Desarrollo Ágil y Modelado

Andrés Djordjalian <andres@indicart.com.ar>
Indicart Carteles Electrónicos y Facultad de Ingeniería, UBA
Para el Simposio Argentino de Sistemas Embebidos (SASE 2010)
Marzo de 2010

Resumen del tutorial

- **Vamos a introducir dos tendencias, originadas en la Ingeniería del Software, que están ganando impulso en embebidos:**

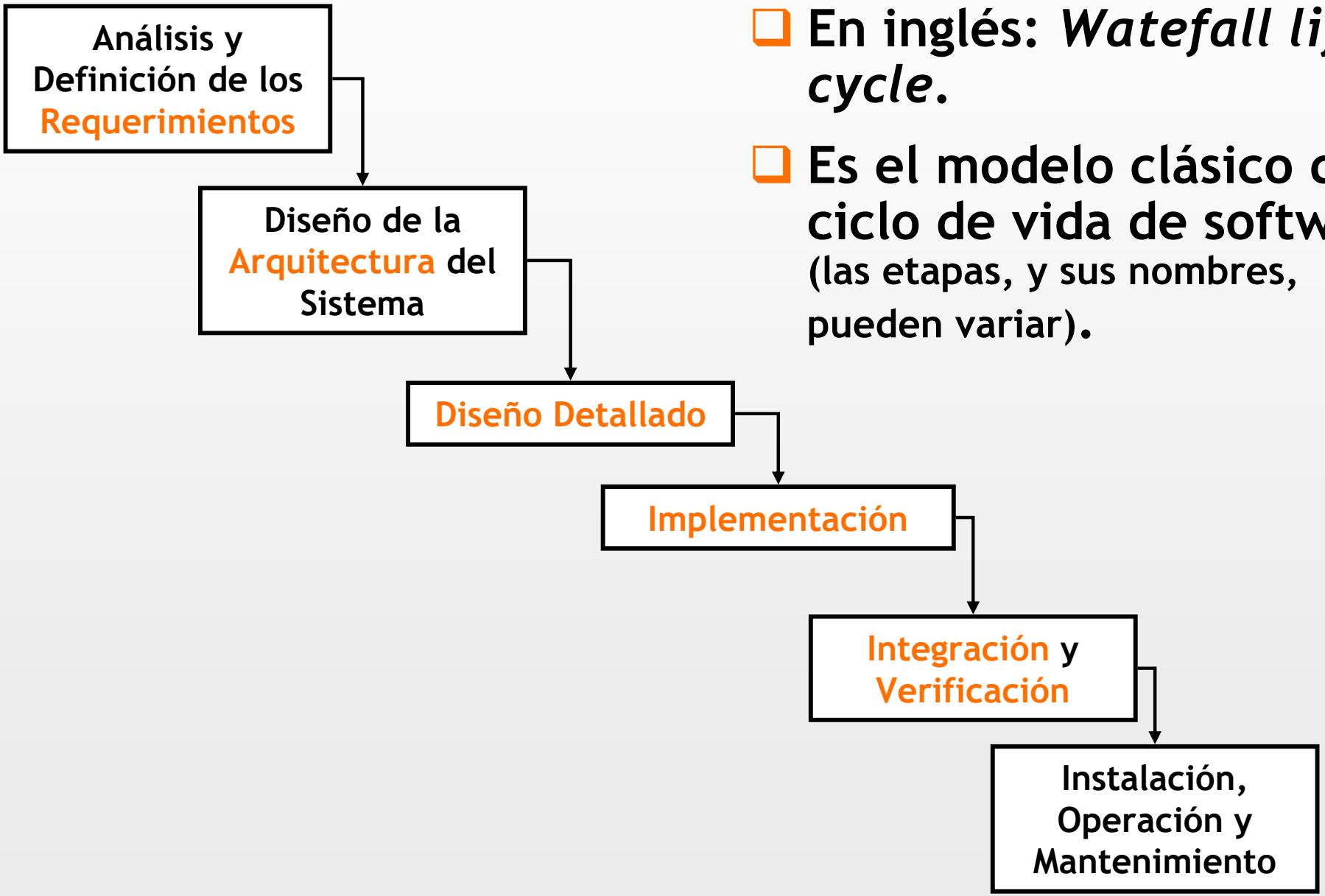
2) **Modelado: Es el trabajo de diseño en alto nivel, por ejemplo mediante gráficos.**

1) **Metodologías Ágiles: Son formas de diseñar software cuyo objetivo es aumentar el valor económico generado por el equipo de desarrollo**
   - Aplicables a embebidos, presumíblemente a hardware también

Éxito Técnico vs. Éxito Económico

- **Como desarrolladores y amantes de los "fierros", frecuentemente nos focalizamos en el éxito técnico**
  - ¿El prototipo funciona? ¿El diseño es ingenioso? ¿Sorprende lo que hace? ¿Fue logrado con poco hard? Etc.
- **Pero los proyectos tiene que lograr éxito económico**
  - ¿El producto hace lo que demanda nuestro **mercado**?
  - ¿Lo estamos lanzando **a tiempo**?
  - ¿Es **confiable**?
  - ¿Es fácil de modificar, para sacar **nuevas versiones**?
  - ¿Podemos **reutilizar** el trabajo en otros diseños?
  - ¿Fueron razonablemente certeras nuestras **estimaciones** de costos y tiempo de entrega?
    - Para las inversiones, la incertidumbre (o sea, *riesgo*) es un "costo"
- **El desarrollo ágil se focaliza en el éxito económico**
  - Pero veamos primero intentos anteriores de organizar los procesos de diseño
    - …que se siguen usando mucho.

Ciclo de Vida en Cascada

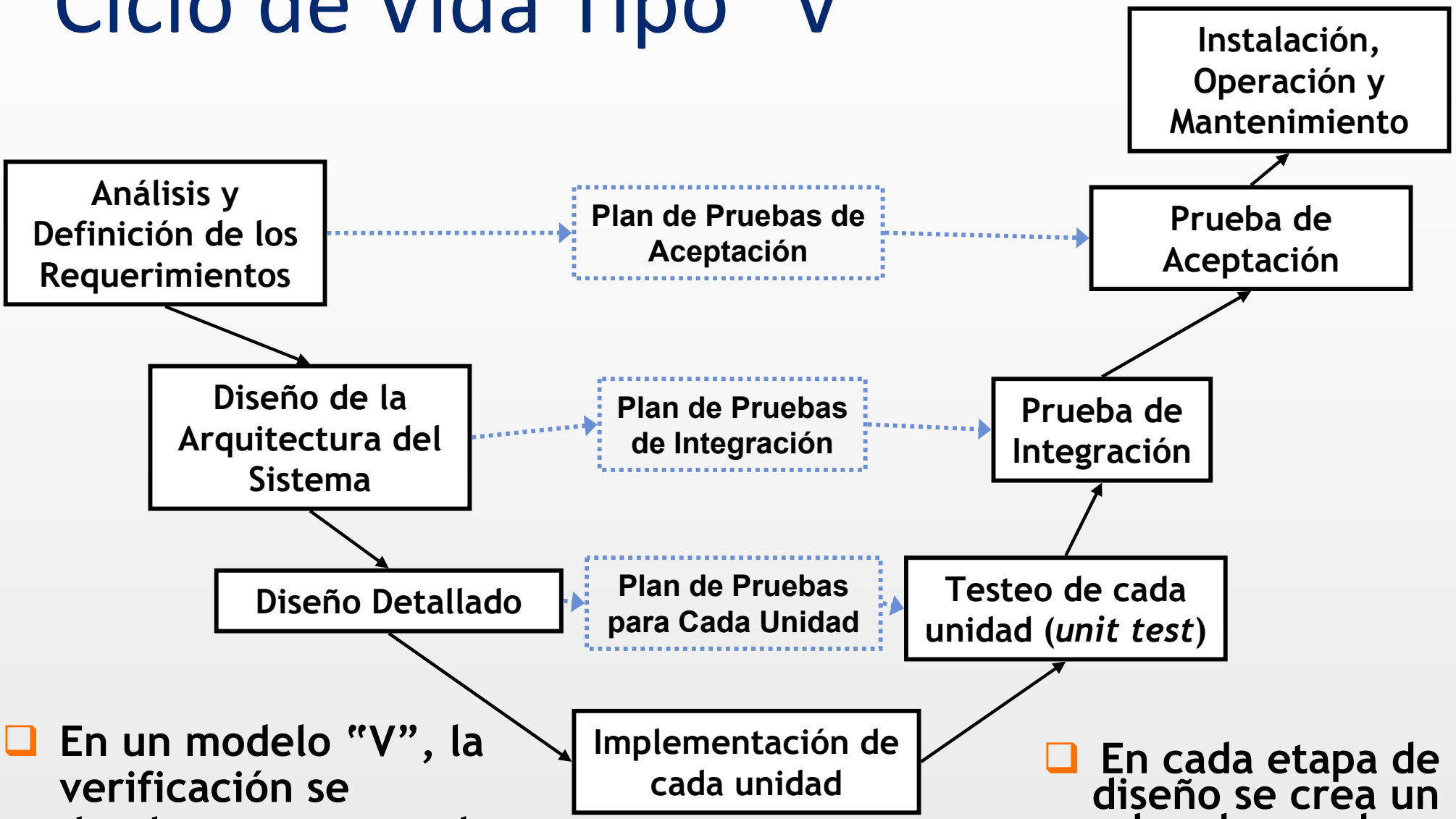


- En inglés: ***Watefall life cycle***.
- Es el modelo clásico de ciclo de vida de software (las etapas, y sus nombres, pueden variar).

Definiciones

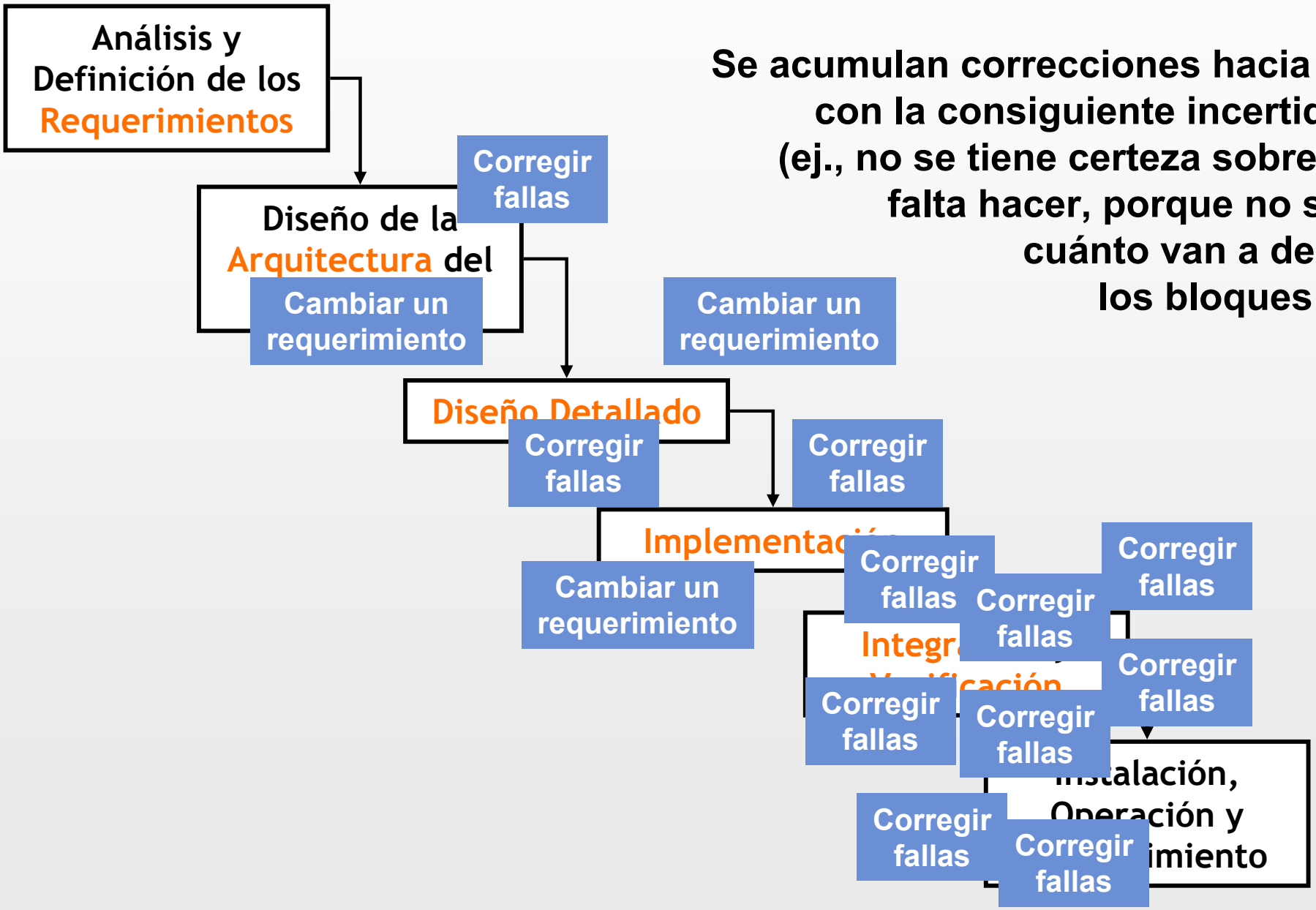
- **Requerimientos: "¿Qué hace el producto?"**
  - Expresado sin ambigüedad y de forma verificable
- **Arquitectura: "¿Cómo está hecho?" (descripto en el nivel de abstracción más alto)**
  - Ej.: Diagrama de bloques; flujo de datos entre estos
- **Implementación: "¿Cómo está hecho?" (descripto en nivel de abstracción bajo)**
  - Ej.: Código; esquemático; dibujo del circuito impreso
- **Diseño Detallado: "¿Cómo está hecho?" (descripto en un nivel de abstracción intermedio)**
  - Ej.: Especificación de las interfases; descripción detallada de la operación
- **Integración de módulos diseñados separadamente**
  - Ej.: Hardware y software
- **Verificación: "¿Funciona como debe?"**

Ciclo de Vida Tipo "V"



- En un modelo "V", la verificación se desglosa en etapas de nivel de abstracción creciente.
- En cada etapa de diseño se crea un plan de pruebas, que es el que guía la etapa de validación que le corresponde.

