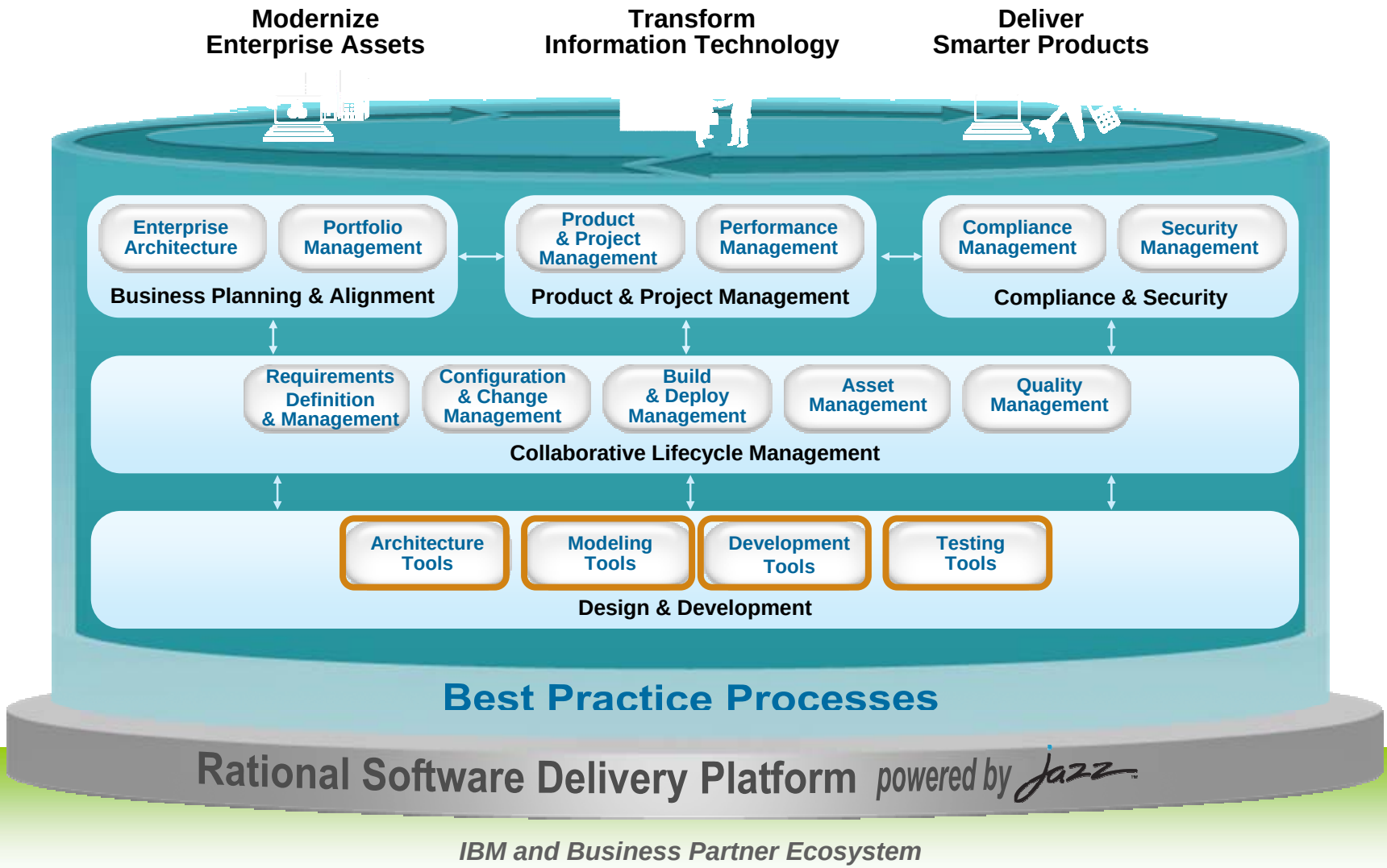
Introducción a IBM Rational Rhapsody

Francisco J. López Minaya
Rational Technical Solution Architect
francisco.lopezminaya@es.ibm.com

Rational. software

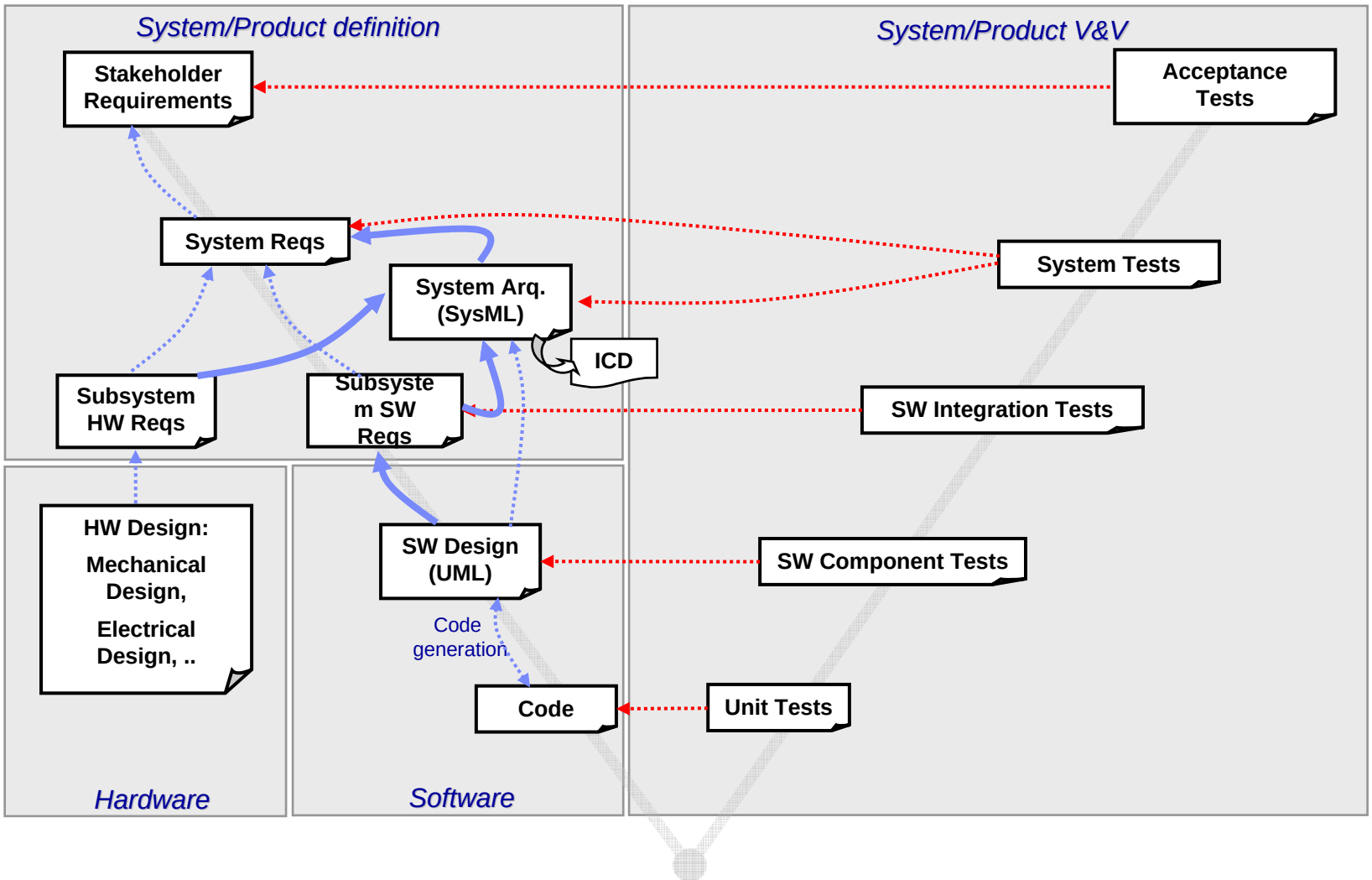
Go to IBM

Rhapsody dentro del portfolio de IBM Rational



IBM and Business Partner Ecosystem

Rhapsody en el desarrollo de sistemas

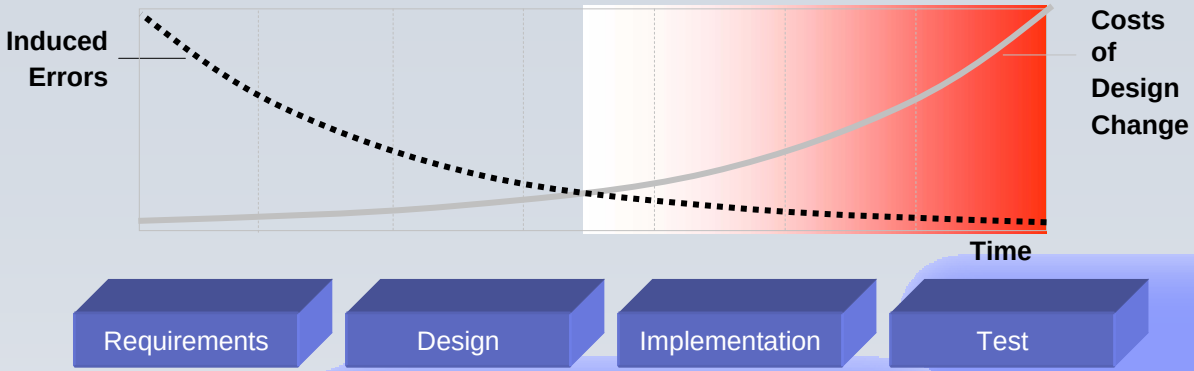


Problemas comunes en el diseño de software y sistemas

- Se encuentran **errores** de diseño demasiado **tarde**, cuando más cuesta solucionarlos.
- O no se encuentran: el **coste** de los errores no encontrados es **muy alto** (retirada o devolución del sistema, penalizaciones, reputación).
- Mala **comunicación** entre diferentes disciplinas: ingenieros de sistemas, ingenieros mecánicos, eléctricos, de software, cada uno habla su propio lenguaje!!!
- **Colaboración** poco eficiente entre equipos de trabajo distribuidos
- Dificultad para gestionar los **cambios** y la evolución continua de los **requisitos**
- Dificultad en la gestión y la integración de **diseños** cada vez más **complejos**, y con sistemas existentes (**Legacy** systems)
- Las **regulaciones y normativas** nos invaden: DO-178B, FDA, …
- **Baja productividad en procesos de prueba**. Procesos lentos y poco ágiles

El tiempo de desarrollo se debe emplear en desarrollar...

El 80% de los costes de desarrollo se emplean en identificar y solucionar errores



$80/defecto

Durante la fase de requisitos

$240/defecto

Durante la fase de diseño y construcción

$960/defecto

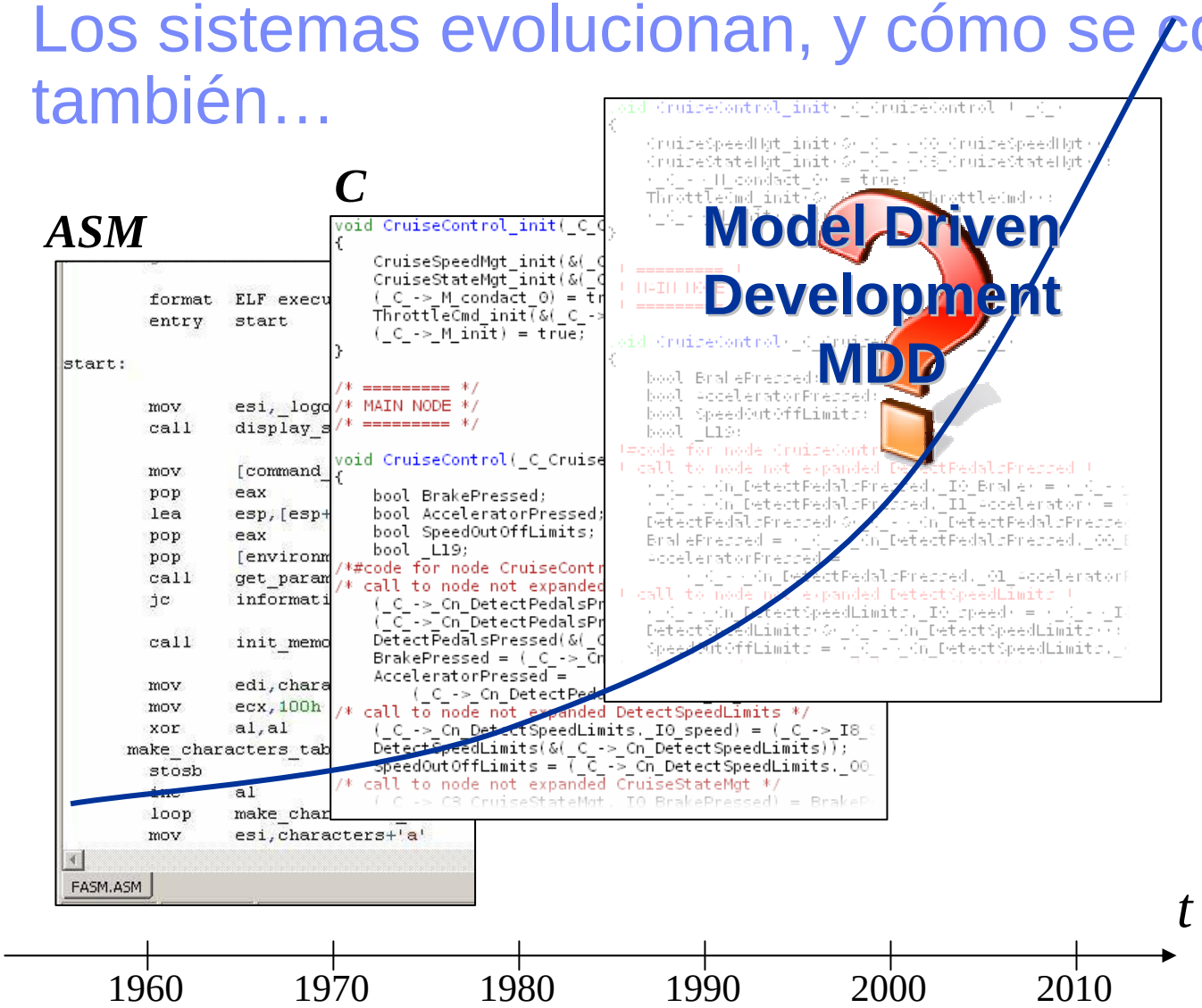
Durante la fase de testing

$7,600/defecto

En producción

Source: 2008 GBS Industry standard study
Defect cost derived in assuming it takes 8 hrs to find, fix and repair a defect when found in code and unit test.
Defect FFR cost for other phases calculated by using the multiplier on a blended rate of $80/hr

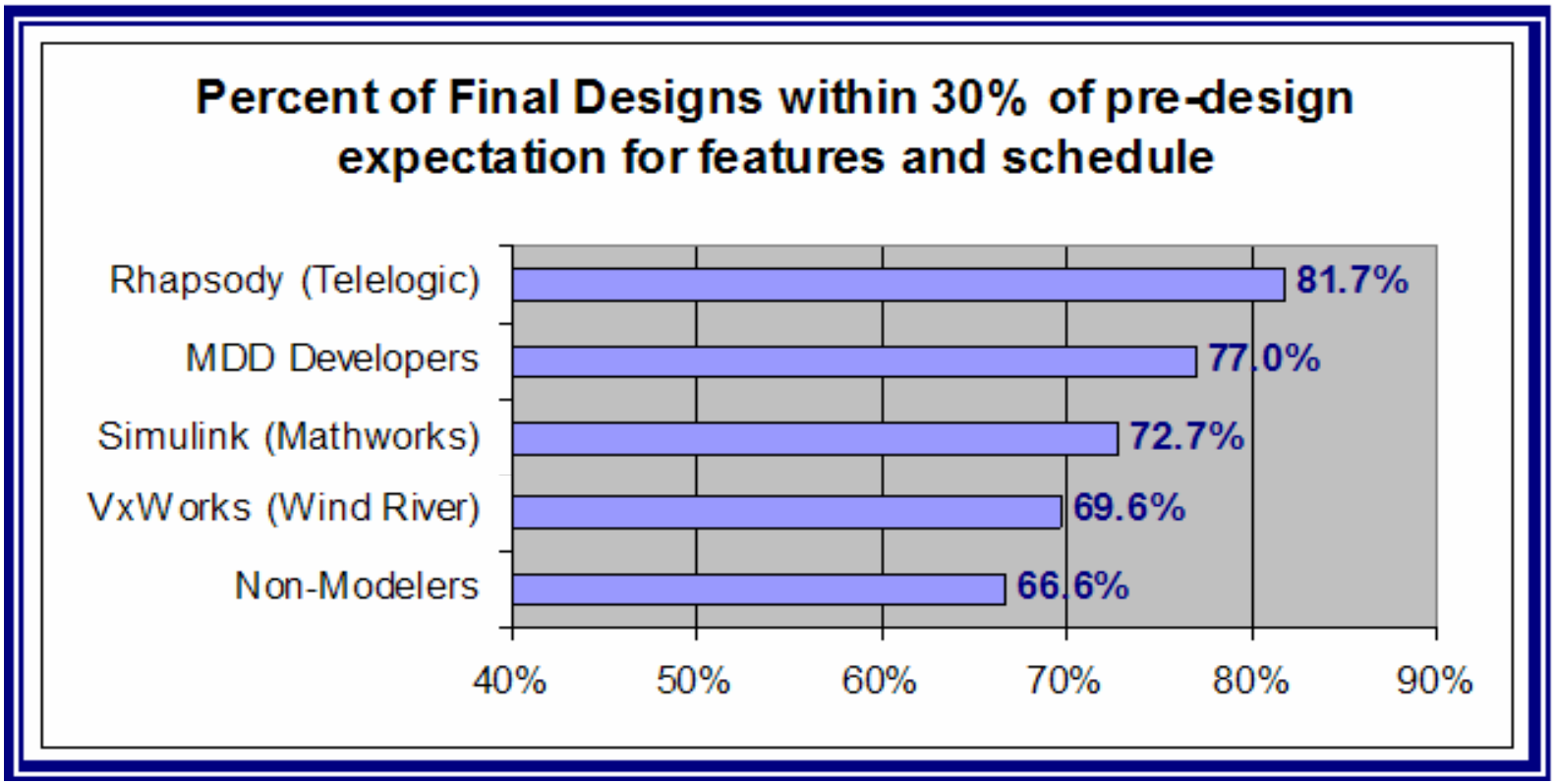
Los sistemas evolucionan, y cómo se construyen también…



Model Driven Development (MDD):

- Mejora la comunicación
- Modelos más fácilmente trazables que el código
- Modelos permiten automatización:
  - ✓ Simulación del modelo
  - ✓ Generación de Código

Model Driven Development es la forma más productiva de desarrollar sistemas: PROBADO!!!

