplicaciones estándares, que sean fácilmente reconocidas comprendidas y utilizadas con

lenguajes de programación de uso en todo el mundo y que pudiesen ejecutarse en diversas plataformas de sistemas operativos.

Base de datos estándar: Existen muchos motores de bases de datos utilizados para el registro de información, la clave para es seleccionar uno que esté basado en un estándar que administre los datos con la misma interfaz que los otros, como "SQL" y "Oracle", entonces estamos hablando de uno que se ha mantenido en el tiempo y representa un modelo a seguir.

Seguridad: En las plataformas de sistemas modernas un software debe contemplar en su definición de usuarios la modalidad "por perfil de usuario", a fin de otorgar acceso a las funcionalidades del sistema, es decir, dos usuarios pueden tener acceso a un mismo módulo pero no a las mismas opciones; como complemento a la seguridad "por perfiles", también debe ser posible obtener la "Trazabilidad" de Auditoria de usuarios, en la práctica que se pueda saber que usuario realizó una determinada acción y en qué momento.

Funcionalidad Web: Es importante en la actualidad que un sistema pueda ser accedido desde Internet, que toda la aplicación puede ser accedida por internet mediante tecnología I.I.S (Internet Information Services) terminal server, ASP u otras. De todos modos con los recursos que proveen los servicios de Internet es posible acceder a toda aplicación.

Respaldo y Soporte: En cuanto al soporte los fabricantes deben contar con un equipo de consultores que atiendan los requerimientos de sus clientes de cualquier parte del Mundo si se trata de una solución internacional. El acceso a los técnicos que más conocen el software debe ser directo.

5.6 Evaluación de la Arquitectura del Software

La Arquitectura de Software de los Sistemas a ser construidos, es un factor importante a tomar en cuenta para lograr que éste tenga un alto nivel de calidad. Poseer una buena Arquitectura de Software es de suma importancia, ya que ésta es el corazón de todo sistema informático y determina cuáles serán los niveles de calidad asociados al sistema.

El propósito de realizar evaluaciones a la arquitectura, es para analizar e identificar riesgos potenciales en su estructura y sus propiedades, que puedan afectar al sistema de software resultante, verificar que los *requerimientos funcionales* y *no funcionales* estén presentes en la arquitectura, así como determinar en qué grado se satisfacen los atributos de calidad. Cabe señalar que los requerimientos no funcionales también son llamados *atributos de calidad.*

No sirve de nada un sistema que no cumple con los atributos de calidad, por lo que diseñar una correcta arquitectura va a determinar el éxito o fracaso de un sistema de software, en la medida que esta cumpla o no con sus objetivos. Debido a esto "Para reducir tales riesgos, y como buena práctica de ingeniería, es recomendable realizar evaluaciones a la arquitectura".

La diferencia entre *evaluar* y *verificar* es que la evaluación se realiza antes de la implementación de la solución. La verificación es con el producto ya construido.

¿Cuáles son los tipos de Arquitectura que se conocen?

Los tipos de arquitectura que se conocen son:

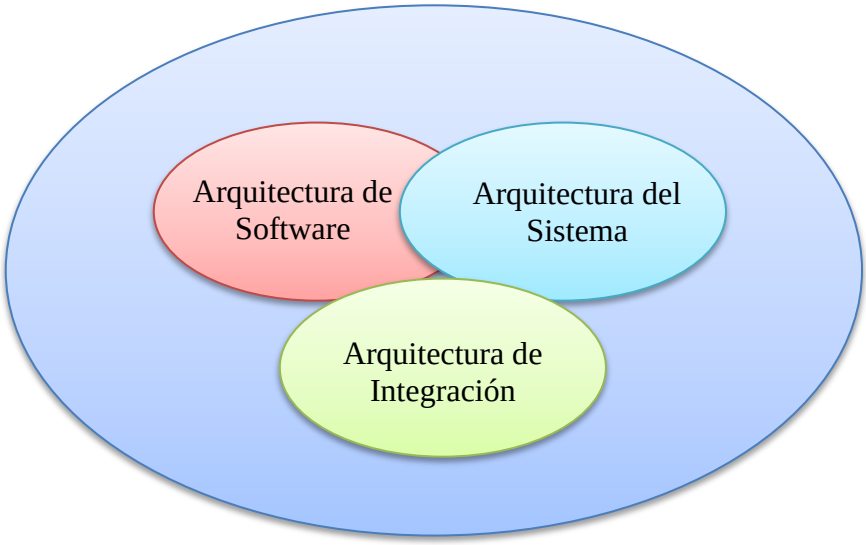


Figura 41: *Tipos de Arquitectura*

¿Cuándo una Arquitectura puede ser evaluada?

Es posible realizarla en cualquier momento, pero se propone dos variantes que agrupan dos etapas distintas: temprana y tarde.

1. **Temprana:** No es necesario que la arquitectura esté completamente especificada para efectuar la evaluación, y esto abarca desde las fases tempranas de diseño y a lo largo del desarrollo.

2. **Tarde:** Cuando ésta se encuentra establecida y la implementación se ha completado. Este caso también se presenta al momento de la adquisición de un sistema ya desarrollado.

¿Quiénes participan en una Evaluación?

Generalmente las evaluaciones a la arquitectura se las hace en forma interna con miembros del equipo de desarrollo, sin embargo puede

haber también situaciones en las que intervengan personas especialistas en el tema o gente externa. Otro que también se interesa por los resultados de una evaluación es el cliente, ya que en dependencia de los resultados puede tomar decisiones de continuar o no con el proyecto.

¿Cuántos métodos de evaluación existen?

Existen varios entre los que tenemos:

1. **ATAM** (Architecture Trade-off Analysis Method): está inspirado en tres áreas distintas: los estilos arquitectónicos, el análisis de atributos de calidad y el método de evaluación SAAM (Software Architecture Analysis Method). El nombre del método ATAM surge del hecho de que revela la forma en que una arquitectura específica satisface ciertos atributos de calidad, y provee una visión de cómo los atributos de calidad interactúan con otros.
   El método de evaluación ATAM comprende nueve pasos, agrupados en cuatro fases (Presentación, Investigación y Análisis, Pruebas y Reporte).

2. **Bosch (2000):** Plantea que el proceso de evaluación debe ser visto como una actividad iterativa, que forma parte del proceso de diseño. Una vez que la arquitectura es evaluada, pasa a una fase de transformación, asumiendo que no satisface todos los requerimientos, luego, la arquitectura transformada es evaluada de nuevo. Este método consta de 5 pasos divididos en dos etapas.

3. **ADR** (Active Design Review): Es utilizado para la evaluación de diseños detallados de unidades del software como los componentes o módulos. Las preguntas giran en torno a la calidad

y completitud de la documentación y la suficiencia, el ajuste y la conveniencia de los servicios que provee el diseño propuesto.

4. **ARID** (Active Reviews for Intermediate Design): Es un método de bajo costo y gran beneficio, es conveniente para realizar la evaluación de diseños parciales en las etapas tempranas del desarrollo. Se basa en ensamblar el diseño de los stakeholders para articular los escenarios de usos importantes o significativos, y probar el diseño para ver si satisface los escenarios.

5. **Losavio (2003):** Es un método para evaluar y comparar arquitecturas de software candidatas, que hace uso del modelo de especificación de atributos de calidad adaptado del modelo ISO/IEC 9126. La especificación de los atributos de calidad haciendo uso de un modelo basado en estándares internacionales ofrece una vista amplia y global de los atributos de calidad, tanto a usuarios como arquitectos del sistema, para efectos de la evaluación.

COMPARACIÓN ENTRE LOS MÉTODOS DE EVALUACIÓN

	ATAM	SAAM	ARID	Bosch(2000)	Losavio(2003)
Atributos de Calidad Contemplados	Modificabilidad Seguridad Confiabilidad Desempeño	Modificabilidad Funcionabilidad	Conveniencia del diseño evaluado	Seleccionados por el arquitecto, de acuerdo a la importancia sobre el sistema	Funcionabilidad Confiabilidad Usabilidad Eficiencia Mantenimiento Portabilidad
Objetos Analizados	Estilos Arquitectónicos, Documentación, Flujo de Datos y Vistas Arquitectónicas	Documentación, y Vistas Arquitectónicas	Especificación de los componentes	Estilos Arquitectónicos, Vistas Arquitectónicas, Patrones Arquitectónicos, Patrones de Diseño y Patrones de Idioma	Especificación de Atributos de Calidad
Etapas del Proyecto en las que se Aplica	Luego que el diseño de la arquitectura ha sido establecido	Luego que la arquitectura cuenta con funcionalidad ubicada en módulos	A lo largo del diseño de la arquitectura	Luego que el diseño de la arquitectura ha sido establecido	Luego que el diseño de la arquitectura ha sido establecido
Enfoques Utilizados	Árbol de Utilidad y lluvia de ideas para articular los requerimientos de calidad. Análisis arquitectónico que detecta puntos sensibles, puntos de balance y riesgos.	Lluvia de ideas para escenarios y articular los requerimientos de calidad. Análisis de los escenarios para verificar funcionalidad o estimar el costo de los cambios.	Revisiones de diseño, lluvia de ideas para obtener escenarios.	Análisis de perfiles	Análisis y comparación de los resultados para las arquitecturas candidatas

Tabla 5: *Comparación entre los métodos de evaluación*

Existen otros métodos de evaluación de arquitectura que evalúan un atributo de calidad específico:

1. **ALMA** (Architecture Level Modifiability Analysis): El atributo de calidad que analiza este método es la *facilidad de modificación*. Es decir la capacidad de un sistema para ser ajustado debido a cambios en los requerimientos, o en el entorno, así como la adición de nuevas funcionalidades. Este método consta de cinco pasos como se muestra en la siguiente figura:

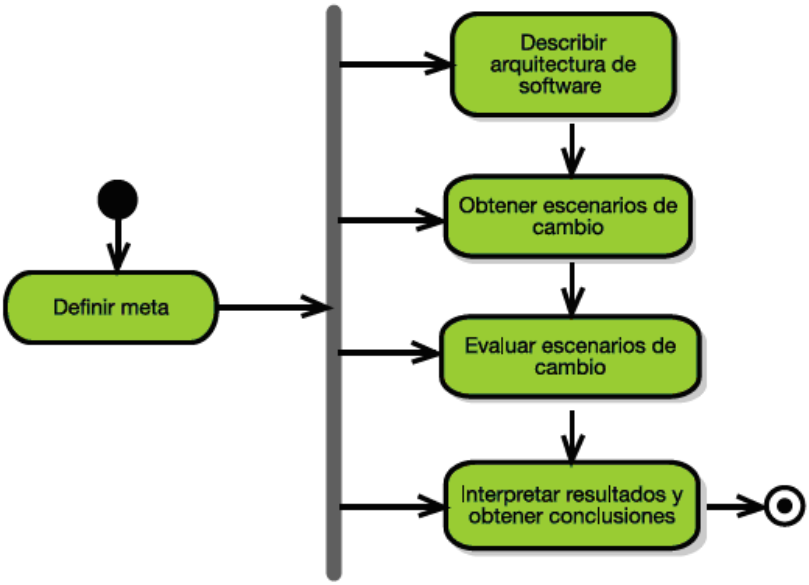


Figura 42: *Método ALMA*

2. **PASA** (Performance Assessment of Software Architecture): El atributo de calidad que analiza este método es el *desempeño*. Se interesa en saber qué tanto tiempo le toma al sistema software responder cuando una o varios eventos ocurren, así como determinar el número de eventos procesados en un intervalo de tiempo dado. Este método también se basa en escenarios y puede aplicarse de forma temprana o tardía. Uno de los requisitos o

precondiciones que presenta, es que la arquitectura debe estar previamente documentada y en caso de que no esté completa se debe extraer el resto de la información a los miembros del equipo.

