



IBM Software Group

# 嵌入式軟體模型導向開發 --以視覺化方式開發系統和軟體 **Rational Rhapsody**

**Rational** software

→ Go to **IBM**

劉正陽(Steven Liu)

Rational高級資訊工程師

IBM軟體事業部

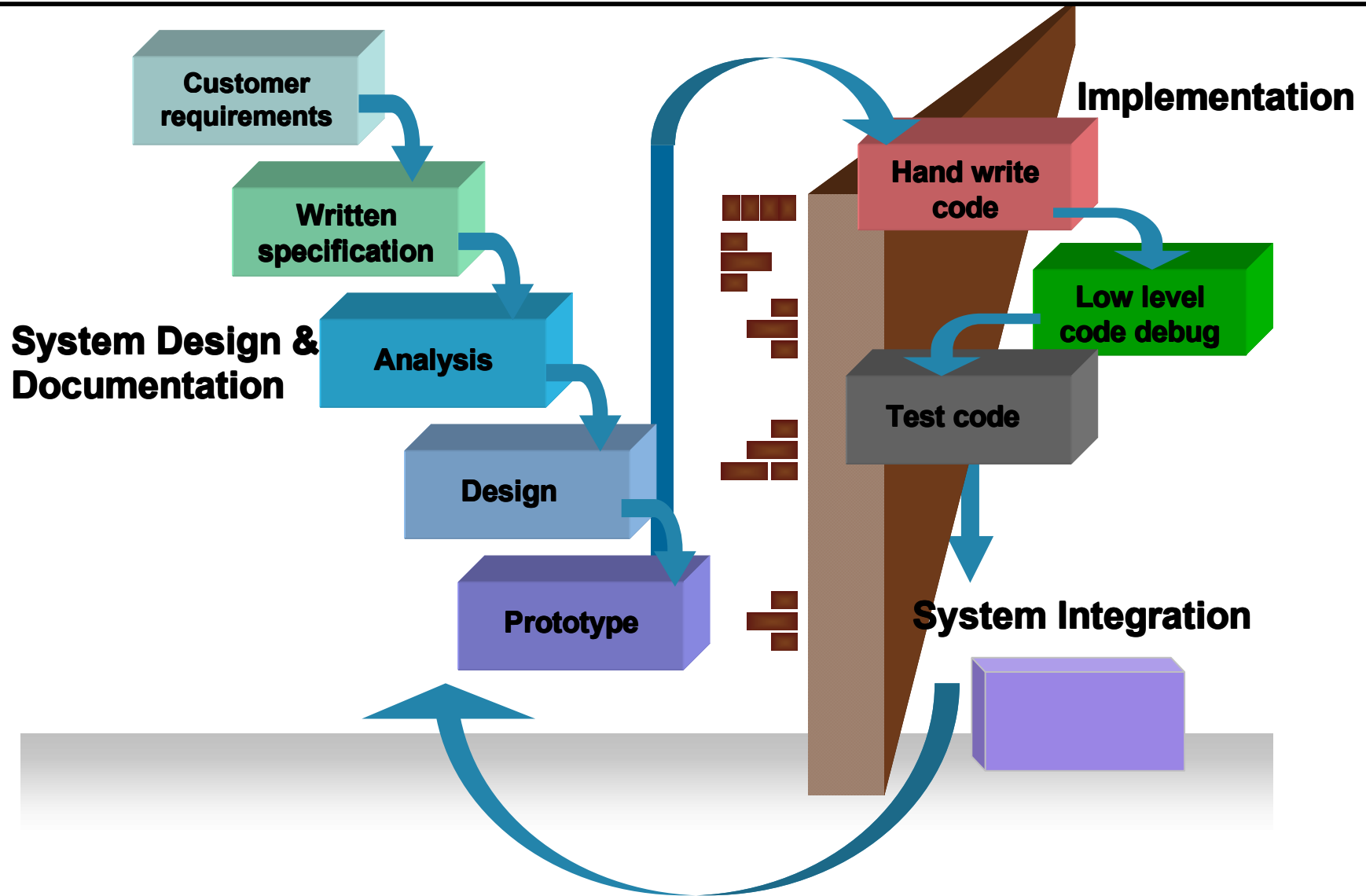
© 2008 IBM Corporation

## 現況分析:

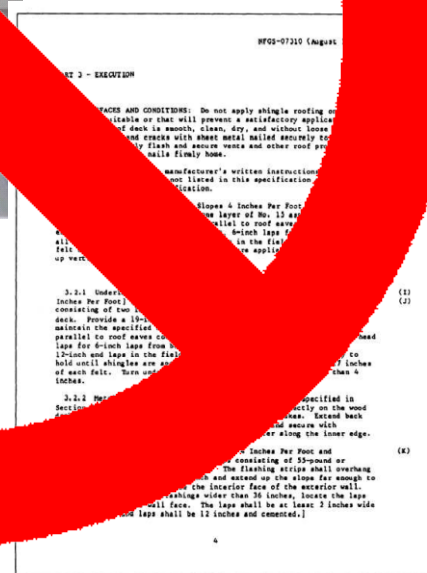
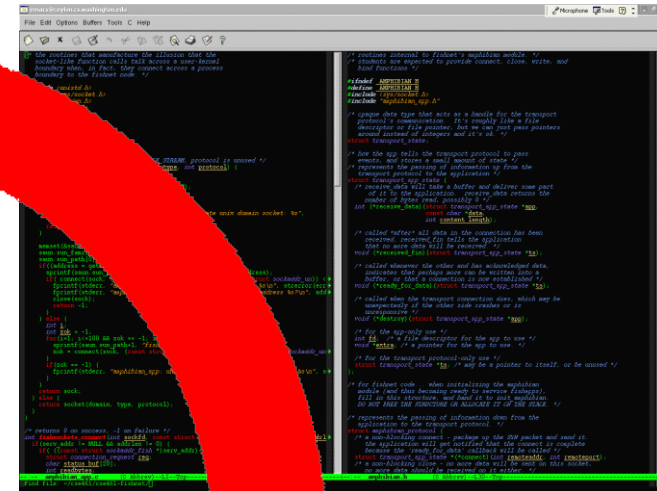
- 遇到問題
  - ▶ 作業系統與應用程式無法分開獨立，只能針對某一作業系統平台開發應用程式。
  - ▶ 嵌入式軟體開發過程中系統分析、設計文件與程式碼無法保持一致性。
  - ▶ 部署到target system才發現問題，除錯困難，耗時費力。
  - ▶ 拿到一個需求規格就立即撰寫程式，無法事先做好規劃與設計。
  - ▶ 若應用程式部署到不同的作業系統環境，必須做大幅度的程式碼修改，耗時費力且增加除錯時間。
  - ▶ 需求規格無法做有效的追蹤，以驗證所開發的系統功能符合需求。