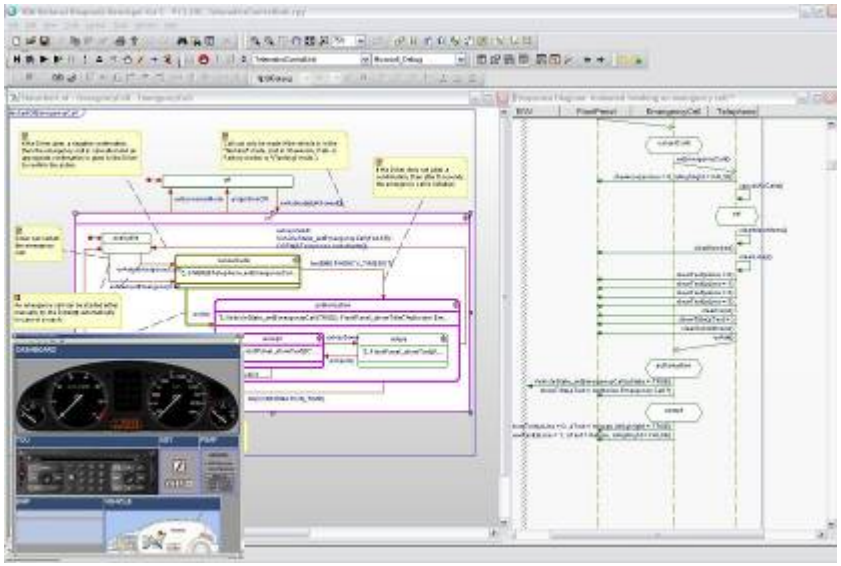


Embedded Market Forecasters

*Documented in "**What Do You Do When the Horse You're Riding Drops Dead? Why Model Driven Design is Emerging as a Preferred Best Practice", March 2007***

IBM Rational Rhapsody®

Model Driven Development para software y sistemas complejos

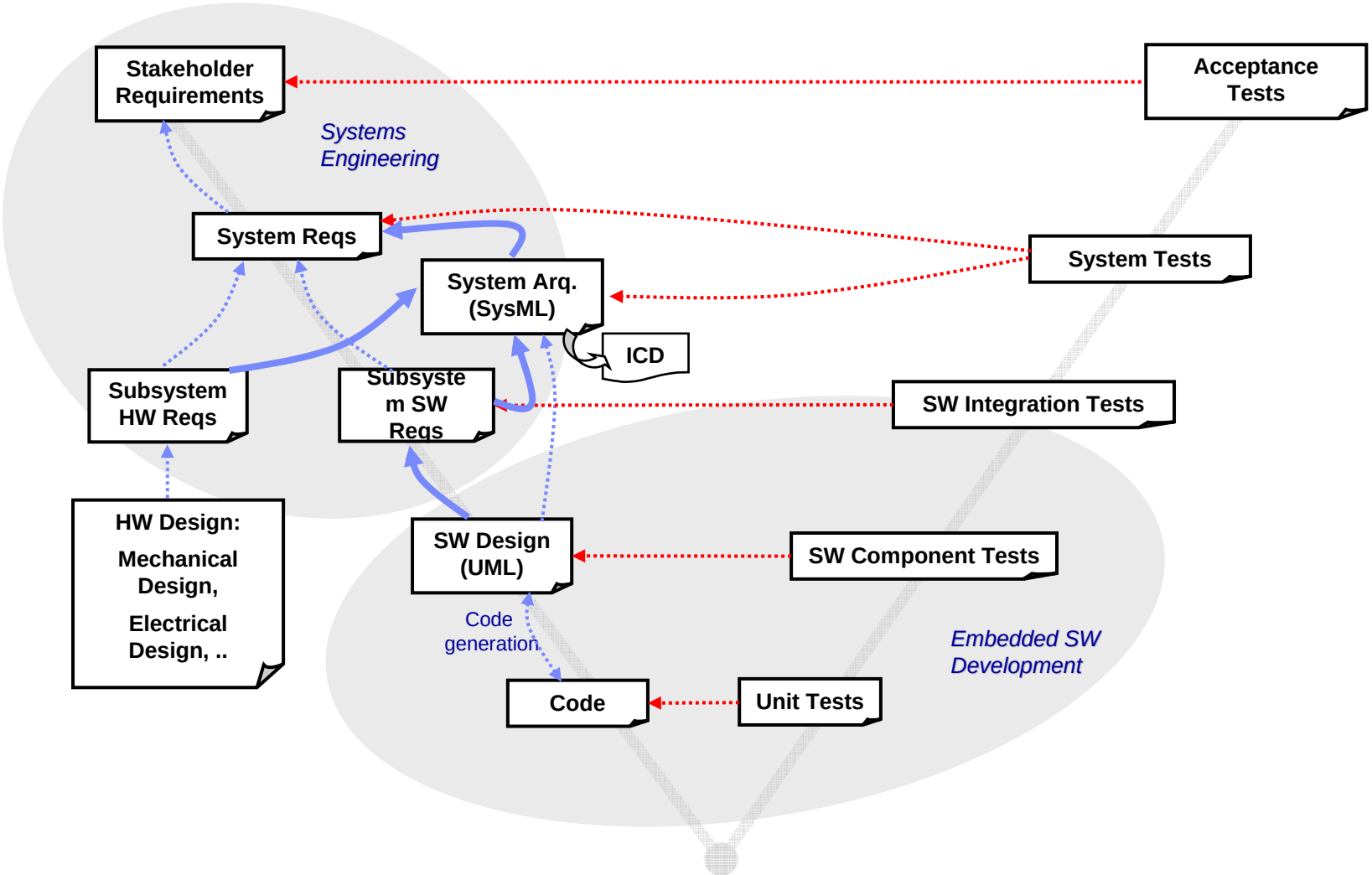


"Working with Rational Rhapsody, Ikerlan-IK4 reports that it is reducing the cost of development by as much as 25 percent and also estimates that they are able to reduce development time by a factor of 10 for each variation in their product line"

Ikerland-IK4, organization specialized in mechanical and electronic energy technologies

- Detección y eliminación temprana de errores en diseño mediante la simulación del sistema (verificación temprana)
- Incrementa la productividad del desarrollador mediante la generación automática de código y de la documentación
- Mejora la comunicación con el cliente y entre disciplinas mediante la utilización de UML/SysML
- Visualización de los requisitos y trazabilidad a elementos del modelo
- Mejora la colaboración entre equipos de desarrollo distribuidos
- Incrementa la productividad del equipo de pruebas: Pruebas dirigidas por el modelo *(Model Driven Testing – MDT)*

Rhapsody en el desarrollo de sistemas

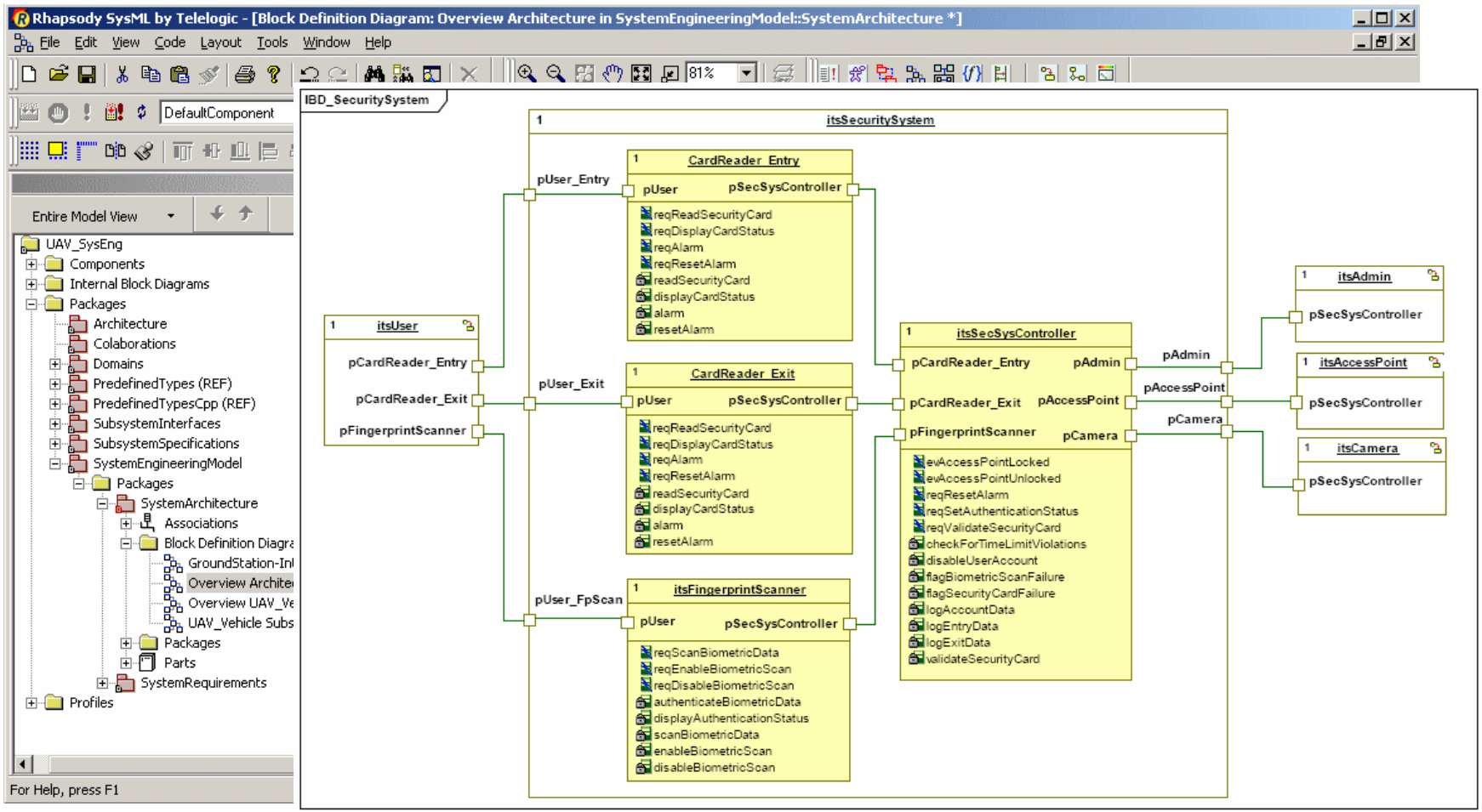


Arquitectura del sistema con Rhapsody

- Systems Modeling Language (SysML):
  - Extensión de UML 2.0 para dar soporte a las actividades y artefactos de Ingeniería de Sistemas
- SysML permite representar los requisitos como parte del modelo
  - Visualización de los requisitos en contexto con el modelo
  - Trazabilidad a requisitos: construcción del sistema correcto, impacto de cambios, mejor comunicación con el cliente y con otras disciplinas, cumplimiento de normativas
- SysML permite diseñar la arquitectura completa del sistema, modelando componentes de cualquier naturaleza (hardware, software, firmware..) y cómo se comunican (interfaces)
  - Mejora la comunicación entre disciplinas
- Simulación de modelos SysML
  - Verificación temprana: Detección de errores en el diseño en la fase de diseño de la arquitectura del sistema

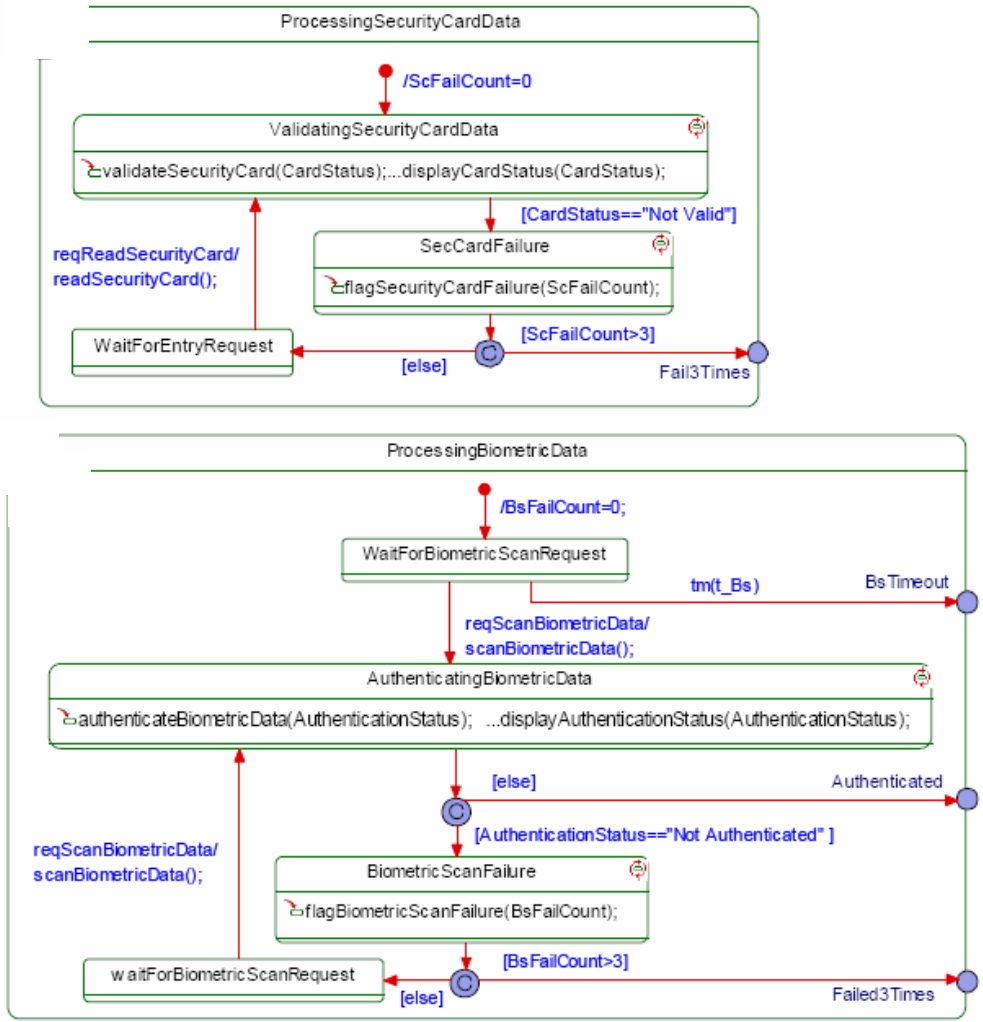
Diseño de la arquitectura del sistema

- Estructura: Block Definition Diagrams e Internal Block Diagrams



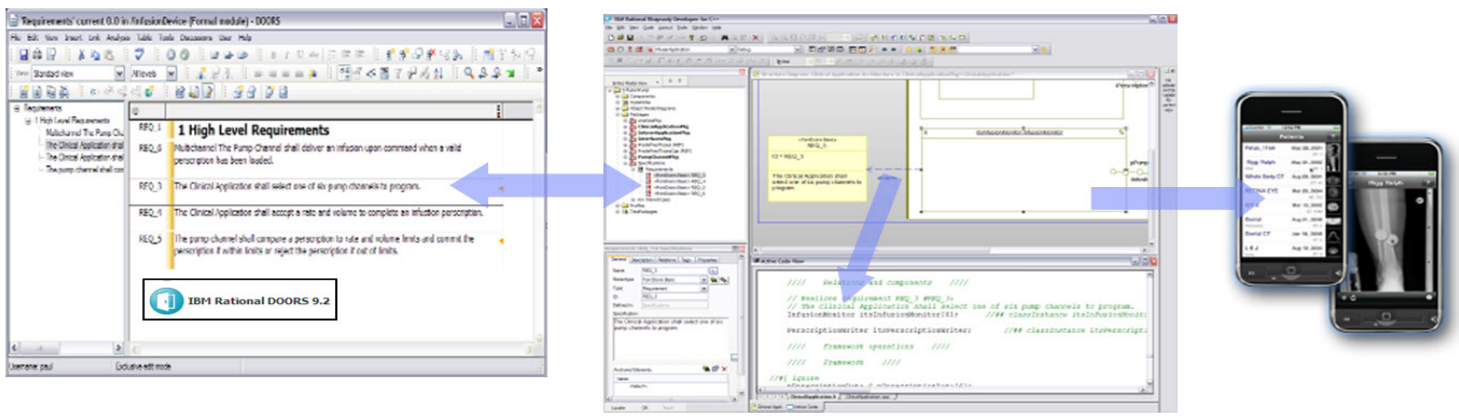
Diseño de la arquitectura del sistema

- Comportamiento:
  - Use Case Diagrams
  - Activity Diagrams
  - Sequence Diagrams
  - Statechart Diagrams



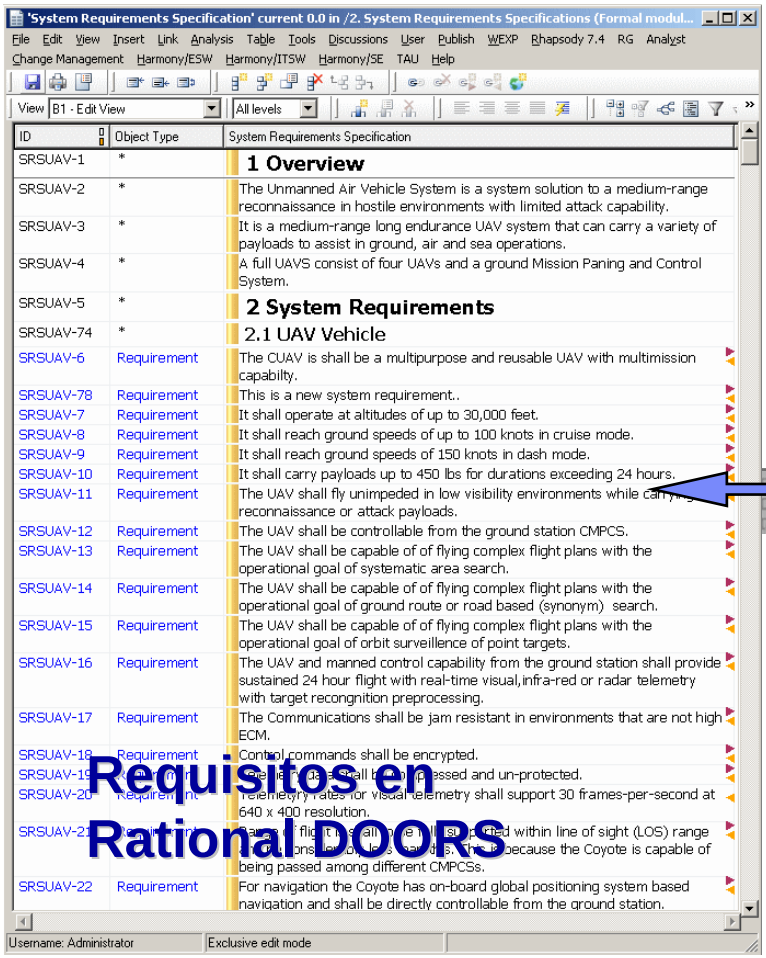
Los requisitos como parte del modelo

- Crear el diseño de la arquitectura del sistema a partir de los requisitos importados
  - Integración bi-direccional con Rational DOORS y RequisitePro
  - Trazabilidad diseño requisitos
  - Visualización de los requisitos a los que responden las funcionalidades implementadas
  - Generación automática de informes de trazabilidad: conformidad con normativas
  - Facilita actividades de análisis:
    - Identificar requisitos no implementados por ningún componente
    - Identificar artefactos de diseño que no responden a ningún requisito



Trazabilidad de los requisitos de sistema al modelo

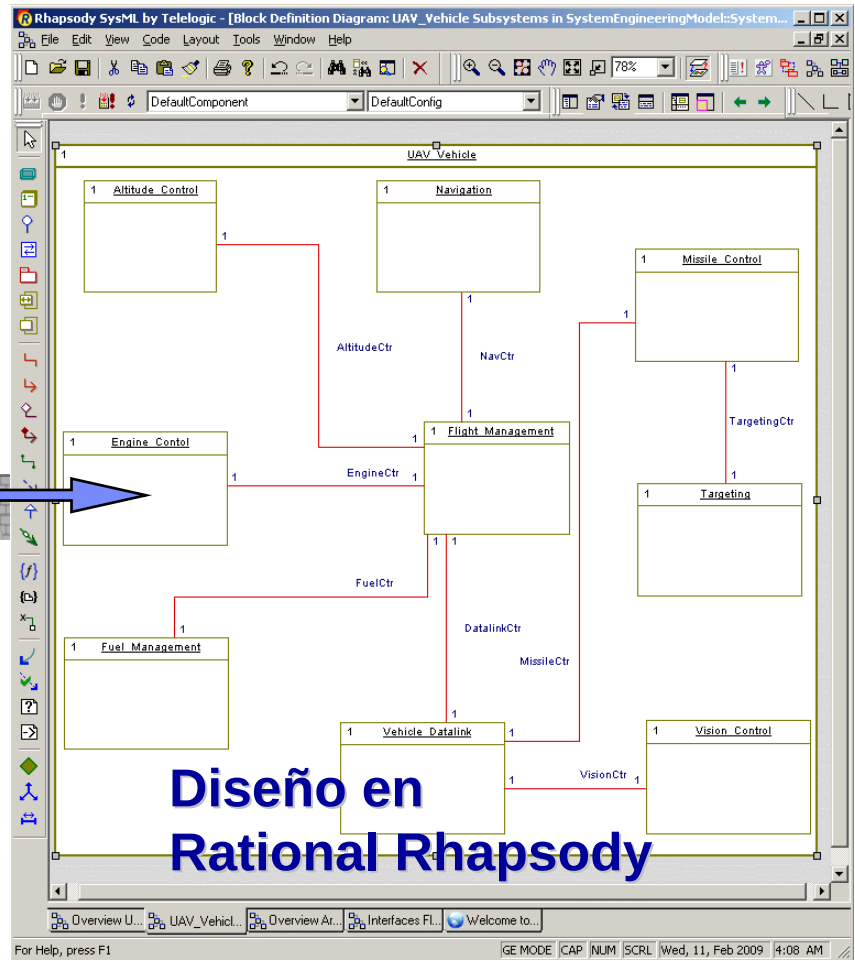
Asegurar que la arquitectura cumple los requisitos del sistema…



Requisitos en Rational DOORS

¿Cómo?