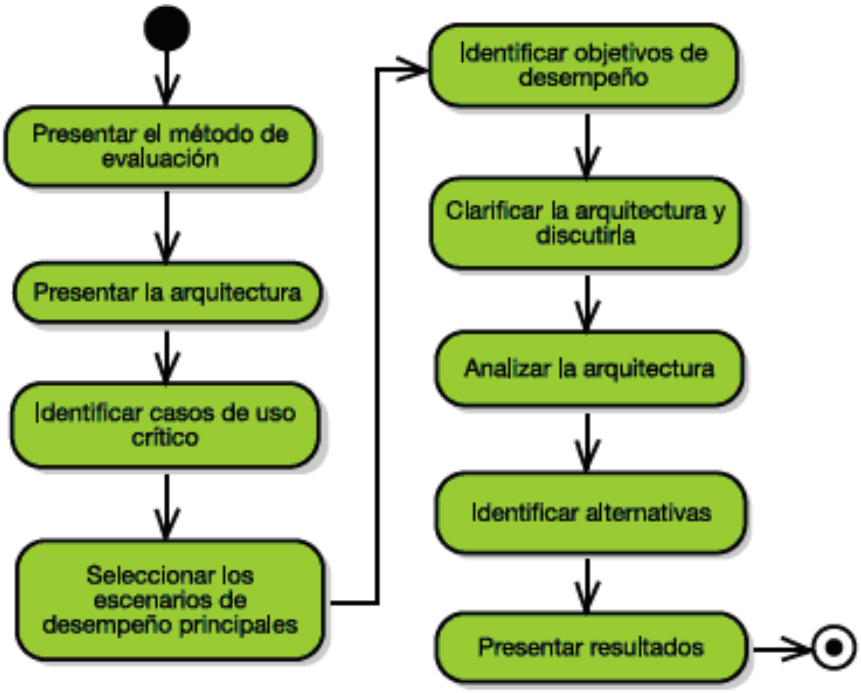


Figura 43: *Método PASA*

3. **SALUTA** (Scenario based Architecture Level Usability Analysis): Es el primer método desarrollado para evaluar arquitecturas desde la perspectiva de la facilidad del uso del sistema.

   Analiza *cuatro atributos* que están directamente relacionados con la facilidad de uso de un sistema de software: *facilidad de aprendizaje, eficiencia de uso, confiabilidad y satisfacción*. El mismo se basa al igual que los dos métodos anteriores en escenarios, que en este caso, son escenarios de uso que agrupan uno a más perfiles de uso valga la redundancia, donde cada uno representa la facilidad de uso requerida por el sistema. Se

recomienda utilizarlo una vez que se ha especificado la arquitectura, pero antes de implementar, consta de cuatro pasos como se muestra a continuación:

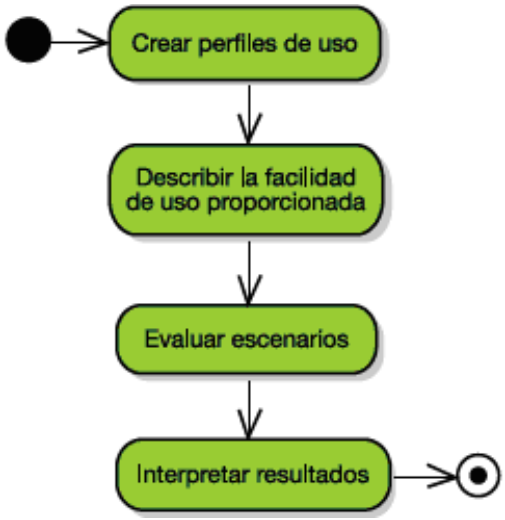


Figura 44: *Método SALUTA*

4. **SNA** (Survivable Network Analysis): Es un método desarrollado por el CERT (Computer Emergency Response Team) que forma parte del SEI (Software Engineering Institute). Este método ayuda a identificar la capacidad de supervivencia en un sistema, analizando su arquitectura. La supervivencia es la capacidad que tiene un sistema para completar su misión a tiempo, ante la presencia de ataques, fallas o accidentes. Para evaluar esta supervivencia SNA utiliza tres propiedades claves: Resistencia, Reconocimiento y Recuperación. Este procedimiento puede ser realizado: después de la especificación de la arquitectura, durante la implementación de esta, o posteriormente.

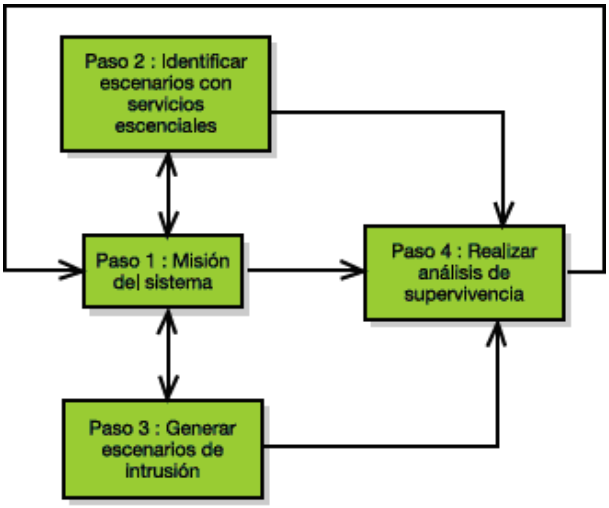


Figura 44: *Método SNA*

COMPARACIÓN ENTRE LOS MÉTODOS

	ALMA	PASA	SALUTA	SNA
Meta	Predecir el costo de mantenimiento, evaluar riesgos, comparación entre arquitecturas.	Analizar la arquitectura con respecto a los objetivos de desempeño de un sistema.	Predecir la facilidad de uso en un sistema analizando la arquitectura.	Identificar la capacidad de supervivencia en un sistema ante la presencia de ataques, fallas o accidentes.
Atributo de calidad	Facilidad de modificación	Desempeño	Facilidad de uso	Supervivencia
Técnica de evaluación	Escenarios de cambio	Escenarios	Escenarios de uso	Escenarios de uso normales, escenarios de intrusión.
Entradas	Especificación de la arquitectura, requerimientos no funcionales.	Especificación de la arquitectura.	Especificación de la arquitectura, requerimientos no funcionales relacionados con la facilidad de uso.	Especificación de la arquitectura
Salidas	Dependiendo de la meta de evaluación se generan los resultados.	Hallazgos encontrados, pasos específicos a seguir y recomendaciones.	Grado de facilidad de uso que soporta la arquitectura evaluada.	Modificaciones recomendadas a la arquitectura y mapa de supervivencia.
Personas involucradas	Arquitecto y equipo de desarrollo	Arquitecto, equipo de desarrollo y administradores del proyecto.	Arquitecto, ingenieros de requerimientos o ingenieros responsables por la facilidad de uso.	Arquitecto, diseñador principal, propietarios del sistema, usuarios.
Duración	No especificado	7 días	No especificado	No especificado
Validación del método	Sistemas de control embebido, sistemas médicos, telecomunicaciones, sistemas administrativos.	Sistemas basados en Web, aplicaciones financieras y sistemas en tiempo real.	Algunos casos de estudio que incluyen principalmente sistemas Web.	Sistemas comerciales y de gobierno.

Tabla 6: *Comparación entre los métodos ALMA, PASA, SALUTA y SNA*

5.7 Evaluación de la Infraestructura del Hardware

Entendemos por infraestructura a los componentes (middleware/sistemas operativos), nodos (hardware) y configuraciones de red que nos permiten alocar a las unidades de despliegue

¿Cuáles son los artefactos que se usan en la Infraestructura?

1) Componentes
   - Sistemas Operativos
   - Virtual Machines
   - Middleware (Application Servers, Containers)
2) Nodos
   - Equipos/Servers/Desktops/Dispositivos/Racks/Chassis
   - Procesadores/Nucleos
   - Memoria
   - Storages
3) Networking
   - Routers/Switchs
   - DAS/NAS
   - Firewalls/DMZ

¿Que es el Hardware Sizing?

Se entiende por *Hardware Sizing* la estimación realizada con el fin de asignar equipamiento para una aplicación, para lo cual se debe contar con la siguiente información previa al análisis:

- Estimado de cantidad de usuarios que van a utilizar la aplicación, (dependiendo del tipo de aplicación), este dato nos definirá las necesidades en cuanto a IO (redes y disco), por ejemplo en

aplicaciones como un datawarehouse, el número de usuarios será reducido, pero las necesidades de IO de disco disco serán grandes, tanto en espacio como en troughput.

- Tipo de aplicación (página Web, Base de datos, OLAP, OLTP, etc), esto nos brindará la base para poder estimar los requerimientos principales a tener en cuenta.
- Espacio requerido en disco, especialmente en bases de datos.
- Crecimiento a lo largo de los años del uso de la nueva aplicación

En base a los datos provistos, se realizará una estimación del hardware que se requiere, generalmente el proveedor del sistema nos dará una lista de hardware y sistema operativos sobre el que se puede ejecutar. De ser posible, es mejor probar el sistema sobre el hardware estimado, con algunos usuarios para corregir cualquier desvío.

5.8 Los instrumentos de evaluación

Entre los instrumentos más usados están las plantillas de evaluación, Checklists, tablas, cuestionarios de valoración en los cuales el resultado indica el grado de conformidad con las afirmaciones propuestas. Es conveniente incluir en las plantillas de evaluación preguntas abiertas que permitan al usuario referirse a aspectos de la aplicación no considerados en las categorías propuestas en las preguntas cerradas.

Los aspectos a evaluar se deben ordenar y clasificar en distintas categorías tales como logro de los objetivos, aspectos técnicos, desarrollo de contenidos. Por otra parte, los cuestionarios utilizados en la evaluación deben estar redactados con un vocabulario sin ambigüedades y adecuado al de los integrantes que se considere.

Durante la evaluación es importante distinguir entre cuatro conceptos muy relacionados pero distintos, entre ellos:

- ***Error:*** acción humana que produce una falta.
- ***Falta*****:** algo que está mal en un producto (modelo, código, documento, etc.).
- ***Fallo*****:** manifestación de una falta.
- ***Defecto*****:** error, falta o fallo.

6. CONCLUSIONES DEL ESTADO DEL ARTE

La era actual de la informática nos permite situarnos en escenarios virtuales simulando situaciones, sucesos y problemas reales. Con la tecnología existente, se facilita el planteamiento de una variedad de cursos de acción para reaccionar frente a las diferentes circunstancias y problemas que se presentan, para de esta manera ofrecer soluciones más viables.

Los juegos de guerra constituyen en la mayoría de los ejércitos la herramienta más importante para la preparación de los comandantes y sus estados mayores, en el proceso de toma de decisiones militares; por lo que el uso de la simulación se constituye hoy en día como la herramienta más aprovechable para economizar recursos, reducir costos y puede ser utilizada para comprender mejor la guerra, permitiendo prácticas en procedimientos u operaciones que serían imposibles de ejecutar con equipo real en tiempo de paz, a la vez que se reduce el impacto del entrenamiento militar en el medio ambiente.

Pero para que esto ocurra el desarrollo de este sistema de simulación debe cumplir con los objetivos planteados, los procesos seguidos en su implementación y las modificaciones o mejoras introducidas, valorando con

ello la discrepancia entre lo diseñado y la realidad, a fin de tomar decisiones sobre el diseño o rediseño del programa en aras a su optimización.

La idea de la mejora se encuentra inevitablemente relacionada con la idea de la evaluación. No es posible mejorar sin evaluar lo que pretendemos, lo que hacemos y lo que conseguimos, la evaluación se convierte así en un instrumento de control y en una parte imprescindible de cualquier tipo de proceso de gestión basado en el esfuerzo y el compromiso de una organización por mejorar, y en un indicador básico de la calidad y de la mejora continua.

La evaluación del software se constituirá en un proceso para determinar el grado de adecuación de dicho software al contexto requerido, para lo cual se consideraran dos tipos de evaluación: una interna y otra externa.

- La *evaluación interna*, se realizara con el equipo responsable del desarrollo del software y permitirá realizar, de ser el caso, los ajustes que se consideren necesarios antes de presentar el producto final a los usuarios finales quienes serán los responsables de realizar la evaluación externa.

- La *evaluación externa* se realizara con los USUARIOS una vez terminado el desarrollo y serán quienes evalúen en base a listas de preguntas, de tipo cerrado (generalmente *"checklist"*) que sean alcanzados los objetivos propuestos, y detectar errores imprevistos. Permitirá además obtener las sugerencias de los usuarios potenciales, quienes serán en definitiva los usuarios finales del software

En el caso de las metodologías no existe un criterio unificado para seleccionar una metodología, por ello, el presente trabajo se orienta a la formulación inicial, en base a la información existente a la fecha, a la

experiencia personal y a la formulación de dos procedimientos al respecto: selección por ***criterios de presencia*** y por ***conocimiento***.