Modelos Waterfall y "V" en la Práctica

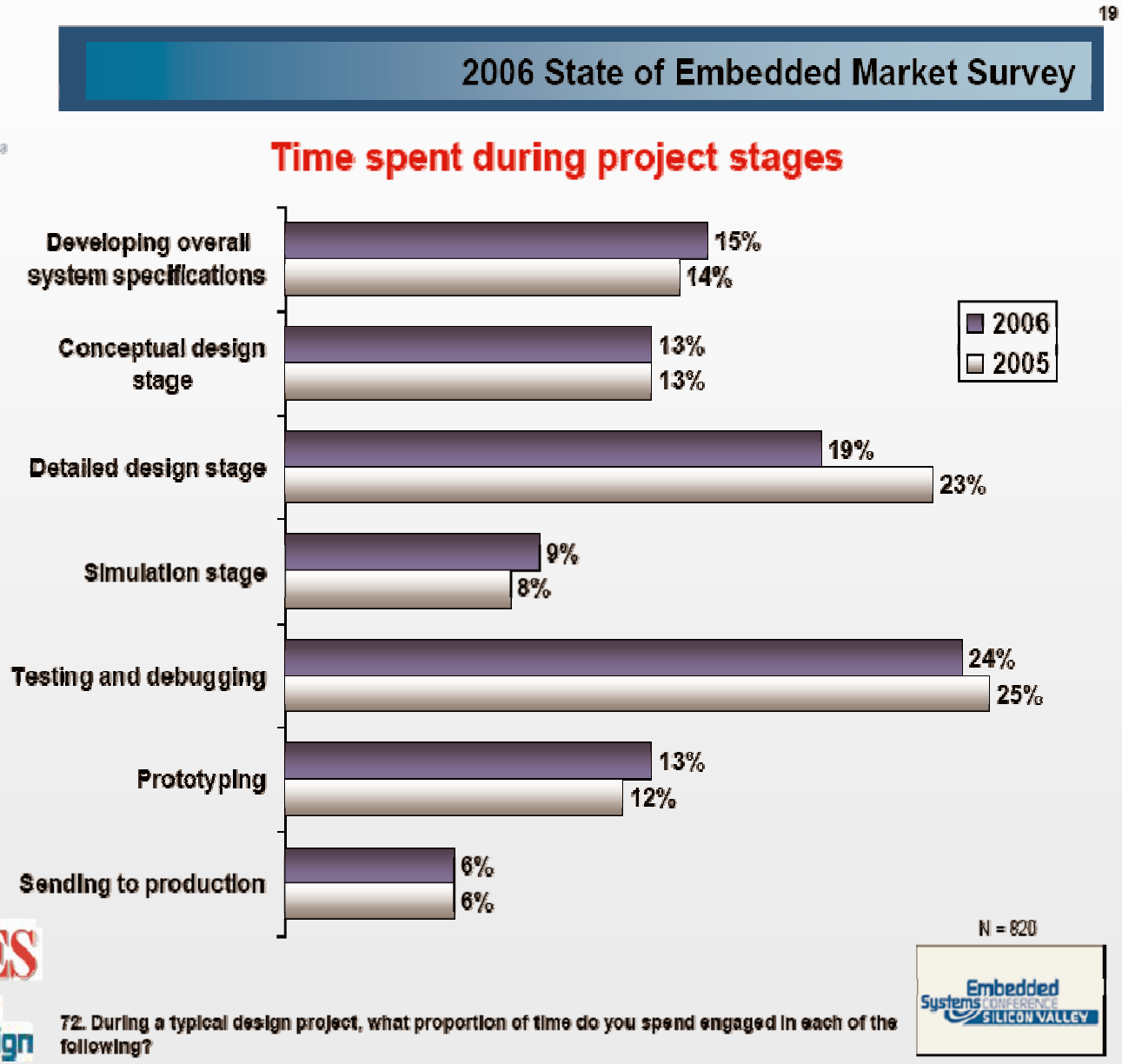


Se acumulan correcciones hacia el final, con la consiguiente incertidumbre (ej., no se tiene certeza sobre cuánto falta hacer, porque no se sabe cuánto van a demandar los bloques azules)

Embebidos: Esfuerzo en Cada Etapa

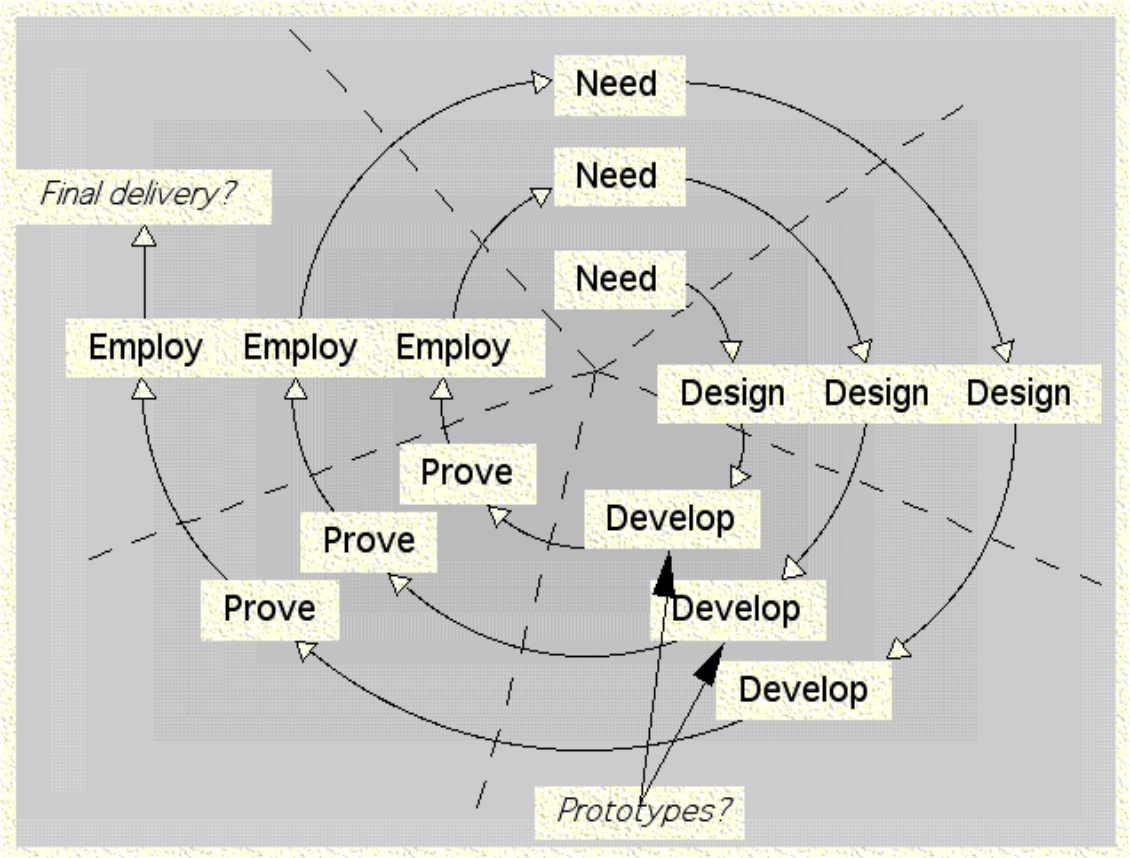
Tiempo utilizado para cada etapa de proyecto:

(2006 State of the Embedded Market Survey: Encuesta a 1217 suscriptos a publicaciones sobre embebidos y visitantes a conferencias.)



Ciclo de Vida Iterativo o Incremental

- En cada iteración se crea un "prototipo", cuya complejidad crece con cada ciclo
- Comparado con waterfall y V, consigue:



World Class Systems Engineering, D.K. Hitchins, disponible el 29/11/08 en <http://www.hitchins.net/WCSE.html>

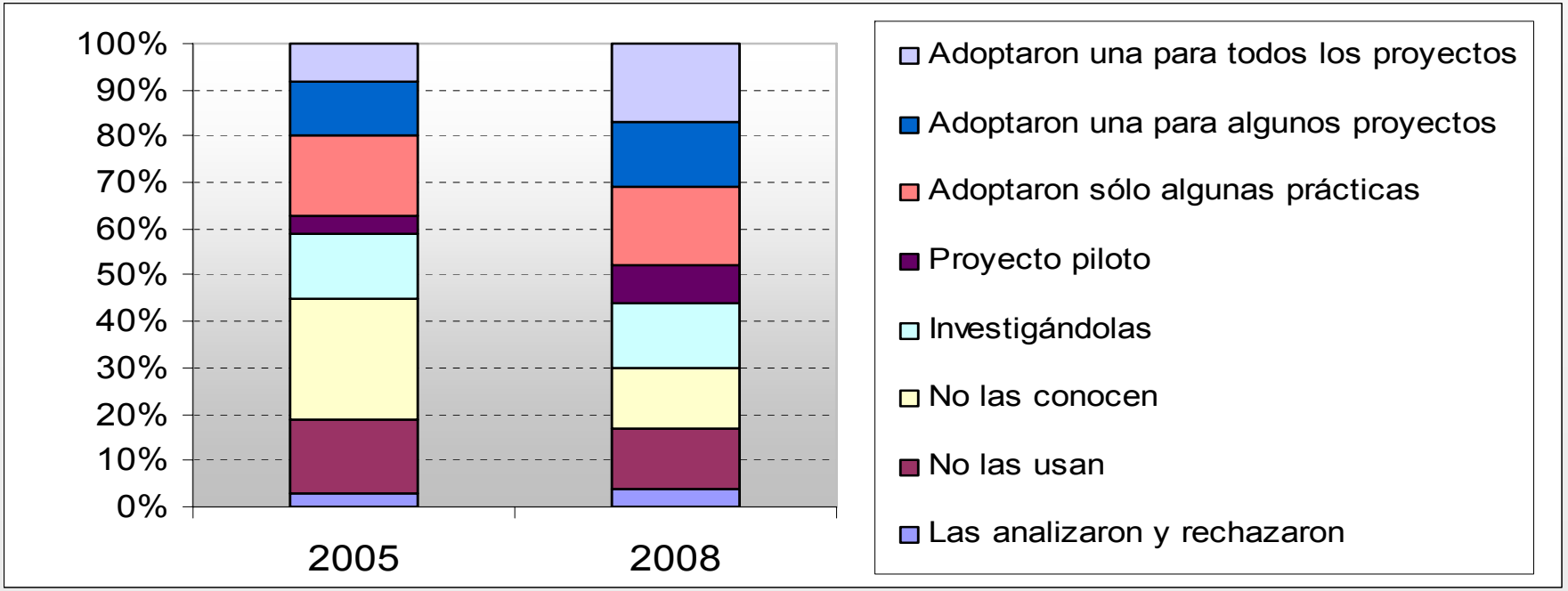
- **Flexibilidad ante cambios de requerimientos = más valor para el usuario.**
- **Corrección más temprana de errores = menor costo.**
- **Mejores métricas de progreso del proyecto = más control (ej., se puede modificar el plan en función del ritmo de avance logrado) = menor riesgo.**

Desarrollo Ágil (*Agile*) de Software

- **Paradigma sobre cómo conviene organizar la construcción de software**
- **Basado en un ciclo de desarrollo iterativo**
  - *Builds* frecuentes, con valor para el usuario, que **colabora** activamente
- **Focalizado en la adaptabilidad**
  - **Aceptar cambios** en los requerimientos, incluso sobre el final del proceso
- **…y en las personas que integran el equipo**
  - Comunicación, auto-organización, motivación, trabajo en equipo
- **Especialmente útil en equipos que no son muy grandes**
- **Las metodologías ágiles más populares son:**
  - **Scrum**
  - **Programación Extrema (*Extreme Programming* o XP)**
    - Vamos a dar ejemplos con esta

Metodologías Ágiles: Creciente interés en el mundo IT

- **"Predicting the Year Ahead"**
  - Un artículo de un blog sobre IT (*Cutter Consortium Blog*)
  - Le pidieron, a 35 especialistas, predicciones breves sobre el IT del 2010
  - 14 de estos 35 mencionan las metodologías ágiles
  - Tomado en marzo de 2010 de http://www.cutter.com/predictions/2010.html
- **Encuesta sobre adopción de metodologías ágiles:**



- Realizada por *Methods & Tools*, con 512 participantes (232 en 2005), tomado en marzo de 2010 de http://www.methodsandtools.com/dynpoll/oldpoll.php?Agile2