Rhapsody Gateway

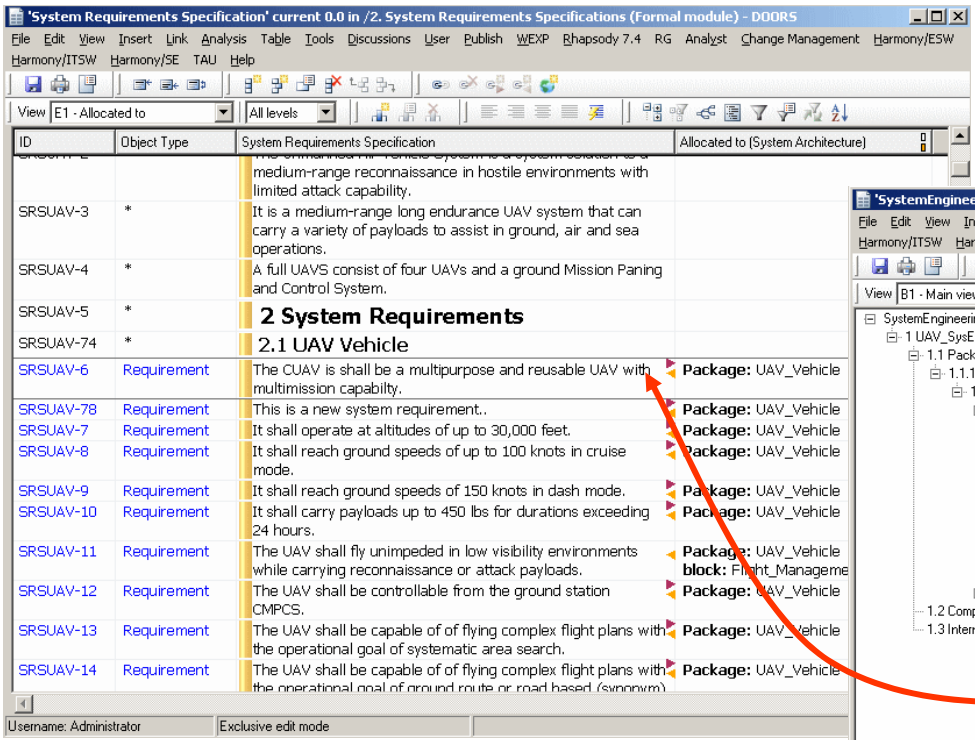


Diseño en Rational Rhapsody

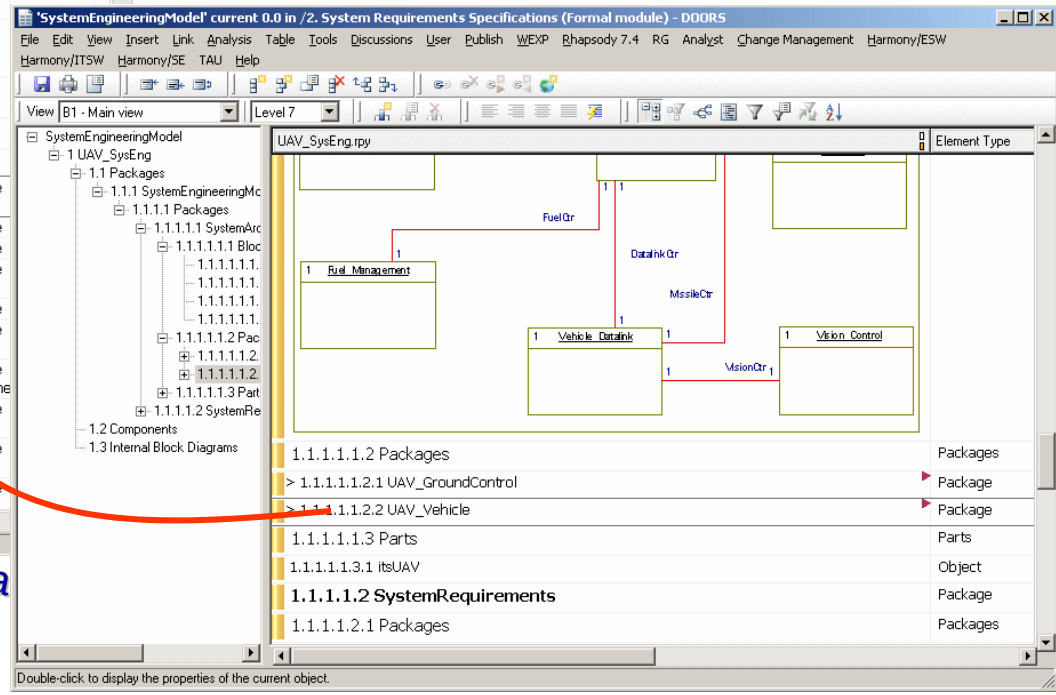
Trazabilidad de los requisitos de sistema al modelo

Asegurar que la arquitectura cumple los requisitos del sistema…

- Las dependencias pueden establecerse desde DOORS ….



Módulo DOORS con una copia de la información del modelo



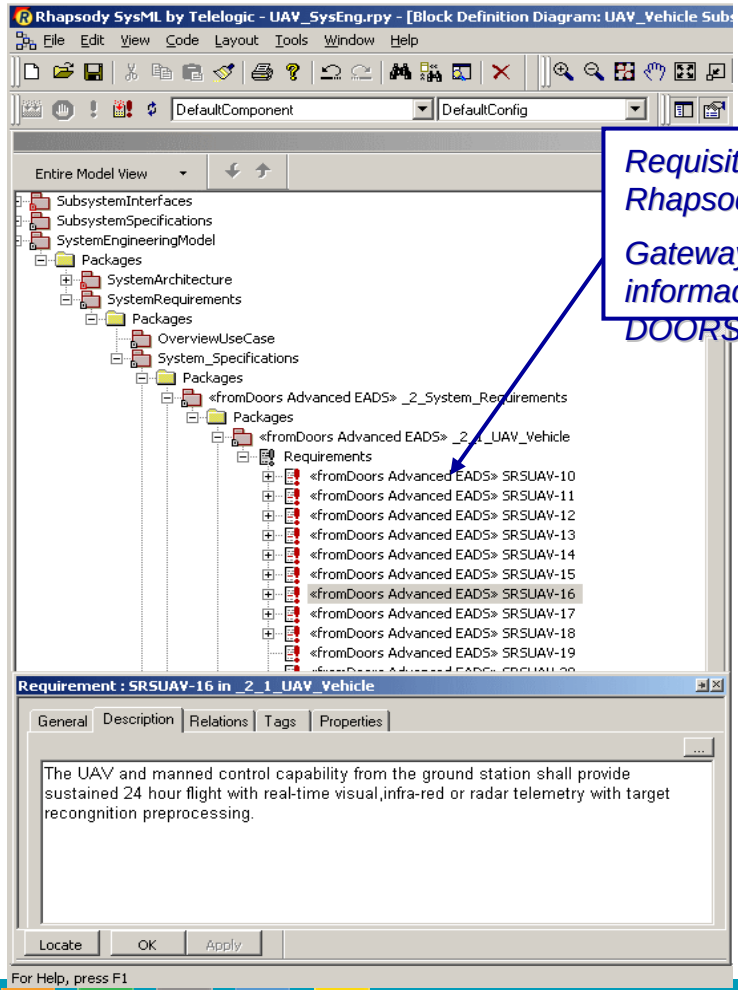
Crear relación desde el modelo a los requisitos

Gateway mantiene actualizada la información del modelo, incluyendo relaciones con reqs

Trazabilidad de los requisitos de sistema al modelo

Asegurar que la arquitectura cumple los requisitos del sistema…

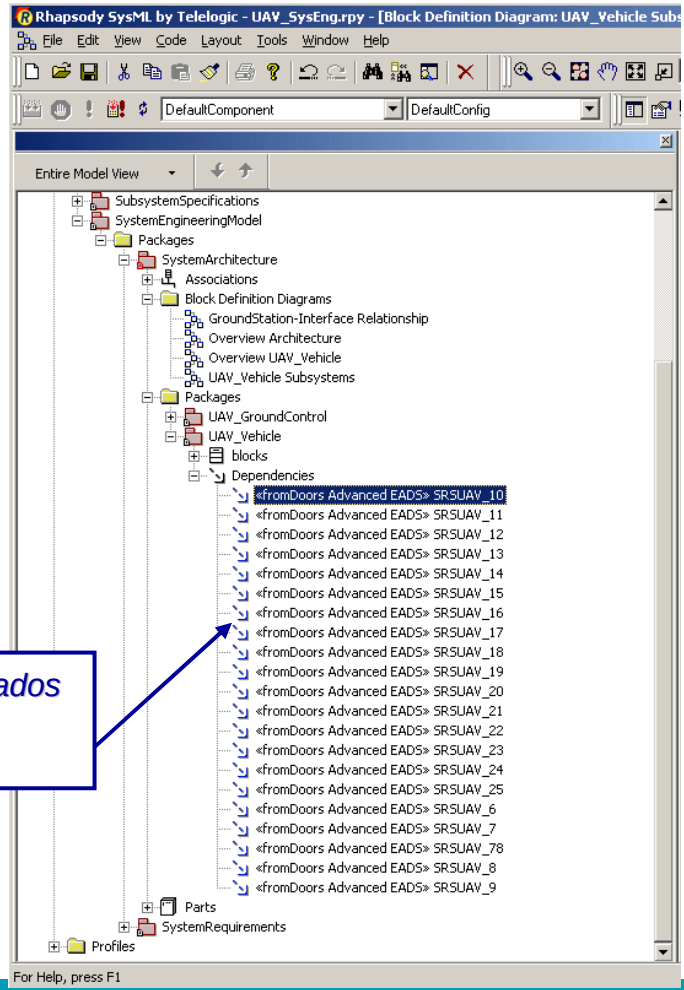
- …o desde Rhapsody



Requisitos en Rhapsody.

Gateway sincroniza la información desde DOORS

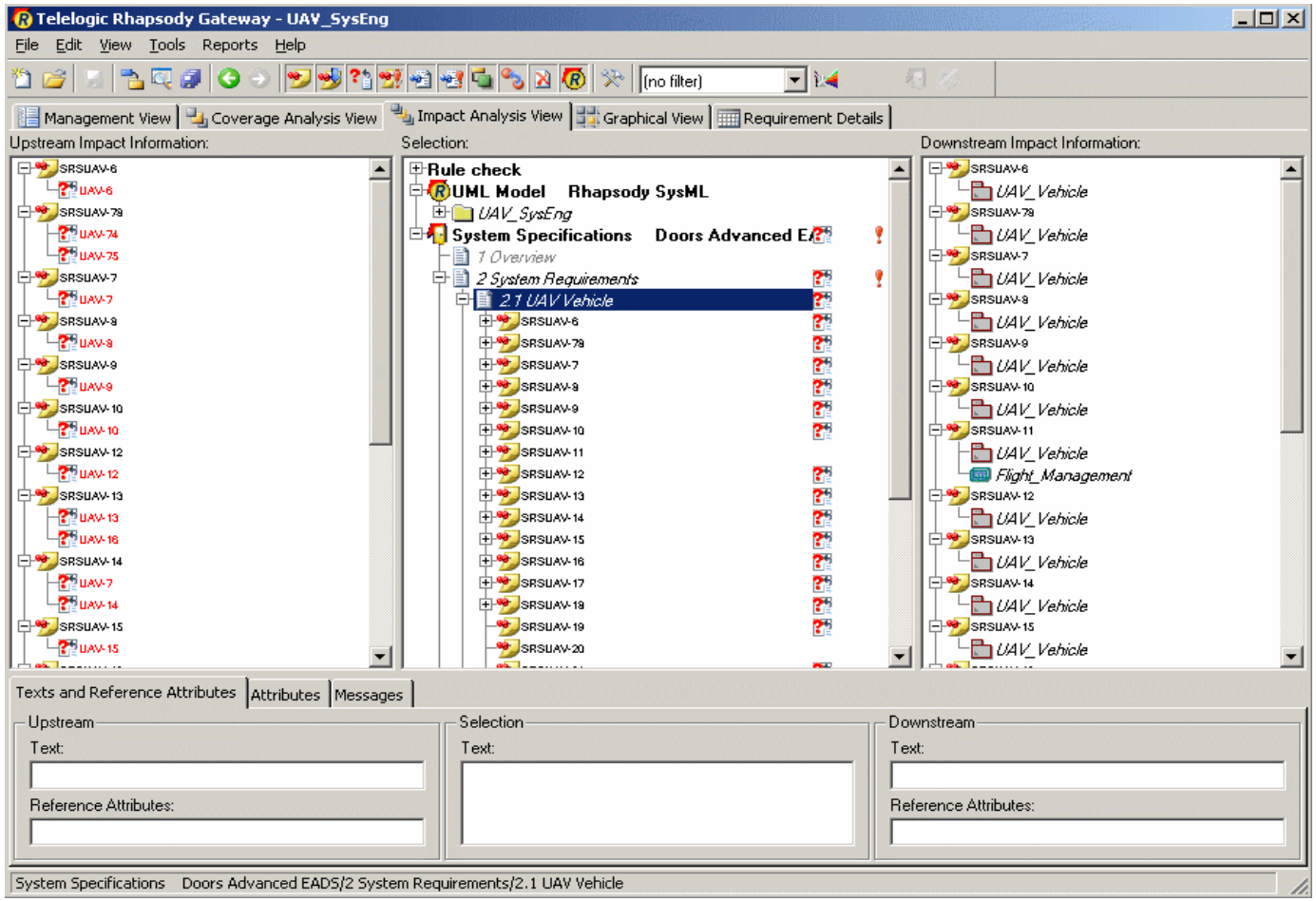
Requisitos relacionados con el paquete "UAV_Vehicle".



Trazabilidad de los requisitos de sistema al modelo

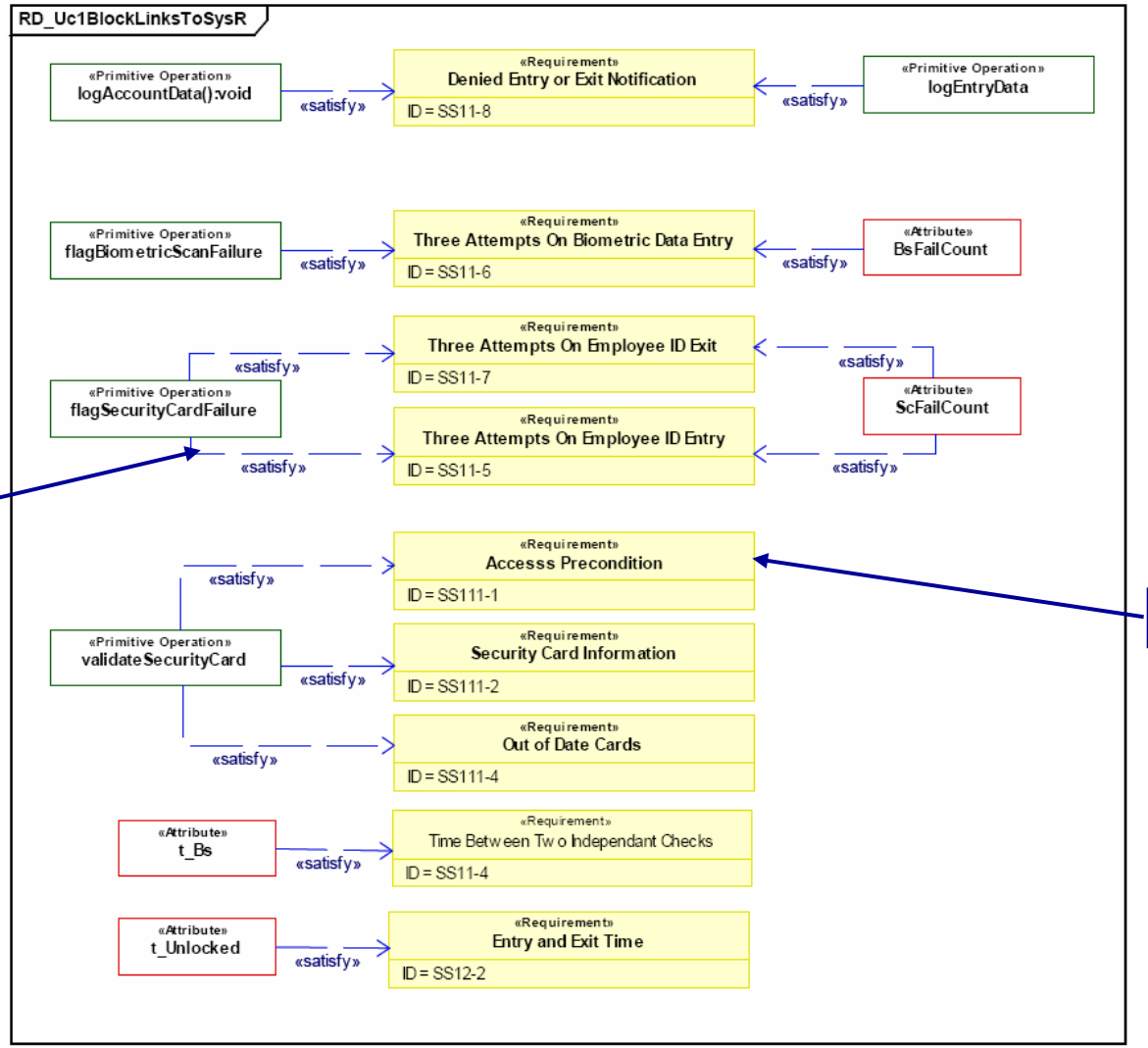
Asegurar que la arquitectura cumple los requisitos del sistema…