# The traditional design process



# Visual Modeling



Expansion	1* PCI Express x16 Slot, 1* PCI Express x1 Slot with CrossFire™ support, 2* PCI Express x1 slots, 2* PCI bus slots
Connectivity	1* eSATA, 8* SATAII, 2* IEEE1394a, 12* USB2.0, 1* FireWire
RAID	RAID 0, RAID 1, RAID 5, RAID 10 with Intel® Matrix Storage Technology and Intel® Rapid Recover Technology
Audio	7 Channel HD Audio
Networking	Digital LAN: Gigabit Ethernet, 802.11n Wireless LAN
Special Features	Onboard BIOS with OC Gear, OC Recovery, Cool Fan Control, Onboard CMOS & On/Off/Reset buttons, JAMM Panel, Cool Pipe (patented solution), 100% Solid Capacitor design, EasyFit.
Accessories	Linux manual, EZ Set-up Guide, Quantum Force product registration card, Quantum Force free gifts: 8* SATA cables + power cables, 1* FDD + IDE cable, 1* 2.5" USB + 1.8" Mini bracket, 1* Fan (optional for North America), IBM CD including Systematic Support, RAID Floppy Disk
Form Factor	ATX

## 하와이

The Kaneohe Hawaii Edition  
1999년 7월 17일 (목요일)

**당뇨치료제 하와이 공급업자 모집**

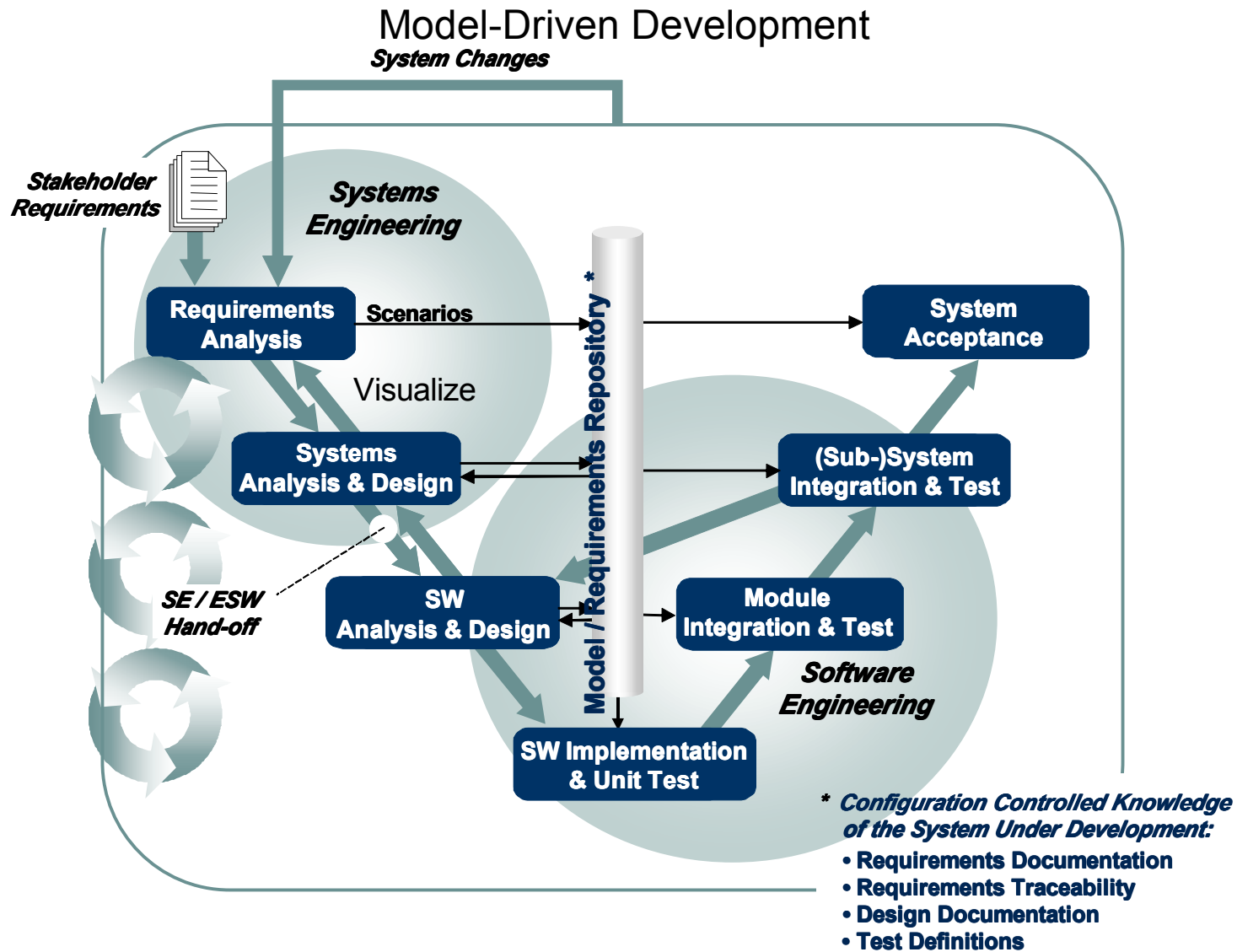
하와이 공약지역에 거주하는 당뇨병 환자들을 위한 당뇨치료제 공급업자를 모집합니다. 모집대상은 하와이 공약지역에 거주하는 당뇨병 환자들을 위한 당뇨치료제 공급업자입니다. 모집기간은 2009년 7월 17일부터 2009년 7월 24일까지입니다. 모집신청은 하와이 공약지역에 거주하는 당뇨병 환자들을 위한 당뇨치료제 공급업자 모집 신청서와 함께 하와이 공약지역에 거주하는 당뇨병 환자들을 위한 당뇨치료제 공급업자 모집 신청서 접수처에 제출합니다. 모집신청서 접수처는 하와이 공약지역에 거주하는 당뇨병 환자들을 위한 당뇨치료제 공급업자 모집 신청서 접수처입니다. 모집신청서 접수처는 하와이 공약지역에 거주하는 당뇨병 환자들을 위한 당뇨치료제 공급업자 모집 신청서 접수처입니다. 모집신청서 접수처는 하와이 공약지역에 거주하는 당뇨병 환자들을 위한 당뇨치료제 공급업자 모집 신청서 접수처입니다.