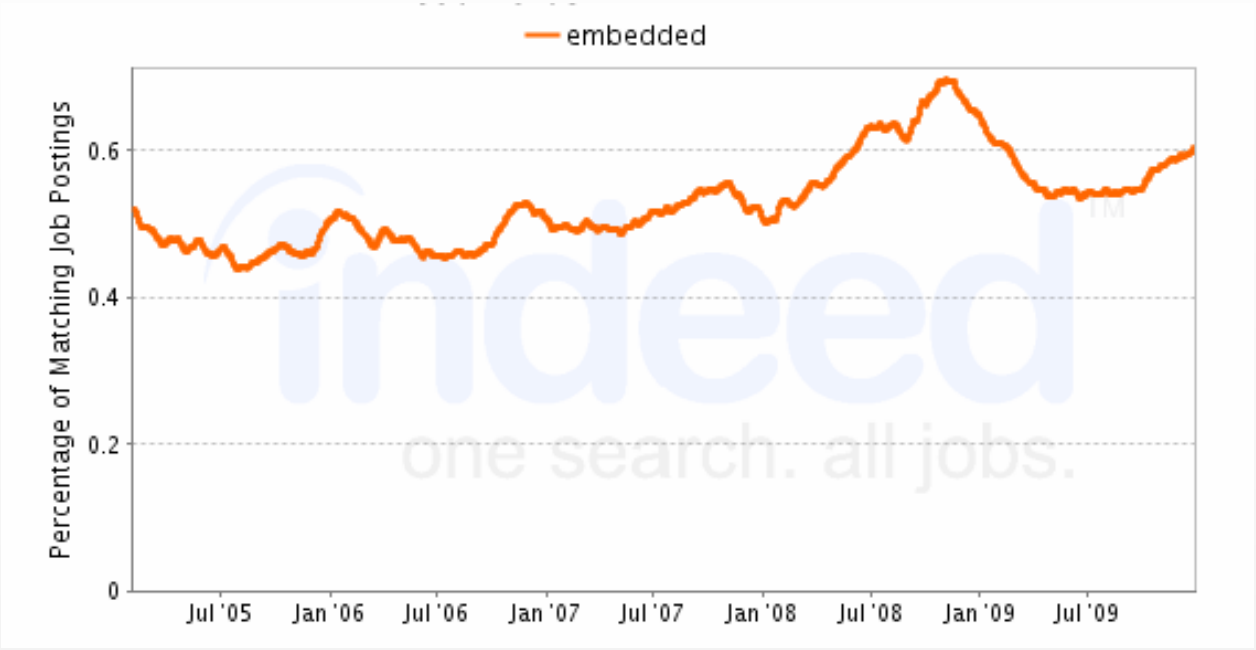
Metodologías Ágiles: Su practicidad para el desarrollo de embebidos

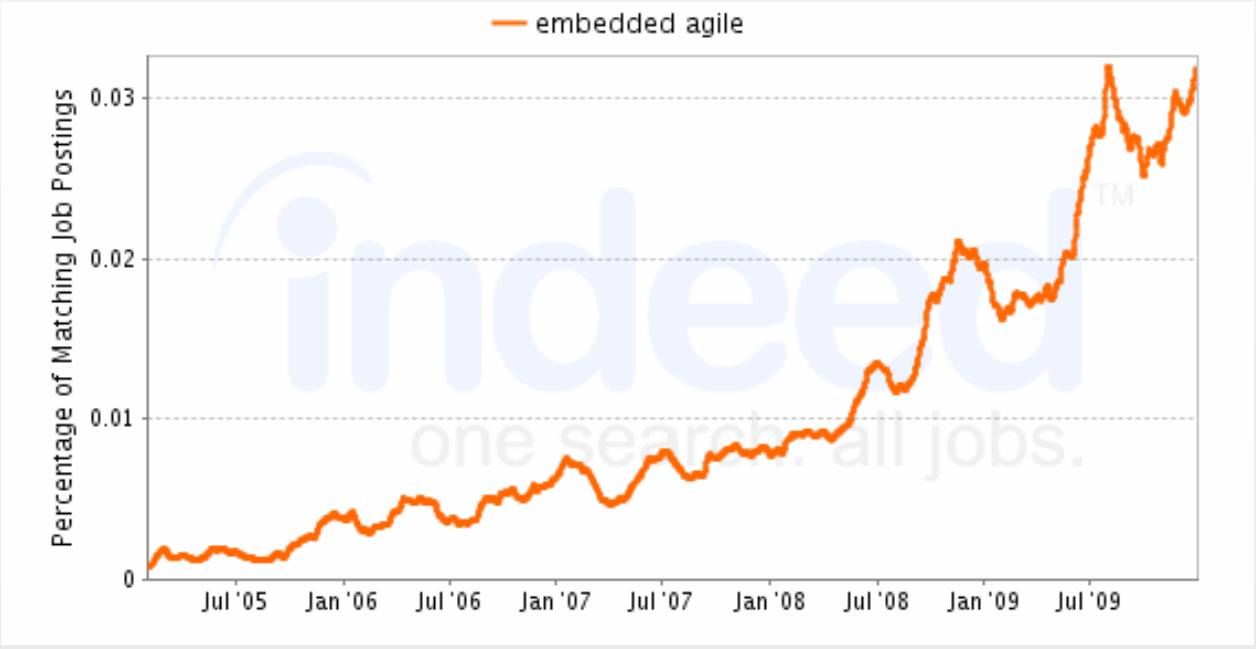
- **El software es una parte significativa del esfuerzo del desarrollo de embebidos, mientras que el diseño de hardware se le parece cada vez más**
  - Se usan mucho los lenguajes de descripción de HW (ej., VHDL, Verilog)
  - Ganan impulso los de verificación de HW (ej., *e*, OpenVera) y de modelado a nivel de transacciones (ej. SystemC)
  - Con FPGAs y simuladores, se pueden obtener prototipos frecuentemente (símil compilar).

- **Las metodologías ágiles se centran en la adaptabilidad y el trabajo en equipo, que resultan particularmente útiles en el co-diseño de hardware y software**

Demanda de “Embedded Agile”

- Avisos en indeed.com cuyo texto incluye “embedded” versus los que incluyen “embedded” y “agile”





El Equipo XP (Programación Extrema)

- **"Clientes"**
  - **Definen** el producto
  - Incluyen un **dueño del producto**
    - Que es el responsable de la visión del producto

- **"Programadores"**
  - Escriben el código, diseñan la arquitectura, etc.
- **Validadores**
  - "Investigan", en busca de defectos
  - Son "optativos", porque los clientes y programadores pueden cumplir sus funciones

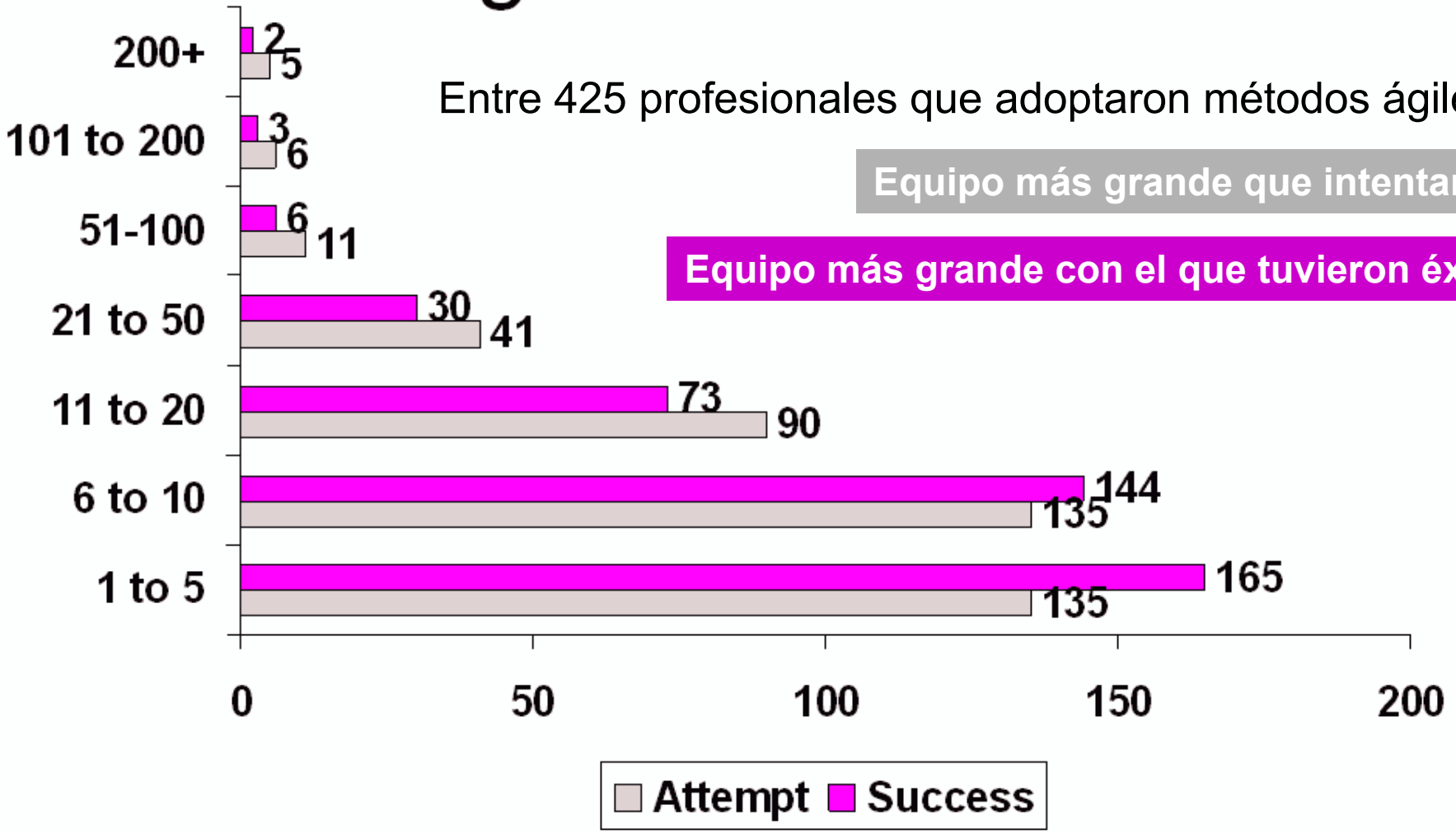
- **Entrenadores**
  - Un **entrenador de programadores**
    - Programador experimentado, para atender consultas, liderar en las decisiones importantes, y evaluar lo que hacen los otros
  - Si hace falta, un **administrador de proyecto**
    - No es imprescindible porque el equipo se auto-organiza
- **Otros**
  - Si hacen falta: escritores, analistas ISO 9000, etc.

Agile Team Size

Entre 425 profesionales que adoptaron métodos ágiles:

Equipo más grande que intentaron

Equipo más grande con el que tuvieron éxito



Los “Clientes”

- **Son los responsables de establecer los requerimientos del sistema**
  - Utilizando las estimaciones de costo que les proveen los “programadores”
- **Entre los “clientes” pueden haber:**
  - Clientes reales de la empresa
  - Expertos de dominio (ej., analistas de negocios, científicos)
  - Diseñadores de interfaces al usuario
- **Proporción típica: 1 cada 3 programadores**
  - O los necesarios para mantener ocupados a los programadores

Plan del *Release*

- **Release es cada "vendible" que se le entrega al usuario final**
- **Se aconseja que demore unos tres meses como máximo**
- **Empieza con una planificación, a cargo de los "clientes"**
  - A cada requisito se le llama ***historia de usuario***
    - Ej.: Un usuario que necesita tal cosa, opera el producto de tal manera, obtiene tal respuesta, etc. Casos puntuales y concretos.
    - …pero dejando los detalles para después
  - Los "programadores" los asisten, estimando el *esfuerzo* (o sea, tiempo) que demoraría cada historia
    - Para que puedan decidir qué dejar para un próximo *release*
  - Obtienen así el ***release plan***

Plan de la Iteración

- **El *release* se divide en iteraciones**
  - De **una semana** c/u, por ejemplo
  - Luego de cada iteración, se tiene un producto demostrable
    - …para ser evaluado por los "clientes" y verificado por los validadores
- **Al principio de cada iteración, los "clientes" determinan qué historias se van a hacer en ella**
  - Empezando por las que son clave; la optimización va al final
  - Este plan puede ser modificado en base a las **estimaciones (de costo) de los programadores**
- **Queda así establecido lo que hace cada equipo en el resto de la iteración:**
  - Los "programadores" implementan esas historias
  - Los "clientes" seleccionan y detallan las de la próxima iteración
    - …y atienden consultas de los programadores
      - O sea que los "clientes" reemplazan la documentación pesada
  - Los validadores (si los hay) ponen a prueba los *builds* que proveen los "programadores"

Los "Programadores"

- **Codifican en pares (*pair programming*)**
  - Habitualmente, en una PC compartida, con teclado y mouse para c/u, más dos notebooks individuales
    - Para averiguar más, googlear "pairing workstation"
  - Es una alternativa a la técnica más tradicional: **revisión de código**
  - El objetivo es lograr un código confiable que pueda ser comprendido por todos
- **Antes de programar cualquier unidad, programan su código de prueba**
  - Esto se llama **Desarrollo Guíado por Pruebas** (*Test-Driven Development*, o TDD)
  - Para estas pruebas automatizadas utilizan ejemplos preparados por los "clientes"
  - En C, el código de prueba de una unidad puede consistir de muchas llamadas a ese módulo, con diferentes parámetros, chequeándose las variables retornadas
  - Como las pruebas se **automatizan**, los proyectos necesitan pocos o ningún validador