- Impacto y cobertura desde Gateway



Visualización de los requisitos y su trazabilidad

Diagramas de Requisitos

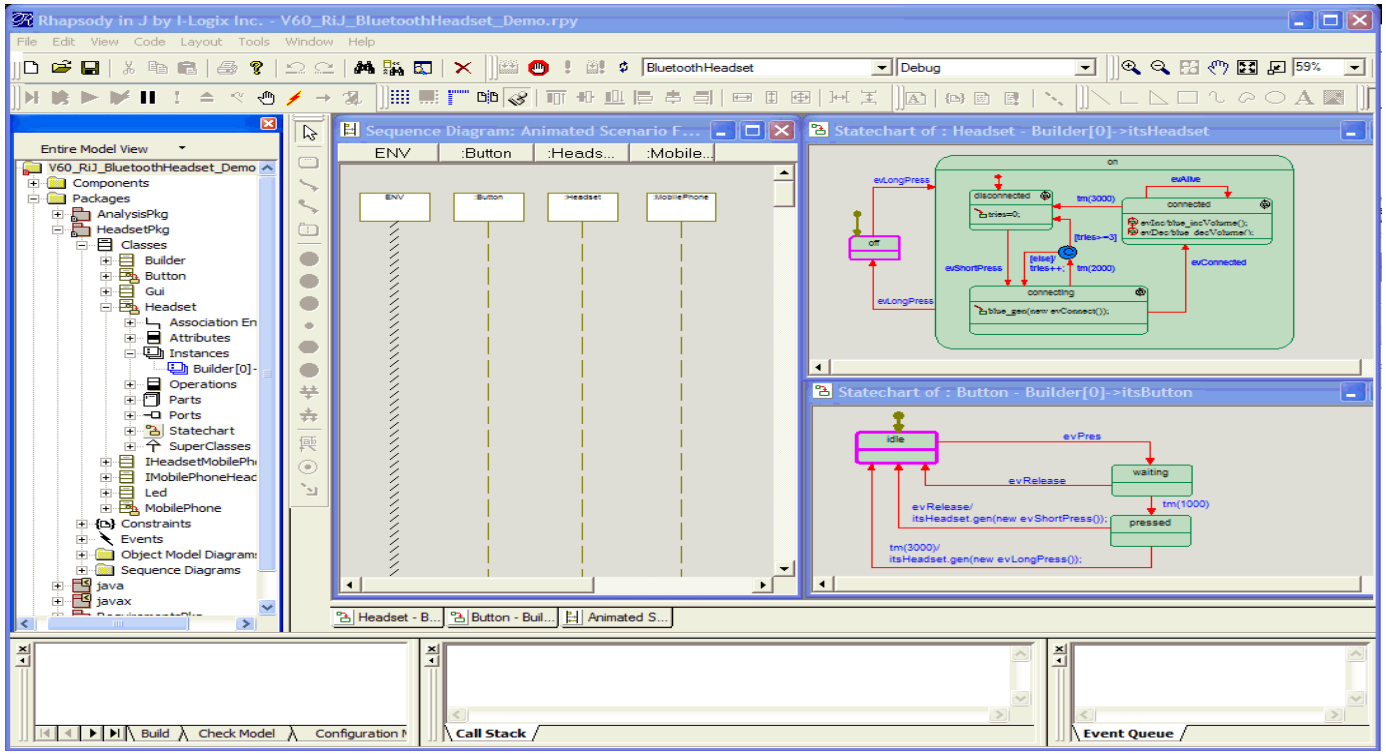


Trazabilidad, relación de satisfacción

Requisito

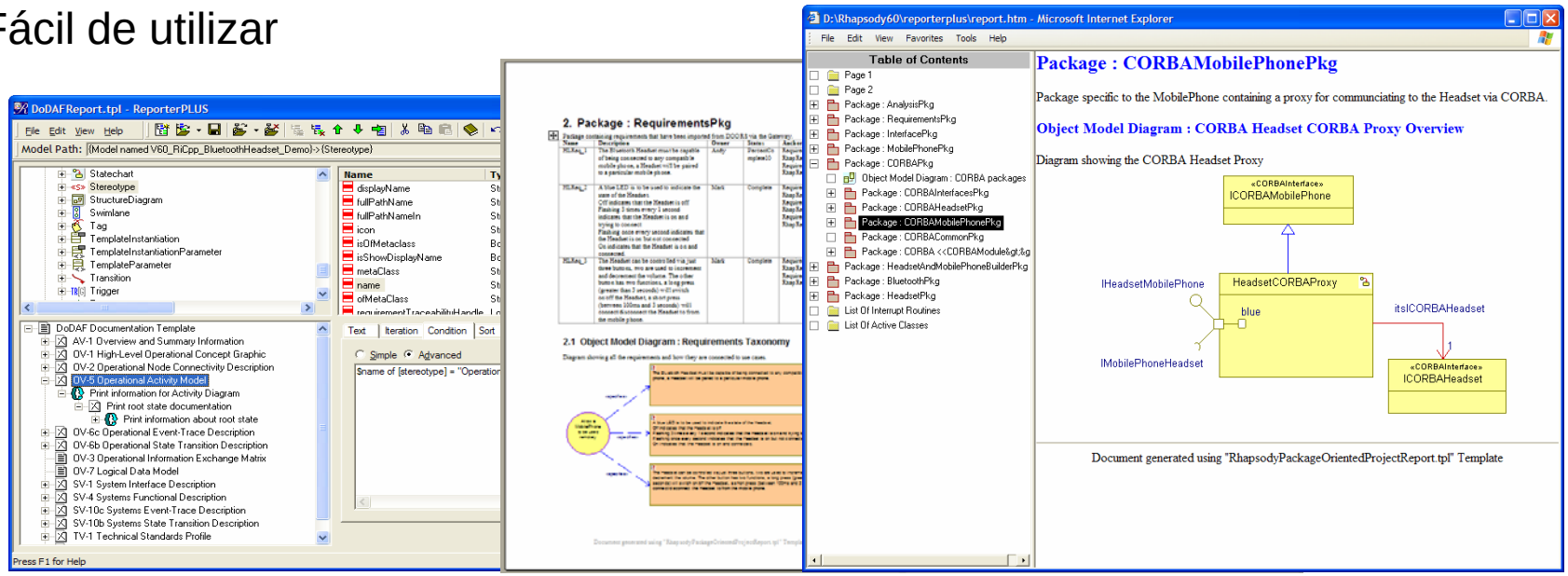
Verificación y validación temprana

- La ejecución del modelo ayuda a detectar errores en las fases tempranas, permite analizar si el comportamiento del sistema es correcto
- Visualización de la ejecución: permite mostrar y comunicar visualmente al cliente el comportamiento del sistema

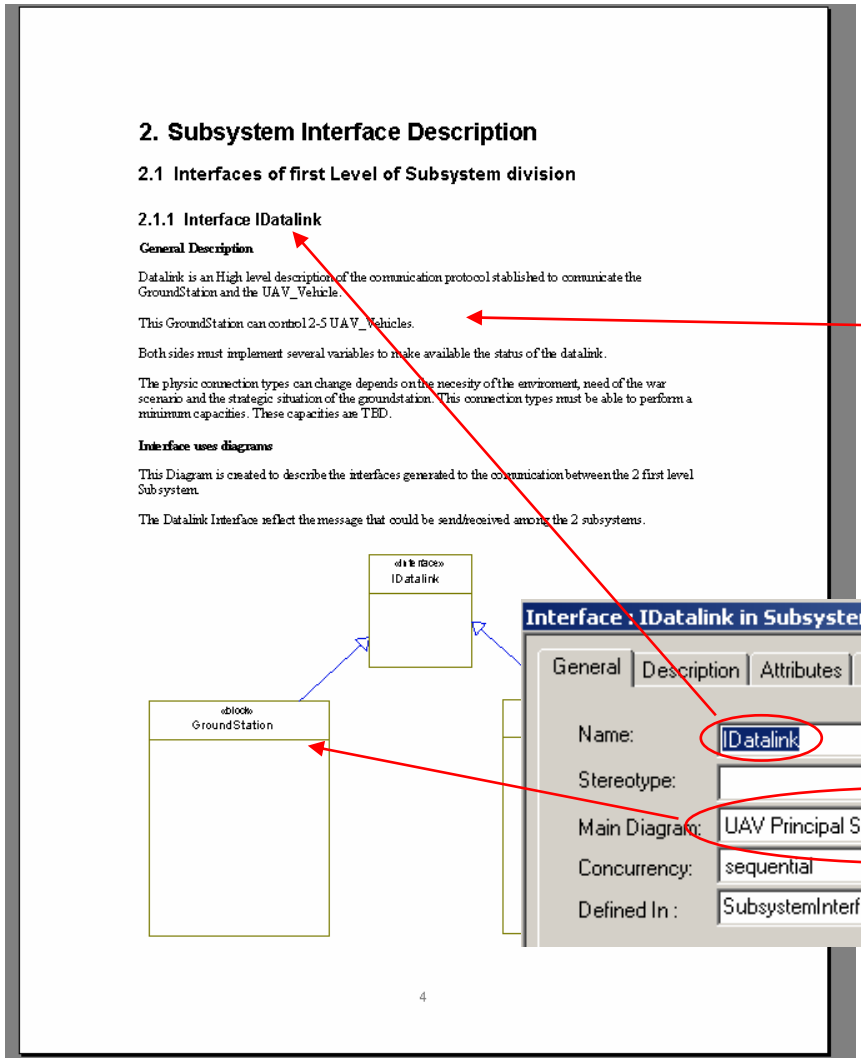


Generación automática de documentación desde el modelo

- Informes personalizables
- Diferentes formatos: HTML, Microsoft® Word®, PowerPoint®, Rich Text Format
  - Crea enlaces para una navegación rápida del informe
- Amplia variedad de plantillas predefinidas
- Fácil de utilizar



Generación de documentación: diagramas, descripción textual..



2. Subsystem Interface Description

2.1 Interfaces of first Level of Subsystem division

2.1.1 Interface IDatalink

General Description

Datalink is an High level description of the comunication protocol stablished to comunicate the GroundStation and the UAV_Vehicle.

This GroundStation can control 2-5 UAV_Vehicles.

Both sides must implement several variables to make available the status of the datalink.

The physic connection types can change depends on the necesity of the enviroment, need of the war scenario and the strategic situation of the groundstation. This connection types must be able to perform a minimum capacities. These capacities are TBD.

Interface uses diagrams

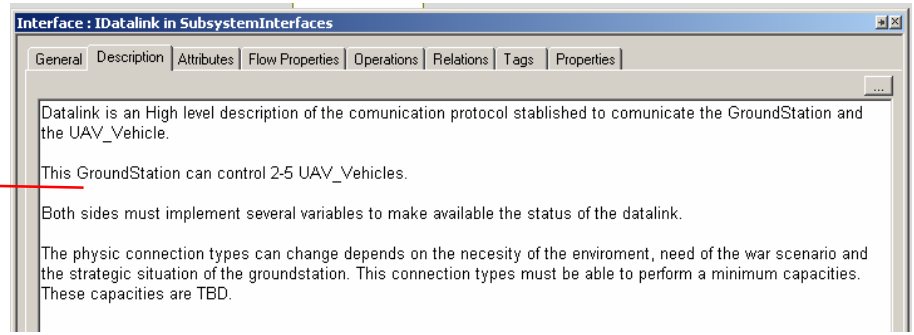
This Diagram is created to describe the interfaces generated to the comunication between the 2 first level Subsystem.

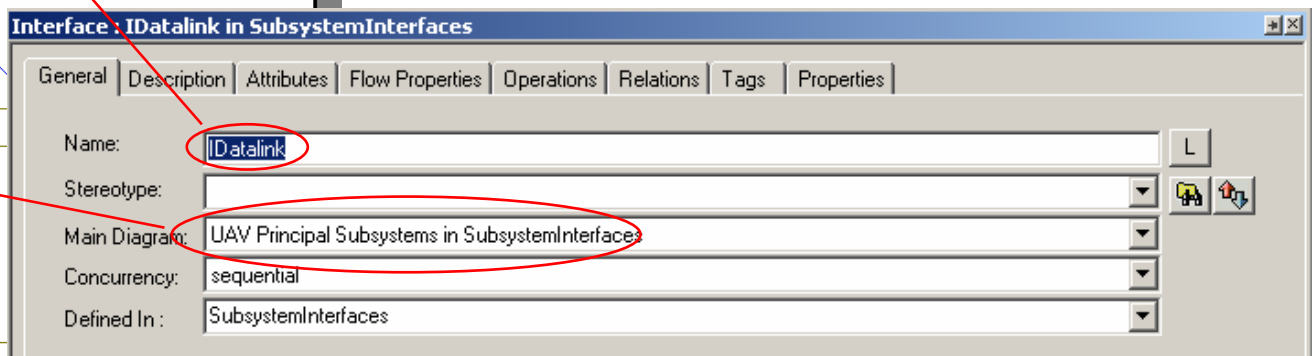
The Datalink Interface reflect the message that could be send/received among the 2 subsystems.

«interface» IDatalink

«block» GroundStation

4





Generación de documentación: mensajes en tablas, tipos de datos…

2.1.1.1 Attributes in the interface IDatalink

Name	Type	Range
LinkType	LinkType	See LinkType
LinkStatus	LinkStatusType	See LinkStatusType
LinkSpeed	int	9600..54000000
CongestionFlag	int	0..100

2.1.1.2 Messages in the Interface IDatalink

In this section will be described the messages that each subsystem must implement to communicate with the rest of the Subsystems that expect to use this interface to communicate.

2.1.1.2.1 Message:InitComms

Description:
Initialize the communications with the requested address.

Fields in this message

Name	Type	Range	Length
OrgAddress	int	0..2e32	32 bits
OrgPort	int	0..2e32	32 bits
DestAddress	int	0..2e32	32 bits
DestPort	int	0..2e32	32 bits
Type	LinkType	See LinkType	4 bits
DLOperation	DataLinkOpType	See DataLinkOpType	3 bits

2.1.1.2.2 Message:CloseComms

Description:
Finalize the communications with the requested address.
Fields in this message

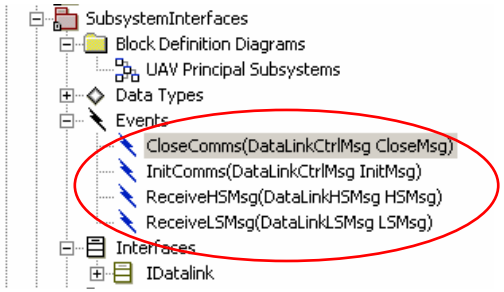
Name	Type	Range	Length
OrgAddress	int	0..2e32	32 bits
OrgPort	int	0..2e32	32 bits
DestAddress	int	0..2e32	32 bits
DestPort	int	0..2e32	32 bits
Type	LinkType	See LinkType	4 bits
DLOperation	DataLinkOpType	See DataLinkOpType	3 bits

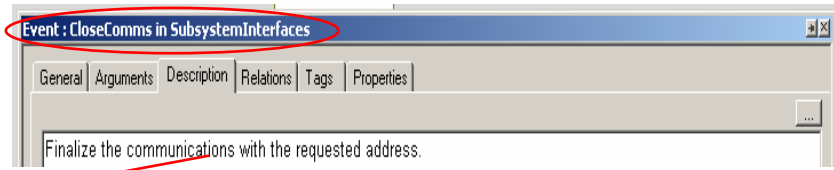
2.1.1.2.3 Message:ReceiveHSMsg

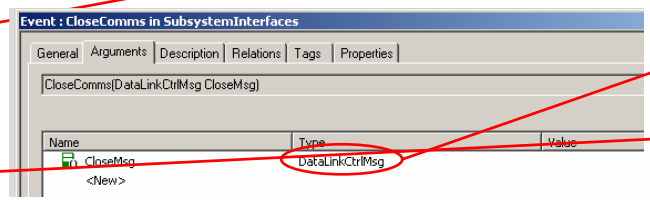
Description:
A Message of High Speed and High priority is sended from an origin address-port pair to a destination address-port pair. These pairs represent the 2 subsystems that is connected.
Fields in this message

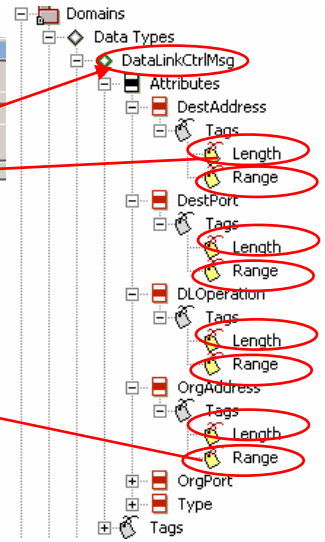
Name	Type	Range	Length
OrgAddress	int	0..2e32	32 bits
OrgPort	int	0..2e32	32 bits
DestAddress	int	0..2e32	32 bits
DestPort	int	0..2e32	32 bits

5



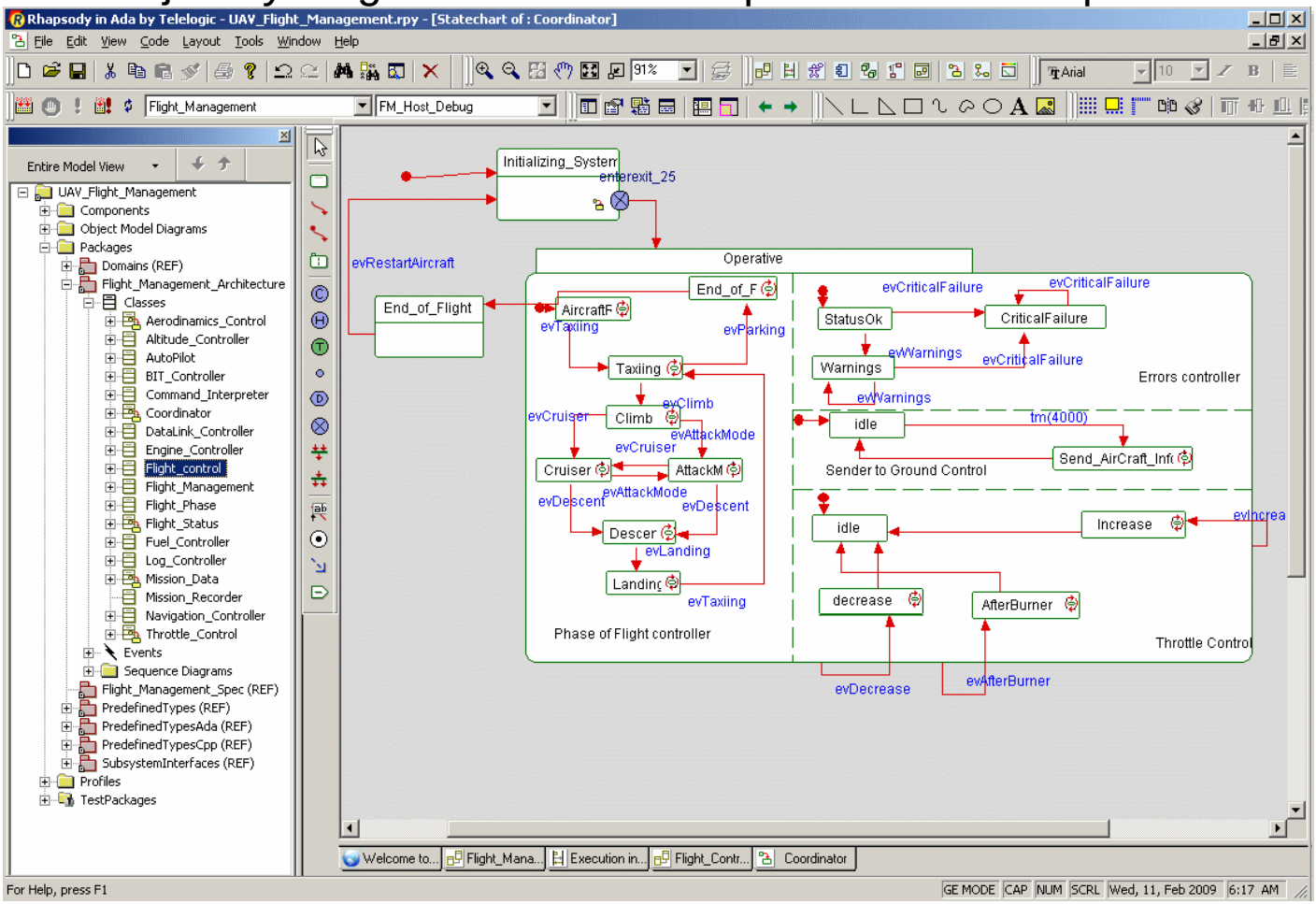






Diseño del software

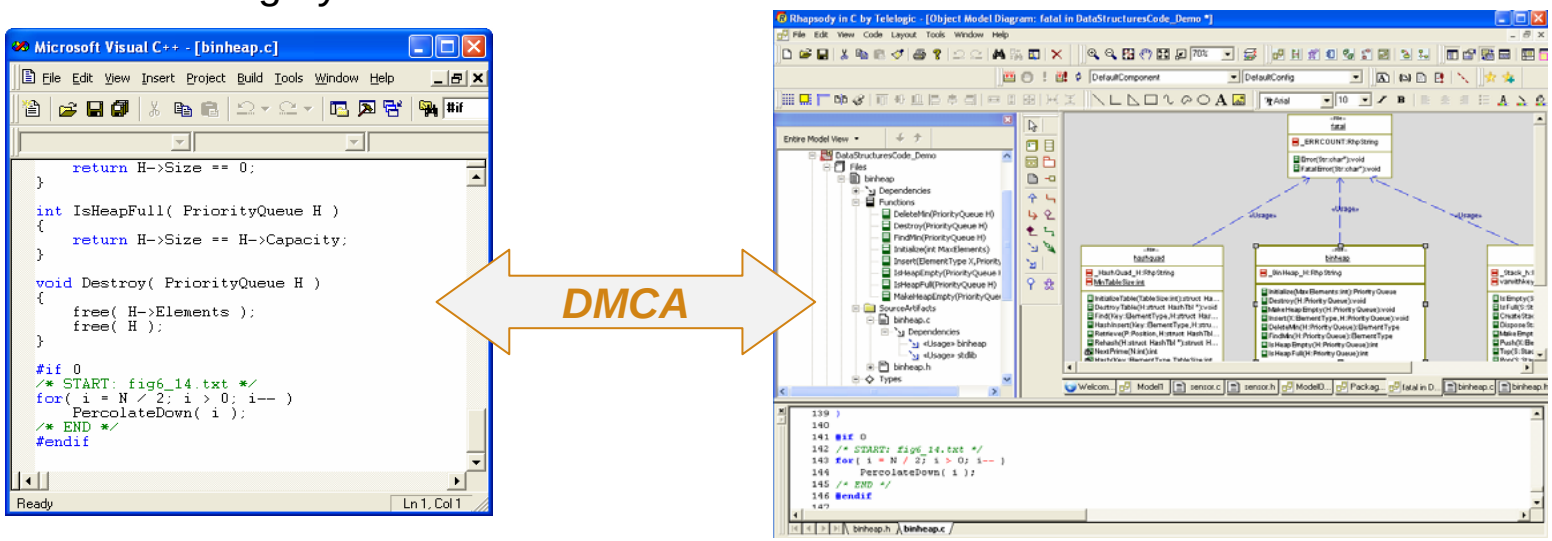
- Diagrama de objetosy diagrama de estados que define el comportamiento