全国日本語学校データベースによるこそ!  
このデータベースでは、全国にある400校以上のすべての日本語学校と日本語教育を行っている大学、短期大学、専門学校の情報を提供しています。データベースには、学校名や、コース、授業料、入学案内などの情報が含まれています。  
なお、このデータベースに掲載されている日本語学校は、すべて(財)日本語教育振興協会の認定を受けています。

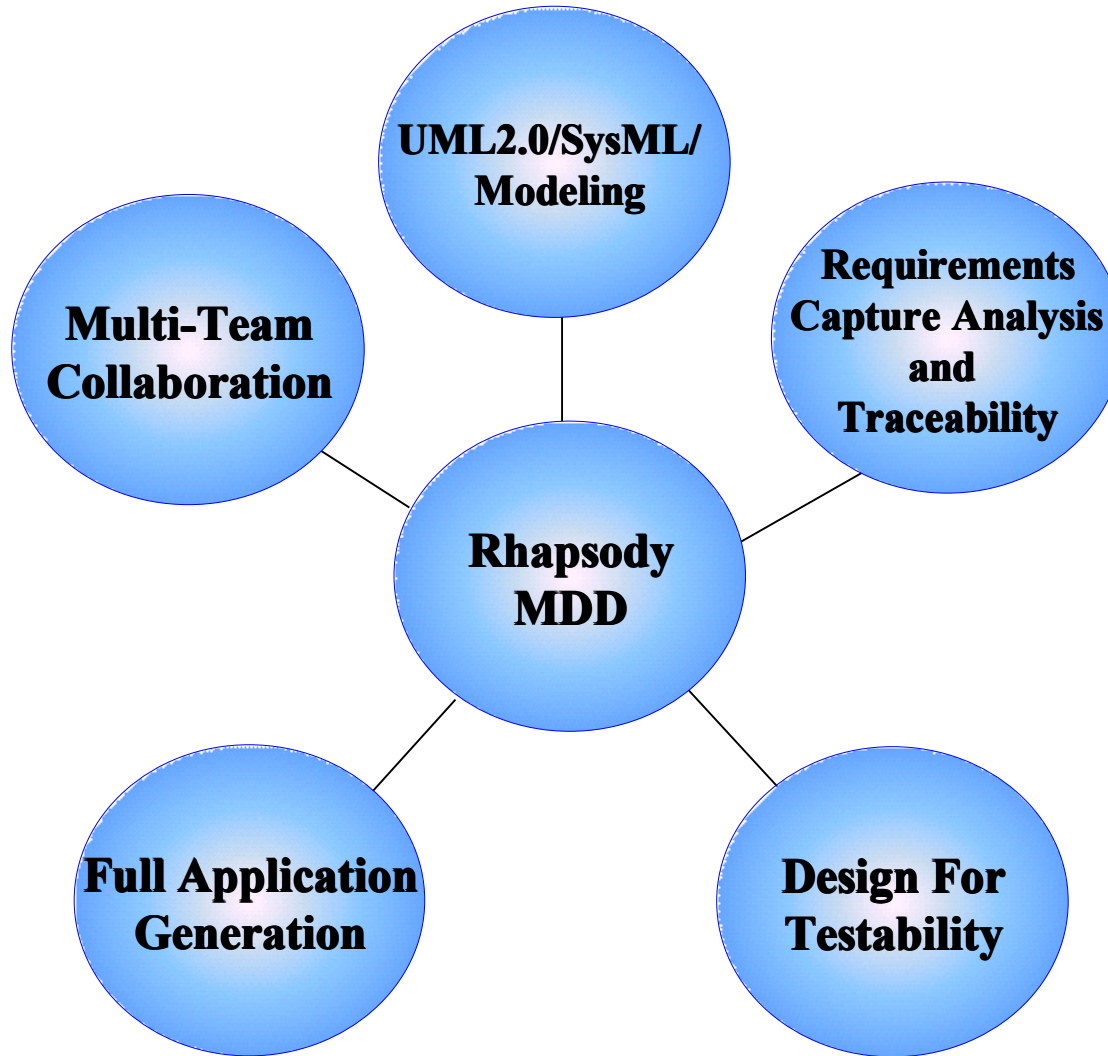
## Watch phone



# Integrated System / Software Development Process



## Marking All These Work



*Rhapsody*<sup>®</sup>



## Conceptual Collaboration in Text

### Developer 1

“Ok. Here’s how it works. Thread A will pass event X to thread B and that will change B’s state to Running from what it was before which was Init. When B changes to Running it will send back an event Y to A and then wait for 2 second and then go back to Idle. Thread A will have started in Idle also and will go to Run after B sends back event Z which happens after the 2 seconds before going to Idle. All this should happen in less then 5 seconds.”

### Developer 2:

“Huh ?” What are you talking about?



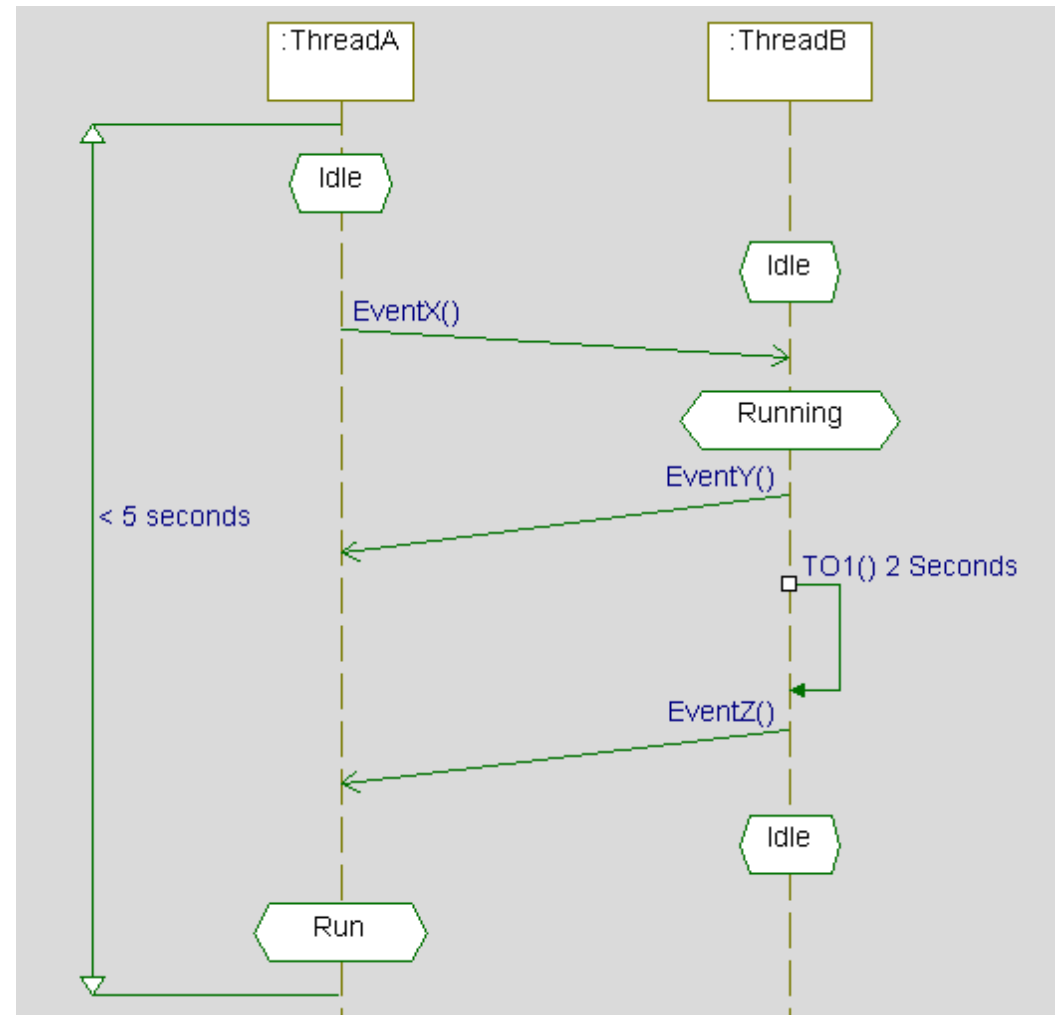
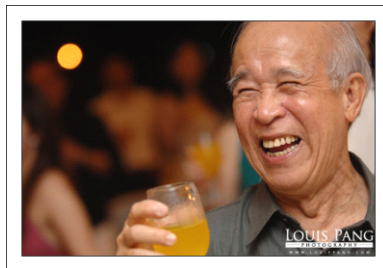
## Conceptual Collaboration in Models

Developer 1

“Here look at this Sequence Diagram.”