Otras Técnicas Importantes

Colaboración

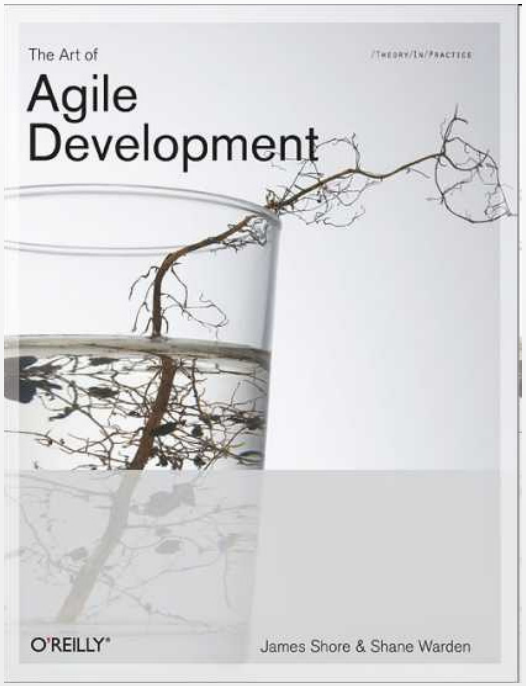
- El ambiente de trabajo debe ser **abierto**
- …con varios **pizarrones** para intercambiar ideas
  - Sugerencia: Sacarles fotos a los diagramas en el pizarrón
- Todo código es de todos
  - Usar un sistema de **control de versiones**

Métricas

- Como **métrica de progreso** del proyecto, se usa la suma del *esfuerzo* de las historias ya implementadas y verificadas
  - Y como métrica de lo que resta por hacerse, se usa la suma del esfuerzo de la historias que faltan
- La cantidad de historias a hacer puede ser **ajustada**, en función de la **velocidad** (o sea, progreso / tiempo) que está logrando el equipo

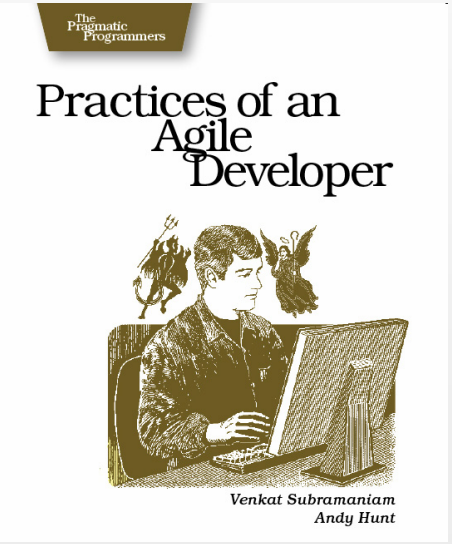
Bibliografía

- **Para ver más: http://www.extremeprogramming.org**



The Art of Agile Development; J.Shore

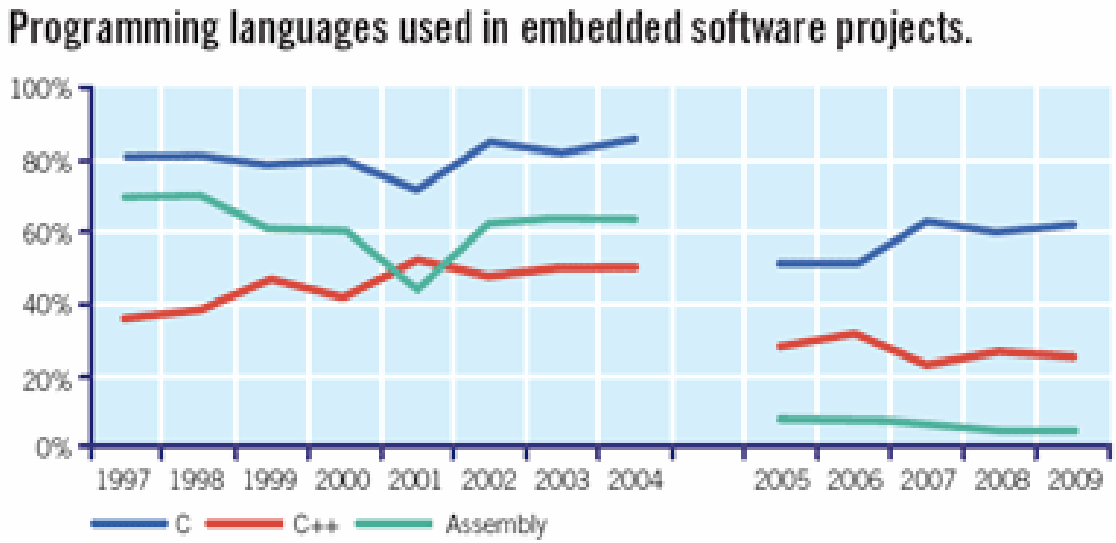
Practices of an Agile Developer; V.Subramanian, A.Hunt



- **Y http://www.agilemodeling.com/ para juntarlo con el próximo tema**
  - …que viene ahora

Nivel de Abstracción Más Alto

- **Hay una tendencia a trabajar en nivel más alto y usar más herramientas y lenguajes (incluyendo los de tipo gráfico).**
- **Ej., principal lenguaje de programación para embebidos:**
  - Ayer: Assembly
  - Hoy: C/C++
  - ¿Mañana: Model-Driven Development (MDD)?
    - MDD es el diseño basado en modelos gráficos, o de otros tipos cercanos al problema (ej., Simulink, Matlab, UML, LabView)



Proyectos de SW embebido: Lenguajes empleados (hasta 2004) / Lenguaje principal (desde 2005)

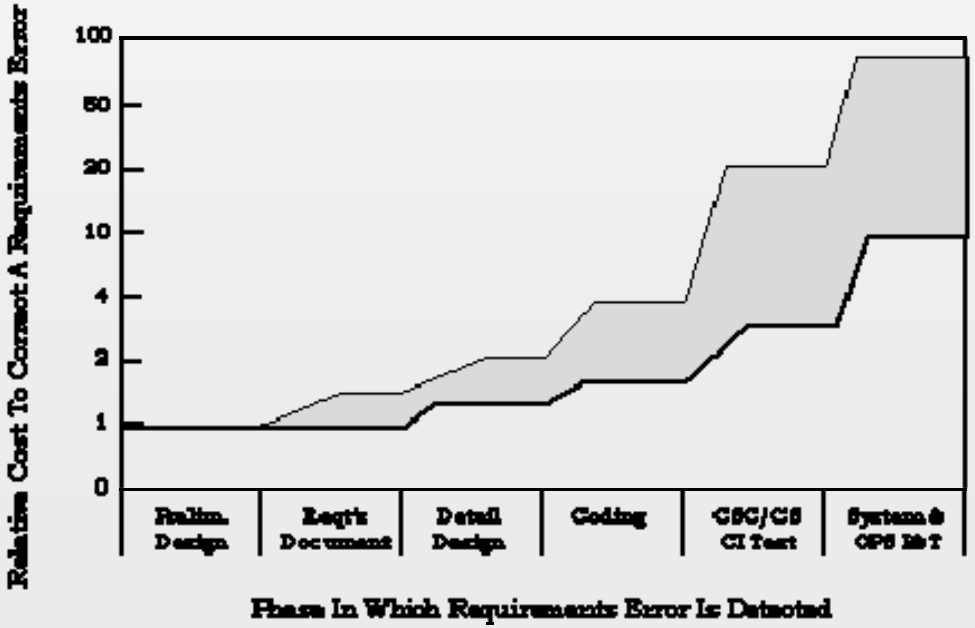
Fuente: http://i.cmpnet.com/embedded/2009/0709_July2009/0709esdBarr01sm.gif

Ventajas de Trabajar en Nivel Alto

- **Es una manera de manejar la complejidad creciente.**
- **La tecnología lo permite**
  - Porque progresaron las herramientas y los componentes.
- **Se potencia el trabajo en equipo**
  - Porque buenos lenguajes evitan malentendidos y facilitan la documentación.
- **Es difícil encontrar profesionales que manejen el dominio del problema y el de la implementación también.**
- **Se pueden corregir errores temprano, sin que produzcan más gastos.**

El Costo de solucionar una falla (ej., en los requerimientos) crece exponencialmente con la demora en hacerlo:

Fuente: nasa.gov

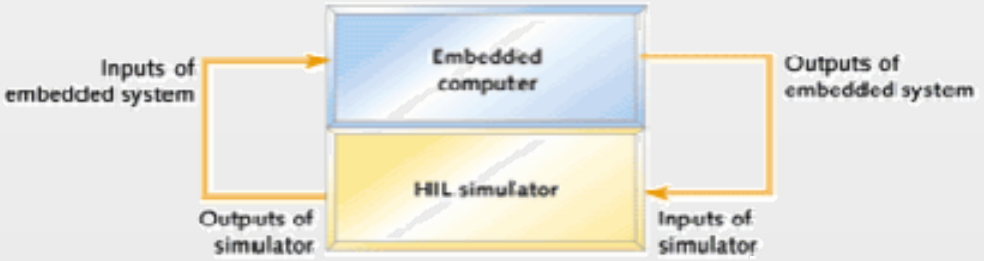


¿Qué es un Modelo?

- **Es la descripción del funcionamiento o la estructura de un sistema, o de alguna de sus partes, en un nivel alto de abstracción**
  - Pueden emplearse en los requerimientos, en la definición de la arquitectura o en el diseño detallado
- **Lo forman uno o más artifacts**
  - Que pueden estar en pizarrón (*whiteboard*), papel, o en un archivo de computadora.
- **Los modelos generalmente están expresados en lenguajes cercanos al problema**
  - Frecuentemente son **gráficos** o **matemáticos**
- **Ejemplos: escribir un código Matlab, diagramar en Simulink o en un diagrama de estados.**

El Modelado Sirve Para:

- **Organizar y comunicar ideas eficientemente.**
  - …al pensar un diseño, hacerlo en equipo, y documentar
- **Encontrar defectos temprano**
  - …si es que se puede ejecutar o chequear formalmente
    - A veces se dice *simular* en lugar de *ejecutar*
    - Las técnicas de chequeo formal intentan *demostrar* que es correcto, como se demuestra un teorema
- **Implementar un sistema embebido**
  - …si contamos con herramientas de MDD
    - MDD=*Model-Driven Development*
      - (vamos a verlo más adelante)
- **Representar el *entorno* de un sistema embebido, para verificarlo**
  - Ej., simulación de “hardware in the loop” (HIL, ver figura)



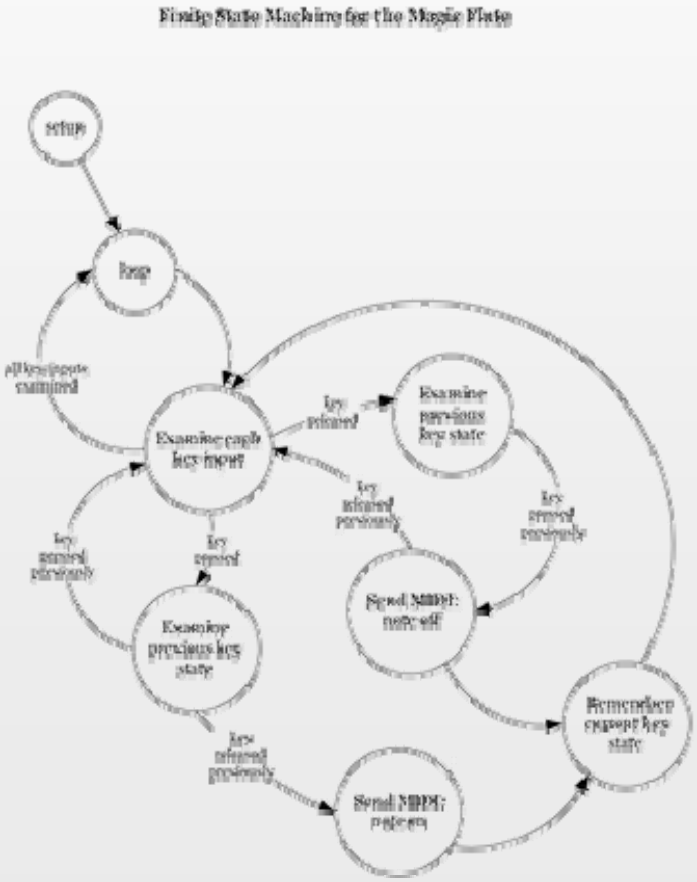
Fuente: embedded.com

Lenguajes de Modelado

- **Son lenguajes artificiales para construir modelos**
  - Ejemplos: El lenguaje del MATLAB o la notación típica de una máquina de estados finitos
- **Propósitos:**
  - Evitar malentendidos
  - Habilitar el uso de herramientas y su interoperabilidad
  - Facilitar modos eficientes de expresar ideas
- **Muchos de estos lenguajes son gráficos**
- **Algunos son de propósito general y otros son *domain-specific***
  - …o sea, especiales para determinados problemas
- **Los hay abiertos, otros son propietarios**
- **Frecuentemente emplean modelos de computación.**