La evaluación de una arquitectura de software pretende medir propiedades del sistema en base a especificaciones abstractas, como por ejemplo los diseños arquitectónicos, por ello, la intención es más bien la evaluación del potencial de la arquitectura diseñada para alcanzar los atributos de calidad requeridos.

La evaluación de una infraestructura de hardware nos permitirá obtener como resultado el análisis de la infraestructura y este nos devolverá parámetros para los distintos componentes de la arquitectura IT a implementar.

CAPITULO II

DESCRIPCIÓN DEL PROBLEMA

7. PRESENTACIÓN DEL PROBLEMA

7.1 Resumen

El empleo de juegos para adiestrar a líderes no es algo nuevo ni revolucionario, a pesar de haber sido inicialmente empleados principalmente para diversión, se descubrió que estos ejercicios eran útiles para el adiestramiento en ejercicios militares, para la formación de líderes y para la toma de decisiones bajo condiciones reales. Estos ejercicios de decisión permiten a los usuarios explorar las alternativas con respecto a problemas planteados, generar discusiones, y practicar la toma de decisiones bajo una variedad de situaciones y condiciones.

Los líderes fortalecen sus habilidades para tomar decisiones rápidamente por practicar la visualización de una situación repetidamente, describiendo esta visualización para otros, y proporcionando orientación a sus subalternos mediante la emisión de órdenes e instrucciones. Normalmente, llevar a cabo este tipo de adiestramiento fortalece la habilidad del individuo para evaluar la situación y emitir órdenes.

7.2 La problemática actual

En la actualidad en la mayoría de los ejércitos todavía se realizan los Juegos de Guerra en forma manual accionando al enemigo subjetivamente y evaluando de forma no muy eficiente las decisiones del Comandante, su Estado Mayor y Plana Mayor, en cada una de las acciones y situaciones de guerra simuladas.

Al no tener automatizadas e identificadas todas las tareas y procedimientos, que conllevan la Planificación, Control y Evaluación de los Juegos de Guerra, los procesos generados se tornan lentos, burocráticos, y casi siempre existe falta de información, lo que dificulta la optimización y ejecución del desarrollo del montaje y ejecución de ejercicios aplicativos de conducción militar y juegos de Guerra; esta modalidad de trabajo ha generado una serie de problemas como:

1. Al ser conducidos de manera manual, se incurre en la subjetividad específicamente durante las evaluaciones, ocasionando que los resultados no se acerquen a la realidad.
2. Inadecuada distribución física de los Cuarteles Generales y puestos de trabajo de los Estados Mayores y Planas Mayores.
3. Dificultad de consolidación de la información que se crea en el entrenamiento de acciones tácticas simuladas.
4. Generación de grandes volúmenes de información innecesaria para el proceso de enseñanza aprendizaje en la conducción de acciones tácticas.
5. No existe una entidad responsable de generar la doctrina institucional que se requiere para de la aplicación de los sistemas de entrenamiento simulado, en las áreas de la conducción militar
6. Dificultad de administrar, controlar y evaluar los ejercicios aplicativos para la formación de los mandos actuales y futuros.
7. No existe un organismo de investigación y desarrollo técnico que permita crear sistemas computacionales que apoyen el entrenamiento y perfeccionamiento del personal militar.

7.3 El desarrollo y su complejidad

Debemos reconocer desde un principio que los juegos de guerra necesariamente tienen un alcance amplio, aplicaciones limitadas y propósitos generalizados. Si bien los resultados de los juegos de guerra conducidos en el pasado se basaban en los datos sintetizados en tablas y cálculos manuales, los actuales modelos de combate y bases de datos computarizadas han llevado el arte de los juegos de guerra a nuevas dimensiones.

Para alcanzar el nivel deseado de análisis, muchas veces se emplean programas especializados, modelos y cálculos informáticos complejos, cartografía específica, así como desarrolladores con un gran conocimiento en diversos tipos de software, para programar los cálculos de un modelo específico.

La tentación de confiar en la computadora, sin considerar a los seres humanos, constituye uno de los mayores retos que se debe enfrentar al desarrollar los juegos de guerra, debido a que los modelos matemáticos desarrollados incorporan los resultados de eventos en diferentes partes del juego, el análisis de estos resultados exige por su parte excelentes aptitudes de los instructores para la evaluación y análisis de una amplia diversidad de operaciones militares. De ahí que es más acertado describir la mayor parte de los juegos de guerra como asistidos por modelos antes que .impulsados por computadora. Si hemos de obtener importantes innovaciones de los juegos de guerra o, cuando menos, si vamos a agudizar nuestro entendimiento del cómo los nuevos conceptos operacionales pueden ser implementados, tales adelantos no se conseguirán a través del programa empleado para producir los cálculos sino que se obtendrán producto del análisis del desarrollo del juego realizado por mentes humanas.

Los juegos de guerra, al igual que la doctrina institucional, la tecnología, y los conceptos operacionales, se someten a un proceso evolutivo que puede resultar en muy diferentes métodos de conducir la guerra, por lo que el software debe ser abierto con componentes modulares que aseguren una fácil expansión (adición de otros centros de control, estaciones de trabajo, funciones adicionales, etc.), así como escalable a fin de que pueda ser extendido con funciones adicionales y permita su supervivencia en el tiempo.

7.4 La problemática y dificultades en la evaluación del software

La evaluación es un proceso susceptible de planificación, que entraña organizar los distintos elementos que intervienen en un juego, sistematizar las fases de su desarrollo, temporalizar las secuencias planificadas y proveer los recursos necesarios para que la misma pueda llevarse a cabo.

Todo ello supone que, además de vencer las resistencias, obstáculos y dificultades que genera su organización y su desarrollo, se debe proveer de un organigrama evaluativo (asignación de tareas y responsabilidades, competencia técnica) y propiciar los instrumentos, técnicas y metodologías idóneas en cada momento, dentro de un modelo, ajustado y coherente con los propósitos evaluativos.

Es en este contexto de referencia en el que queremos ubicarnos a la hora de analizar la problemática y las dificultades que entraña el proceso de evaluación, tanto en su planificación como en su realización práctica. Se trata con ello de evidenciar los elementos que deben ser objeto de atención en el proceso de configuración y desarrollo de la evaluación y no tanto el justificar la dificultad de la misma.

También habrá que considerar que los programas son deudores de la forma como se confeccionaron y de los objetivos y finalidades que se les asignaron. Una indefinición de metas o la falta de claridad en las acciones que el propio programa plantea es lógico que sean una fuente de problemas y, en último extremo, una dificultad para la evaluación.

Al referirnos a los problemas en la evaluación de programas, estamos haciendo alusión a aquellos obstáculos que podemos encontrar a lo largo del proceso evaluativo, que afectan tanto a la viabilidad, factibilidad y utilidad de la misma, y, por tanto, repercuten en el proceso de recogida de información, la emisión del juicio de valor y en la consecuente toma de decisiones.

Para una mejor comprensión de las diferencias entre las diferentes tecnologías que se utilizan para realizar juegos de guerra, a continuación se realiza un cuadro comparativo en donde se hace un análisis de las mismas:

SISTEMAS DE SIMULACION (ECUADOR, COLOMBIA)	**SISTEMAS COMPUTARIZADOS (BRASIL, CHILE, ARGENTINA, EL SALVADOR)**
Se atiene a los estándares de la OTAN para simulación, incluyendo los protocolos de comunicación para simulaciones DIS. y HLA.	Necesitaría ser desarrollado.
Se basa en una arquitectura de componentes: **Sensores:** proveen modelos del ambiente simulado. **Controladores:** utilizan la información provista por los sensores para desempeñar tareas específicas. **Actuadores**: son los componentes que realizan los cambios al ambiente simulado.	Necesita ser desarrollado por personal técnico especializado de varias áreas y fusión de tecnologías.
Dispone de una base de entidades preestablecidas que son configurables para la simulación de varios sistemas y armamentos.	Necesita ser desarrollado y requiere recursos humanos y técnicos especializados.

Configuración de escenarios para múltiples combinaciones: hombre vs hombre, hombre vs. Equipo Pc, o equipo vs equipo para obtener y analizar los mejores resultados posibles de un escenario.	Necesita ser desarrollado, es un módulo propio de un simulador.
Pueden simular el rendimiento, poder de combate y las cualidades de nuevos sistemas de armamento antes de ser comprados (Adquisición Basada en Simulación).	Necesita ser desarrollado, es un módulo propio de un simulador.
Las características de los sensores, los modelos de daños, las conductas (acciones), pueden configurarse en el motor, por medio de variables parametrizables, editando los archivos que contienen la información de estos parámetros, sin necesidad de tener que realizar para esto tareas de programación.	Necesita ser desarrollado y posteriormente requiere de tareas adicionales de programación, es un módulo propio de un simulador.
Son sistemas que proveen varios modelos de distintos tipos de entidades, de superficie, aéreas, terrestres, de forma humana, etc.	Necesita ser desarrollado, es un módulo propio de un simulador.
El nivel de adaptación y personalización según utilidades (módulos) del programa.	Muy adaptable a necesidades locales en vista de que se puede programar.
Ofrece tratamiento de los datos cartográficos transparente para el usuario (no necesita realizar ningún tratamiento, el software se encarga).	Se necesita adaptar los datos cartográficos al software, requiere especialistas cartográficos.
No requiere personal para la importación de la cartografía disponible en el IGM porque trabaja con estándares internacionales.	Requiere personal para preparar la cartografía de acuerdo a los requerimientos del desarrollo del sistema.
La arquitectura del motor de simulación almacena los datos relacionados a la configuración de todas las entidades en una base de datos de parámetros, los cuales se cargan al momento de ser añadidos al escenario.	Se dispone de una base de datos relacional para almacenar, organizar, administrar, recuperar y efectuar las consultas sobre los datos del juego, sean estos gráficos y no gráficos.
El encargado de gestionar los recursos es el Kernel (Motor) y no necesita un administrador de Base de Datos.	La estructura de los datos se maneja a través de las tablas, registros, atributos y necesita un administrador de Base de Datos.

Los modelos matemáticos están incluidos en el Kernel	Los modelos matemáticos a desarrollarse permitirán realizar las operaciones y cálculos necesarios para ejecutar y simular la acción ordenada por el usuario.
Interfaces graficas propias - GUI	Interfaces graficas GUI a desarrollarse dependientes de otra herramienta. Requiere más tiempo y técnicos especializados.
Lenguaje de programación para personalización único.	Lenguaje de programación para personalización variado. Se necesita de personal técnico adicional.
No requiere administración y mantenimiento especializado para la red de datos, backup de base de datos.	Se necesita personal para la administración, mantenimiento de la base de datos y base cartográfica.
El tiempo que se requiere para el diseño y la personalización es menor.	Se requiere mayor tiempo para: análisis, diseño, desarrollo e implementación de la aplicación.
Los escenarios reaccionan de manera dinámica a los cambios de las circunstancias y/o elementos particulares, y el efecto de éstos cambios se puede visualizar de manera inmediata en el escenario (Una unidad se desplaza en forma dinámica con efectos visuales simulados).	Para resolver un movimiento de una Unidad, se utilizan menús que permiten interactuar con el programa de modelos y procesos y mediante el uso de un lenguaje de consulta, se relacionan con la Base de Datos. (Una unidad se desplaza realizando saltos de ubicación).
La arquitectura del sistema permite a los usuarios tener la facilidad de añadir, reemplazar, o modificar las dinámicas de los vehículos, las conductas y tácticas, los modelos de daños, los sensores, las reacciones, y las características de las entidades, incluyendo el armamento.	Necesita ser desarrollado por personal técnico especializado de varias áreas y fusión de tecnologías, es un módulo propio de un simulador.
Tiene capacidades de generación y ejecución de escenarios de campos de batalla para Guerra Regular, la cual puede también mediante programación, ampliar su espectro para abarcar escenarios de Guerra Irregular y Guerra Urbana.	Necesita ser desarrollado por personal técnico especializado de varias áreas y fusión de tecnologías, es un módulo propio de un simulador.
Opera bajo tácticas, procedimientos y doctrinas de combate que han sido sometidas a un proceso de verificación y validación.	Necesita ser desarrollado por personal técnico especializado de varias áreas y fusión de tecnologías.