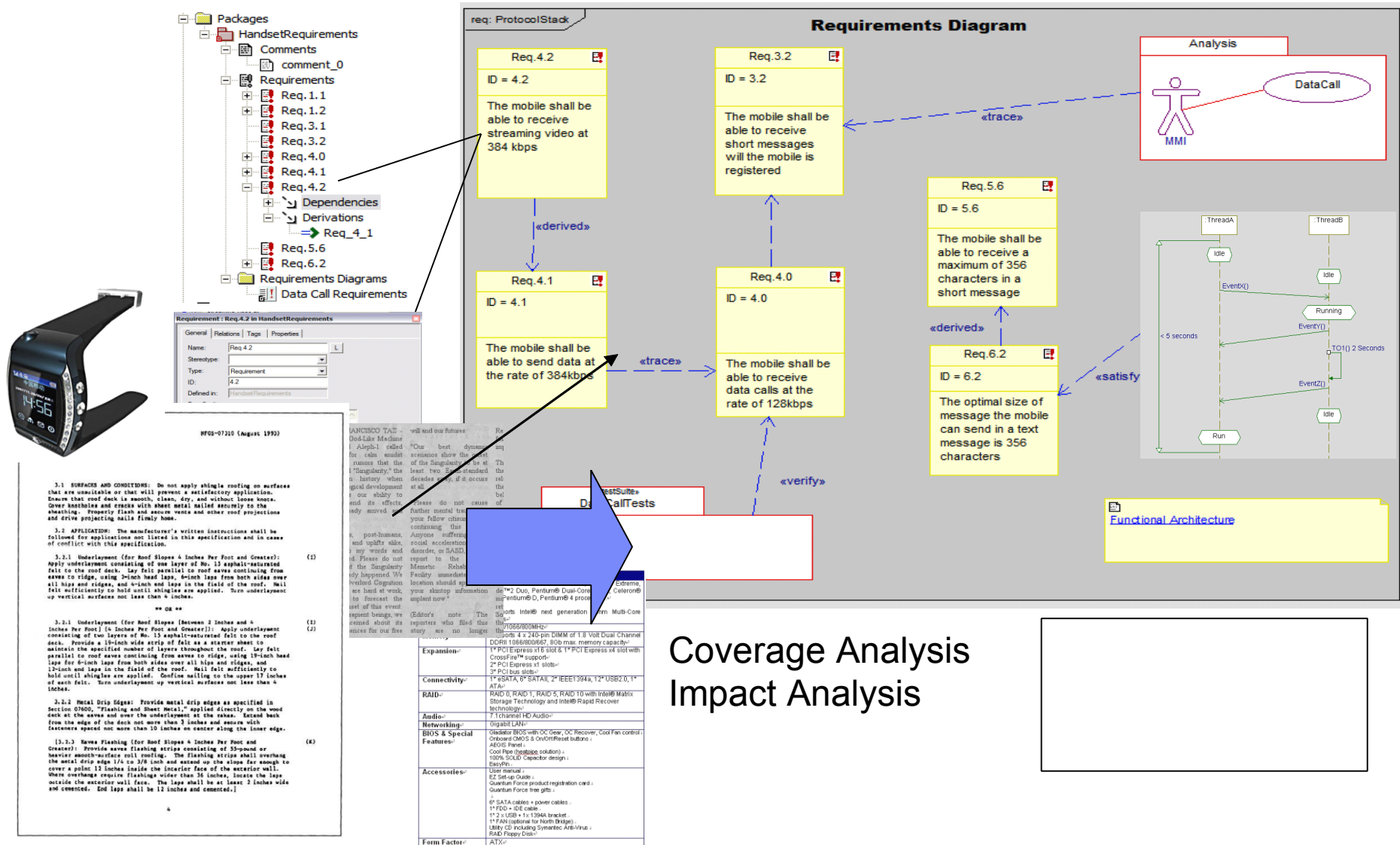
Developer 2

“Ahhh, now I see!”