Modelos de Computación

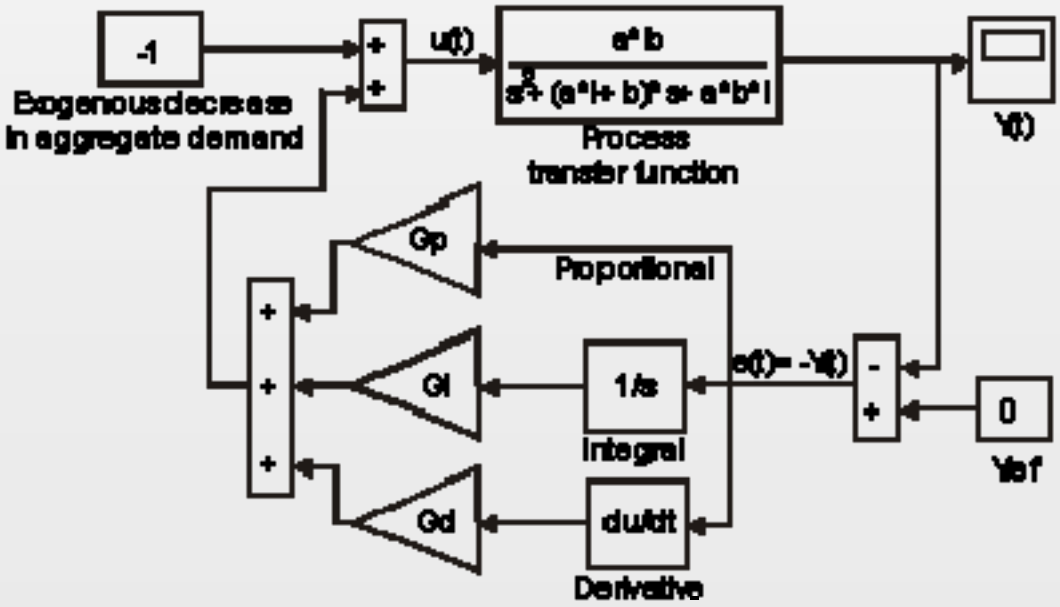
- **Son definiciones abstractas de máquinas capaces de computar**
- **Ejemplo: una máquina de estados finitos (FSM)**



- **Algunos lenguajes de modelado incluyen, por ejemplo diagramas de estado**
  - Ej: los **StateCharts** de UML, o la extensión Stateflow de Simulink
  - A veces les incorporan elementos nuevos y/o una sintaxis particular

Modelo de Flujo de Datos

- **Representación gráfica de cómo se mueven los datos entre los distintos procesos o componentes**
  - Como la que se usa en DSP
- **También se le dice Data-Flow Diagram (DFD)**
- **Puede ser en tiempo discreto o continuo**

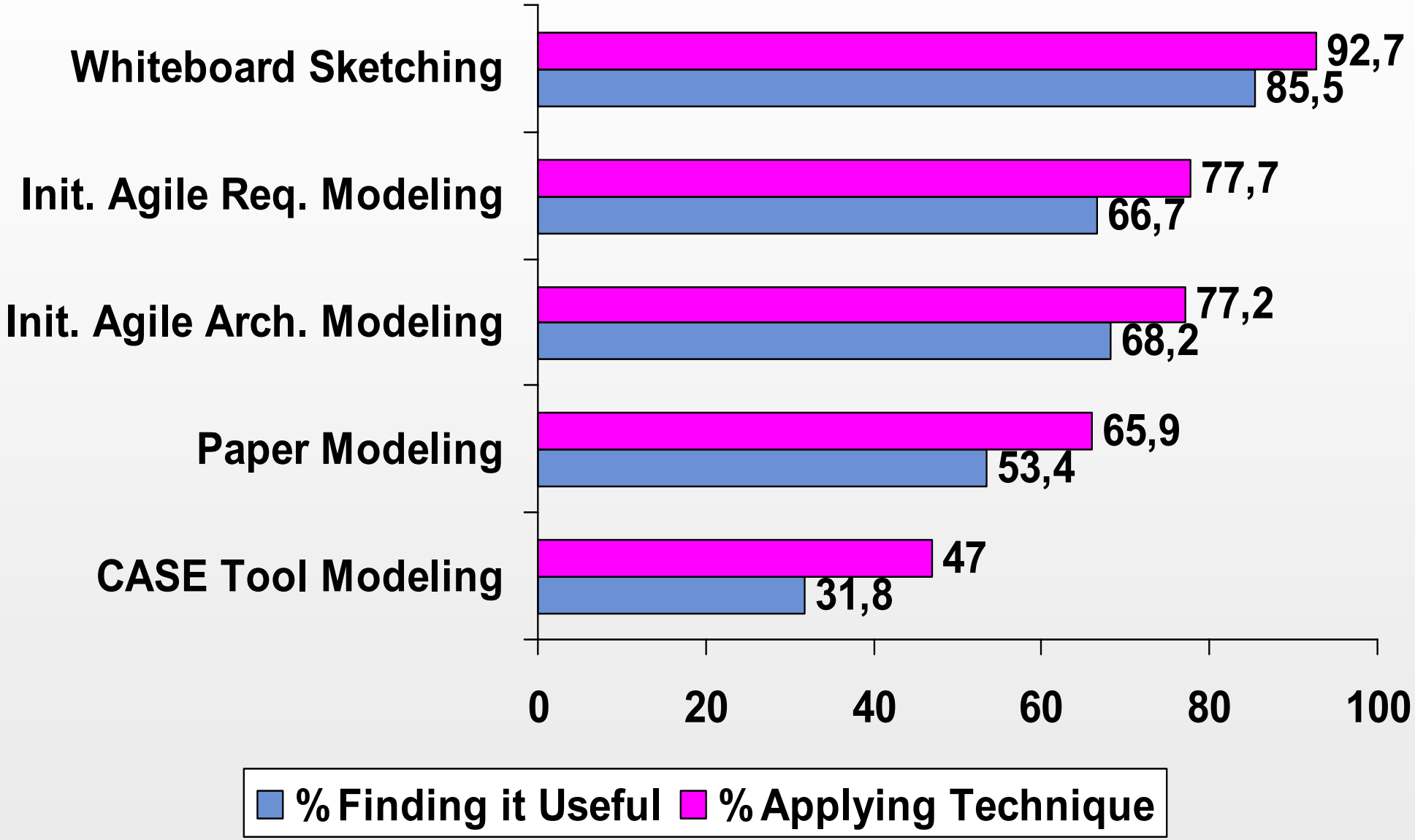


Utilización de Lenguajes de Modelado en la Industria

- **2006 State of the Embedded Market Survey: Encuesta a 1217 suscriptos a publicaciones sobre embebidos y visitantes a conferencias.**
- **Preguntas:** "My current embedded project uses…" y "My next embedded project is likely to use…"

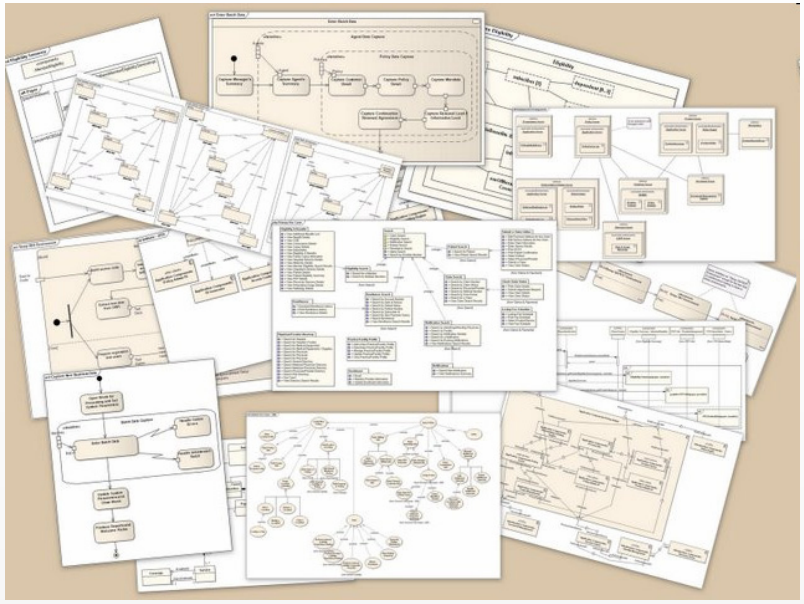
Current Project	2006 %	2005 %
SystemC or other "hardware C" language	19	16
UML	15	17
Other hardware/software codesign or coverficiation tool	15	11
SimuLink or other modeling language	12	12
None of the above	53	58
Next Project		
UML	23	26
SystemC or other "hardware C" language	21	21
SimuLink or other modeling language	17	16
Other hardware/software codesign or coverficiation tool	18	17
None of the above	44	44

Survey Says: Agilists are Modeling

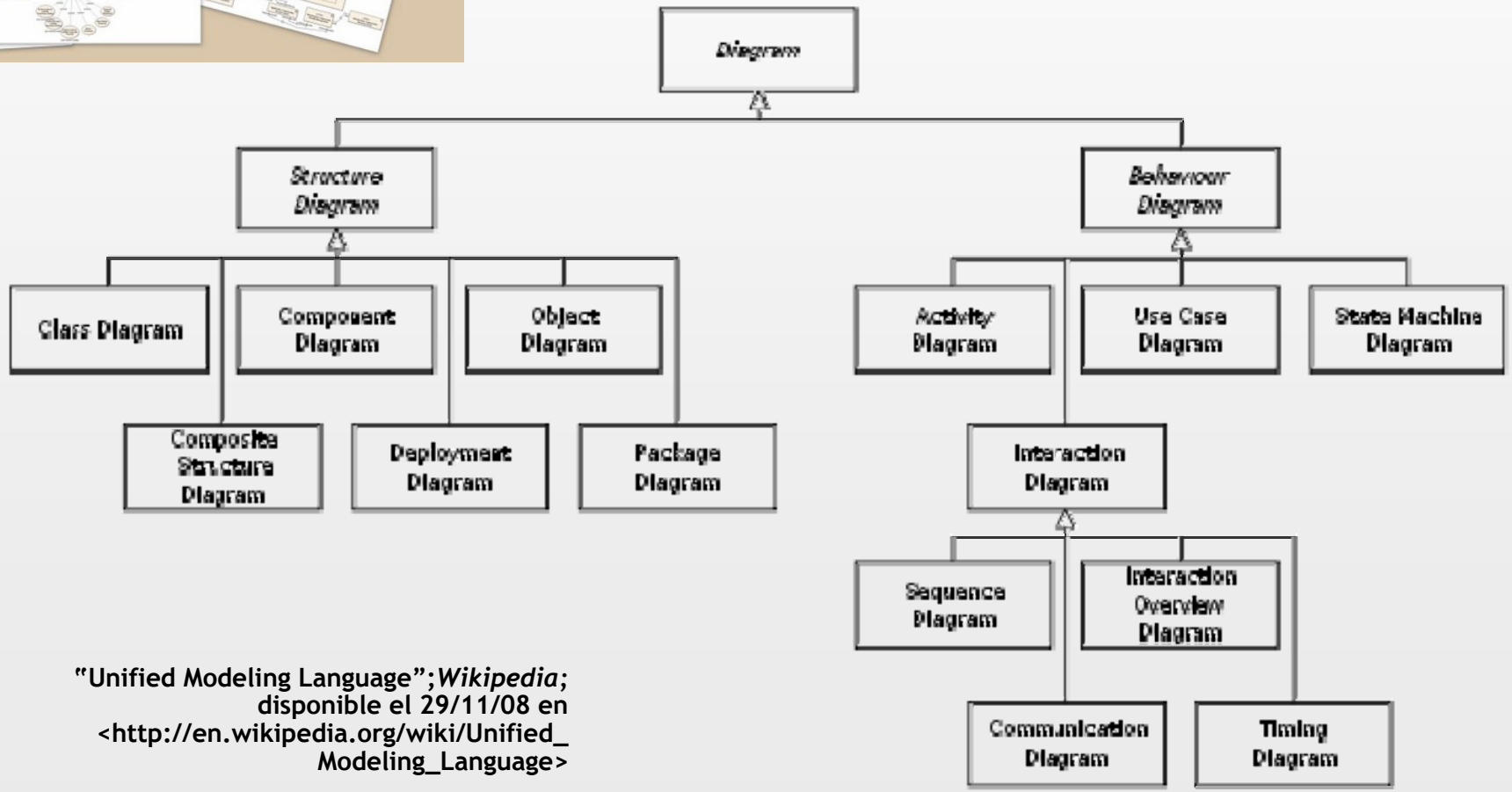


Unified Modeling Language (UML)

- **Lenguaje de modelado estándar en la industria del software**
  - Lanzado en 1996. La versión actual es la 2 (2005).
  - Emplea muchos símbolos que ya se usaban comúnmente desde antes.
  - Está basado en orientación a objetos pero también puede aprovecharse con otros paradigmas.
- **Dada su influencia, hay lenguajes de modelado ajenos al software, y domain-specific, que se basan en UML**
  - Ej.: Systems Modeling Language (SysML)
- **Tiene mecanismos de extensión**
  - Además, usarlo flexiblemente es una práctica frecuente, en lugar de respetar estrictamente la norma.
  - Con estos mecanismos se creó un profile de UML ejecutable (i.e., que se puede simular) llamado **Exectuable UML** o **xUML**.