Trabaje como usted quiera

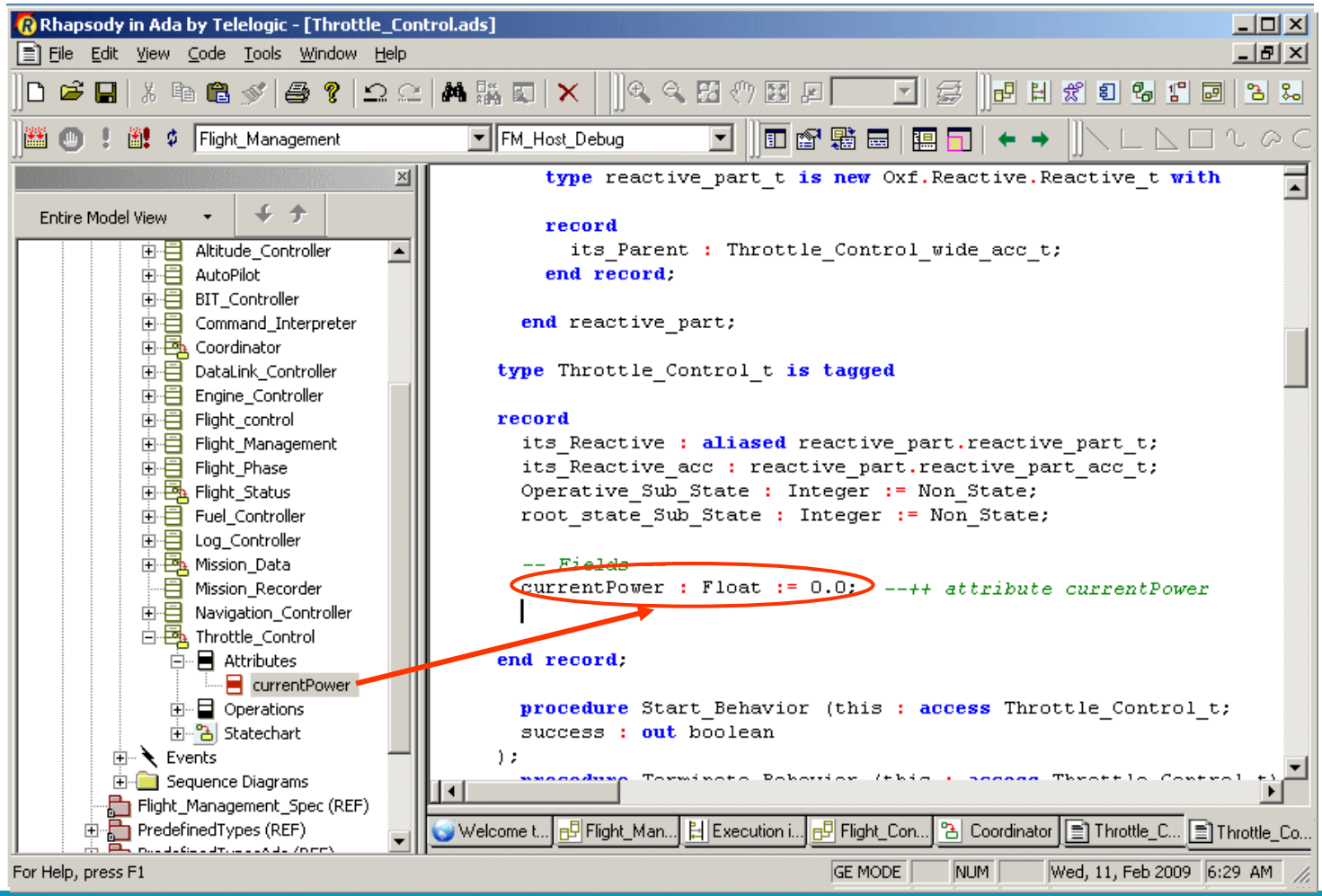
Dynamic Model Code Associativity - DMCA

- El código es otra vista del modelo
- A nivel del modelo o de código
  - Reduce la curva de aprendizaje
  - Incremento de la productividad
- Diseño y código se mantienen sincronizados: “dynamic model code associativity” (DMCA)
  - Cambie el modelo y el código cambiará automáticamente
  - Cambie el código y el modelo cambiará automáticamente



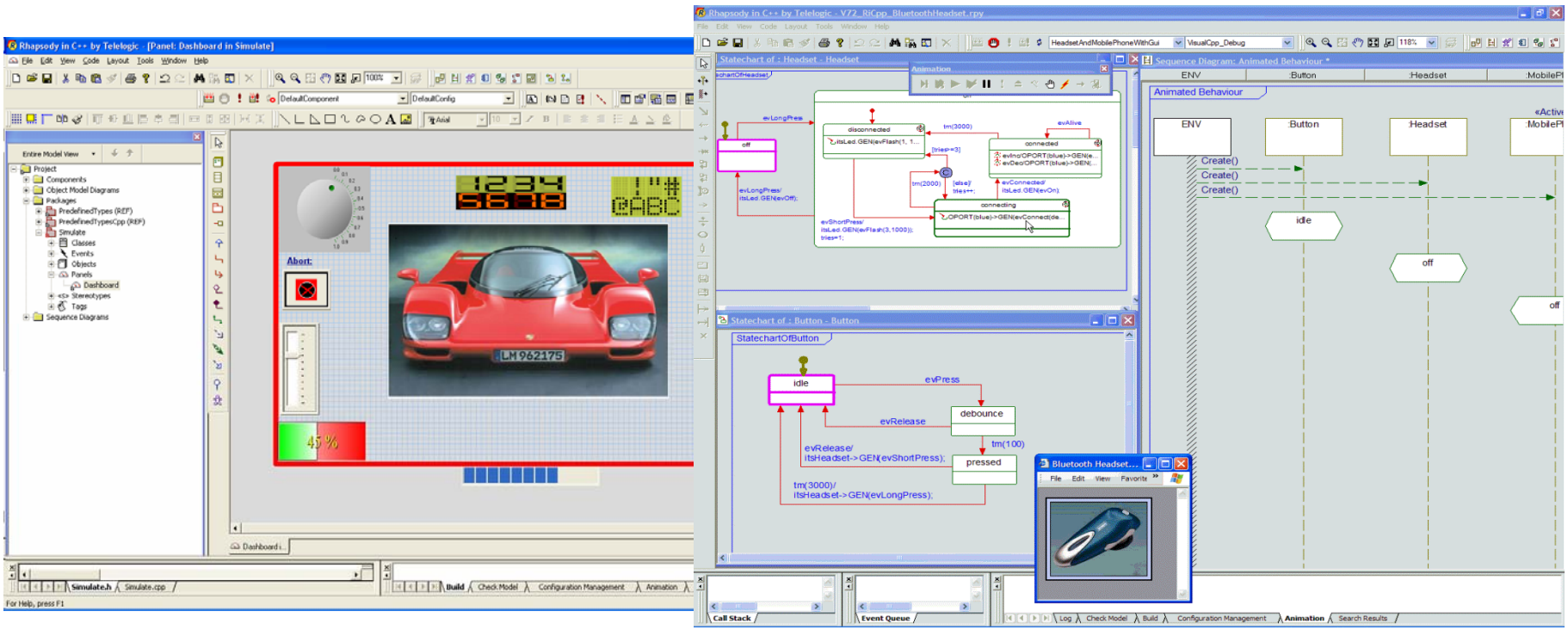
Si no quiere, no es necesario codificar…

- Añadir detalle al modelo a través de formularios



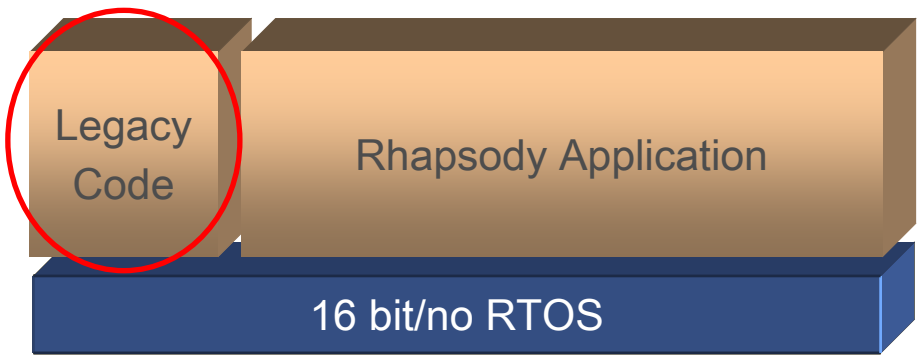
Paneles gráficos para la simulación y animación

- Creación de maquetas de la interfaz para comunicar eficazmente el comportamiento de diseño destinado a los clientes
- Modificar, monitorizar y analizar los valores de datos durante la simulación para ayudar a garantizar que el diseño es correcto en una fase temprana del proceso



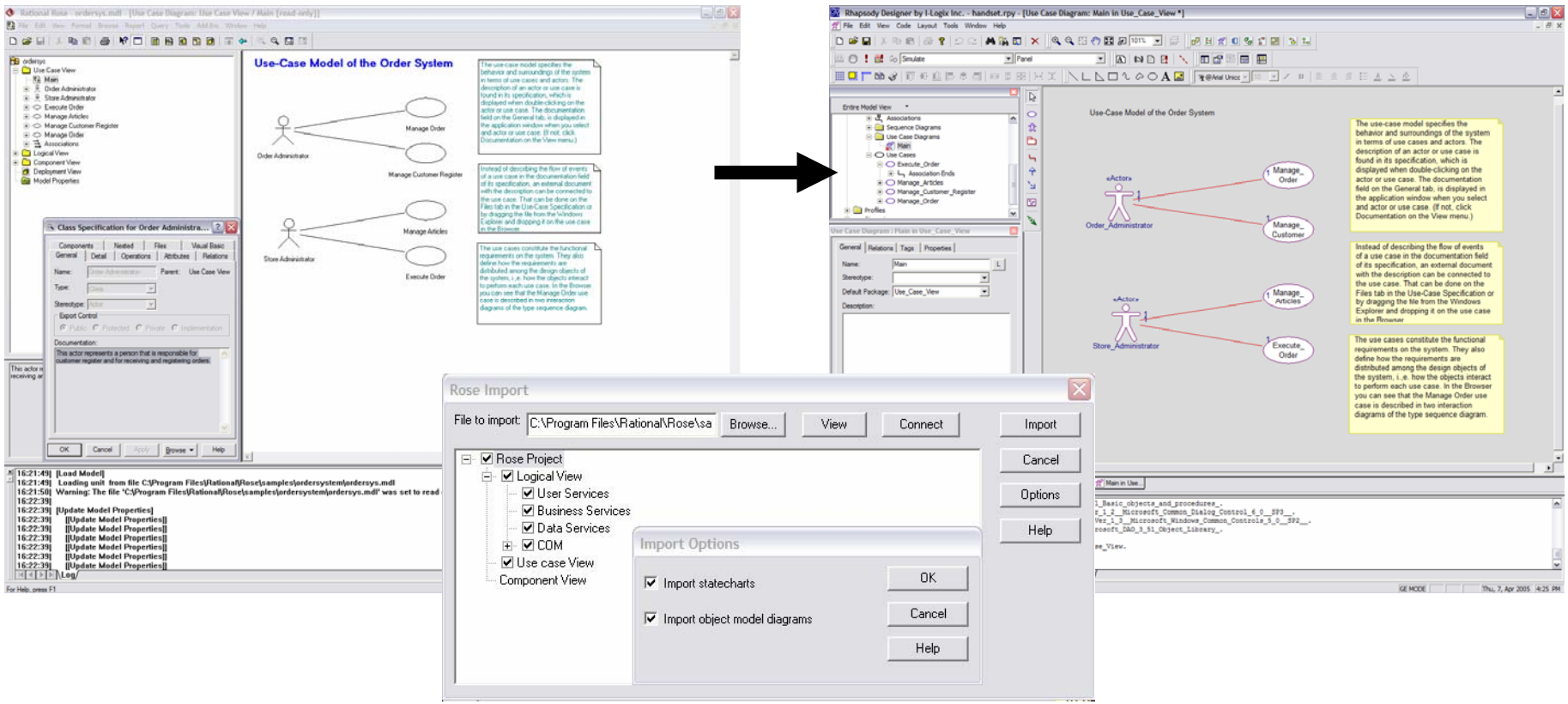
Generación automática de código desde el modelo

- Generar código completo de aplicaciones C, C++, Java y Ada
- Desplegar rápidamente su diseño en cualquier plataforma de destino.
- Reutilizar código externo:
  - Referenciado: visualización de código externo
  - Incluido automáticamente en el modelo mediante ingeniería inversa



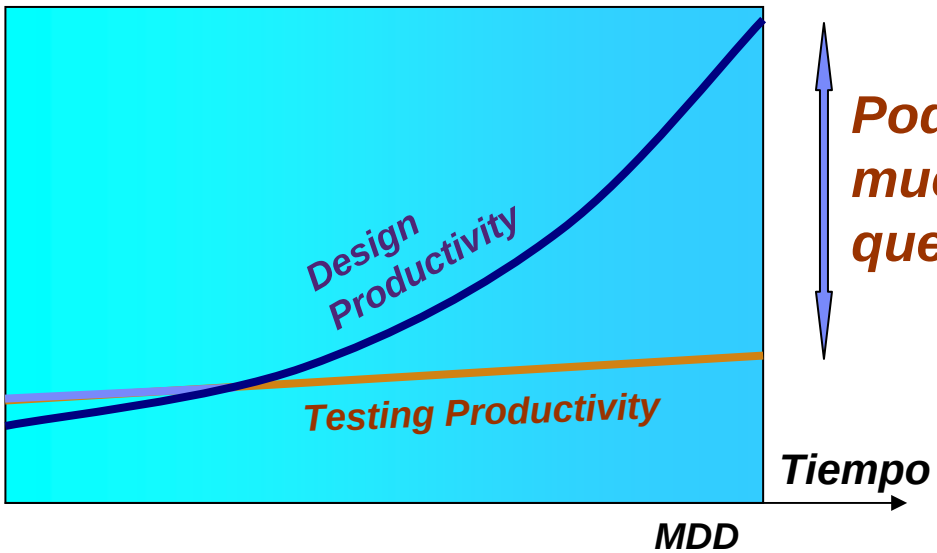
Reutilización de modelos ya existentes

- Importación y reutilización de modelos existentes en otras herramientas Rational o XMI-compliant



Proceso de pruebas poco productivo…

- Está comprobado que la adopción de MDD mejora la productividad de los desarrolladores
- El equipo de pruebas sigue siendo mucho menos productivo que el de desarrollo: pruebas a nivel de código!!!



- ¿Por qué no transladar la idea de MDD a Testing?

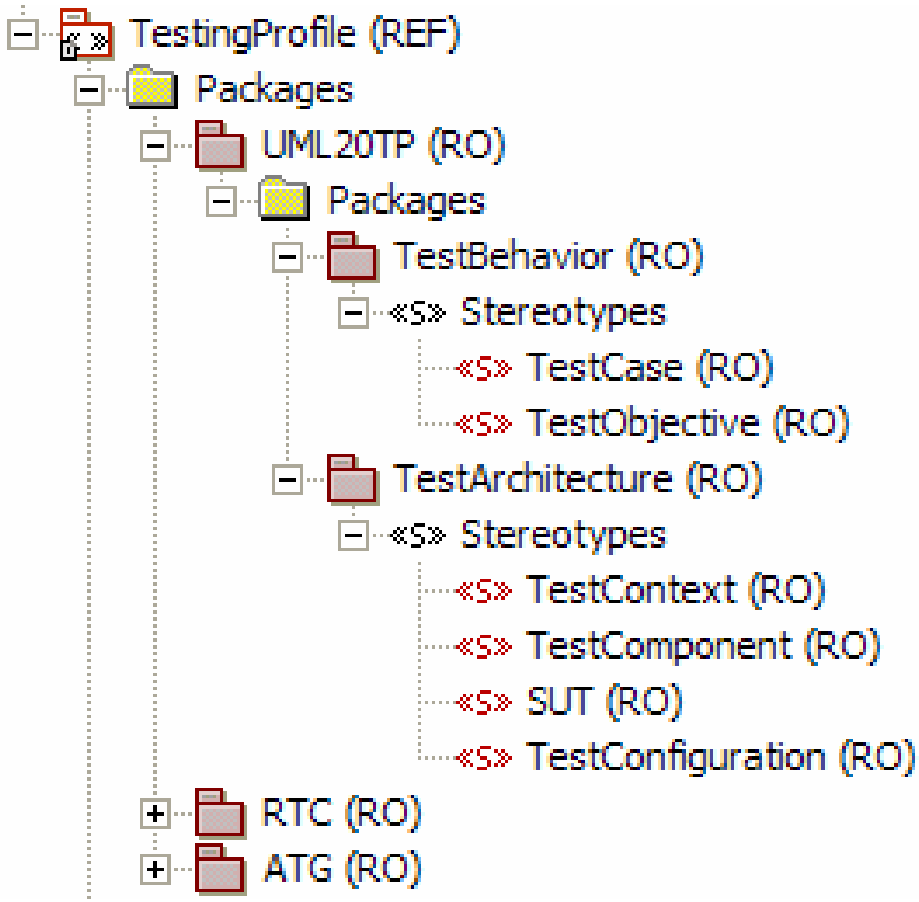
Model Driven Testing

Pruebas dirigidas por modelos (Model Driven Testing - MDT)

- Wikipedia, la enciclopedia libre en la World Wide Web (www), se refiere a *model driven testing* como "*software testing where test cases are derived in whole or in part from a model that describes some (if not all) aspects of the system under test (SUT)*"
- UML es el lenguaje para modelado, mientras que los tests derivados del modelo se representan utilizando UML Testing Profile (UTP)
- UTP se puede considerar como un lenguaje para visualizar, especificar, analizar, construir y documentar todos los artefactos involucrados en un proceso de pruebas.