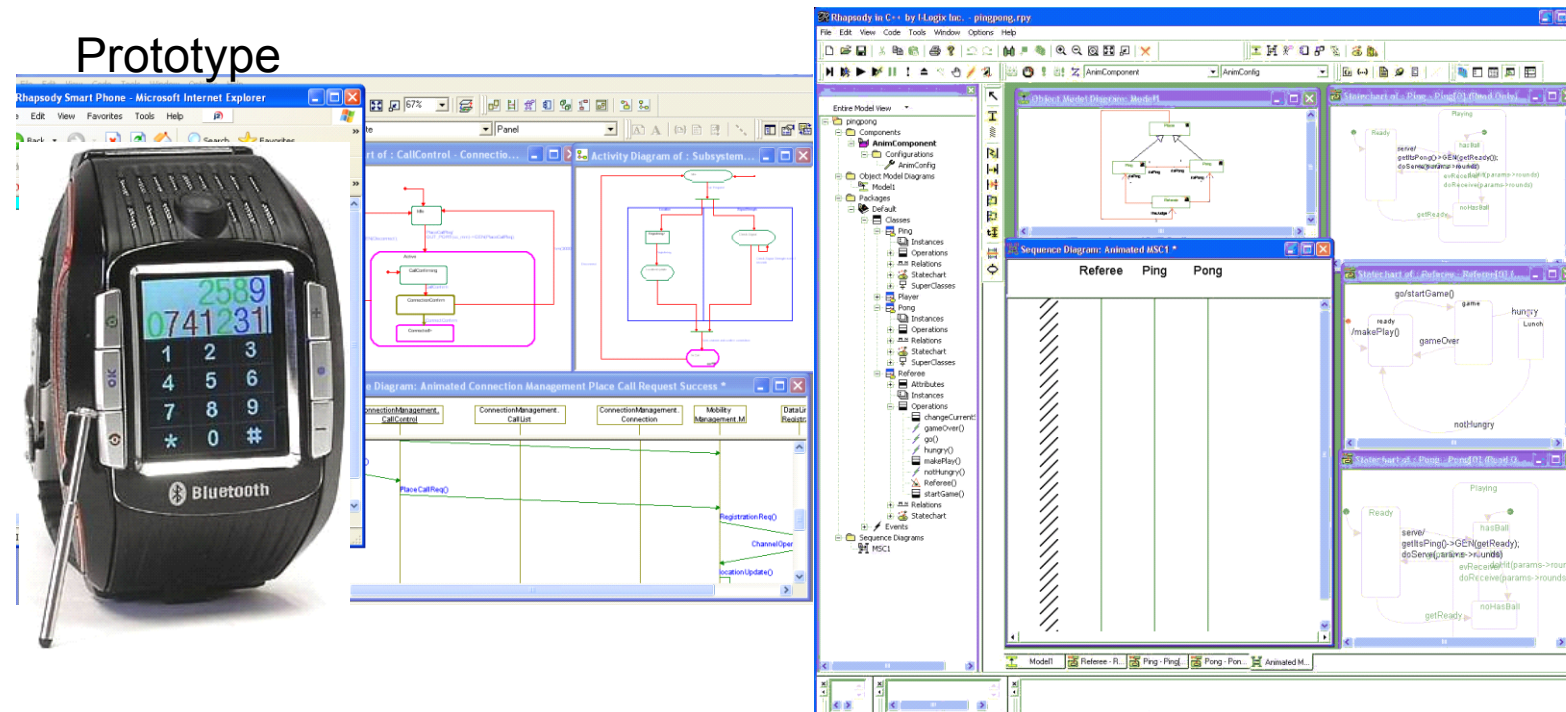


# Requirement Traceability



## Executable models (Simulation and Animation)

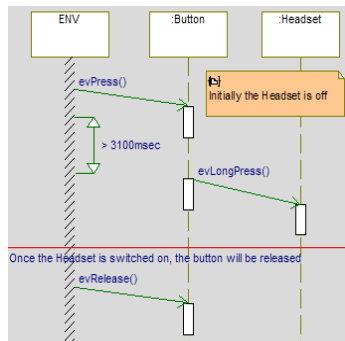
*You can't test what you can't execute!*



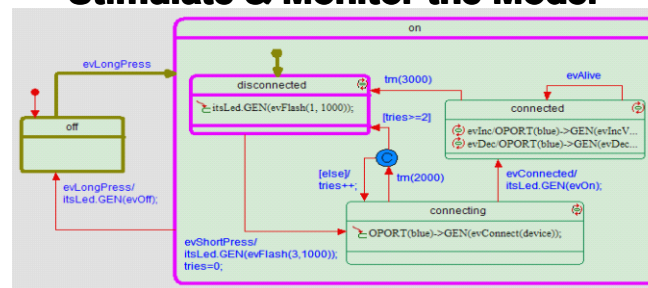
# Requirements Based Testing

- Use requirement scenarios to validation the design
- Automatically run multiple scenarios
- Easily identify errors

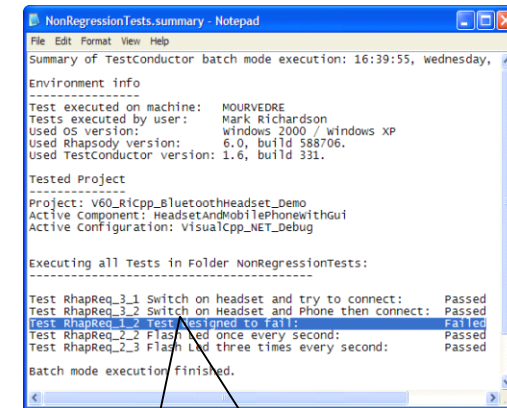
## Test Scenario



## Stimulate & Monitor the Model



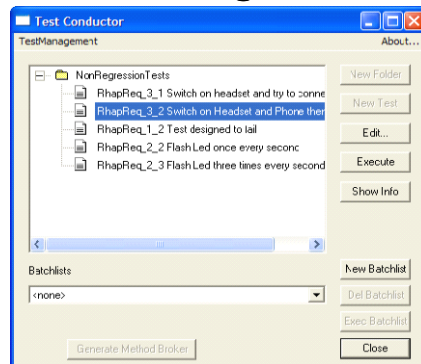
## Test Results



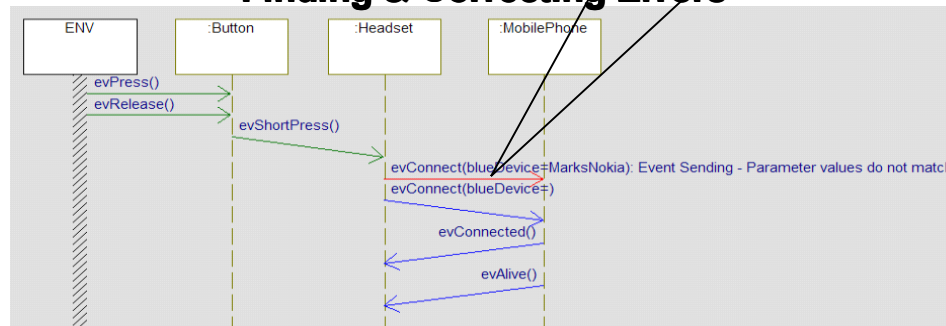
*Failed Test*

*Unexpected result*

## Test Configuration



## Finding & Correcting Errors



## Full Application Generation

---

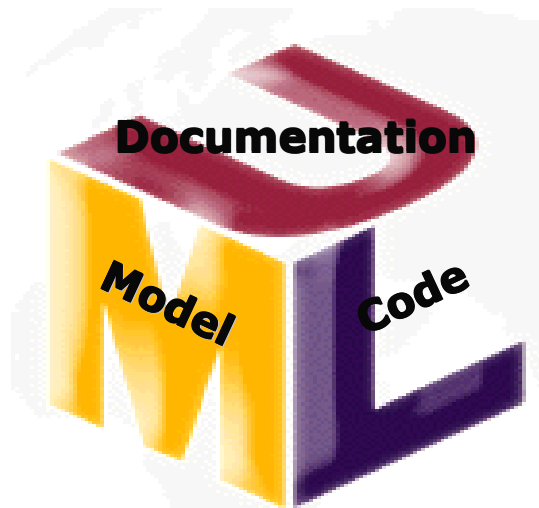
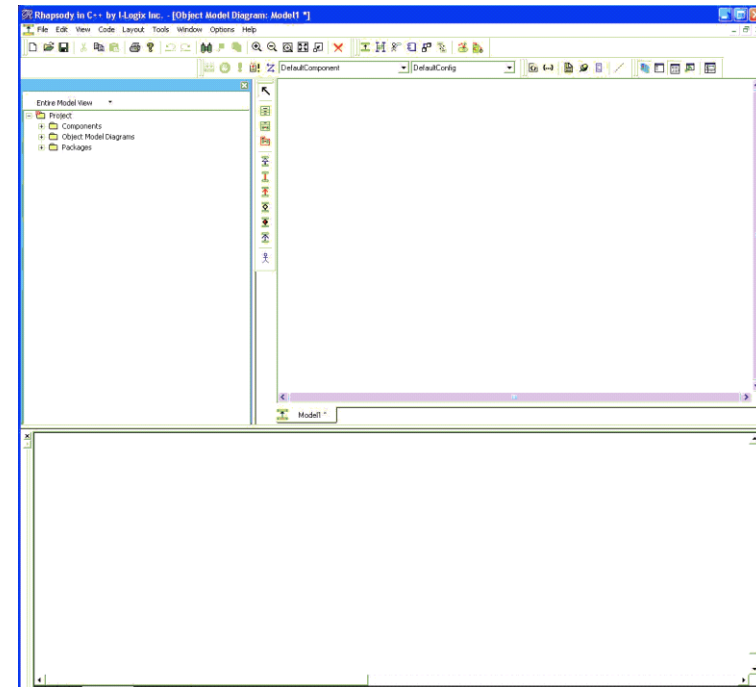
- Rhapsody leverages *all* structural and behavioral model views to produce an executable application
  - State machines: event driven behavior
  - Activity diagrams: algorithms and process flows
  - Generates all construction artifacts (e.g. Makefiles)
- Support for
  - ▶ C, C++, Ada and Java
  - ▶ Size/Speed tradeoffs
  - ▶ Coding style options
- **Seamless Reuse** of existing code and models (IP)
- **Dynamic Model Code Associativity (DMCA)** gives you the ability to work the way you want
- **The Real Time Framework** enables rapid application deployment onto any RTOS or systems with no RTOS