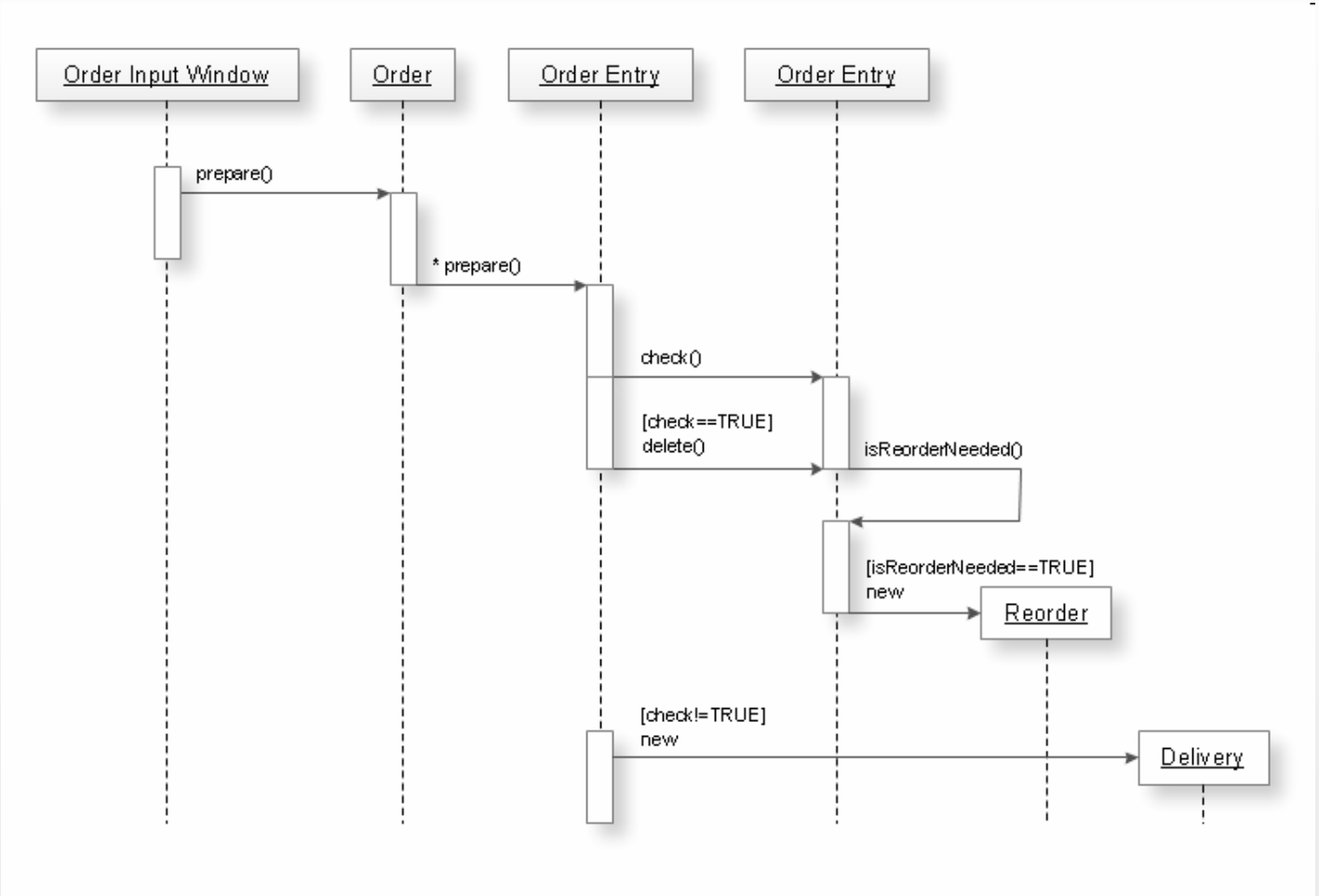
Diagramas del UML 2.0



"Unified Modeling Language";*Wikipedia*; disponible el 29/11/08 en <http://en.wikipedia.org/wiki/Unified_Modeling_Language>

Diagrama de Secuencia

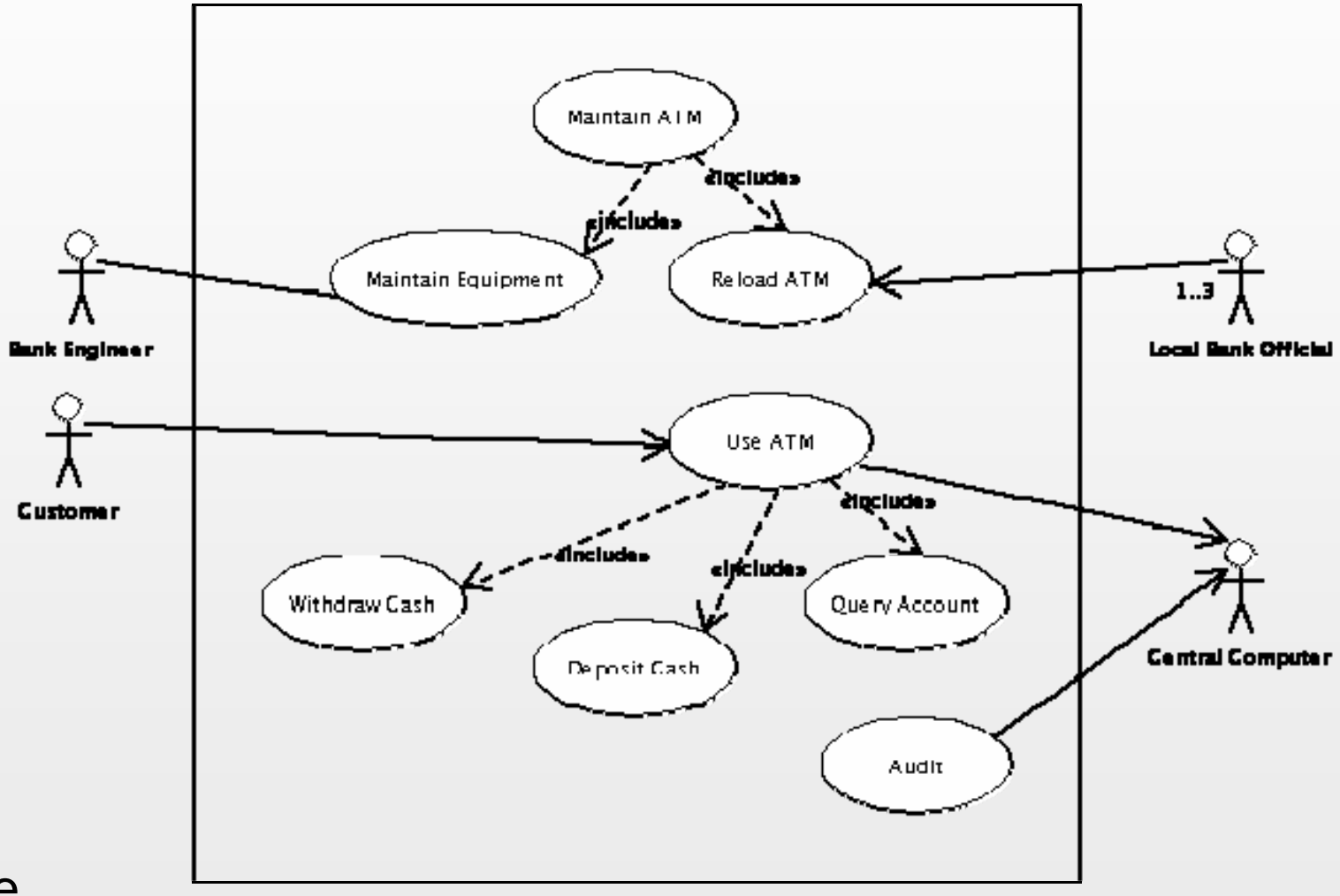
- Muestran las comunicaciones entre varios objetos, a lo largo del tiempo
- La línea punteada se hace rectángulo cuando el objeto está "activo"



Fuente: http://www.conceptdraw.com/en/products/cd5/ap_uml_tool.php

Diagrama de Casos de Uso

- En inglés: *Use case diagram*
- Muestra a los usuarios del sistema (o sea, *actores*), sus objetivos al utilizarlo (o sea, *casos de uso*), y las relaciones entre los casos de uso
  - Un actor no necesariamente es una persona



ArgoUML User Manual, A. Ramirez et al. ; disponible el 29/11/08 en <http://argouml-stats.tigris.org/documentation/printablehtml/manual/argomanual.html>

StateCharts

- **Son diagramas de estados, mejorados**

FSM

\+ Jerarquía

\+ Concurrencia

StateCharts

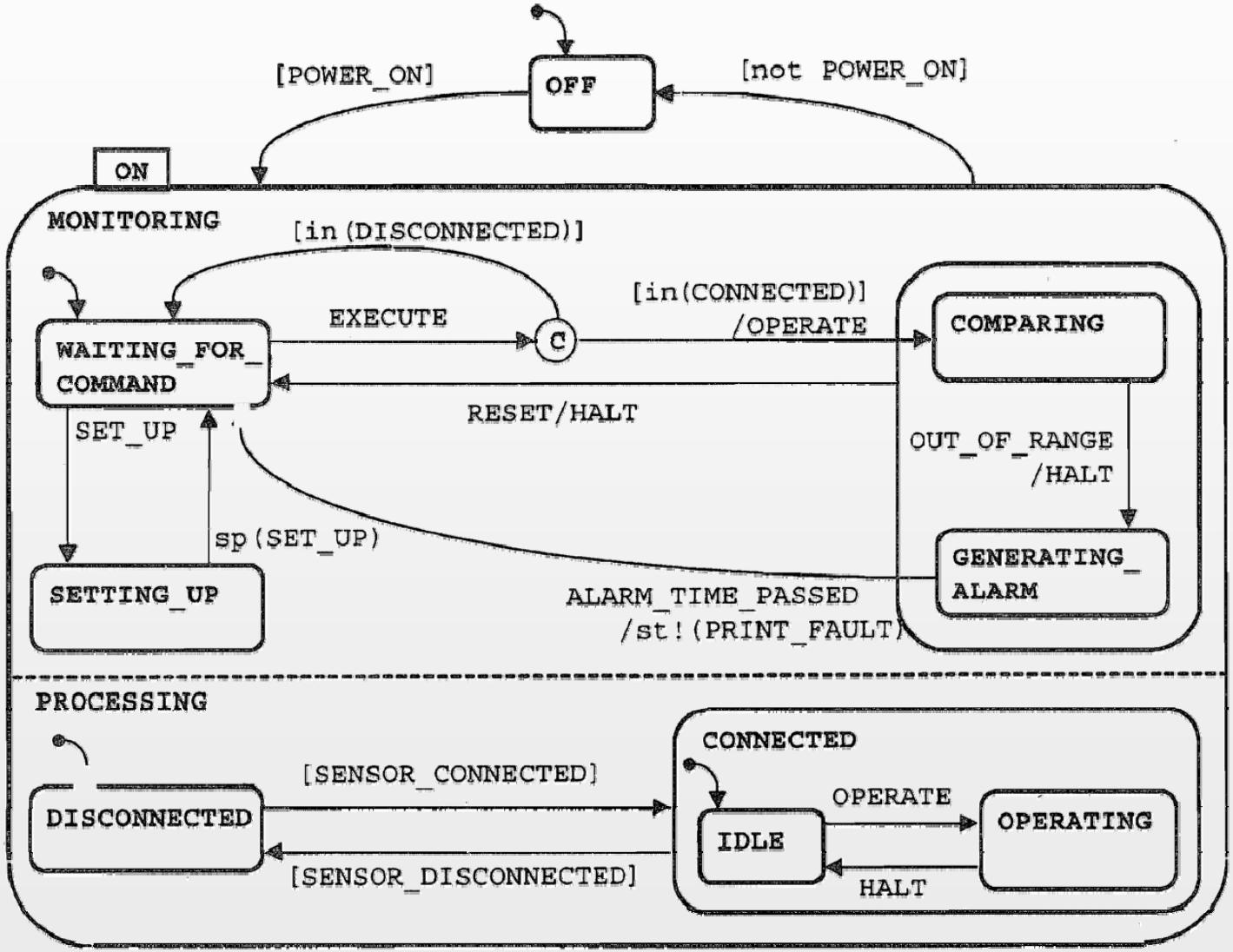
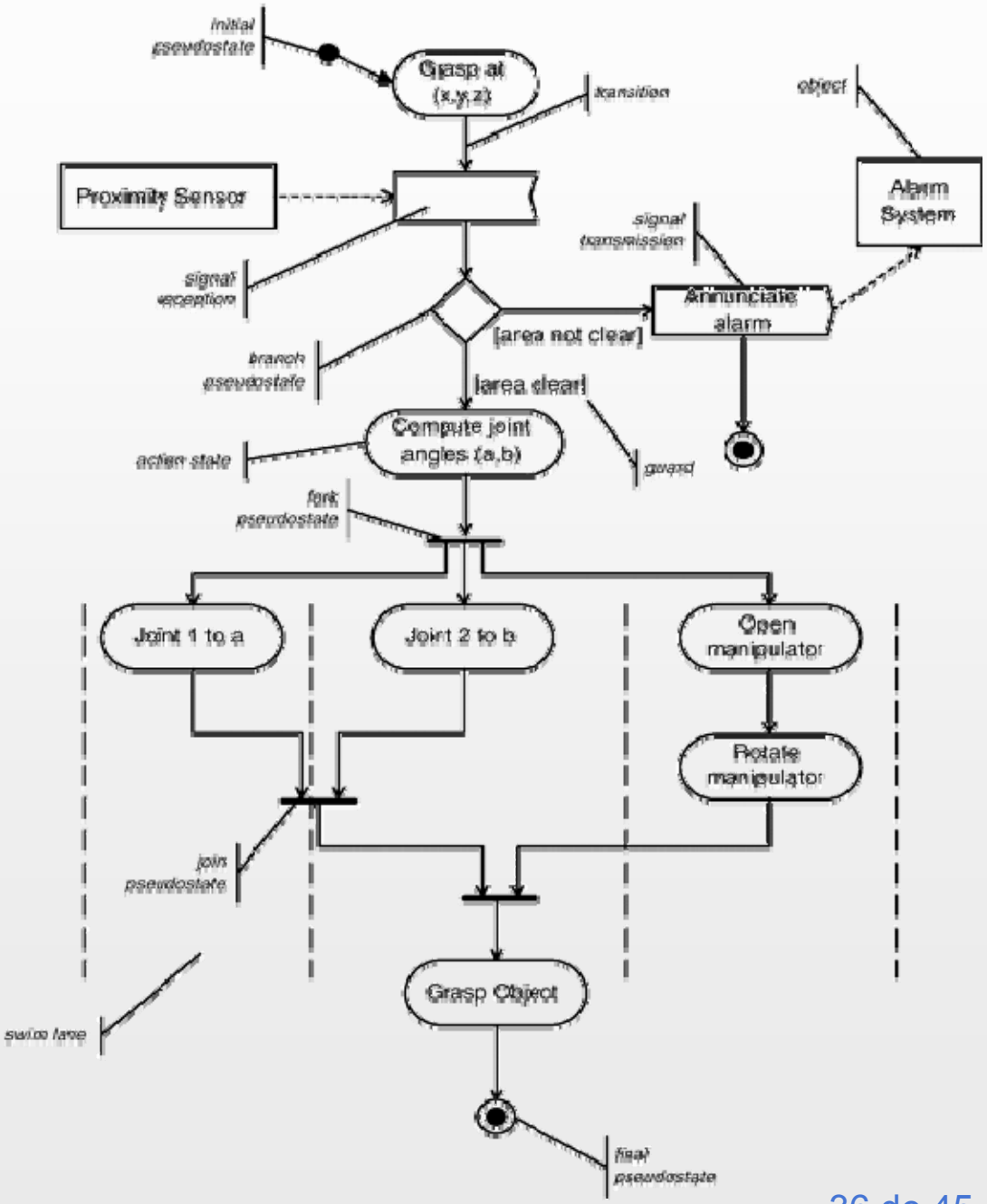