Permite modificar escenarios y evaluar diferentes alternativas.	Permite modificar escenarios y evaluar diferentes alternativas.
Permite la repetición de ejercicios sin pérdida de equipo o personal.	Permite la repetición de ejercicios sin pérdida de equipo o personal.
Simula batallas en Ejercicios Asistidos por Computadoras.	Simula batallas en Ejercicios Asistidos por Computadoras.
Provee apoyo a ejercicios desde Pelotón a División.	Provee apoyo a ejercicios desde Pelotón a División.
Graba un registro de la información y resultados del ejercicio de combate que luego se repite en un foro de análisis y critica	Graba un registro de la información y resultados del ejercicio de combate que luego se repite en un foro de análisis y critica.
Opera con una base de datos de terrenos y armamentos que son específicos del país.	Opera con una base de datos de terrenos y armamentos que son específicos del país
Opera en computadoras personales utilizando una arquitectura abierta y de protocolos estándar en la red de informática.	Opera en computadoras personales utilizando una arquitectura abierta y de protocolos estándar en la red de informática
Simula operaciones tácticas para evaluar conceptos que existen o que están planeados.	Simula operaciones tácticas para evaluar conceptos que existen o que están planeados
Permite la participación en ejercicios dentro de una red global de informática.	Permite la participación en ejercicios dentro de una red global de informática
Trabaja con representaciones exactas de áreas geográficas (base de datos del terreno) al igual que con modelos exactos de edificios dentro de estas áreas geográficas.	Trabaja con representaciones exactas de áreas geográficas (base de datos del terreno) al igual que con modelos exactos de edificios dentro de estas áreas geográficas
Los comportamientos de las unidades simuladas y las características de los armamentos han pasado por un proceso de verificación.	Los comportamientos de las unidades simuladas y las características de los armamentos han pasado por un proceso de verificación
Arquitectura abierta a programación externa.	Arquitectura abierta a programación externa

Tabla 7: *Comparación entre los sistemas Simulados y Computarizados*

CAPITULO III

SOLUCIÓN PROPUESTA

8. PROPUESTAS DE SELECCION

8.1 PROPUESTA PARA LA SELECCIÓN DEL TIPO DE EVALUACIÓN A REALIZAR

En este documento se plantea como producto inicial la selección del tipo de evaluación a realizar, para establecer un recurso que aporte criterios y orientaciones generales, como base de la propuesta de un modelo de evaluación de programas. Los distintos tipos de evaluación que se va a utilizar estarán determinados por la siguiente tabla:

Según el momento en que se evalúa	**SI / NO**
- Ex ante	
- Durante	
- Final, ex post o de impacto	
Según las funciones que cumple	
- Formativa	
- Sumativa	
- De impacto	
Según la procedencia de los evaluadores	
- Externa	
- Interna	
- Mixta	
- Autoevaluación	

Según el aspecto objeto de evaluación o contenidos	
- Las necesidades o contexto	
- El diseño o planificación	
- El proceso y desarrollo del programa	
- Resultados o productos	

La información de cada uno de los ítems se encuentra detallada en el numeral 5.4 - Tipo de evaluación.

8.2 PROPUESTA PARA LA EVALUACIÓN DEL ENTORNO INFORMÁTICO

Esta escala de evaluación se establece con la finalidad de aportar un instrumento donde se realice una valoración del entorno informático, que permita tomar decisiones para su adquisición y uso posterior a partir de conceptos y análisis técnicos; si es óptimo, si tiene proyección futura, si tiene un buen respaldo o si simplemente es un producto de software más sin estándares en su diseño, por lo tanto se proporciona una lista de control (conjunto de ítems organizados según ciertos criterios, que guían el proceso de selección y se valoran con un SI o NO).

La propuesta de evaluación de entornos informáticos que se va a utilizar estará determinada por la siguiente tabla:

		SI / NO
1	Diseño estándar	
2	Creado por especialistas	
3	Lenguaje estándar	
4	Base de datos estándar	
5	Seguridad	
6	Funcionalidad Web	
7	Respaldo y Soporte	

Para ponderar los valores en porcentajes se considerara un total de 100% para los 7 ítems considerados como SI. En caso de existir valores con NO se calculara a través de una regla de tres, a fin de obtener un valor en términos de porcentaje.

80 % - 100 % ACEPTABLE

60 % - 79 % MEDIANAMENTE ACEPTABLE (con correcciones)

40 % - 59% NO ACEPTABLE

La información de cada uno de los ítems se encuentra detallada en el numeral 5.5 Evaluación del entorno informático.

8.3 PROPUESTA DE LOS FACTORES QUE PERMITIRAN DETERMINAR LA CALIDAD DEL SOFTWARE

La calidad del *software* es medible y varía de un sistema a otro o de un programa a otro. La obtención de un *software* con calidad implica la utilización de metodologías o procedimientos estándares para el análisis, diseño, programación y prueba del *software* que permitan uniformar la filosofía de trabajo, en aras de lograr una mayor confiabilidad, mantenibilidad y facilidad de prueba, a la vez que

eleven la productividad, tanto para la labor de desarrollo como para el control de la calidad del *software*.

Operaciones del producto	**Características operativas**	**SI / NO**
Corrección ¿Hace lo que se le pide?	El grado en que una aplicación satisface sus especificaciones y consigue los objetivos encomendados por el cliente.	
Fiabilidad ¿Lo hace de forma fiable todo el tiempo?	El grado que se puede esperar de una aplicación lleve a cabo las operaciones especificadas y con la precisión requerida.	
Eficiencia ¿Qué recursos hardware y software necesito?	La cantidad de recursos hardware y software que necesita una aplicación para realizar las operaciones con los tiempos de respuesta adecuados	
Integridad ¿Puedo controlar su uso?	El grado con que puede controlarse el acceso al software o a los datos a personal no autorizado.	
Facilidad de uso ¿Es fácil y cómodo de manejar?	El esfuerzo requerido para aprender el manejo de una aplicación, trabajar con ella, introducir datos y conseguir resultados.	

Revisión del producto	**Capacidad para soportar cambios**	**SI / NO**
Facilidad de mantenimiento ¿Puedo localizar los fallos?	El esfuerzo requerido para localizar y reparar errores	
Flexibilidad ¿Puedo añadir nuevas opciones?	El esfuerzo requerido para modificar una aplicación en funcionamiento.	
Facilidad de prueba ¿Puedo probar todas las opciones?	El esfuerzo requerido para probar una aplicación de forma que cumpla con lo especificado en los requisitos.	

Transición del producto	Adaptabilidad a nuevos entornos	SI / NO
Portabilidad ¿Podré usarlo en otra máquina?	El esfuerzo requerido para transferir la aplicación a otro hardware o sistema operativo.	
Reusabilidad ¿Podré utilizar alguna parte del software en otra aplicación?	Grado en que partes de una aplicación pueden utilizarse en otras aplicaciones.	
Interoperabilidad ¿Podrá comunicarse con otras aplicaciones o sistemas informáticos?	El esfuerzo necesario para comunicar la aplicación con otras aplicaciones o sistemas informáticos	

Para ponderar los valores en porcentajes se considerara un total de 100% para los 11 ítems considerados como SI. En caso de existir valores con NO se calculara a través de una regla de tres, a fin de obtener un valor en términos de porcentaje.

80 % - 100 % ACEPTABLE

60 % - 79 % MEDIANAMENTE ACEPTABLE (con correcciones)

40 % - 59% NO ACEPTABLE

8.4 PROPUESTA DE SELECCIÓN DE UNA METODOLOGÍA DE DISEÑO Y DESARROLLO

No existe una metodología de software universal. Las características de cada proyecto (equipo de desarrollo, recursos, etc.) exigen que el proceso sea configurable. La metodología también está influenciada por consideraciones de tamaño y estructura de la organización. La organización previa al desarrollo tiene dos opciones:

- Crear y desarrollar la metodología apropiada (métodos de gestión, técnicas de desarrollo y soporte automatizado).

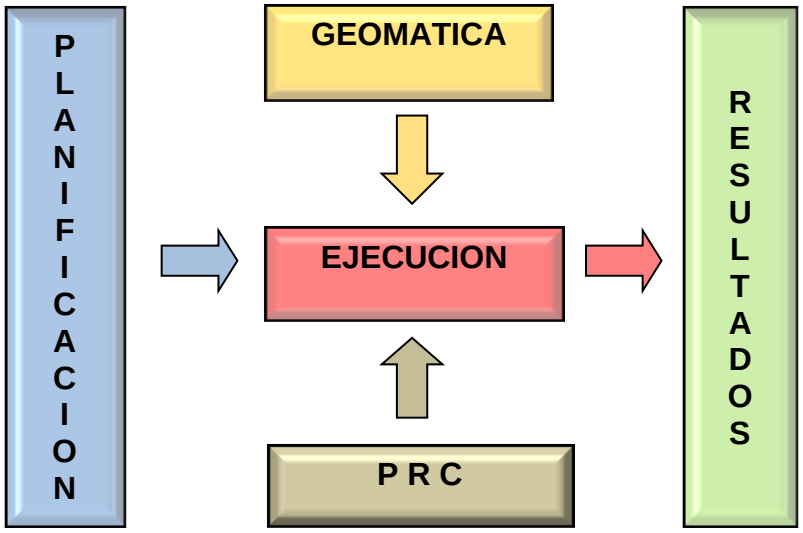
- Analizar y evaluar las metodologías existentes y adoptar la que más se ajuste a sus necesidades.

La metodología que se seleccione va a permitir establecer el camino para desarrollar software de manera sistemática, proporcionando un estándar de trabajo a la organización; la idea no es tratar de ver cuál es mejor o peor, sino de cuando usar la una y cuando la otra. La metodología definirá ***Quién*** debe hacer ***Qué***, ***Cuándo*** y ***Cómo*** debe hacerlo.

La finalidad de la selección de la metodología de desarrollo será garantizar la eficacia (p.ej. cumplir los requisitos iniciales) y la eficiencia (p.ej. minimizar las pérdidas de tiempo) en el proceso de generación de software.

Un sistema para juegos de guerra deberá ser concebido como un sistema auto-contenido, que inicialmente no requiera información directa de otros sistemas pero que a futuro pueda requerir de interfaces de integración para que cada entidad evaluadora (o institución con requisitos especiales) pueda interactuar con el sistema sin la necesidad de contar con una interface de usuario.

De forma general el siguiente diagrama muestra la descomposición de un sistema de juegos de guerra y los módulos básicos que se requerirán desarrollar o evaluar:



En el momento de adoptar un estándar o seleccionar una metodología, se han de considerar unos requisitos deseables, por lo que seguidamente se proponen una serie de criterios de evaluación de dichos requisitos.

1. **La metodología deberá ajustarse a los objetivos generales del desarrollo**

   Cada aproximación al desarrollo de software estará basada en objetivos, por ello la metodología que se elija debe recoger el aspecto filosófico de la aproximación deseada, es decir que los objetivos generales del desarrollo deben estar implementados en la metodología de desarrollo.

2. **La metodología deberá cubrir el ciclo entero de desarrollo de software**

   Para ello la metodología ha de realizar las siguientes etapas:

   - Investigación

- Análisis de requisitos
- Diseño

3. **La metodología deberá integrar las distintas fases del ciclo de desarrollo**

   - **Rastreabilidad.** Es importante poder referirse a otras fases de un proyecto y fusionarlo con las fases previas. Es importante poder moverse no sólo hacia adelante en el ciclo de vida, sino hacia atrás de forma que se pueda comprobar el trabajo realizado y se puedan efectuar correcciones.

   - **Fácil interacción entre etapas del ciclo de desarrollo**. Es necesaria una validación formal de cada fase antes de pasar a la siguiente. La información que se pierde en una fase determinada queda perdida para siempre, con un gran impacto en el sistema resultante.

4. **La metodología deberá incluir la realización de validaciones**

   La metodología deberá detectar y corregir los errores cuanto antes, uno de los problemas más frecuentes y costosos es el aplazamiento de la detección y corrección de problemas en las etapas finales del proyecto. Cuanto más tarde sea detectado el error más caro será corregirlo; por lo tanto cada fase del proceso de desarrollo de software deberá incluir una actividad de validación explícita.