## Model Code Associativity

### *Rhapsody works the way you do*

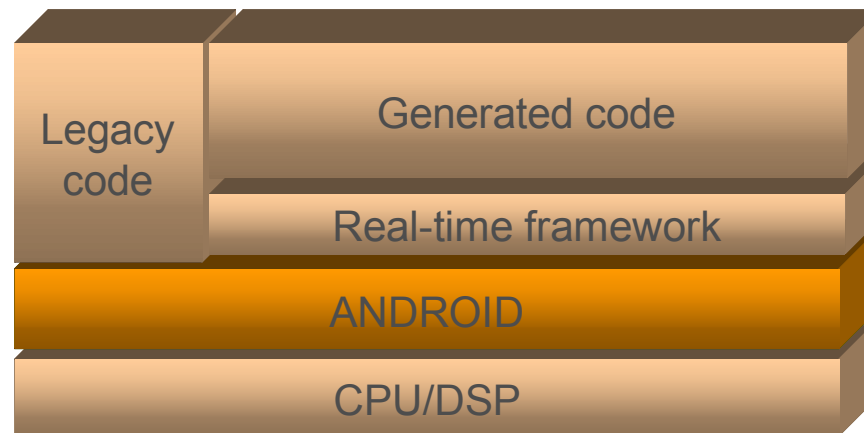
- Design, Code and Documentation are always kept in sync
- Freedom to work at code level or design level
- Change one view, the others ***change automatically***
- *Critical for real-time embedded software development*



## The Rhapsody Real-Time Framework

### *Rhapsody provides an executable real-time framework*

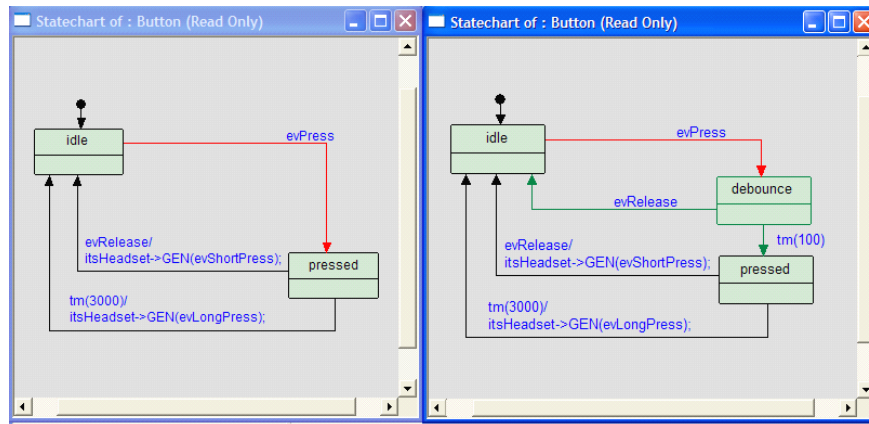
- Most applications are over 50% “housekeeping code” which is redeveloped every time you create a system.
- A *framework* is a partially completed application.
  - ▶ **you** customize and specialize for **your** application.
  - ▶ All source code is provided.
- A *real-time framework* is an:
  - ▶ integrated set of design patterns
  - ▶ optimized for embedded applications





# Rational Team Concert enables distributed teams to perform as one through integrated collaboration, process and tools.

- Real time, in-context collaboration
  - Make software development more automated, transparent and predictive
- "Think and work in unison"
  - Integrated planning, source control, work item and build management



*Open and extensible on*

- ✓ Collaborate in context
- ✓ Right-size governance
- ✓ Day one productivity

## IBM Rational Team Concert



transparent *integrated presence*  
 wikis OPEN real-time reporting  
 chat automated hand-offs Web 2.0  
*custom dashboards* automated data gathering  
**EXTENSIBILITY** Eclipse plug-ins services  
 architecture **FREEDOM TO CREATE**

JAZZ TEAM SERVER



## Business Value 商業效益:

- 提高嵌入式軟體開發的生產力，開發人員可以專注於模型的設計與應用程式的邏輯設計，而非底層與作業系統相關的設計工作。
- 確保需求規格的涵蓋度，提高客戶滿意度。
- 視覺化的建模方式，有效提升專案開發人員的溝通能力，而非以往傳統用程式碼做為討論的方式。
- 模型可重複使用，自動產生出程式碼，節省開發時程，降低人工撰寫程式的錯誤率。
- 加速在不同作業系統環境的部署時間，反應市場快速變遷，提升競爭力。
- 系統分析、設計文件與程式碼永遠維持同步一致性，減少人員異動的困擾與維護的困難。
- 提早於模型設計階段透過模型的模擬執行發掘系統的問題，有效降低上線的風險。





## 範例展示 - Car System

- Attribute
  - ▶ 速度
- Operation
  - ▶ 點火
  - ▶ 熄火
  - ▶ 加速
  - ▶ 減速
  - ▶ 踩煞車



# 範例展示 - 建立 Use Case Diagram

The screenshot displays the Rational Rhapsody environment for creating a Use Case Diagram. The main workspace shows a diagram with an actor labeled 'driver' connected to a use case labeled 'moving passenger' inside a system boundary labeled 'Car System'. A red callout box points to the use case with the text '描述Car System功能'. The left sidebar shows a project browser with a tree view containing 'Components', 'Configurations', 'Test', 'Hyperlinks', 'Object Model Diagrams', 'Model1', 'Packages', 'Default', 'Actors', 'Classes', 'Car', 'Attributes', 'Operations', 'Statedchart', 'Events', 'Use Cases', 'PredefinedTypes (REF)', 'PredefinedTypesCpp (REF)', 'Settings', and 'Use Case Diagrams'. The bottom status bar shows 'GE MODE | CAP | NUM | SCRL | Fri, 15, Jan 2010 | 10:27 AM'.