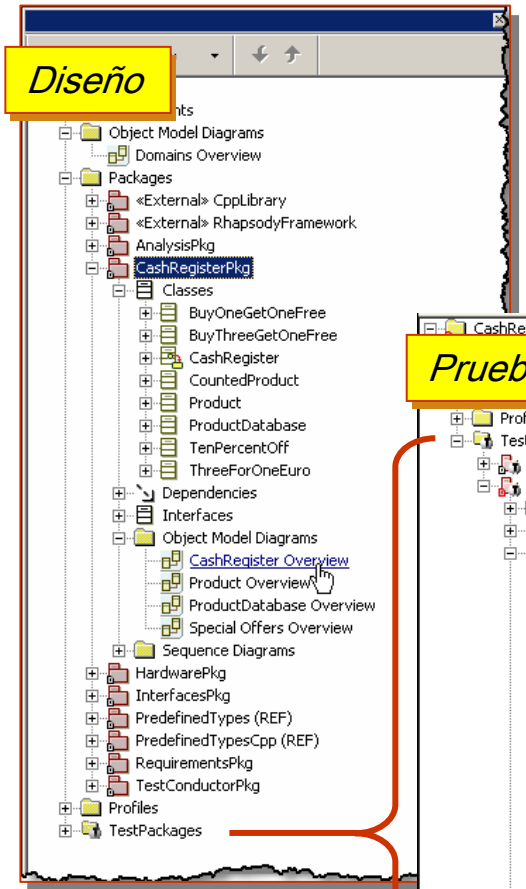
UML Testing Profile

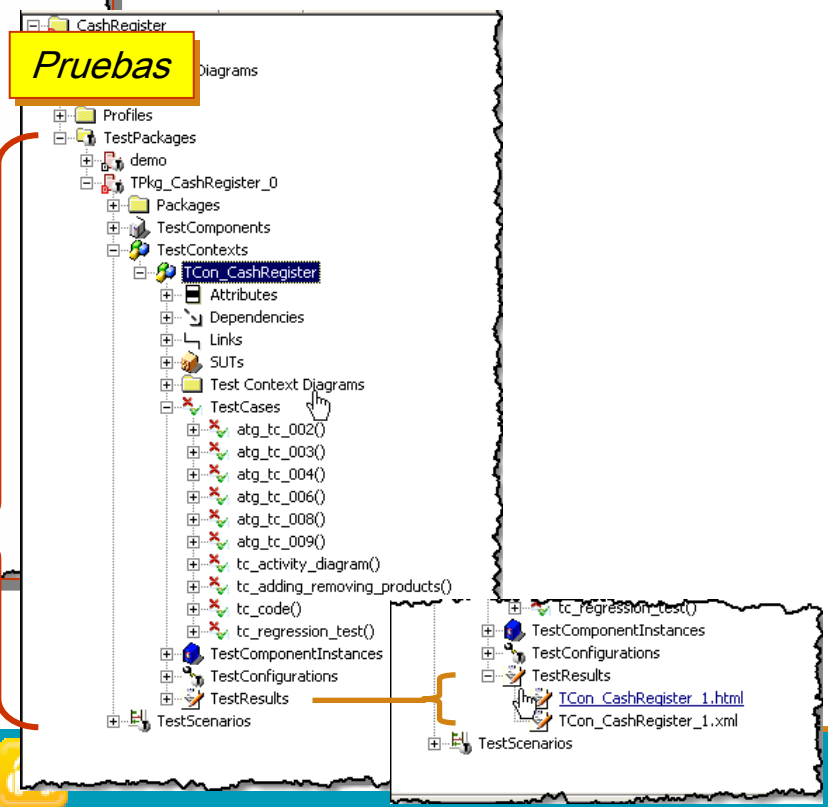
- El UML 2 Testing Profile extiende UML con conceptos específicos de pruebas:
  - **Test Behavior :** actividades a realizar durante la prueba y resultados
  - **Test Architecture :** elementos del modelo a probar y sus relaciones con otro elementos



Procesos de diseño y pruebas completamente integrados

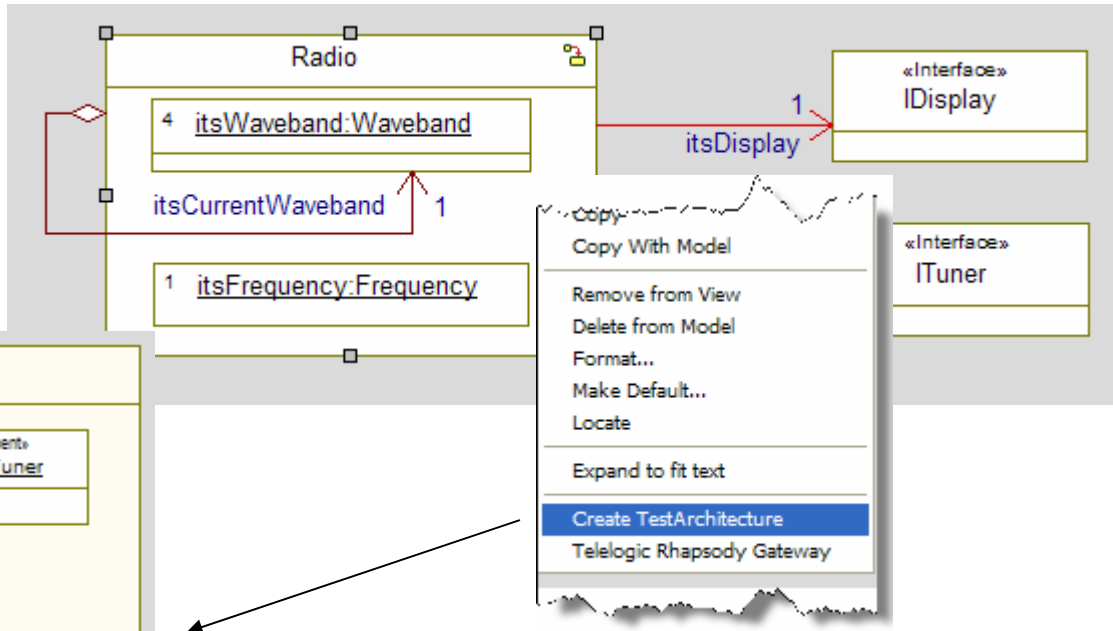
- Fácil navegación entre el diseño y los artefactos de prueba, son parte del mismo modelo
- Trazabilidad de requisitos a test cases
- Diseño y pruebas siempre sincronizados y actualizados
- Generación automática de informes de resultados

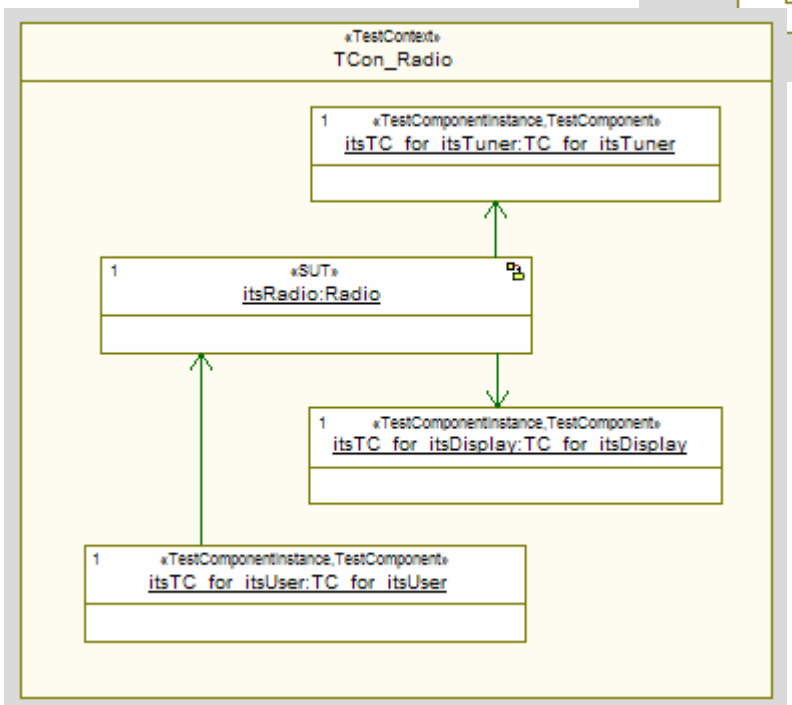




Proceso de pruebas: Crear arquitectura de pruebas

- Crear Test Architecture
  - Creación automática
  - Actualización automática, o
  - Actualización manual

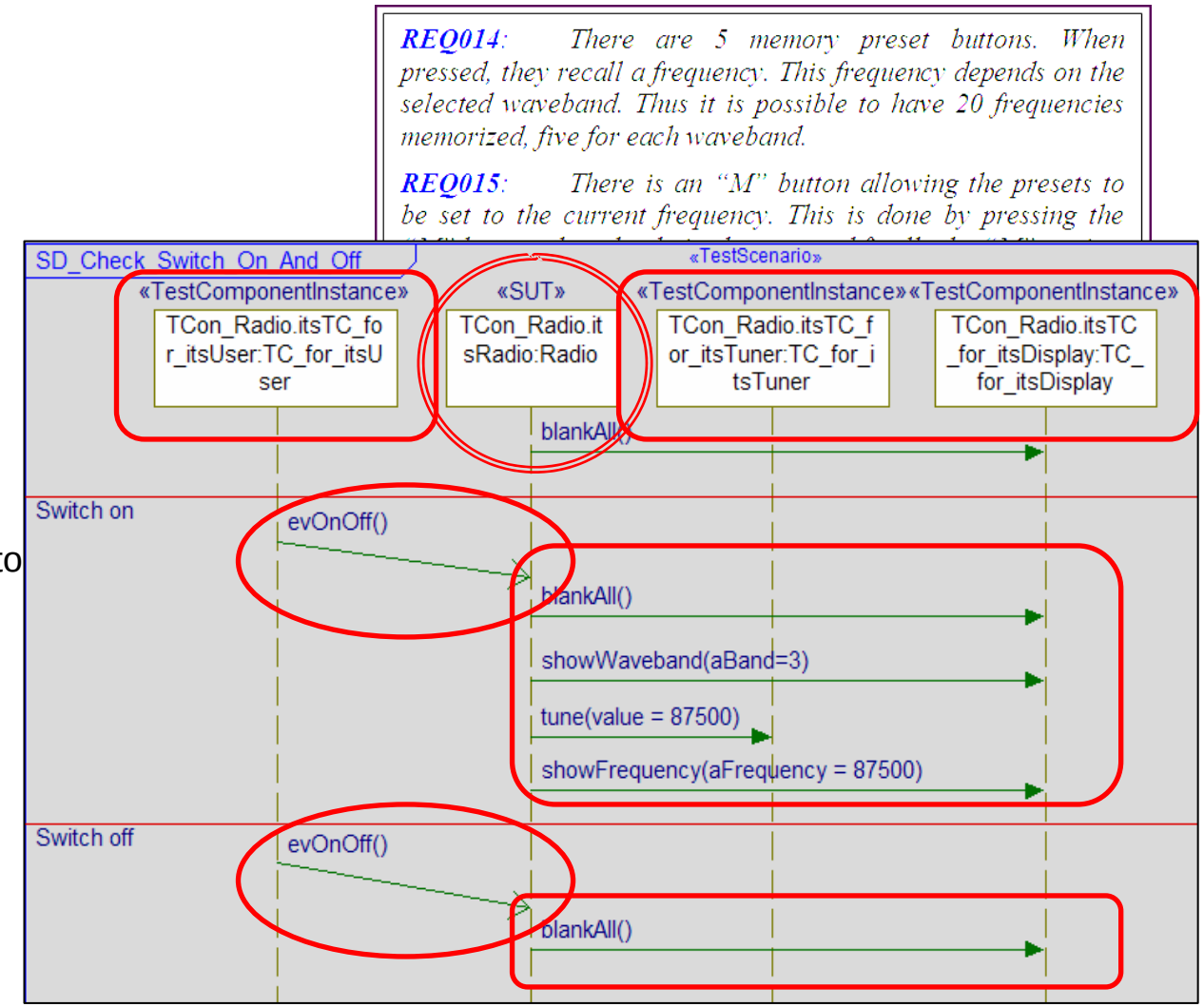




- Graphical Test Architecture
  - Es sólo arquitectura, todavía no hay casos de prueba

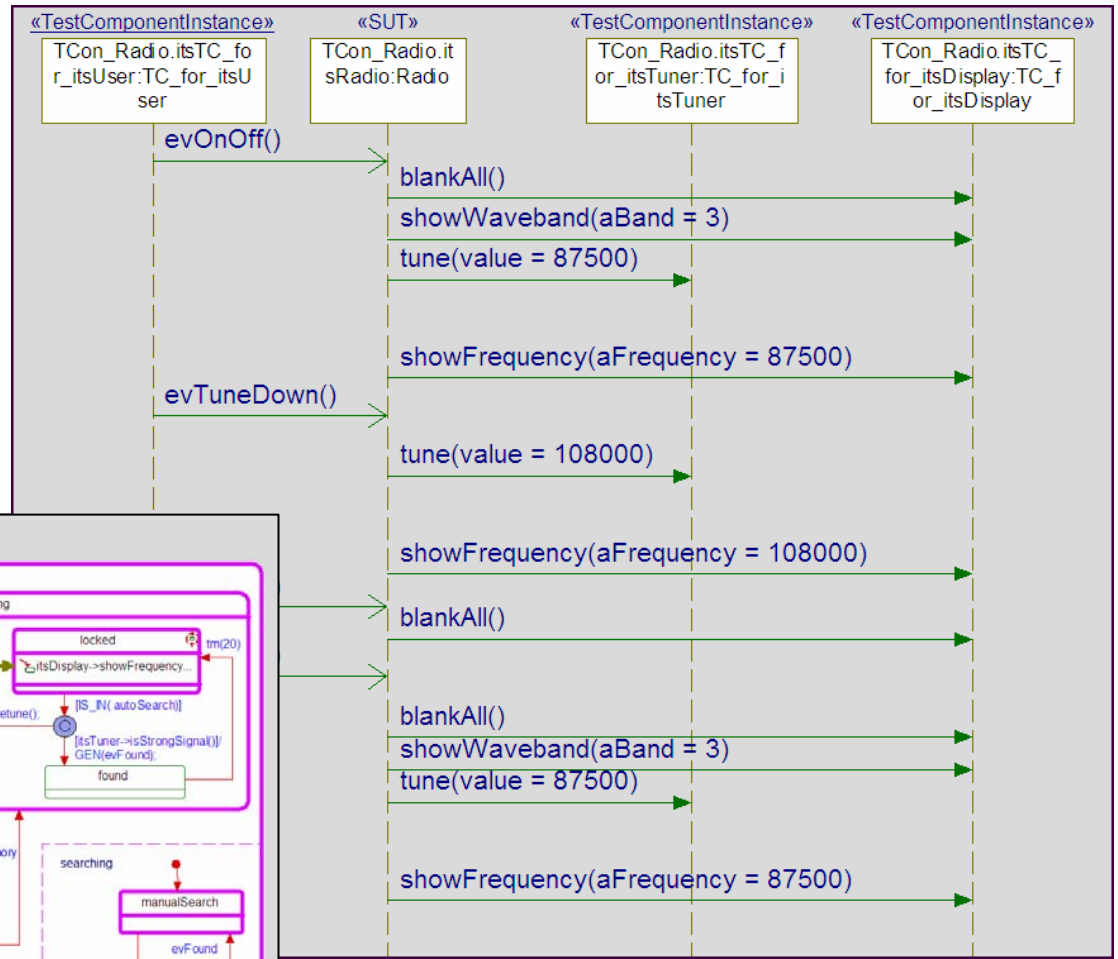
Proceso de pruebas: creación de casos de prueba

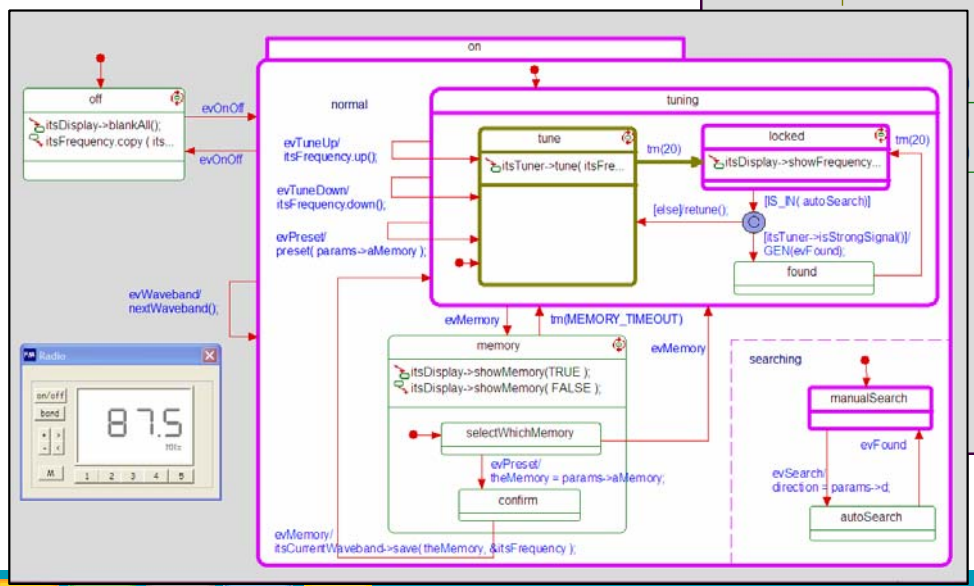
- Crear Test Architecture
- **Casos de prueba como diag de secuencia**
  - Manualmente
    - En base a
      - Requisitos Textuales
      - Experiencia previa o en el dominio
    - Se define el comportamiento esperado del elemento a probar o SuT (System under Test)



Proceso de pruebas: creación de casos de prueba

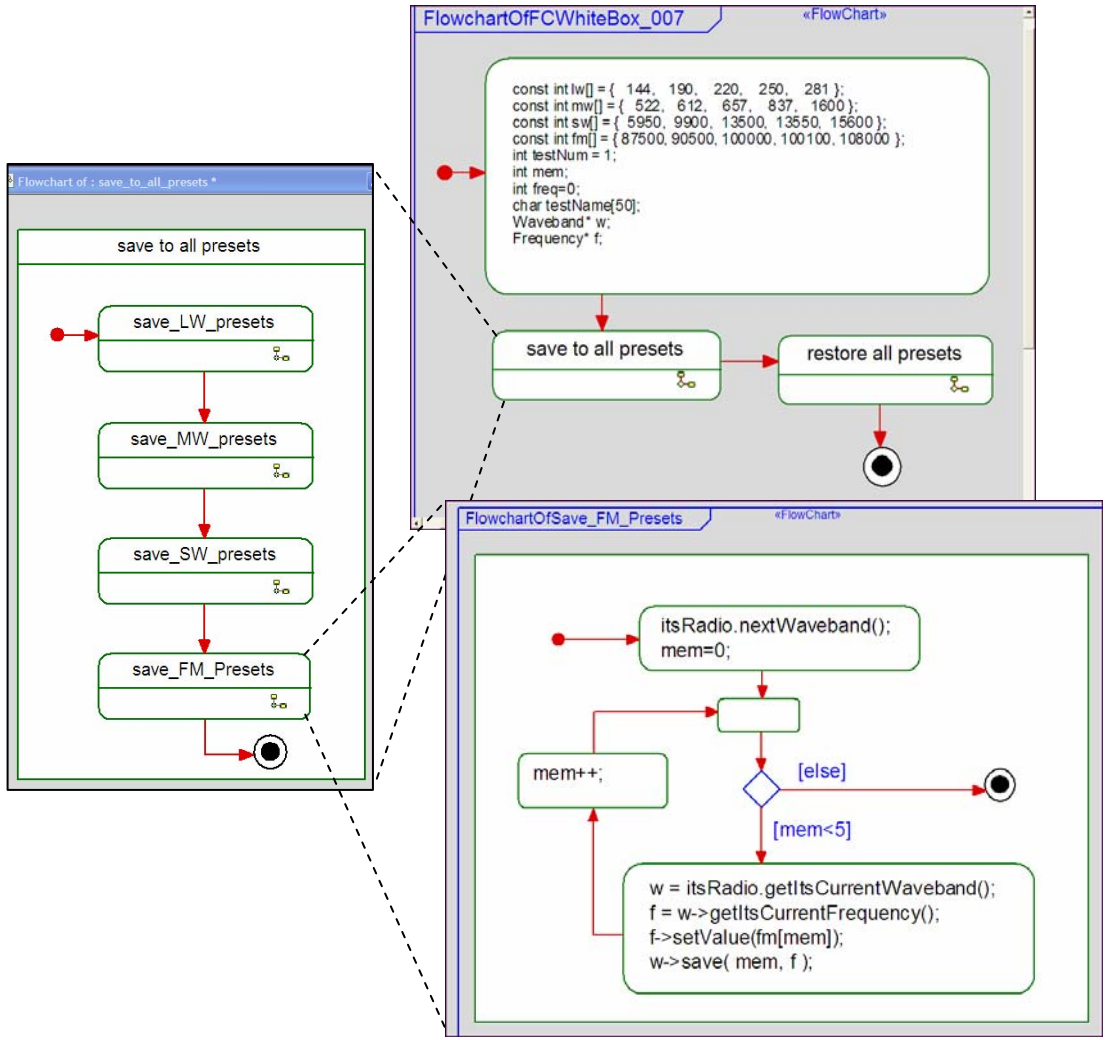
- Crear Test Architecture
- **Casos de prueba como diag de secuencia**
  - Manualmente
  - Grabación de simulaciones (diagramas de secuencia animados)