5. **La metodología deberá soportar la determinación de la exactitud del sistema a través del ciclo de desarrollo.**

   La exactitud del sistema implica muchos asuntos, incluyendo la correspondencia entre el sistema y sus especificaciones, así como que el sistema cumple con las necesidades del usuario. Por ejemplo, los métodos usados para análisis y especificación del sistema deberían colaborar a terminar con el problema del entendimiento entre los informáticos, los usuarios, y otras partes implicadas. Esto implica una comunicación entre usuario y técnico, amigable y sencilla, exenta de consideraciones técnicas.

6. **La metodología deberá ser la base de una comunicación efectiva.**

   Deberá ser posible gestionar a los informáticos, y éstos deben ser capaces de trabajar conjuntamente. Ha de haber una comunicación efectiva entre analistas, programadores, usuarios y gestores, con pasos bien definidos para realizar progresos visibles durante la actividad del desarrollo.

7. **La metodología deberá funcionar en un entorno dinámico orientado al usuario.**

   A lo largo de todo el ciclo de vida del desarrollo se deberá producir una transferencia de conocimientos hacia el usuario. La clave del éxito es que todas las partes implicadas han de intercambiar información libremente.

   La participación del usuario es de importancia vital debido a que sus necesidades evolucionan constantemente. Por otra parte la

adquisición de conocimientos del usuario le permitirá la toma de decisiones correctas. Para involucrar al usuario en el análisis, diseño y administración de datos, es aconsejable el empleo de técnicas estructuradas lo más sencillas posible. Para esto, es esencial contar una buena técnica de diagramación.

8. **La metodología deberá especificar claramente los responsables de los resultados**

Deberá especificar claramente quienes son los participantes de cada tarea a desarrollar, deberá detallar de una manera clara los resultados de los que serán responsables.

9. **La metodología deberá poder emplearse en un entorno amplio de proyectos software**

- **Variedad.** Una empresa deberá adoptar una metodología que sea útil para un gran número de sistemas que vaya a construir. Por esta razón no es práctico adoptar varias metodologías en una misma empresa.

- **Tamaño, vida.** Las metodologías deberán ser capaces de abordar sistemas de distintos tamaños y rangos de vida.

- **Complejidad.** La metodología deberá servir para sistemas de distinta complejidad, es decir puede abarcar un departamento, varios departamentos o varias empresas.

- **Entorno**. La metodología deberá servir con independencia de la tecnología disponible en la empresa.

10. La metodología se debe poder enseñar

Incluso en una organización sencilla, serán muchas las personas que la van a utilizar, incluso los que se incorporen posteriormente a la empresa. Cada persona deberá entender las técnicas específicas de la metodología, los procedimientos organizativos y de gestión que la hacen efectiva, las herramientas automatizadas que soportan la metodología y las motivaciones que subyacen en ella.

11. La metodología deberá estar soportada por herramientas CASE

La metodología deberá estar soportada por herramientas automatizadas que mejoren la productividad, tanto del ingeniero de software en particular, como la del desarrollo en general. El uso de estas herramientas reduce el número de personas requeridas y la sobrecarga de comunicación, además de ayudar a producir especificaciones y diseños con menos errores, más fáciles de probar, modificar y usar.

12. La metodología deberá soportar la eventual evolución del sistema

Normalmente durante su tiempo de vida los sistemas tienen muchas versiones, pudiendo durar incluso más de 10 años. Existen herramientas CASE para la gestión de la configuración y otras denominadas "Ingeniería inversa" para ayudar en el mantenimiento de los sistemas no estructurados, permitiendo estructurar los componentes de éstos facilitando así su mantenimiento.

13. **La metodología deberá contener actividades conducentes a mejorar el proceso de desarrollo de software.**

    Para mejorar el proceso es básico disponer de datos numéricos que evidencien la efectividad de la aplicación del proceso con respecto a cualquier producto software resultante del proceso. Para disponer de estos datos, la metodología debe contener un conjunto de mediciones de proceso para identificar la calidad y coste asociado a cada etapa del proceso. Es ideal el uso de herramientas CASE.

De igual manera los atributos deseables que se deberán considerar para el producto final son:

1. **Adecuación**: El sistema satisface las expectativas del usuario.

2. **Mantenibilidad**: Facilidad para realizar cambios una vez que el sistema está funcionando en la empresa del cliente.

3. **Usabilidad**: Es el grado de dificultad en aprender a manejar el sistema por parte de un usuario que no tiene por qué ser programador. Irónicamente se puede decir que este atributo es inversamente proporcional a la resistencia al cambio.

4. **Fiabilidad**: Es la capacidad de un sistema de funcionar correctamente durante un tiempo dado. La diferencia con la corrección es que aquí interesa el tiempo, es decir, no se trata del número absoluto de defectos en el sistema sino de los que se manifiestan en un intervalo de tiempo. Interesan sobre todo:

a. **MTBF**: *Mean Time Between Failures* (Tiempo medio entre fallos)

b. **Disponibilidad**: Probabilidad de que el sistema esté funcionando en un instante dado.

5. **Corrección**: Densidad de defectos mínima.

6. **Eficiencia**: El sistema es capaz de realizar su tarea con el mínimo consumo de recursos necesario.

8.5 Selección de una Metodología de Diseño Estructurada

Esta metodología que surgió a mediados de los años 70, se centraba en las aplicaciones de sistemas de información, luego a mediados de los 80 se introdujeron mejoras que proporcionaban una notación adecuada para los aspectos de control y de comportamiento de los problemas de tiempo real

Existe una gran cantidad de proyectos implementados utilizando estas metodologías, generalmente orientados a la manipulación de datos y gestión. Estas metodologías funcionaban muy bien con los lenguajes de programación estructurados, como por ejemplo el COBOL, C, Pascal, FORTRAN y Modula 2.

La metodología estructurada está orientada a procesos, es decir, se basa en la estructuración y descomposición funcional de problemas en unidades más pequeñas interrelacionadas entre sí. En el caso de pequeños programas, estos principios de organización son eficientes. El programador sólo tiene que crear una serie de instrucciones en un

lenguaje de programación, compilar en la computadora y ésta, a su vez, ejecuta estas instrucciones.

Cuando los programas se vuelven más grandes, cosa que lógicamente sucede, cuando aumenta la complejidad del problema a resolver, la lista de instrucciones aumenta considerablemente, de modo tal que el programador tiene muchas dificultades para controlar ese gran número de instrucciones.

Los programadores pueden controlar, de modo normal, unos centenares de líneas de instrucciones. Para resolver este problema los programas se descompusieron en unidades más pequeñas que adoptaron el nombre de *funciones* (*procedimientos*, *subprogramas* o *subrutinas* en otros lenguajes de programación). De este modo un programa orientado a procedimientos se divide en funciones, de modo que cada función tiene un propósito bien definido y resuelve una tarea concreta, y se diseña una interfaz claramente definida (el prototipo o cabecera de la función) para su comunicación con otras funciones.

Esta metodología clásica presenta ciertos problemas, que han ido haciéndose cada vez más graves, a medida que se construían aplicaciones y sistemas informáticos más complejos, entre los que destacan los siguientes:

- El principal inconveniente de este método de programación es que se obtiene un único bloque de programa, que cuando se hace demasiado grande puede resultar problemático el manejo de su código fuente

- Es difícil modificar y extender los programas, pues suele haber datos compartidos por varios subprogramas, que introducen interacciones ocultas entre ellos

- Es difícil mantener los programas. Casi todos los sistemas informáticos grandes tienen errores ocultos, que no surgen a la luz hasta después de muchas horas de funcionamiento.

- Es difícil reutilizar los programas. Es prácticamente imposible aprovechar en una aplicación nueva las subrutinas que se diseñaron para otra.

Hoy en día las aplicaciones informáticas son mucho más ambiciosas que las necesidades de programación existentes en los años 70 y 80, principalmente debido a las aplicaciones gráficas por lo que las técnicas de programación estructurada no son suficientes. Ello ha llevado al desarrollo de nuevas técnicas, tales como la programación orientada a objetos y el desarrollo de entornos de programación que facilitan la programación de grandes aplicaciones.

Pretender utilizar esta metodología no solo que resultaría infructuoso sino que el único lenguaje que sobrevive y que fue utilizado por Chile por más de 20 años es C y que actualmente se lo conoce como C++ el cual está orientado al objeto; los otros lenguajes de programación casi han desaparecido.

Con la programación estructurada elaborar programas para computador seguirá siendo una labor que demande esfuerzo, creatividad y cuidado.

8.6 Selección de una metodología de diseño Orientada a Objetos

Esta metodología es una aproximación posterior. La orientación a objetos al ser más reciente cuenta con mayor número de adeptos y es previsible que termine sustituyendo a la anterior. Además cuenta con una serie de ventajas:

1. Están basadas en componentes, lo que significa que es más fácil reutilizar código hecho por terceras personas.

2. Es fácil de mantener debido a que los cambios están más localizados.

La mentalidad que subyace al diseño estructurado es: ¿Cómo se puede dividir el sistema en partes más pequeñas que puedan ser resueltas por **algoritmos** sencillos y qué información se intercambian?. En el diseño orientado a objetos la idea es sin embargo: ¿Cuáles son los tipos de **datos** que hay que utilizar, que características tienen y como se relacionan?

La orientación a objetos supone un paradigma distinto del tradicional (no necesariamente mejor o peor) que supone focalizar la atención en las estructuras de datos. El lenguaje **C++** fue una ampliación de C para que soportara objetos, resultó muy eficiente y también muy complejo. **Java** es otro lenguaje orientado a objetos derivado del C++ pero con la idea de ser más sencillo.

El análisis, diseño y programación orientada a objetos, ha sido desarrollado para responder a las necesidades de flexibilidad en los Sistema de información basados en computadora. La encapsulación, herencia y polimorfismo, tienen como objeto proporcionar sistemas

complejos con mecanismos para un rápido, fácil y confiable mantenimiento y cambio de los programas.

Aunque el desarrollo Orientado a Objetos típico involucra una fase de análisis y diseño más amplia, esta inversión se traduce en menores costos de operación de los sistemas que es probable que requiera una gran actividad de mantenimiento.

Existen varias metodologías orientadas a objetos, a pesar que tienen variantes entre ellas, todas trabajan con el mismo paradigma por tanto se basan en los mismos fundamentos de modelación de objetos. Sin embargo, no existe una técnica que se haya definido como estándar para este paradigma.

Criterios de Evaluación para la selección de una Metodología Estructurada u Orientada a Objetos.

ORD.	Criterios de Evaluación Metodología Estructurada u Orientada a Objetos	SI / NO
1	La metodología se ajusta a los objetivos generales del desarrollo?	
2	La metodología cubre el ciclo entero de desarrollo de software? ❑ Investigación ❑ Análisis de requisitos ❑ Diseño	
3	La metodología integra las distintas fases del ciclo de desarrollo? ❑ Rastreabilidad ❑ Fácil interacción entre etapas del ciclo de desarrollo.	
4	La metodología incluye la realización de validaciones?	

5	La metodología soporta la determinación de la exactitud del sistema a través del ciclo de desarrollo?	
6	La metodología es la base de una comunicación efectiva?	
7	La metodología funciona en un entorno dinámico orientado al usuario?	
8	La metodología especifica claramente los responsables de resultados?	
9	La metodología se puede emplear en un entorno amplio de proyectos software? ❑ Variedad. ❑ Tamaño, vida. ❑ Complejidad. ❑ Entorno.	
10	La metodología se puede enseñar?	
11	La metodología esta soportada por herramientas CASE?	
12	La metodología soporta la eventual evolución del sistema?	
13	La metodología contiene actividades conducentes a mejorar el proceso de desarrollo de software?	

Estos criterios de evaluación nos van a permitir ponderar en función de la importancia que los *distintos involucrados* en el desarrollo de software le den a cada una de estas características; estas ponderaciones permitirán crear una matriz de ponderación para las metodologías seleccionadas para el desarrollo de software.

La escala que se va a utilizar para ponderar deberá contener los siguientes valores:

1 = Totalmente en desacuerdo

2 = En desacuerdo

3 = Indeciso o neutral

4 = De acuerdo

5 = Totalmente de acuerdo

Esta escala a su vez estará ponderada en relación al porcentaje, donde 5 representara el 100 %, 4 el 80 %, 3 el 60 %, 2 el 40 % y 1 el 20 %.