Diagrama de Actividad

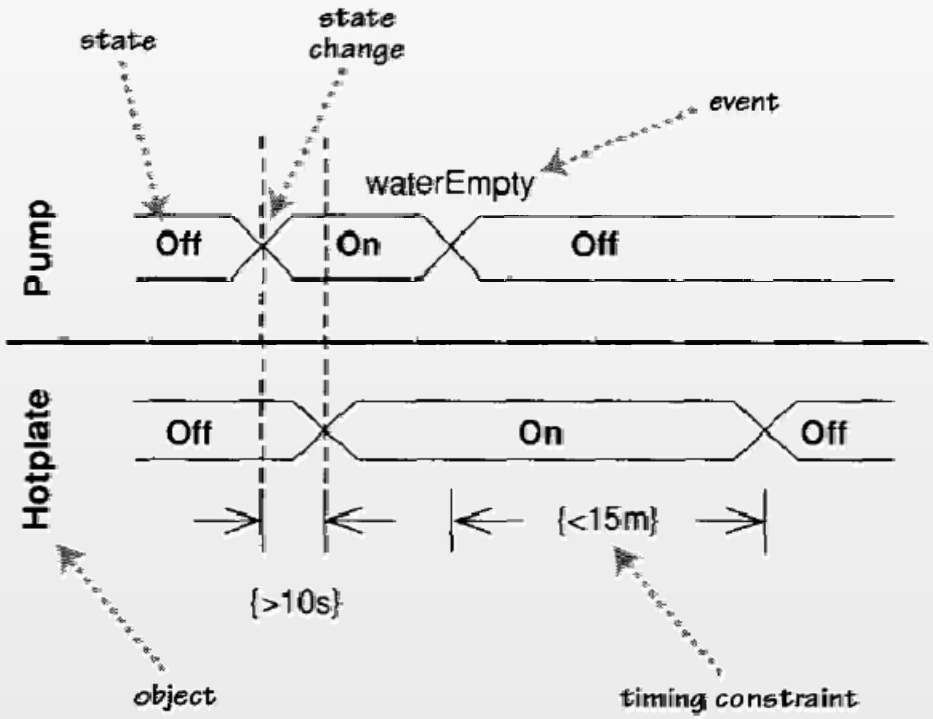
- **Mezcla de los viejos diagramas de flujo con concurrencia basada en tokens, estilo redes de Petri.**
  - Ojo, es un diagrama de flujo *de control*, no de datos



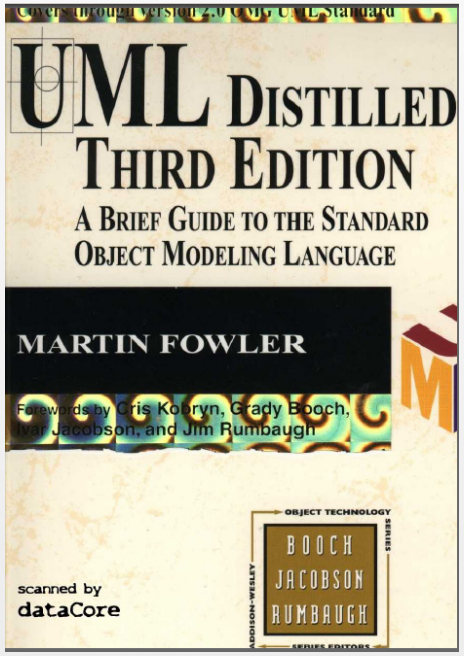
Real-Time Design Patterns: Robust Scalable Architecture for Real-Time Systems; p. 28; B.P. Douglass; Addison Wesley; 2002

Diagrama de Temporización

- O *timing diagram*
- Como los que se usan en digitales



Timing diagram showing states as areas

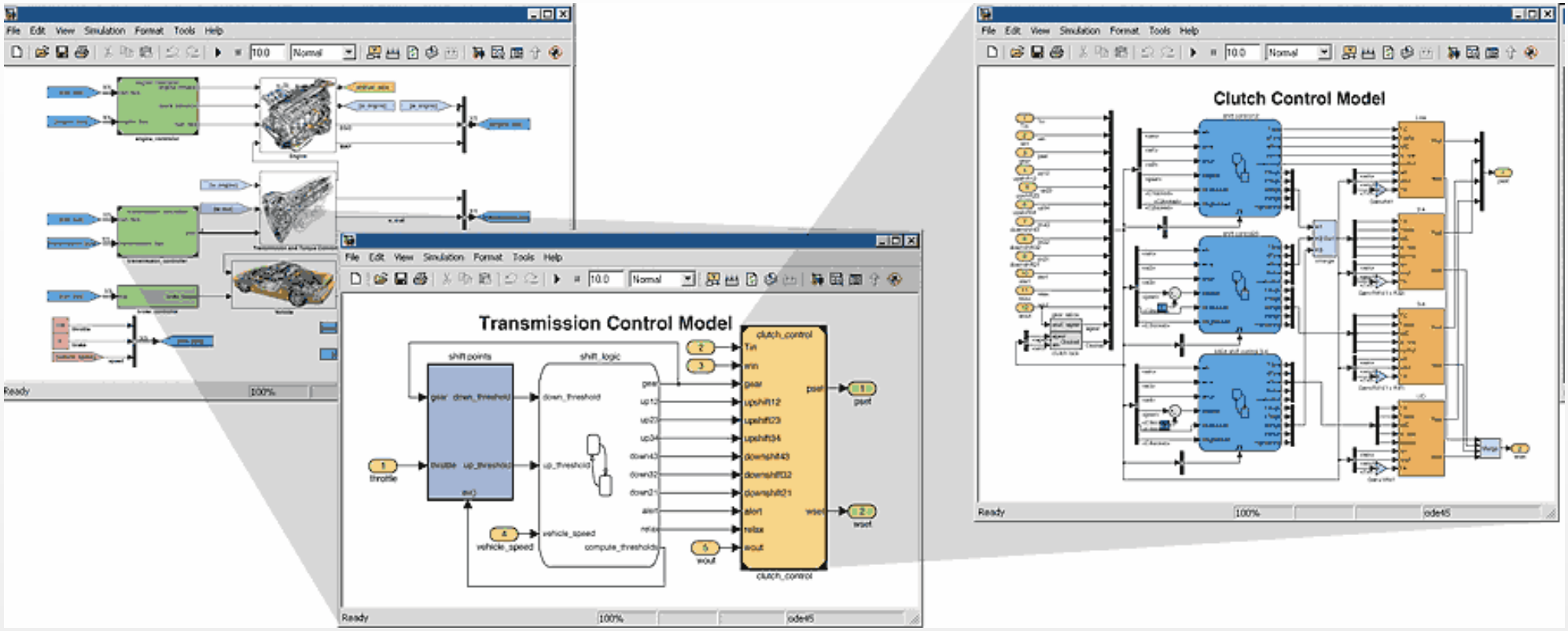


El diagrama se tomó de este libro, que ofrece un resumen detallado (160 páginas) del UML 2

Desarrollo Basado en Modelos

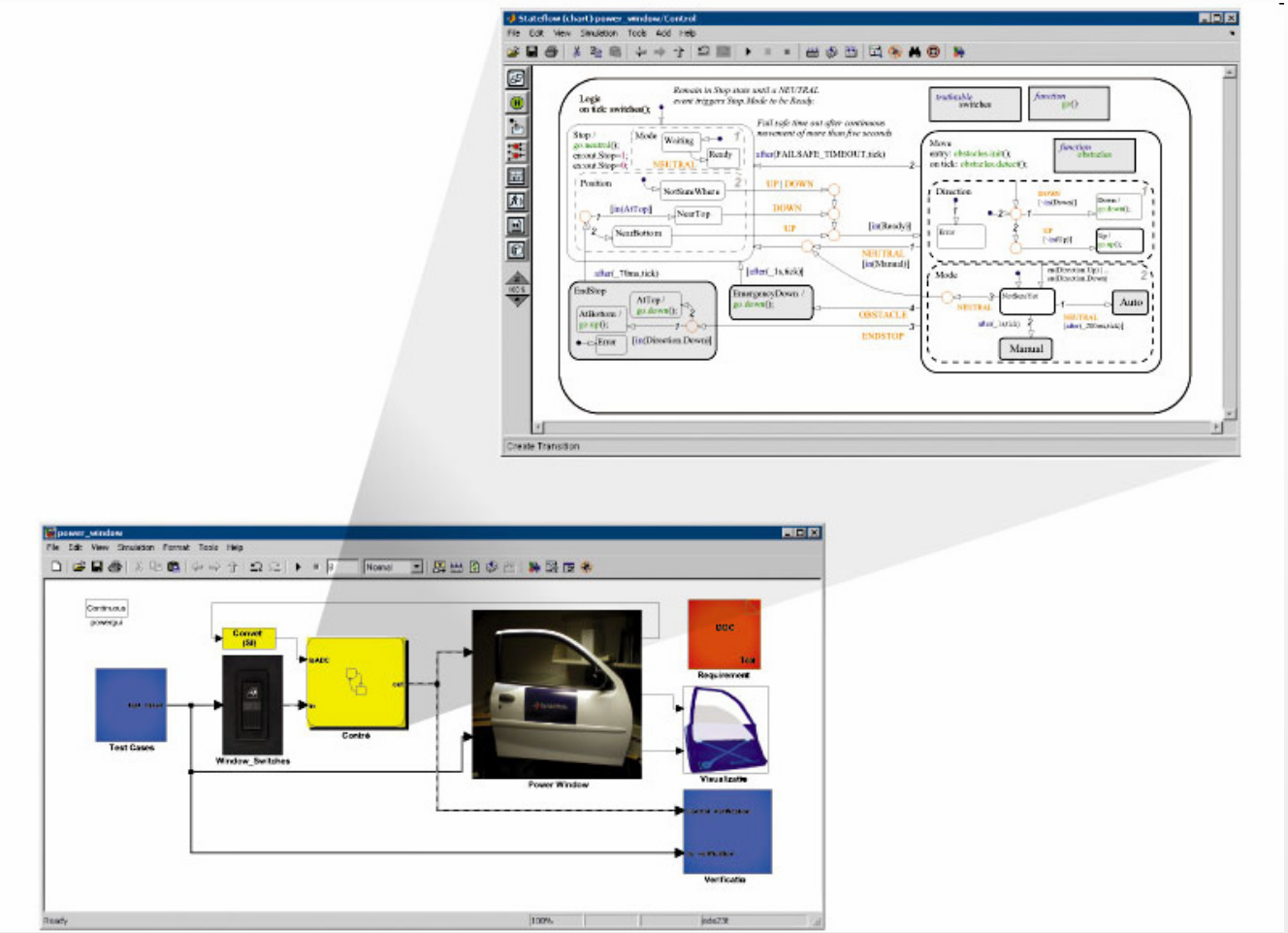
- **Significa que modelos ejecutables sean el "código fuente" principal de los desarrollos.**
- **Se lo llama Model-Driven Architecture (MDA) o Model-Driven Development (MDD)**
  - "There are many views and opinions about what MDA is and is not." (Alan Brown, Staff, IBM)
- **Tratan la provisión de herramientas basadas en modelos, para creación, transformación, testeo, análisis, simulación, ing. inversa, etc.**
  - Ejemplo de MDD para aplicaciones embebidas generales: **Telelogic Rhapsody** (de IBM)
- **A este tipo de herramientas antes se las categorizaba como *CASE* (Computer-Aided Software Engineering)**
  - El término CASE fue tan abusado que ahora se lo utiliza poco.