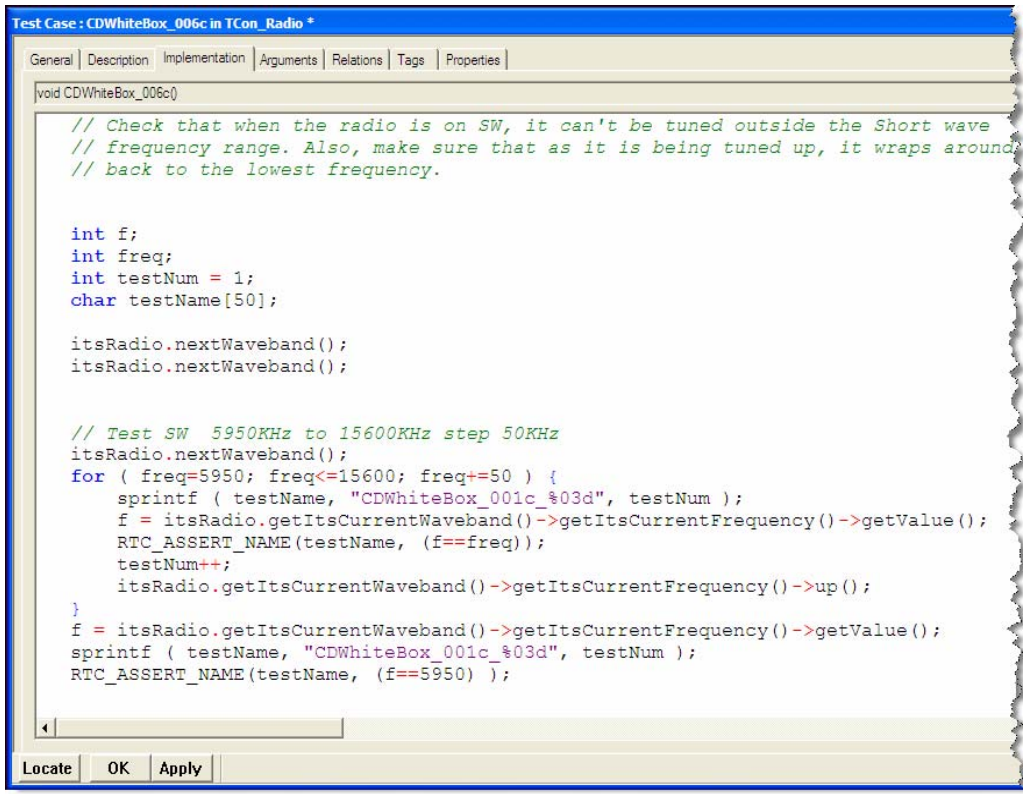
Proceso de pruebas: creación de casos de prueba

- Crear a Test Architecture
- Casos de prueba como diag de secuencia
  - Manualmente
  - Grabación de simulaciones
- **Casos de prueba como diagramas de flujo**
  - “Programación gráfica”
  - Intuitivos, potentes y fáciles de usar



Proceso de pruebas: creacion de casos de prueba

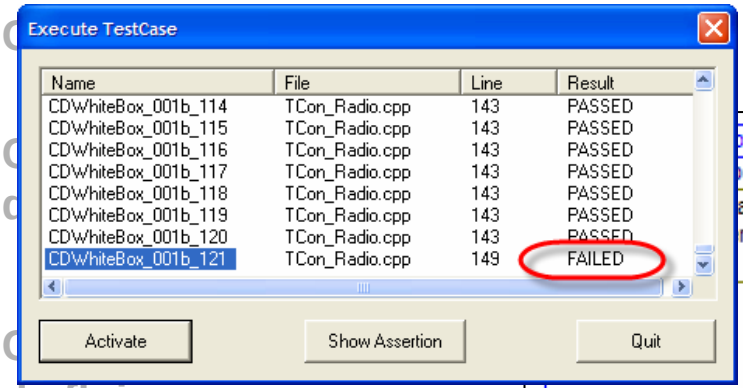
- **Crear Test Architecture**
- **Casos de prueba como diag de secuencia**
  - Manualmente
  - Grabación de simulaciones
- **Casos de prueba como diagramas de flujo**
  - “Programación gráfica”
  - Intuitivos, potentes y fáciles de usar

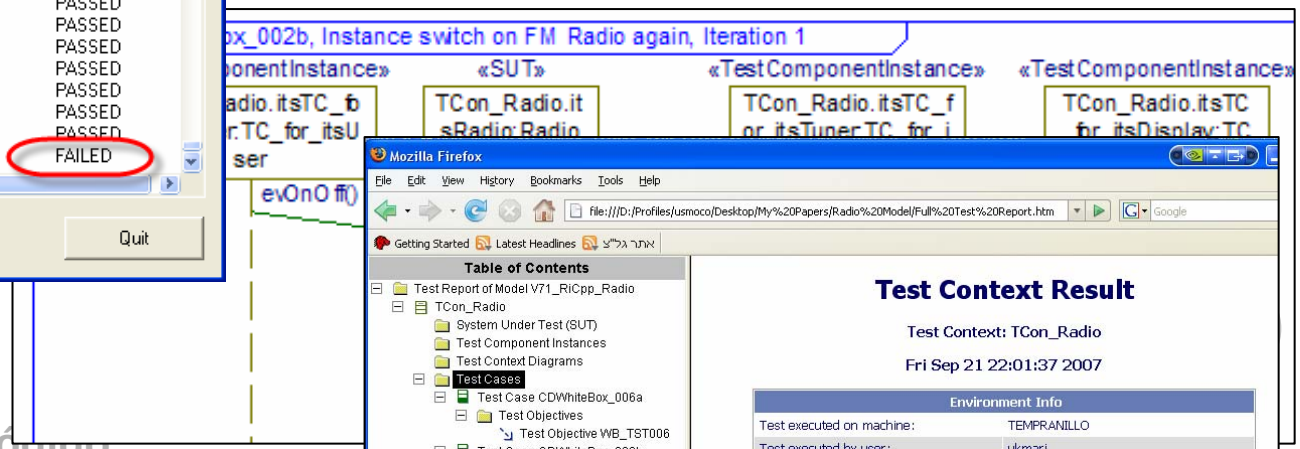


- **Casos de prueba como código**
  - El comportamiento del caso de prueba se puede introducir directamente codificando

Proceso de pruebas: ejecucion y resultados

- C
- C
- C
  de flujo
- Casos de prueba como código
- **Ejecución e informe de resultados**
  - Comportamientos inesperados se resaltan

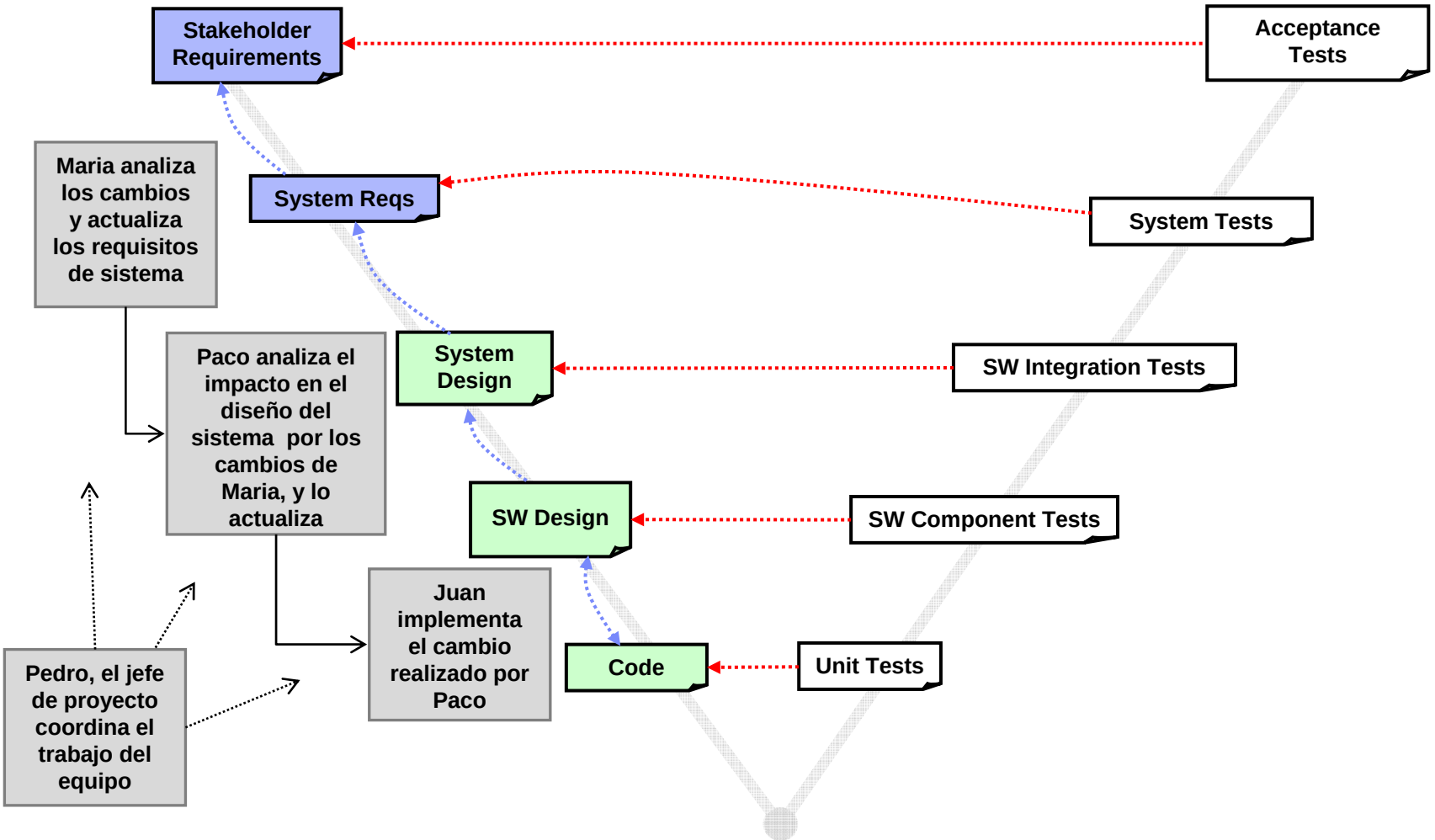




Colaboración

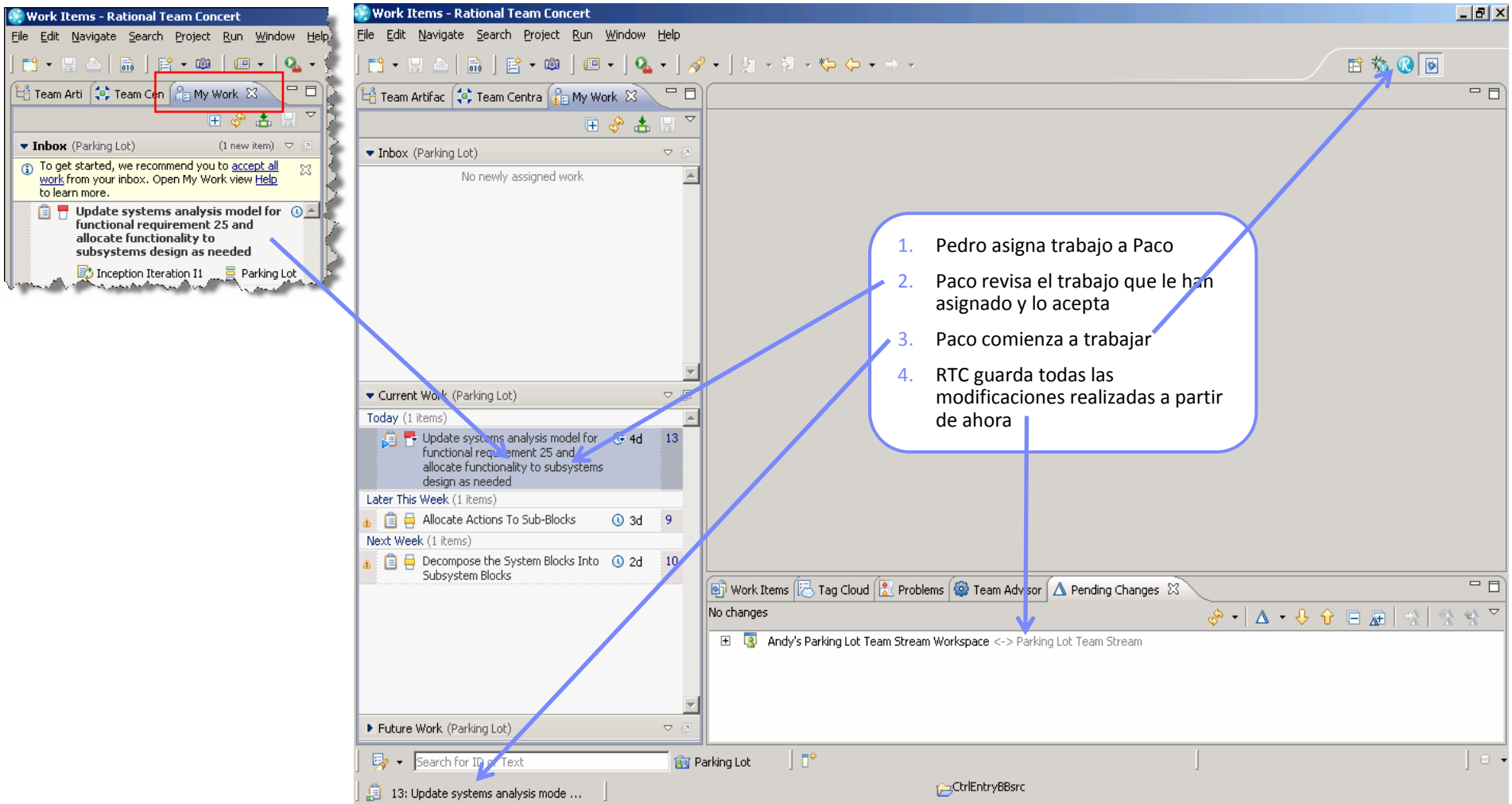
- Soporte a equipos de desarrollos de cualquier tamaño
  - En el mayor despliegue hay 1200 usuarios
- Gestión del desarrollo en paralelo con comparador gráfico de diferencias entre modelos (también merge)
- Integración con herramientas de control de configuración de software como Rational ClearCase®, Rational Synergy™y Rational Team Concert

Colaboración – Integración con Rational Team Concert



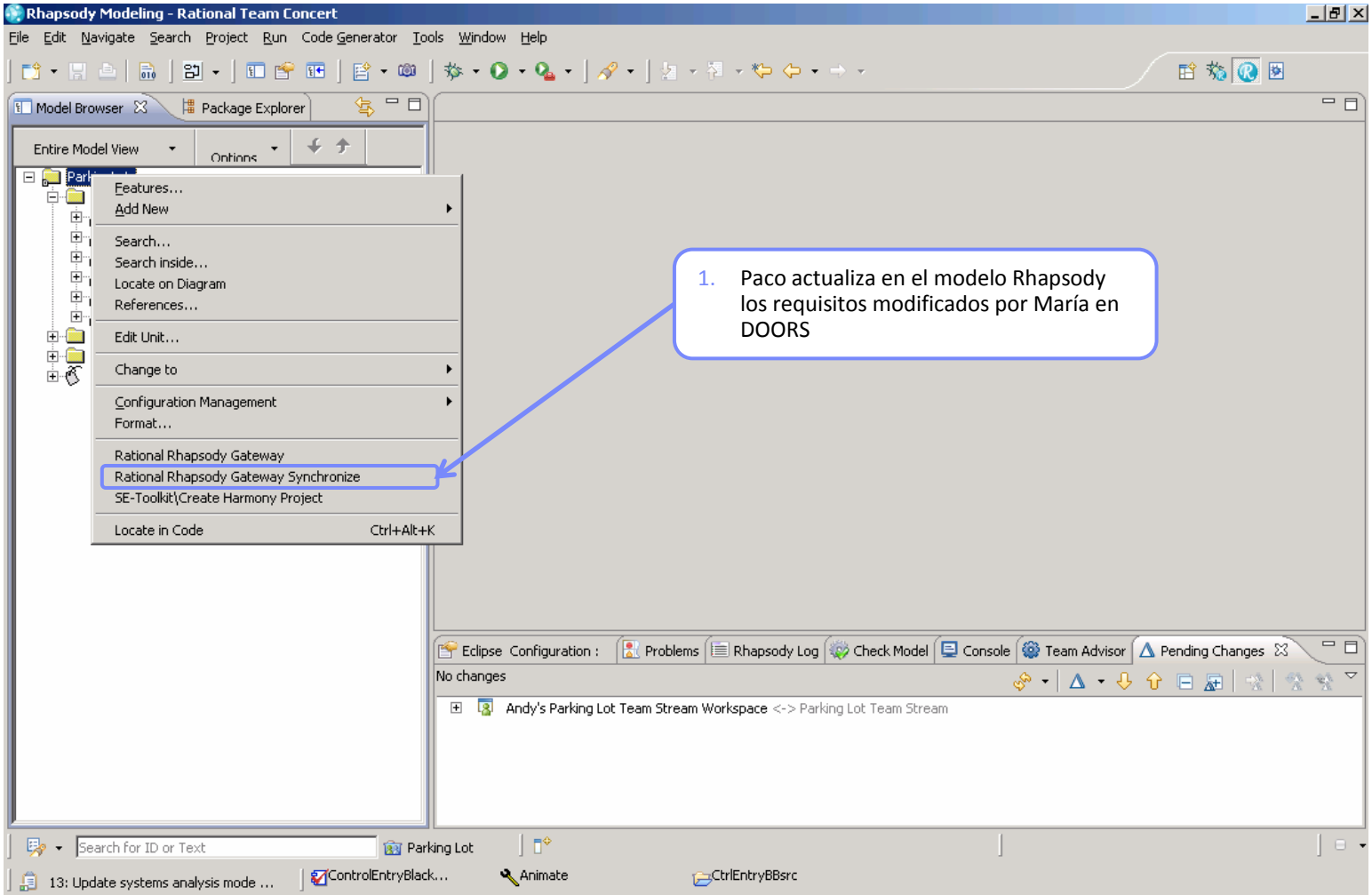
Colaboración – Integración con Rational Team Concert

Escenario ejemplo



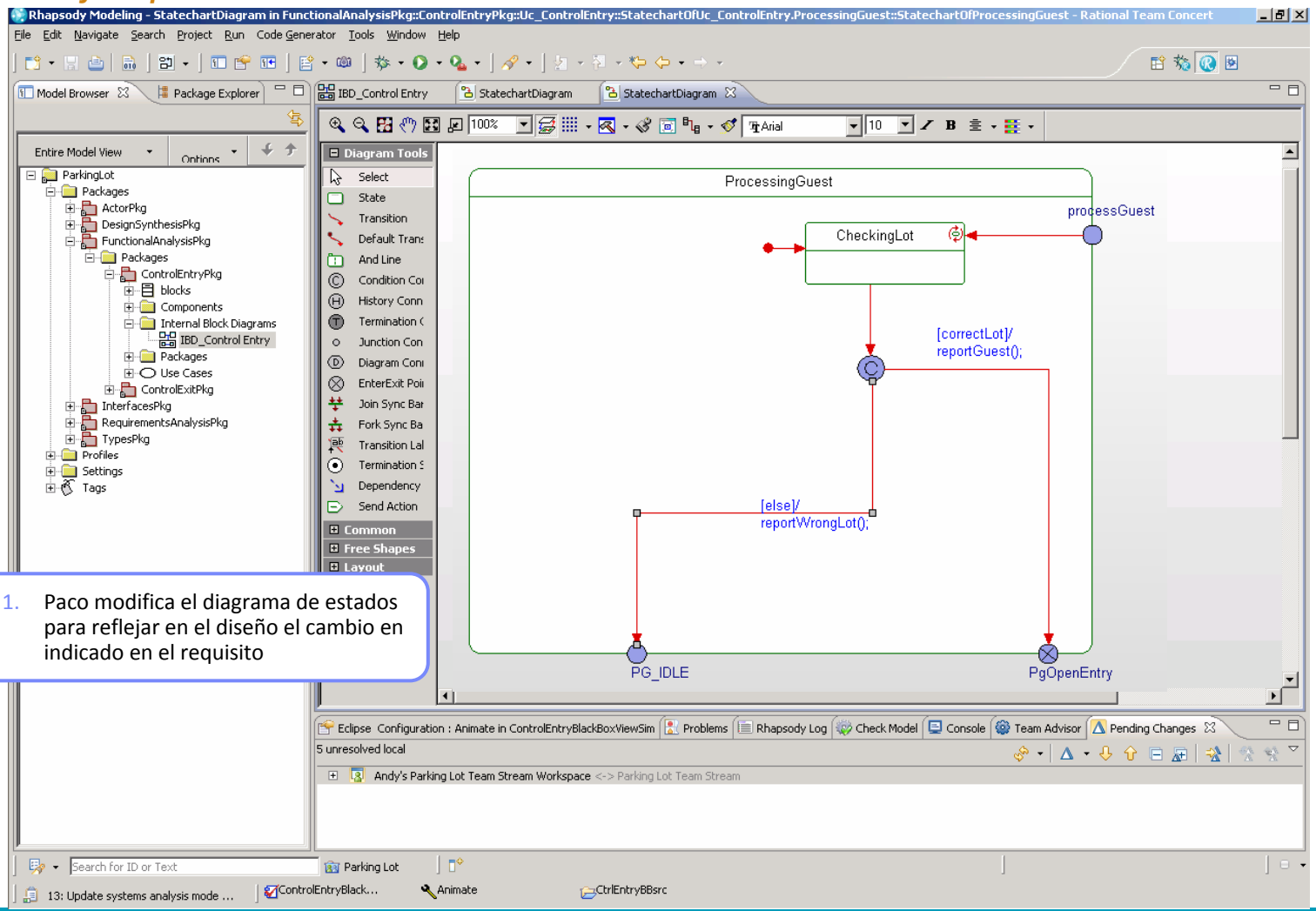
Colaboración – Integración con Rational Team Concert