# 範例展示 - 建立 Class diagram

The screenshot displays the Rational Rhapsody IDE. The main window shows a UML Class Diagram for a class named 'Car'. The class has a private attribute 'speed:int=0' and several public methods: 'ignitionOn():void', 'ignitionOff():void', 'accelerate():void', 'decelerate():void', 'brake():void', and 'evTurnOn()'. A red callout box points to the methods section with the text '描述Car System的屬性與操作' (Describe Car System's attributes and operations).

Below the diagram, the C++ code is visible in a separate window. A red callout box points to the code with the text '產生出程式碼' (Generate code). The code snippet is as follows:

```

030 /// package Default
031
032 /// class Car
033 class Car : public OMReactive {
034     //// Friends ////
035
036 public :
037

```

# 範例展示 - 建立 Statechart Diagram

The screenshot displays the Rhapsody in C++ interface for a statechart diagram titled "Statechart of : Car". The diagram shows the following structure:

- idle** state: Initial state, transitions to **EngineOn** on `evTurnOn/ignitionOn()` and back to **idle** on `evTurnOff/ignitionOff()`.
- EngineOn** state: Contains two sub-states:
  - stationary**: Transitions to **moving** on `evAcc/accelerate()`.
  - moving**: Transitions back to **stationary** on `[0==speed]`.

Below the diagram, a code editor shows the following C++ code snippet:

```

127
128  /// statechart_method
129  virtual IOxFReactive::TakeEventStatus rootState_processEvent ();
130
131  // idle:
132  /// statechart_method
133  inline bool idle_IN() const;
134
    
```

Two red callout boxes with white text are present:

- A box pointing to the code editor with the text: **產生出程式碼** (Generate code).
- A box pointing to the statechart diagram with the text: **描述Car System的行為狀態** (Describe the behavior state of the Car System).

## 範例展示 - 執行模擬測試

於設計階段進行模擬測試

提供UI介面方便於測試

```

stateDiagram-v2
    state EngineOn {
        state idle
        state stationary
        state moving
        idle --> stationary : evTurnOn/ignitionOn()
        stationary --> idle : evTurnOff/ignitionOff()
        stationary --> moving : evAcc/accelerate()
        moving --> stationary : [0==speed]
    }
    
```

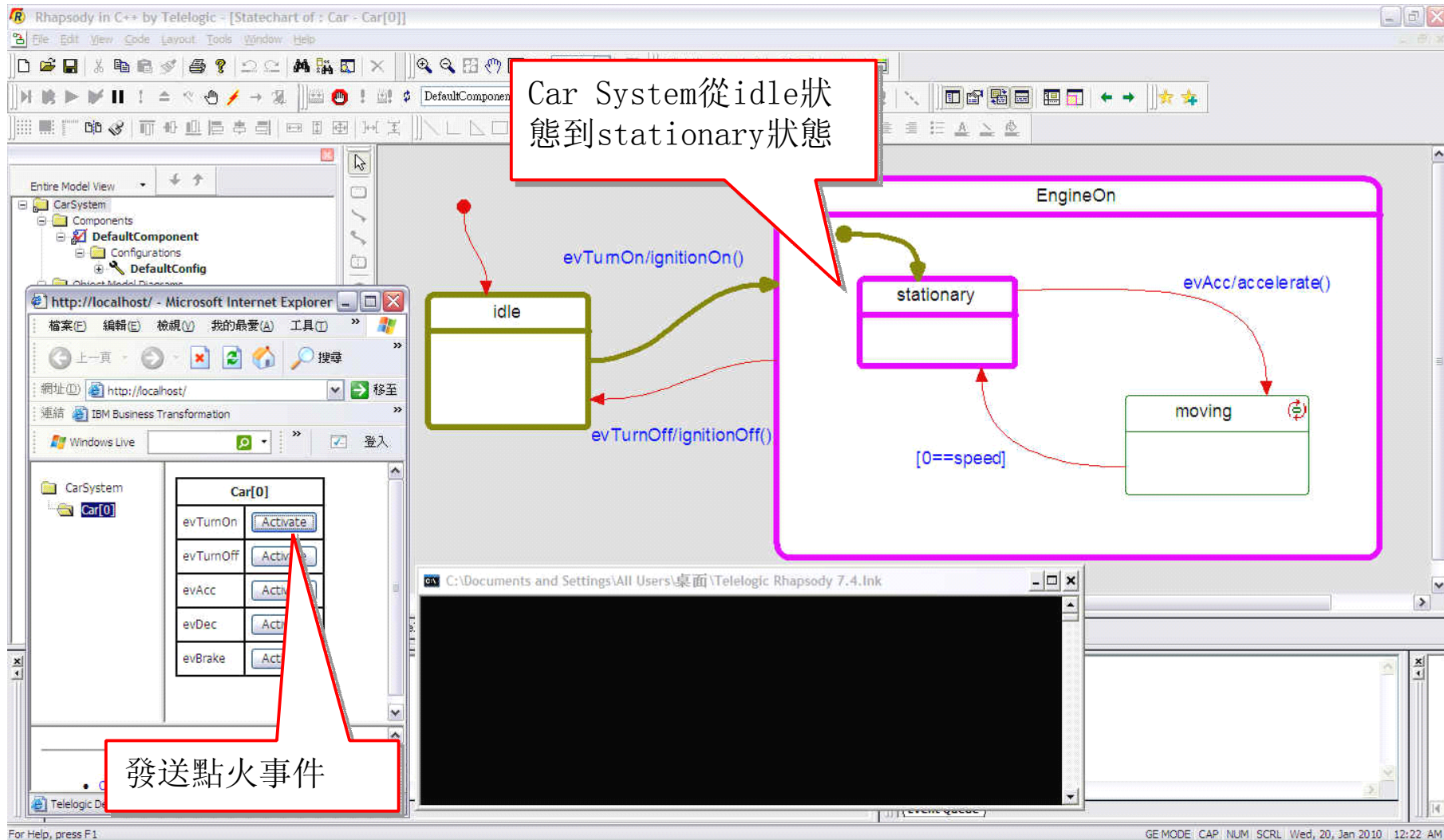
Car[0] UI Controls:

evTurnOn	Activate
evTurnOff	Activate
evAcc	Activate
evDec	Activate
evBrake	Activate

Console Output: [0] (Event Queue)

System Information: GE MODE | CAP | NUM | SCRL | Wed, 20, Jan 2010 | 12:21 AM

# 範例展示 - Car System Turn On



# 範例展示 - Car System 加速

The screenshot displays the Rhapsody in C++ interface for a statechart of a Car System. The statechart includes an `idle` state, an `EngineOn` state containing `stationary` and `moving` states, and a transition from `stationary` to `moving` triggered by the event `evAcc/accelerate()`. A guard condition `[0==speed]` is associated with this transition. The `idle` state has transitions to `EngineOn` (labeled `evTurnOn/ignitionOn()`) and back to `idle` (labeled `evTurnOff/ignitionOff()`). The `moving` state has a self-loop labeled `[0==speed]`.