1 = 20 %

2 = 40 %

3 = 60 %

4 = 80 %

5 = 100 %

Para la obtención de los resultados se deberá crear una hoja en Excel de tal forma que nos permita ir agregando y obteniendo los resultados de forma automática las ponderaciones respectivas para cada una de las variables.

CRITERIOS DE EVALUACIÓN PARA LA SELECCIÓN DE UNA METODOLOGÍA (EJEMPLO)

VARIABLES	1	2	3	4	5	20 %	40 %	60 %	80 %	100 %	TOTAL
La metodología se ajusta a los objetivos generales del desarrollo?			1	3	3	0	0	0,1429	0,4286	0,4286	**0,857**
La metodología cubre el ciclo entero de desarrollo de software?			1	1	5	0	0	0,1429	0,1429	0,7143	**0,914**
La metodología integra las distintas fases del ciclo de desarrollo?				3	4	0	0	0	0,4286	0,5714	**0,914**
La metodología incluye la realización de validaciones?			1	3	3	0	0	0,1429	0,4286	0,4286	**0,857**
La metodología soporta la determinación de la exactitud del sistema a través del ciclo de desarrollo?			2	3	2	0	0	0,2857	0,4286	0,2857	**0,800**
La metodología es la base de una comunicación efectiva?		1		3	3	0	0,1429	0	0,4286	0,4286	**0,829**
La metodología funciona en un entorno dinámico orientado al usuario?		2	2	1	2	0	0,2857	0,2857	0,1429	0,2857	**0,686**
La metodología especifica claramente los responsables de resultados?		1		3	3	0	0,1429	0	0,4286	0,4286	**0,829**
La metodología se puede emplear en un entorno amplio de proyectos software?				4	3	0	0	0	0,5714	0,4286	**0,886**
La metodología se puede enseñar?			1	5	1	0	0	0,1429	0,7143	0,1429	**0,800**
La metodología esta soportada por herramientas CASE?			3	3	1	0	0	0,4286	0,4286	0,1429	**0,743**
La metodología soporta la eventual evolución del sistema?		1	1	2	3	0	0,1429	0,1429	0,2857	0,4286	**0,800**
La metodología contiene actividades conducentes a mejorar el proceso de desarrollo de software?			3	2	2	0	0	0,4286	0,2857	0,2857	**0,771**

Tabla 8: Tabla de cálculo de las ponderaciones de las variables

Las ponderaciones se obtendrán a partir de los valores asignados por los desarrolladores integrantes del equipo. Estos valores deberán ser sumados para obtener el total por cada una de las variables; a continuación se divide para el total:

1 / 7 = 0,1429
2 / 7 = 0,2857
3 / 7 = 0,4286
4 / 7 = 0,5714
5 / 7 = 0,7143

Para obtener el total de cada una de las variables, procedemos a sacar el porcentaje asignado a cada uno de los valores ponderados y finalmente los sumamos.

TOTAL = (0,1429 * 60 %) = 0,08574
(0,4286 * 80 %) = 0,04899
(0,4286 * 100 %) = 0,4286

TOTAL = 0,857

Del análisis de la matriz de ponderaciones de las variables, se desprende que se obtuvieron 5 factores en mayor escala, los que a continuación se presentan:

VARIABLES	PUNTOS
La metodología cubre el ciclo entero de desarrollo de software?	0,914
La metodología integra las distintas fases del ciclo de desarrollo?	0,914
La metodología se puede emplear en un entorno amplio de proyectos software?	0,886

La metodología se ajusta a los objetivos generales del desarrollo?	0,857
La metodología incluye la realización de validaciones?	0,857

De los valores obtenidos se puede observar que para los involucrados en el desarrollo la prioridad lo tiene el cubrir el ciclo entero de desarrollo de software e integrar las distintas fases del ciclo de desarrollo, para luego darle importancia al empleo en entornos amplios de proyectos software y ajustar la metodología a los objetivos generales del desarrollo y a las validaciones.

Sin embargo, estos resultados dejan por fuera factores que de alguna manera son de gran importancia a la hora de diseñar una metodología como es la eventual evolución del sistema.

Una vez obtenidos los resultados para la ponderación de las variables y su ordenamiento según su grado de importancia (determinado por las personas involucradas en el desarrollo), se procede a diseñar la matriz de evaluación de metodologías.

Los pasos previos que se deberán seguir para la generación de esta matriz serán los siguientes:

1. Identificar los factores claves de éxito
2. Asignar la ponderación a cada factor clave de éxito
3. Se asigna fortaleza o debilidad a cada factor por competidor
   3.1 Debilidad grave = 1
   3.2 Debilidad menor = 2
   3.3 Fortaleza menor = 3
   3.4 Fortaleza importante = 4
4. Calcular los resultados ponderados
5. Sumar resultados

MATRIZ DE EVALUACION DE METODOLOGIAS

FACTORES CLAVES DE EXITO	PONDERACION	ESTRUCTURADA		ORIENTADA A OBJETOS	
		PUNTUACION	RESULTADO PONDERADO	PUNTUACION	RESULTADO PONDERADO
La metodología se ajusta a los objetivos generales del desarrollo?	0,857	1	0,857	2	1,714
La metodología cubre el ciclo entero de desarrollo de software?	0,914	3	2,742	3	2,742
La metodología integra las distintas fases del ciclo de desarrollo?	0,914	3	2,742	4	3,656
La metodología incluye la realización de validaciones?	0,857	2	1,714	3	2,571
La metodología soporta la determinación de la exactitud del sistema a través del ciclo de desarrollo?	0,800	1	0,800	2	1,600
La metodología es la base de una comunicación efectiva?	0,829	1	0,829	3	2,487
La metodología funciona en un entorno dinámico orientado al usuario?	0,686	2	1,372	3	2,058
La metodología especifica claramente los responsables de resultados?	0,829	1	0,829	2	1,658
La metodología se puede emplear en un entorno amplio de proyectos software?	0,886	1	0,886	3	2,658
La metodología se puede enseñar?	0,800	2	1,600	4	3,200
La metodología esta soportada por herramientas CASE?	0,743	3	2,229	4	2,972
La metodología soporta la eventual evolución del sistema?	0,800	1	0,800	4	3,200
La metodología contiene actividades conducentes a mejorar el proceso de desarrollo de software?	0,771	1	0,771	4	3,084
			18,171		**33,600**

Tabla 9: Matriz de evaluación de metodologías

Los resultados muestran que la metodología Orientada a Objetos es claramente la que sobresale al momento de seleccionar una metodología para desarrollo de software.

Actualmente es una de las metodologías que mas se usa para el desarrollo ya que es una nueva forma de concebir los lenguajes de programación al incorporar bibliotecas de clases y otros componentes reutilizables de manera que se puedan adaptar a otros sistemas, además es fácil de dividir el sistema en varios subsistemas independientes y se fomenta la reutilización de componentes.

8.7 Selección de Metodologías Agiles

En el caso de las metodologías ágiles no existe un criterio unificado para seleccionar una metodología, por ello, el presente trabajo se orienta a la formulación inicial, en base a la información existente a la fecha, a la experiencia personal y a la formulación de dos procedimientos al respecto: selección por ***criterios de presencia*** y por ***conocimiento***.

- **Selección de metodologías ágiles, por criterios de presencia**

  Una metodología dispone de **training**, si se encuentra alguna institución, organización o compañía que ofrezca formación sobre la metodología.

  Se considera que una metodología tiene **comunidad**, si se ha formado una comunidad relevante o si está asociada a la *Agile Alliance*, soportándola y cumpliendo sus principios. Se consideran los proyectos realizados, en su mayoría por metodologías que se han aplicado en empresas privadas y por lo tanto no existe mucha documentación

pública al respecto. Por lo tanto, determinar esta clasificación, requiere de una búsqueda exhaustiva.

– **Aplicación del criterio de selección por presencia**

Esta selección se la realiza con el grupo de desarrolladores, acerca de su conocimiento sobre el uso de metodologías, para el efecto se utiliza un grupo de 5 metodologías (o mas), que serán evaluadas para este criterio de selección.

Para determinar la presencia, de las metodologías en Internet, se realiza búsquedas en Google, Yahoo y Live. Sobre el resultado, se asignan 5 puntos al mayor, y 1 punto al menor.

Para determinar las metodologías de *mayor documentación*, se consideran como documentos, los Libros en español, libros en inglés y papers que hablen sobre la aplicación de la metodología. Siguiendo el mismo método, se asigna 5 puntos al mayor y 1 punto al menor.

En el caso de la *Certificación y Training*, se ha busca si hay instituciones que certifiquen la implementación de la metodología, así como también si hay entrenamiento o capacitación en la misma. Como no es posible hacer diferencias en cuanto a la certificación, se asigna el mismo puntaje a las metodologías que tienen Certificación y Training (5 puntos) y 3 puntos las metodologías que contienen sólo training.

En cuanto a *Comunidades*, la mayoría pertenece a la Agile Alliance, pero hay algunas que tienen sus propias comunidades, alianzas e intensa actividad a su alrededor. A estas metodologías se

les asigna 5 puntos, porque no es posible diferenciar entre estas comunidades el número de miembros. A las metodologías que solo pertenecen a la Agile Alliance, se les asigna 2 puntos.

En cuanto a *proyectos de software y presencia empresarial*, se asigna 5 puntos a la metodología que presenta más proyectos y un punto a la que presenta menos.

– **Resultado de la aplicación del criterio de selección por presencia**

Se prepara un cuadro resumen con los resultados de la selección. Para cada metodología evaluada, se coloca la puntuación que se ha obtenido de la clasificación. La sumatoria de cada clasificación determinara la metodología ágil que se debería utilizar, por tener una mejor puntuación.

METODOLOGÍAS AGILES, POR CRITERIOS DE PRESENCIA

Metodología	**Mayor presencia en Internet**	**Mejor documentación**	**Certificadas y con training**	**Comunidades**	**Presencia empresarial**	**Proyectos de SW**	**Total**
Agile Project Management	2	1	3	5	1	1	11
Dynamic Systems Development Method	1	3	5	5	4	4	22
Scrum	5	2	5	5	5	5	**27**
Test Driven Development	3	4	3	2	2	2	16
Extrem Programming	4	5	3	2	3	3	19
Total	**15**	**15**	**19**	**19**	**15**	**15**	**95**

Tabla 10: *Resumen Metodologías Agiles, por criterios de presencia*

- **Selección de metodología, por criterios de conocimientos**

  En función del grupo de trabajo o de diseño, se consideran los siguientes criterios en función de los conocimientos que el equipo de desarrollo tenga de las metodologías a evaluar. Estos criterios son:

  - Grado de conocimiento
  - Soporte orientado a objetos
  - Adaptable a cambios
  - Basado en casos de uso
  - Posee documentación adecuada
  - Facilita la integración entre las etapas de desarrollo
  - Relación con UML
  - Permite desarrollo software sobre cualquier tecnología

  En función de los conocimientos que el equipo tenga, se establecen los pesos para cada criterio. Se propone la siguiente tabla de pesos:

    - 20% Por el Grado de conocimiento
    - 15% Si es adaptable a cambios
    - 15% Si posee documentación adecuada
    - 10% para el resto de criterios

METODOLOGÍAS AGILES, POR CRITERIOS DE CONOCIMIENTOS

Criterio	**%**	**XP**	**Scrum**	**Cristal Methods**	**FDD**	**RUP**	**DSDM**	**ASD**
Grado de conocimiento	20	10	10	5	5	15	10	10
Soporte Orientado a Objetos	10	10	10	10	10	10	10	10
Adaptable a cambios	15	15	15	15	10	10	15	10
Basado en casos de uso	10	5	5	5	5	10	5	10
Posee documentación adecuada	15	10	10	10	10	15	15	15
Facilita la integración entre las etapas de desarrollo	10	10	10	10	10	10	10	10
Relación con UML	10	8	10	10	10	10	8	8
Permite desarrollo de SW sobre cualquier tecnología	10	10	10	10	5	10	10	10
Total	100	78	80	75	65	**90**	83	83

Tabla 11: *Resumen Metodologías Agiles, por criterios de conocimientos*

- Extreme Programming (XP)
- Scrum.
- Cristal Methods (CM)
- Feature Driven Development (FDD)
- Proceso Unificado Rational (RUP)
- Dynamic Systems Development Method (DSDM).
- Adaptive Software Development (ASD)

En esta evaluación, la metodología se selecciona de acuerdo al puntaje que recibe por parte del equipo de desarrollo.