Escenario ejemplo



Colaboración – Integración con Rational Team Concert

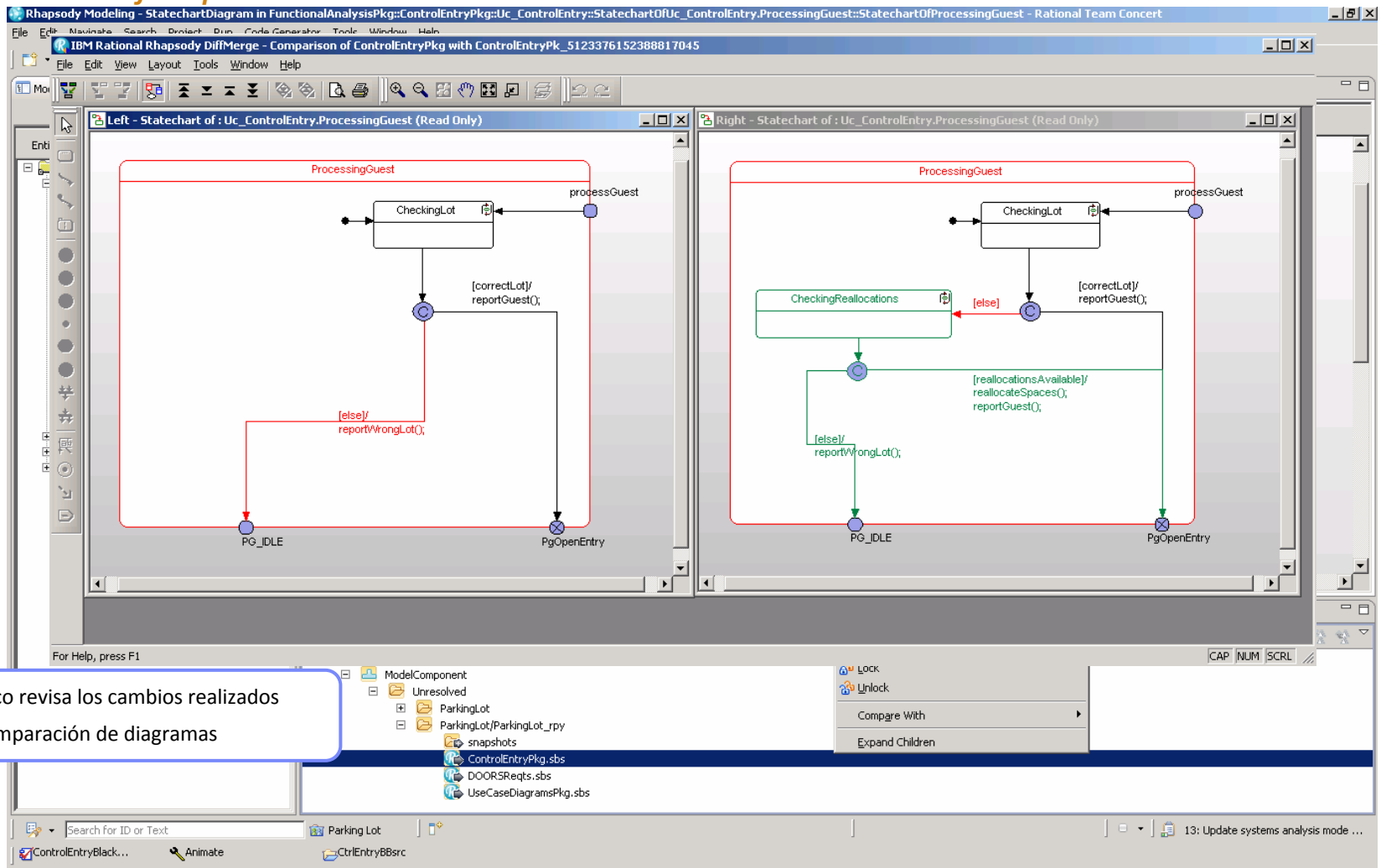
Escenario ejemplo



1. Paco modifica el diagrama de estados para reflejar en el diseño el cambio en indicado en el requisito

Colaboración – Integración con Rational Team Concert

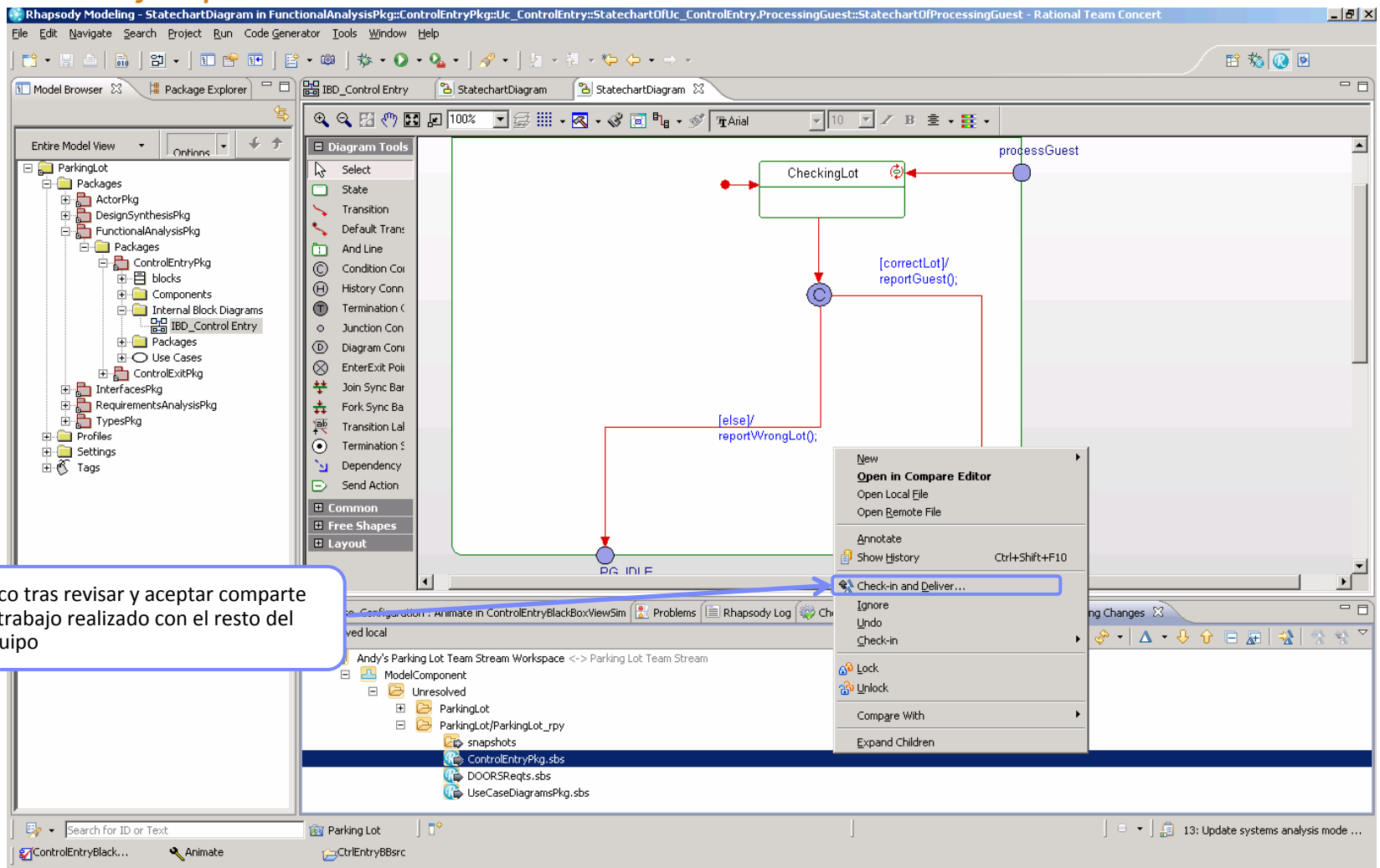
Escenario ejemplo



1. Paco revisa los cambios realizados
2. Comparación de diagramas

Colaboración – Integración con Rational Team Concert

Escenario ejemplo

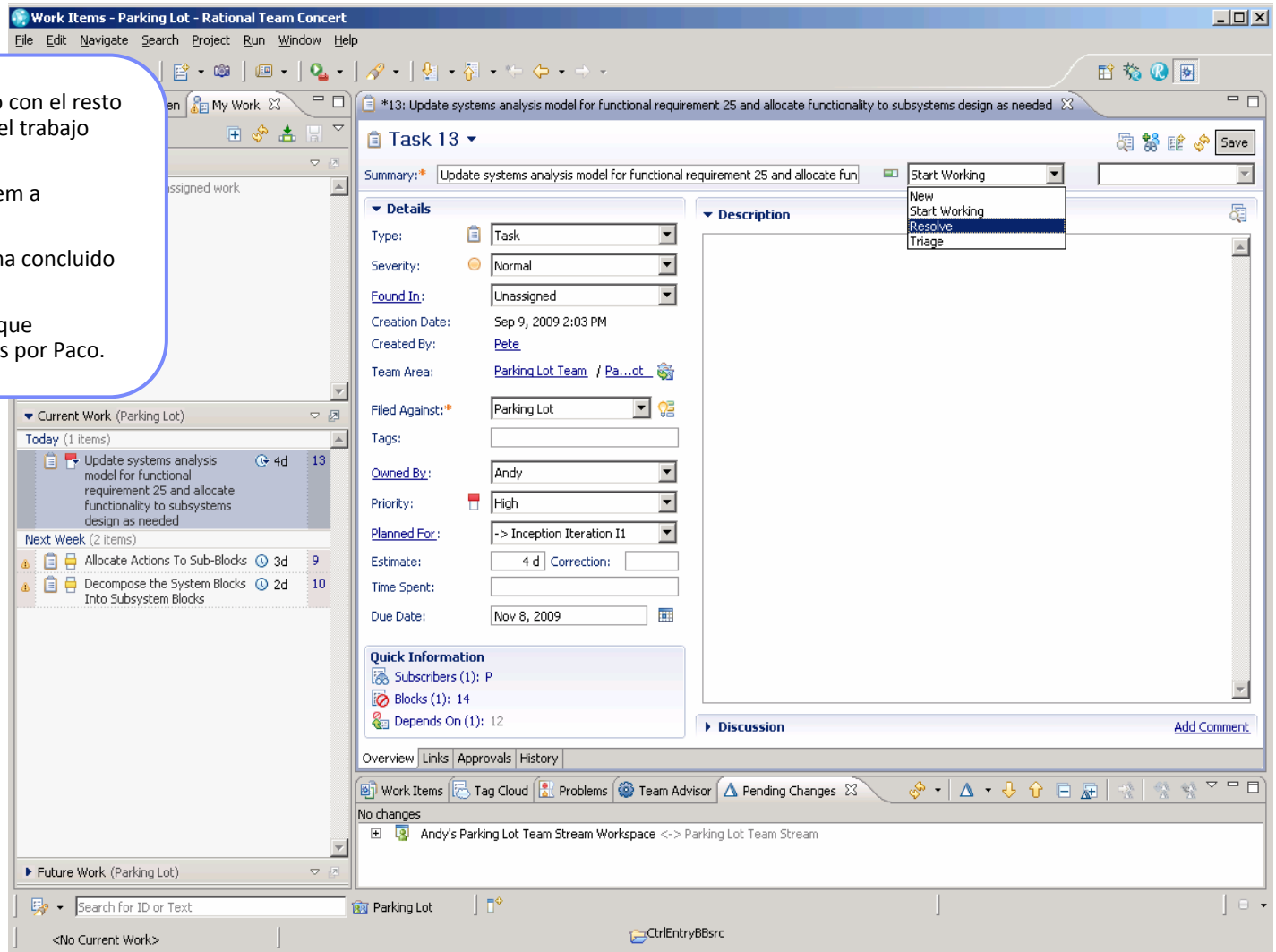


1. Paco tras revisar y aceptar comparte el trabajo realizado con el resto del equipo

Colaboración – Integración con Rational Team Concert

Escenario ejemplo

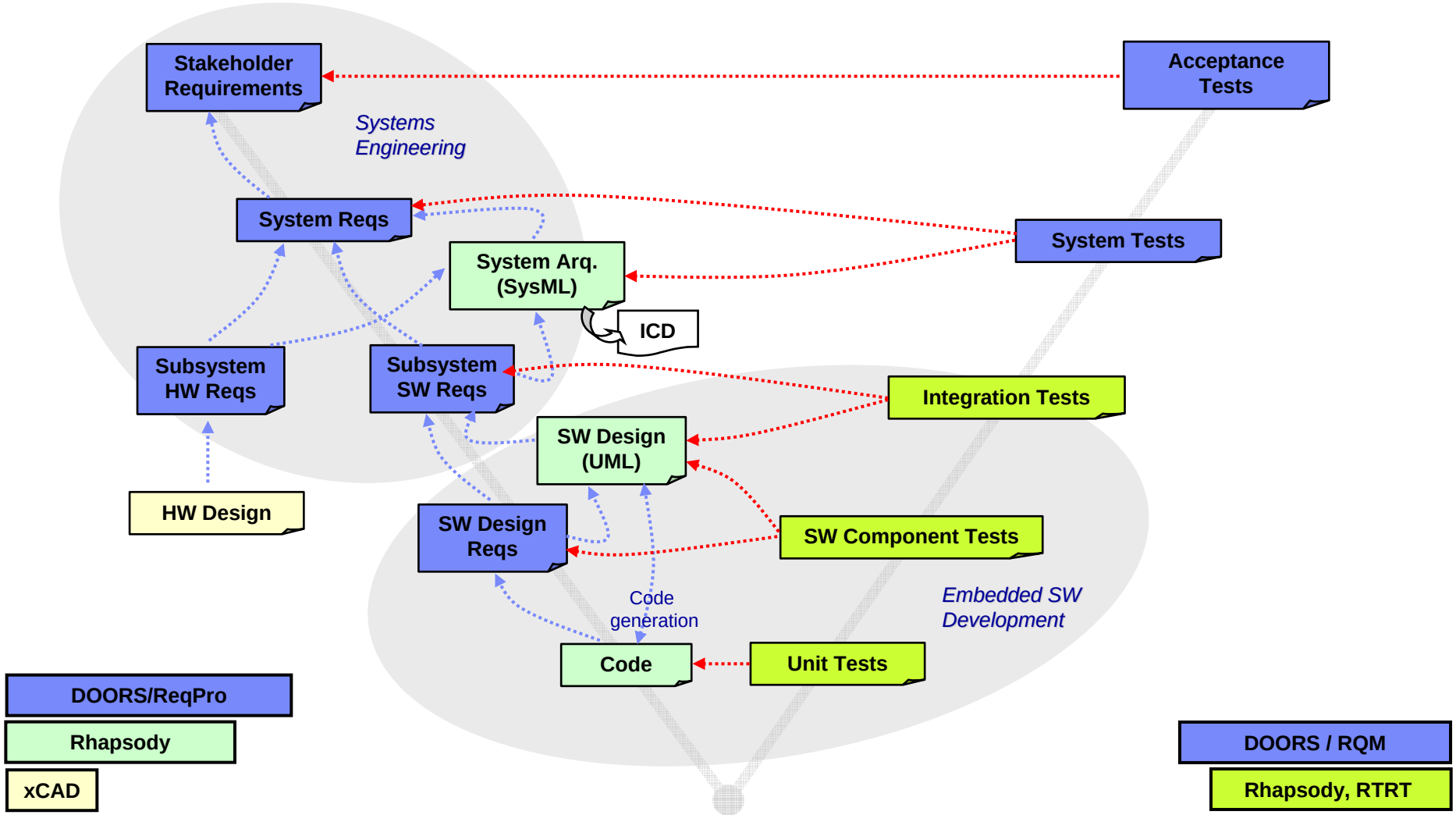
1. Tras compartir el trabajo realizado con el resto del equipo Paco da por concluido el trabajo “Stop Working”
2. Paco cambia el estado del work item a “Resolve”
3. Pedro, es notificado de que Paco ha concluido su trabajo
4. Pedro, asigna trabajo a Juan para que implemente los cambios realizados por Paco.



Resumen - IBM Rational Rhapsody®

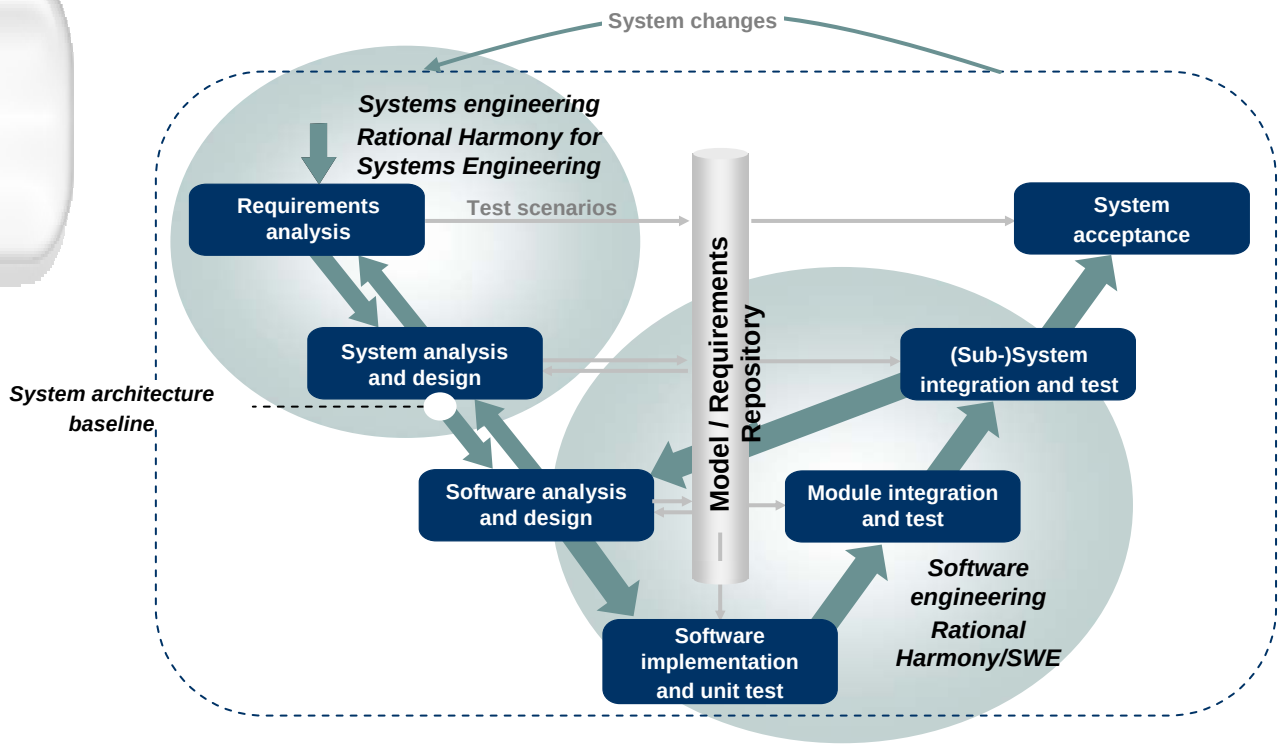
- Solución MDD de IBM Rational para el diseño y desarrollo de sistemas complejos, embebidos y de tiempo real
- Detección y eliminación temprana de errores en diseño mediante la simulación del sistema (verificación temprana)
- Incrementa la productividad del desarrollador mediante la generación automática de código y de la documentación, y DMCA.
- Mejora la comunicación con el cliente y entre disciplinas mediante la utilización de UML/SysML
- Visualización de los requisitos y trazabilidad al modelo y trazabilidad:
  - Asegura que el diseño cumple los requisitos
  - Impacto de cambios
- Mejora la colaboración entre equipos de desarrollo distribuidos
- Incrementa la productividad del equipo de pruebas:
  Pruebas dirigidas por el modelo *(Model Driven Testing – MDT)*

Proceso, herramientas…



… y Rational Harmony

Harmony es la librería de buenas prácticas para el desarrollo de software y sistemas



Dos versiones:

- Rational Harmony/SE
- Rational Harmony/SWE

Francisco J. López Minaya

Rational Technical Solution Architect

francisco.lopezminaya@es.ibm.com