Simulink®



- **Es un entorno para modelado (gráfico) de los fabricantes de Matlab**
  - Pueden embeberse código Matlab en un modelo Simulink, y viceversa.
    - También puede embeberse código en C, Fortran o Ada
  - Puede usarse Matlab para analizar la salida de un modelo Simulink
- **Los modelos son ejecutables**
  - O sea que sirven para detectar y corregir fallas, temprano
    - Además de ser útiles para diseñar y documentar
- **Trabaja con diagramas de flujo de datos, en tiempo discreto o continuo**
  - Por lo tanto, es especialmente útil para DSP y control

Stateflow® (para Simulink)



- Es un plug-in para poder incluir statecharts en los modelos que se diseñan con el Simulink

Real-Time Workshop® (para Simulink)

- **Plug-in que "compila" el modelo en código C o C++ para microcontroladores**
  - Con llamadas al sistema operativo que se elija, o a ninguno
- **Genera código para**
  1. Inicialización
  2. Implementación de los algoritmos del modelo
  3. Instrumentación (para ajustar parámetros y "data logging")
- **Video de demo en:**
  http://www.mathworks.co.jp/products/demos/rtw/introduction/index.html
- **Está pensado también para simulación de tipo *hardware in the loop* (HIL)**
  - O sea, simular el *entorno* que interactúa con un sistema embebido

Fuente: embedded.com

Telelogic Rhapsody® (IBM)

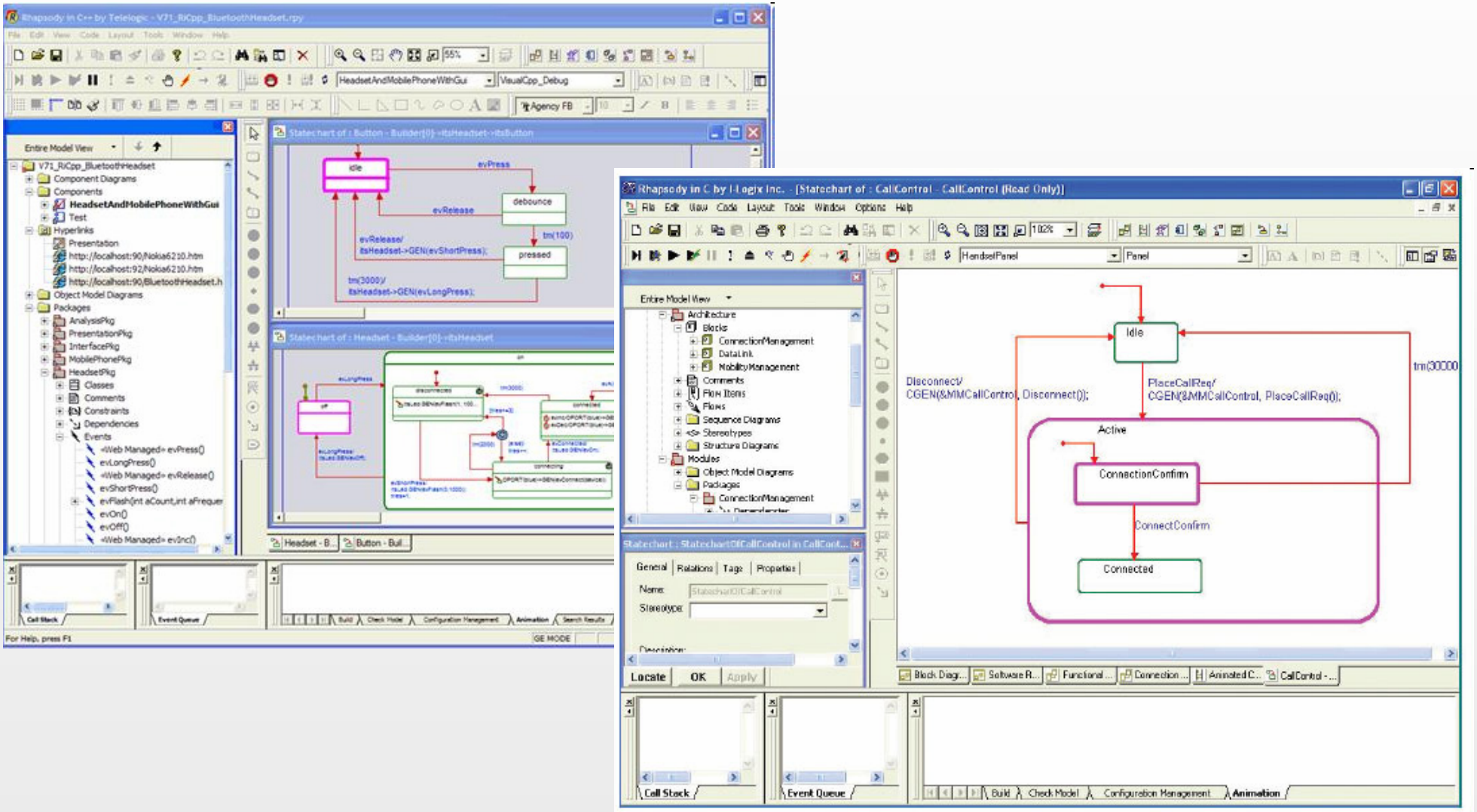
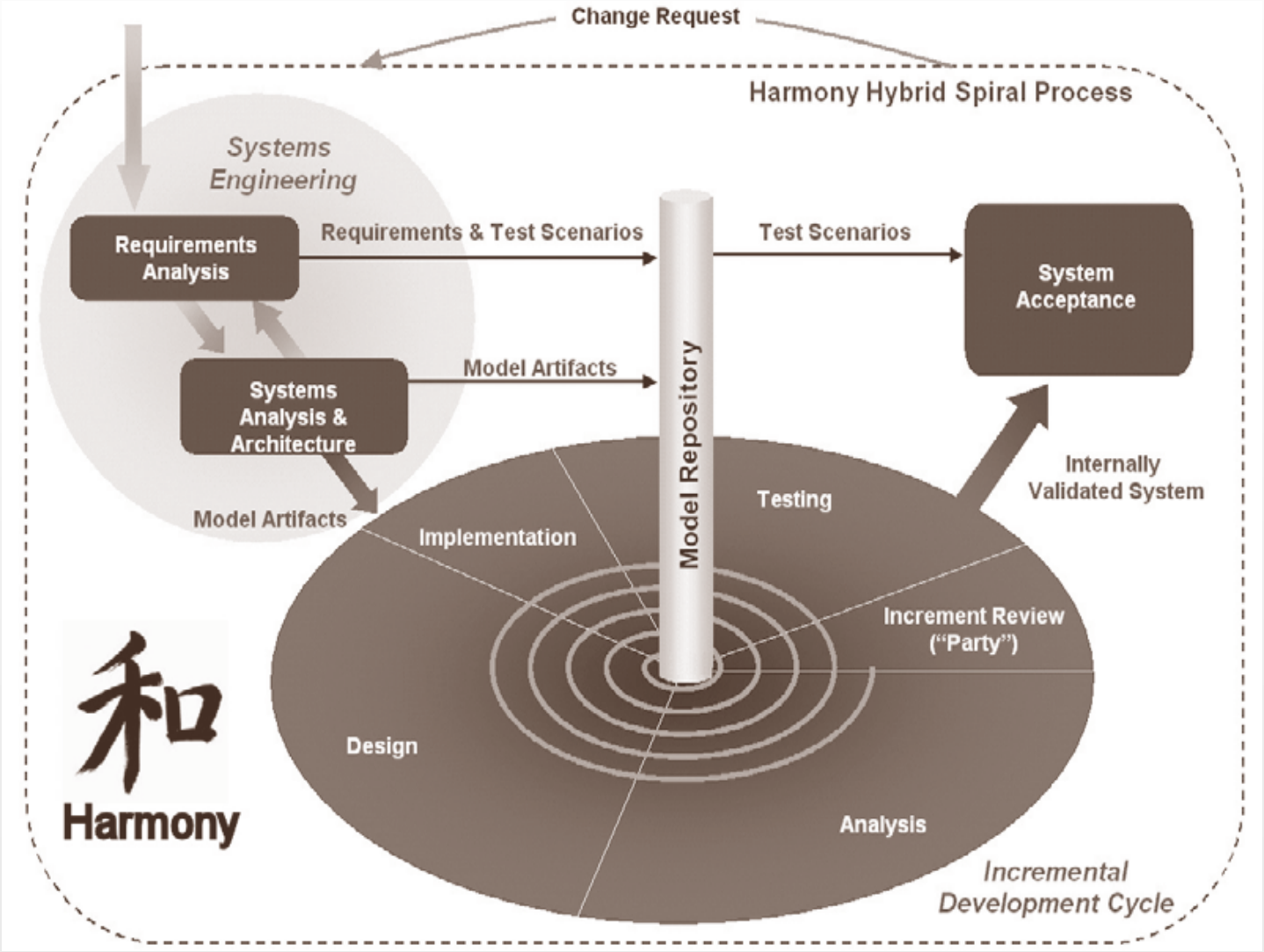


Figure 1 The recently introduced Version 7.0 Rhapsody from Telelogic generates C, C++, or Java code directly from a standard UML model.

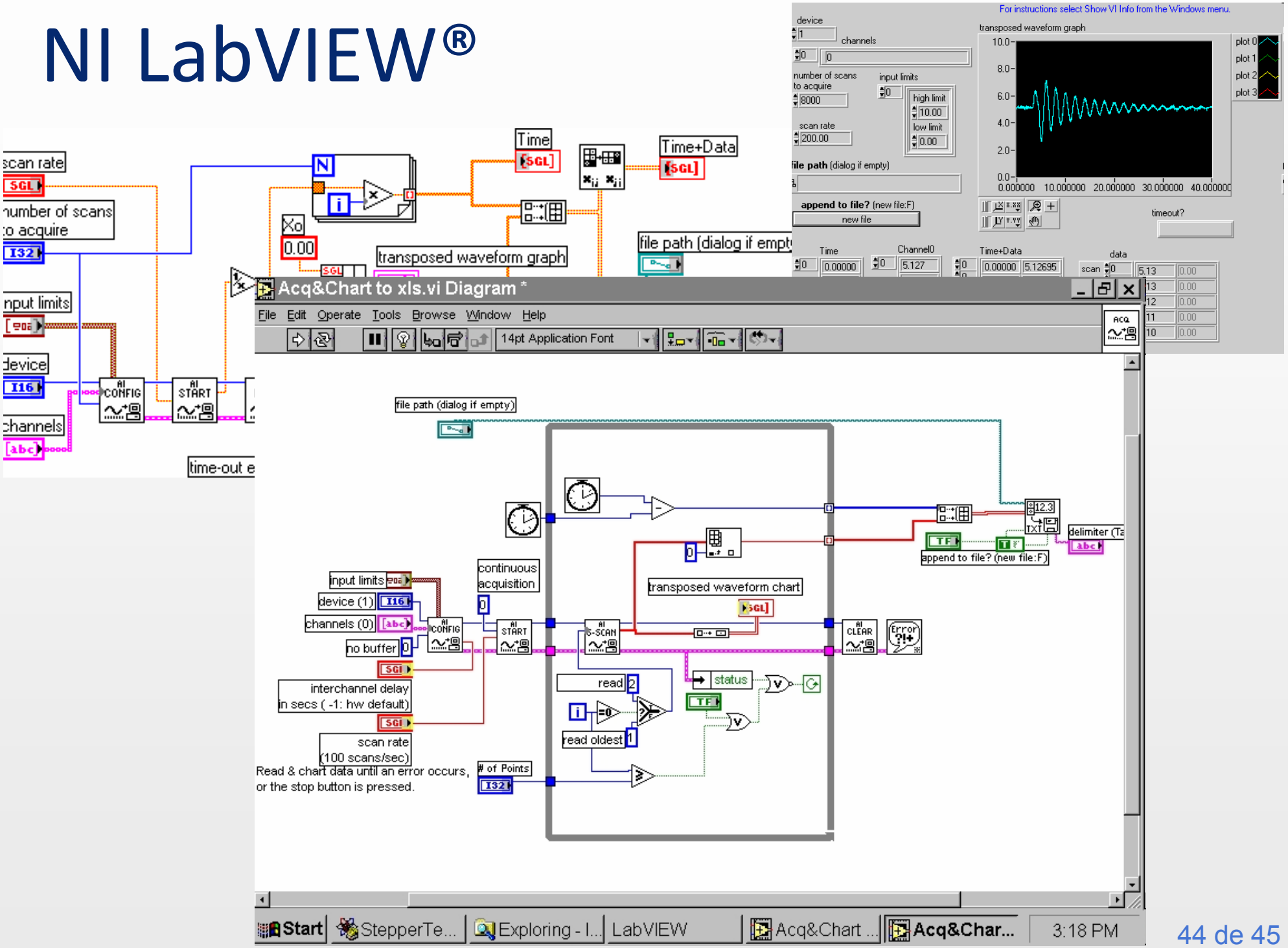
Metodología Asociada al Rhapsody

Harmony Process

- De I-Logix, que fue adquirida por Telelogic, que fue adquirida por IBM
- Emplea UML, y modelo de ciclo de vida que combina espiral y V



NI LabVIEW®


Acq&Chart to xls.vi Diagram *
File Edit Operate Tools Browse Window Help
14pt Application Font
file path (dialog if empty)
continuous acquisition
transposed waveform chart
input limits
device (1)
channels (0)
no buffer
interchannel delay in secs (-1: hw default)
scan rate (100 scans/sec)
Read & chart data until an error occurs, or the stop button is pressed.
# of Points
read
read oldest
status
append to file? (new file:F)
delimiter
transposed waveform graph
Time
Time+Data
Start
StepperTe...
Exploring - I...
LabVIEW
Acq&Chart ...
Acq&Char...
3:18 PM

¡Gracias!

Desarrollo Ágil y Modelado

Andrés Djordjalian <andres@indicart.com.ar>
Indicart Carteles Electrónicos y Facultad de Ingeniería, UBA

Para el Simposio Argentino de Sistemas Embebidos (SASE 2010)

Marzo de 2010