También se puede utilizar la matriz de evaluación de metodologías por el método de ponderaciones de las variables.

Los pasos previos que se deberán seguir para la generación de esta matriz serán los siguientes:

1. Identificar los factores claves de éxito
2. Asignar la ponderación a cada factor clave de éxito
3. Se asigna fortaleza o debilidad a cada factor por competidor
   - 3.1 Debilidad grave = 1
   - 3.2 Debilidad menor = 2
   - 3.3 Fortaleza menor = 3
   - 3.4 Fortaleza importante = 4
4. Calcular los resultados ponderados
5. Sumar resultados

VARIABLES	POND.	XP		Scrum		Cristal Methods		FDD		RUP		DSDM		ASD	
		PUNT.	RESULT. POND.	PUNT.	RESULT. POND.	PUNT.	RESULT. POND.	PUNT.	RESULT. POND.	PUNT.	RESULT. POND.	PUNT.	RESULT. POND.	PUNT.	RESULT. POND.
La metodología se ajusta a los objetivos generales del desarrollo?	0,857	4	3,428	3	2,571	2	1,714	1	0,857	4	3,428	4	3,428	3	2,571
La metodología cubre el ciclo entero de desarrollo de software?	0,914	3	2,742	3	2,742	2	1,828	2	1,828	4	3,656	4	3,656	3	2,742
La metodología integra las distintas fases del ciclo de desarrollo?	0,914	3	2,742	2	1,828	3	2,742	3	2,742	4	3,656	4	3,656	3	2,742
La metodología incluye la realización de validaciones?	0,857	4	3,428	3	2,571	3	2,571	3	2,571	3	2,571	3	2,571	2	1,714
La metodología soporta la determinación de la exactitud del sistema a través del ciclo de desarrollo?	0,800	3	2,400	3	2,400	2	1,600	4	3,200	4	3,200	3	2,400	4	3,200
La metodología es la base de una comunicación efectiva?	0,829	4	3,316	3	2,487	2	1,658	3	2,487	3	2,487	3	2,487	3	2,487
La metodología funciona en un entorno dinámico orientado al usuario?	0,686	4	2,744	4	2,744	3	2,058	4	2,744	3	2,058	3	2,058	2	1,372
La metodología especifica claramente los responsables de resultados?	0,829	4	3,316	3	2,487	3	2,487	4	3,316	3	2,487	3	2,487	3	2,487
La metodología se puede emplear en un entorno amplio de proyectos software?	0,886	2	1,772	2	1,772	1	0,886	3	2,658	3	2,658	2	1,772	2	1,772
La metodología se puede enseñar?	0,800	4	3,200	4	3,200	3	2,400	3	2,400	4	3,200	2	1,600	1	0,800
La metodología esta soportada por herramientas CASE?	0,743	2	1,486	2	1,48	2	1,486	4	2,972	4	2,972	2	1,486	3	2,229
La metodología soporta la eventual evolución del sistema?	0,800	3	2,400	3	2,400	2	1,600	2	1,600	4	3,200	3	2,400	4	3,200
La metodología contiene actividades conducentes a mejorar el proceso de desarrollo de software?	0,771	2	1,542	2	1,642	3	2,313	3	2,313	4	3,084	2	1,542	2	1,542
			34,516		**30,23**		**25,343**		**31,688**		**38,657**		**31,543**		**28,858**

Tabla 12: *Tabla de cálculo de las ponderaciones de las variables*

Los métodos Ágiles son estrategias de desarrollo de software que promueven prácticas que son adaptativas en vez de predictivas, centradas en la gente o en los equipos, orientadas hacia prestaciones y hacia la entrega, de comunicación intensiva, y que requieren que el negocio se involucre en forma directa. Las metodologías ágiles están muy alineadas tanto en los principios como en las prácticas para el desarrollo de software en ambientes que requieren un alto grado de adaptabilidad.

- La Metodología **XP**, se recomienda para proyectos medianos y pequeños de corto plazo.
- La Metodología **Scrum**, se recomienda para proyectos medianos y pequeños de corto plazo, óptima para equipos de trabajo de hasta 8 personas
- La Metodología **RUP** es más adaptable para proyectos medianos y grandes (de largo plazo) donde la complejidad amerite el uso de la Orientación a Objetos.
- La Metodología **Cristal Methods** se recomienda para proyectos pequeños, medianos y grandes.
- La Metodología **DSDM**, se recomienda para proyectos medianos y pequeños de presupuestos limitados y agendas ocupadas y apretadas
- La Metodología **FDD,** se recomienda para proyectos pequeños de corto plazo, solo cubre las fases de diseño y construcción.
- La Metodología **ASD**, se recomienda para proyectos medianos y pequeños.

Podemos concluir además, que lo más importante antes de elegir la metodología que se usara para la implementación del software, es determinar el alcance que tendrá y luego de ahí ver cuál es la que más se acomoda a la aplicación.

8.8 La elección del ciclo de vida

Esta evaluación se la realiza de la misma manera que la selección de metodologías y se selecciona de acuerdo al puntaje que se recibe por parte del equipo de desarrollo.

Los pasos previos que se deberán seguir para la generación de esta matriz serán los siguientes:

1. Identificar los factores claves de éxito
2. Asignar la ponderación a cada factor clave de éxito
3. Se asigna fortaleza o debilidad a cada factor por competidor
   - 3.1 Debilidad grave = 1
   - 3.2 Debilidad menor = 2
   - 3.3 Fortaleza menor = 3
   - 3.4 Fortaleza importante = 4
4. Calcular los resultados ponderados
5. Sumar resultados

De los distintos modelos de ciclo de vida existentes, para el presente caso se han seleccionado solamente cuatro (los mas comunes), la elección de cada uno de ellos dependerá de la naturaleza del proyecto, de la aplicación, de los métodos a usar y de los controles y entregas requeridos.

VARIABLES	POND.	CASCADA		PROTOTIPADO		LINEAL		ESPIRAL	
		PUNT.	RESULT. POND.	PUNT.	RESULT. POND.	PUNT.	RESULT. POND.	PUNT.	RESULT. POND.
La metodología se ajusta a los objetivos generales del desarrollo?	0,857	4	3,428	3	2,571	2	1,714	1	0,857
La metodología cubre el ciclo entero de desarrollo de software?	0,914	3	2,742	3	2,742	2	1,828	2	1,828
La metodología integra las distintas fases del ciclo de desarrollo?	0,914	3	2,742	2	1,828	3	2,742	3	2,742
La metodología incluye la realización de validaciones?	0,857	4	3,428	3	2,571	3	2,571	3	2,571
La metodología soporta la determinación de la exactitud del sistema a través del ciclo de desarrollo?	0,800	3	2,400	3	2,400	2	1,600	4	3,200
La metodología es la base de una comunicación efectiva?	0,829	4	3,316	3	2,487	2	1,658	3	2,487
La metodología funciona en un entorno dinámico orientado al usuario?	0,686	4	2,744	4	2,744	3	2,058	4	2,744
La metodología especifica claramente los responsables de resultados?	0,829	4	3,316	3	2,487	3	2,487	4	3,316
La metodología se puede emplear en un entorno amplio de proyectos software?	0,886	2	1,772	2	1,772	1	0,886	3	2,658
La metodología se puede enseñar?	0,800	4	3,200	4	3,200	3	2,400	3	2,400
La metodología esta soportada por herramientas CASE?	0,743	2	1,486	2	1,48	2	1,486	4	2,972
La metodología soporta la eventual evolución del sistema?	0,800	3	2,400	3	2,400	2	1,600	2	1,600
La metodología contiene actividades conducentes a mejorar el proceso de desarrollo de software?	0,771	2	1,542	2	1,642	3	2,313	3	2,313
			34,516		**30,23**		**25,343**		**31,688**

Tabla 13: *Matriz de ponderaciones del Ciclo de Vida*

Del cuadro anterior se puede ver que el modelo en cascada resulto ser el que mas puntaje ha obtenido, lo que quiere decir que este seria el modelo a utilizar.

Para poder realizar este tipo de evaluación se requiere que el equipo de desarrollo tenga un buen nivel de conocimientos sobre modelos de ciclos de vida, de ahí es donde se podrán obtener resultados satisfactorios. Para poder definir cual es el modelo más adecuado, se debe realizar un análisis tomando como referencia la siguiente tabla:

Modelo de proceso	Desempeño con requisitos y arquitectura no predefinidos	Produce software altamente fiable	Gestión de riesgos	Permite correcciones sobre la marcha	Visión del progreso por el Cliente y el Jefe del proyecto
Cascada	Bajo	Alto	Bajo	Bajo	Bajo
Prototipado	Alto	Medio	Medio	Alto	Alto
Espiral	Alto	Alto	Alto	Medio	Medio
Incremental	Bajo	Alto	Medio	Bajo	Bajo

Tabla 14: *Cuadro comparativo de modelos del ciclo de vida*

A continuación se citan algunas características que también se deberán tomar en cuenta previa la selección de un modelo de ciclo de vida:

Modelo de prototipado

- No modifica el flujo del ciclo de vida
- Reduce el riesgo de construir productos que no satisfagan las necesidades de los usuarios
- Reduce costos y aumenta la probabilidad de éxito
- Exige disponer de las herramientas adecuadas

- No presenta calidad ni robustez
- Una vez identificados todos los requisitos mediante el prototipo, se construye el producto de ingeniería.

Modelo de cascada

- Requiere establecimiento explícito de requerimientos desde el comienzo
- Exige al cliente/usuario gran paciencia
- Dificulta la incorporación de modificaciones durante el desarrollo
- Se tarda mucho tiempo en pasar por todo el ciclo
- Perpetúa el fracaso de la industria del software en su comunicación con el usuario final.
- El mantenimiento se realiza en el código fuente
- Las revisiones de proyectos de gran complejidad son muy difíciles
- Impone una estructura de gestión de proyectos

Modelo en espiral

- Trata de mejorar los ciclos de vida clásicos y prototipos.
- Permite acomodar otros modelos
- Incorpora objetivos de calidad y gestión de riesgos
- Elimina errores y alternativas no atractivas al comienzo
- Permite iteraciones, vuelta atrás y finalizaciones rápidas
- Cada ciclo empieza identificando:
  - Los objetivos de la porción correspondiente
  - Las alternativas
  - Restricciones

- Cada ciclo se completa con una revisión que incluye todo el ciclo anterior y el plan para el siguiente

Modelo Incremental

- Se descompone el sistema en incrementos.
- Cada incremento culmina con un producto operacional
- Asume que los requerimientos son conocidos
- Los requerimientos son priorizados
- Al iniciar un incremento se congelan los requerimientos, asumiendo que los cambios serán abarcados en incrementos posteriores
- Disminuye los riesgos y permite manejar mejor los errores
- Útil en proyectos críticos a gran escala
- Permite entregar un subproducto útil más tempranamente
- Los primeros incrementos sirven como prototipos
- Permite evaluar la aceptación usuaria del producto
- Disminuye el riesgo de fracaso del proyecto

8.9 Propuesta de evaluación de la Arquitectura de software

La Arquitectura de Software es diseño de nivel estratégico, no es solo lógico sino también físico y organizacional, contempla decisiones funcionales y técnicas, además contiene las estrategias para resolver los requerimientos NO funcionales. Puede ser vista como la estructura o estructuras del sistema que comprende componentes de Software, propiedades externas de esos componentes y la interacción entre ellos.

Cada ente evaluador podrá seleccionar la metodología o método de evaluación de arquitecturas que considere útil a sus propósitos, algunas de ellas ya fueron explicadas en el capitulo anterior, a

continuación y en base a la experiencia se presentan los pasos básicos para realizar una evaluación de la arquitectura de software de un único producto como es el juego de guerra simulado:

1. **Preparación y presentación**

   a. **Contexto** (Se define el conjunto de elementos, las relaciones entre si y se grafica)
      - *Módulos:* Son las unidades conceptuales de primer orden que guían el desarrollo.
      - *Componentes:* son los bloques técnicos que permiten llevar a cabo la funcionalidad de los módulos. Pueden existir componentes desarrollados internamente, de terceros, contenedores y software de base.
      - *Conectores:* Representan la interacción con los demás, a través de conectores y como se va a establecer esa interacción, por lo que pueden existir diferentes maneras.

   b. **Tipo de evaluación:** Temprana o Tarde.

   c. **Definición del equipo de Evaluación:** Equipo de desarrollo, gente externa, especialistas en el tema, el cliente

   d. **Presentación de la arquitectura:** Es un diagrama esquemático que representa las ideas prestablecidas y los módulos/componentes candidatos de un sistema o arquitectura, principalmente deberá estar basada en UML y deberá tener al menos los siguientes elementos:
      - Actores o Roles Principales
      - Los módulos/componentes principales
      - Los Nodos principales

- Repositorios de Datos
- Como fluye la información
- Las zonas de red

2. Realización

a. Se revisan los requerimientos:

- *Funcionales:* definen lo que el usuario necesita
  - Procesos de negocio: como se llevan a cabo a través de la organización.
  - Casos de uso: definen la interacción entre un actor y el sistema.
- *No funcionales (atributos de calidad):* definen la calidad y las características que los sistemas deben soportar (performance, disponibilidad, seguridad, testeabilidad, modificabilidad, usabilidad)

b. Se evalúa las restricciones:

- *Del negocio:* tiempo en el mercado, costo beneficio, tiempo de vida útil del sistema.
- *De la Arquitectura:* integración con otros sistemas, robustez, posibles cambios

c. Se verifica si es razonable la solución y permite cumplir con los requerimientos.
d. Identificar y analizar los riesgos técnicos (documentándolos)
e. Se discuten problemas y observaciones

3. Resultados

a. Listar las observaciones que terminen en un riesgo (malas decisiones) y no riesgo (buenas decisiones)
b. Recomendaciones

Una evaluación de una Arquitectura de Software (AS) no nos da un SI o un NO, tampoco nos indica si es buena o mala, o tal vez nos da una calificación. Simplemente nos indica donde está el riesgo, es decir las fortalezas y las debilidades identificadas en la AS.

Después de una evaluación de una AS, se pueden tomar algunas decisiones como: si se puede seguir el proyecto con las áreas de debilidad dadas en la evaluación o si hay que reforzar la AS o si hay que comenzar de nuevo toda la AS.

8.10 Propuesta de evaluación de la infraestructura del hardware

El resultado del análisis de la infraestructura nos devolverá parámetros para los distintos componentes de la arquitectura IT a implementar:

1) **Servidores:** Los servidores son ordenadores capaces de correr aplicaciones, espacio de almacenamiento o servicios de comunicaciones al resto de equipos conectados a una red. Su potencia de proceso, su arquitectura optimizada del chipset y la velocidad de los discos, diferencian un simple PC de un auténtico servidor. Los servidores, por las funciones que desempeñan, son elementos críticos en la estructura de red, ya que el resto de los equipos dependerán de su correcto funcionamiento.

2) **CPU**: Se deberá contar con métricas estándar (SAPS, TPC, etc) de los diferentes vendedores de HW, a fin de comparar las potencias de los distintos CPU. Los más utilizados son:

- Intel x86/x64
- Intel Itanium2
- Sun SPARC T Series
- Sun UltraSPARC64

3) **Memoria**: Generalmente se estima en base a la cantidad de cores y el tipo de aplicación.

4) **Disco**: Se debe estimar en base a dos parámetros principales:
   - **Espacio**: Espacio necesario para la aplicación y crecimiento asociado.
   - **Throughput**: Velocidades requeridas de Lectura y/o Escritura. En base a este parámetro definiremos el tipo de RAID a implementar sobre los distintos files systems.

5) **Red**:
   - **Ancho de banda**: Cantidad de datos que pueden ser transportados por un medio de un equipo a otro. Se mide en Mbit/Gbit sobre segundo.
   - **Latencia**: El tiempo necesario para que un paquete de red viaje de un equipo a otro

Más allá de definir los parámetros básicos (CPU, Memoria, Disco y redes), es importante definir una arquitectura IT escalable, que nos permita incrementar la capacidad de cómputo a medida que la aplicación lo requiera. Las principales alternativas para armar una arquitectura escalable son:

1) **Adquisición de HW escalable**: Solución en general costosa, ya que los equipos que soportan escalabilidad vertical cuestan entre un 20% y un 30% más.

2) **Clusterizado Activo - Activo o balanceo de carga de la solución**: Este tipo de soluciones brindan varias ventajas como ser: simplicidad en la escalabilidad horizontal, alta disponibilidad, performance. Por otro lado, son soluciones que requieren inversión ya que generalmente el HW requerido se duplica. Es aplicable a proyectos con alto impacto en el negocio que requieren disponibilidad 7x24.

3) **Virtualización**: Es la forma más simple y económica de brindar escalabilidad vertical / horizontal, simplemente se potencia el virtual (hasta la potencia total del HW físico) o se agrega otro equipo virtual a la granja y se incluye en el balanceo (generalmente se manejan templates de estos virtuales, por lo que la generación es muy simple). El costo es relativamente bajo ya que los equipos generalmente utilizados para armar las granjas de virtualización son low-entry. Adicionalmente se puede contar con Alta Disponibilidad, con la arquitectura IT necesaria.

4) **Herramientas para detección de problemas:** estas herramientas nos permitirán controlar los recursos adquiridos, entre estas tenemos:
   - Monitores de Red (WireShark, routers, switches, etc)
   - Paneles de control de los application server y sistemas operativos
   - Monitores de la VM (JMX, Visual VM)
   - Profiling (JProfile, YourKit, dotTrace)
   - Logs

CAPITULO IV

CONCLUSIONES FINALES

Los Juegos de Guerra son de raigambre militar y se han usado para preparar a los líderes militares a enfrentarse a circunstancias imprevistas en el combate y pueden ser utilizados para comprender mejor la guerra, tradicionalmente fueron utilizados como diversión, hoy son empleados en cualquier organización como medio de adiestramiento, educación y desarrollo de habilidades y destrezas profesionales.

Aunque la teoría del caos nos dice que la guerra nunca va a ser completamente predecible, nos dice igualmente que se puede utilizar la simulación para identificar centros de gravedad. Es importante mencionar que en los juegos de guerra se pretende fortalecer las competencias en la toma de decisiones bajo las condiciones normales de una operación militar.

Desde los inicios de las ciencias de las matemáticas y la física, el hombre ha hecho uso de herramientas de simulación para poder así dar respuesta a los diversos planteamientos que se le presentaban.

El entrenamiento con militares es una de las principales tareas que se han beneficiado con la llegada de los simuladores. Estos simuladores proporcionan lo necesario para que los objetivos a alcanzar en el campo de combate sean los más eficaces. Pero el principal factor para desarrollar simuladores está en el asunto económico, debido a los costos que se incurren cuando de entrenar todo un batallón se trata. Los simuladores permiten entonces la Toma de decisiones y el entrenamiento y educación a bajo costo.

Sin embargo el desarrollo de simuladores para entrenamiento militar, puede a veces no cumplir con sus objetivos y en vez de permitir la economía de los recursos que se gastan en un entrenamiento puede resultar mucho más costoso, ya que se va a incurrir en gastos no previstos y en muchos casos sin solución.

Es por esos que se requiere una metodología para evaluar estos desarrollos a fin de permitir coherencia en tiempos, equipo, insumos, recursos humanos, infraestructura, software y equipos, de tal manera que se pueda tener resultados exitosos y cumplir con los objetivos planteados.

El desarrollo de esta tesis pretende ser un aporte para la evaluación de juegos de guerra simulados, ya que los requerimientos funcionales y no funcionales son difíciles de comprender para equipos de desarrollo compuestos por personal civil y en ciertos casos militar.

Para llevar a cabo este trabajo se recabó información exhaustiva de metodologías de desarrollo estructuradas y orientadas a objetos, metodologías ágiles, áreas de conocimiento de ingeniería de software, ciclos de vida de los sistemas, prácticas de desarrollo ágil y otras disciplinas dentro de la informática ya que como se trata de un sistema software no podría ser una excepción.

9.1 Aportes del presente trabajo

1) Se ha presentado una metodología de evaluación para los desarrollos de los programas de software para juegos de guerra simulados utilizando conceptos de evaluaciones metodológicas universales.

2) Se ha visto, que la situación actual es compleja, en tanto existe una gran cantidad de lenguajes de programación que posibilitan diferentes alternativas de desarrollo, así como también, los avances en cuanto a la tecnología informática, que permiten utilizar recursos impensados una década atrás.

3) Es necesario evitar las problemáticas concernientes al diseño, desarrollo e implementación de los programas simulados para juegos de guerra y se ha propuesto esta tesis como una de las posibles metodologías, para solución a dichas necesidades.

4) Se ha tomado en cuenta las opiniones del personal de alumnos de la AGE (academia de guerra del ejército) usuarios del CEOTAS (centro de entrenamiento operativo táctico simulado), quienes han venido utilizando el sistema por el lapso de 4 años e hicieron las sugerencias que han permitido realizar las mejoras respectivas.

9.2 Líneas de trabajo futuras.

Los juegos de guerra, al igual que la doctrina institucional, la tecnología, y los conceptos operacionales, constantemente se someten a un proceso evolutivo que puede resultar en muy diferentes métodos de conducir la guerra. A partir de las problemáticas detectadas y de los resultados obtenidos se pueden derivar las siguientes líneas de trabajo futuras:

1. Desarrollo basado en metodologías agiles con diferentes combinaciones de ciclos de vida del software, estableciendo así comparaciones con la solución propuesta en este trabajo y con ponderación de resultados.

2. Otra de las posibles líneas de trabajo, que se deriva al respecto de los sistemas simulados para juegos de guerra, es la adaptación de los programas de mercado de acuerdo a las necesidades específicas de cada institución, considerando que el software desarrollado para estas necesidades especiales, es escaso y caro.

3. Una línea de trabajo de fondo en los desarrollos con base informática sería, el nivel de capacitación de los usuarios para cada nivel desarrollado, así como también la difusión a cada estamento militar a fin de realizar la comprobación de los diferentes planes que se disponen en cada unidad.

4. Por último, se ha dejado como tema central, la investigación con respecto a inteligencia artificial para el desarrollo de los niveles estratégicos y operativos en los que se deberá aplicar la herramienta informática y los procesos respectivos.

5. ANEXOS

Anexo I: Caso practico: Evaluación del CEOTAS

6. REFERENCIAS BIBLIOGRÁFICAS

Air & Space Power Journal - Español Otoño Trimestre 1995 ¿QUÉ SIGNIFICA LA TEORÍA DEL CAOS EN LA GUERRA? Mayor David Nicholls, USAF Mayor Todor Tagarev, Fuerza Aérea De Bulgaria.

Military Review l Julio-Agosto 2003 – El Sistema de Entrenamiento Táctico Computarizado del Ejercito de Chile

Reseña histórica del CETAC – 1994 - Marlon Gutiérrez Ayala - Teniente PA. - Historiador del CETAC

Aerospace Power Journal - Español Primer Trimestre 2002 - Coronel Bobby J. Wilkes, USAF - Un Concepto para la Guerra Operacional

La matemática de la guerra - Gustavo Herren (especial para ARGENPRESS.info) (Fecha publicación:23/07/2006)

Díaz Herráiz, E. y Rodríguez Martín, V. (2002): *La evaluación en Servicios Sociales*, en Fernández García, T. y Ares Parra, A. (coords): Servicios Sociales: Dirección, gestión y planificación. Ciencias Sociales Alianza Editorial. Madrid.

Evaluando Arquitecturas de Software. Parte 1. Panorama General. Gómez, Omar Salvador Gómez. 01, México : Brainworx S.A, 2007. 1870-0888.

Erika Camacho, Fabio Cardeso, Gabriel Nuñez. Arquitecturas de Software. 2004.

Gustavo Andrés Brey, Gastón Escobar, Nicolas Passerini y Juan Arias. Arquitectura de Proyectos de IT. Evaluación de Arquitecturas. Buenos Aires : Universidad Tecnológica Nacional. Facultad Regional de Buenos Aires - Departamento de Sistemas, 2005.