In the background, a control panel for `Car[0]` is visible with buttons for `evTurnOn`, `evTurnOff`, `evAcc`, `evDec`, and `evBrake`. A red callout points to the `evAcc` button with the text "發送加速事件".

The console window at the bottom shows the following output:

```

The car is at speed : 1
The car is at speed : 2
The car is at speed : 3
The car is at speed : 4
The car is at speed : 5
The car is at speed : 6
The car is at speed : 7
The car is at speed : 8
    
```

A red callout points to this console output with the text "顯示Car System正在加速".

Another red callout points to the `moving` state in the statechart with the text "Car System從stationary狀態到moving狀態".

# 範例展示 - Car System 減速

The screenshot displays the Rhapsody in C++ interface for a Car System statechart. The statechart shows three states: 'idle', 'stationary', and 'moving'. The 'moving' state is highlighted with a red box. A transition from 'moving' to 'stationary' is labeled with the event 'evAcc/accelerate()' and a guard condition '[0==speed]'. A transition from 'stationary' to 'idle' is labeled with 'evTurnOff/ignitionOff()'. A transition from 'idle' to 'stationary' is labeled with 'evTurnOn/ignitionOn()'. The 'moving' state is also highlighted with a red box. A red callout box points to the 'moving' state with the text 'Car System仍處於moving狀態'. Another red callout box points to the console output with the text '顯示Car System正在減速'. A third red callout box points to the 'evDec' button in the control panel with the text '發送減速事件'. The console output shows the car's speed decreasing from 1 to 5.



# 範例展示 - Car System 煞車

The screenshot displays the Rhapsody in C++ interface for a statechart titled "Statechart of: Car - Car[0]". The statechart features three states: "idle", "stationary", and "moving".

- idle** (green box) transitions to **stationary** on the event `evTurnOn/ignitionOn()`.
- stationary** (purple box) transitions to **moving** on the event `evAcc/accelerate()`.
- moving** (olive box) transitions back to **stationary** when the condition `[0==speed]` is met.
- stationary** transitions back to **idle** on the event `evTurnOff/ignitionOff()`.

Annotations and UI elements:

- A red callout box points to the transition from **stationary** to **idle**, containing the text: "Car System從moving狀態回到stationary狀態".
- A red callout box points to the **stationary** state, containing the text: "顯示Car System已經煞車".
- A red callout box points to the "evBrake" button in the control panel, containing the text: "發送煞車事件".
- The control panel (bottom left) shows buttons for `evTurnOn`, `evTurnOff`, `evAcc`, `evDec`, and `evBrake`, each with an "Activate" button.
- The console window (bottom center) displays the following output:

```
The car is at speed : 1
The car is at speed : 2
The car is at speed : 3
The car is at speed : 4
The car is at speed : 5
The car is at speed : 6
The car is at speed : 7
The car is at speed : 8
The car is at speed : 7
The car is at speed : 6
The car is at speed : 5
The car is braked !!!
```

# 範例展示 - Car System Turn Off

The screenshot displays the Rhapsody in C++ interface for a statechart titled "Statechart of: Car - Car[0]". The statechart consists of an **idle** state, an **EngineOn** composite state, and a **moving** state. The **EngineOn** state contains two sub-states: **stationary** and **moving**. Transitions are as follows:
 

- idle** to **EngineOn**: triggered by `evTurnOn/ignitionOn()`.
- EngineOn** to **idle**: triggered by `evTurnOff/ignitionOff()`.
- stationary** to **moving**: triggered by `evAcc/accelerate()`.
- moving** to **stationary**: triggered by `[0==speed]`.

 A console window at the bottom shows the following output:
 

```

    The car is at speed : 1
    The car is at speed : 2
    The car is at speed : 3
    The car is at speed : 4
    The car is at speed : 5
    The car is at speed : 6
    The car is at speed : 7
    The car is at speed : 8
    The car is at speed : 7
    The car is at speed : 6
    The car is at speed : 5
    The car is braked !!!
    
```

 A control panel on the left shows a table for **Car[0]** with buttons for `evTurnOn`, `evTurnOff`, `evAcc`, `evDec`, and `evBrake`. The `evTurnOff` button is highlighted with a red box and labeled "發送熄火事件" (Send engine stop event). A red callout box points to the `evTurnOff/ignitionOff()` transition, stating "Car System從stationary狀態回到idle狀態" (Car System returns from stationary state to idle state).

# 範例展示 - 自動產生報表

The screenshot displays the Rhapsody in C++ interface. The main window shows a statechart for a car engine with states: idle, stationary, and moving. Transitions are labeled with events like evTumOn/ignitionOn(), evAcc/accelerate(), and =speed]. A ReporterPLUS Wizard dialog box is open, titled "ReporterPLUS Wizard : Select Task". The dialog asks "What would you like to do?" and lists several options: Generate HTML Page, Generate Microsoft PowerPoint Presentation, Generate Microsoft Word Document, Generate RTF File, and Generate Text File. A red arrow points from the dialog box to a text box containing the Chinese text "自動產生多種格式報表". The bottom of the window shows a code editor with C++ code for the statechart.

```
127
128  /// statechart_method
129  virtual IOxfReactive::TakeEventStatus r
130
131  // idle:
132  /// statechart_method
133  inline bool idle_IN() const;
134
```

自動產生多種格式報表

# 範例展示 - 自動產生報表目錄

report1.doc - Microsoft Word

Table of Contents

- 1. Use Case Diagram Information ..... 5
  - 1.1 Use Case Diagram name: Car System..... 5
- 2. Object Model Diagram Information ..... 5
  - 2.1 Object Model Diagram name: Model1 ..... 5
- 3. Components Information ..... 6
  - 3.1 Component Name:DefaultComponent ..... 6
    - 3.1.1 Configuration information for Component: DefaultComponent ..... 6
      - 3.1.1.1 Test Configuration ..... 6
    - 3.1.2 File information for Component: DefaultComponent ..... 6
      - 3.1.2.1 Files ..... 6
- 4. Package Information ..... 6
  - 4.1 Package: Default ..... 6
    - 4.1.1 Class Information for Package: Default ..... 6
      - 4.1.1.1 Class name: Car ..... 6
        - 4.1.1.1.1 Attribute Information for Class: Car ..... 6
          - 4.1.1.1.1.1 Attribute Name: speed ..... 7
        - 4.1.1.1.2 Operation information for Class: Car ..... 7
          - 4.1.1.1.2.1 Operation name: ignitionOn ..... 7
          - 4.1.1.1.2.2 Operation name: ignitionOff ..... 7
          - 4.1.1.1.2.3 Operation name: accelerate ..... 7
          - 4.1.1.1.2.4 Operation name: decelerate ..... 8
          - 4.1.1.1.2.5 Operation name: brake ..... 8
        - 4.1.1.1.3 EventReception information for Class: Car ..... 8
          - 4.1.1.1.3.1 Event Reception name: evTurnOn ..... 8
          - 4.1.1.1.3.2 Event Reception name: evTurnOff ..... 8
          - 4.1.1.1.3.3 Event Reception name: evAcc ..... 8
          - 4.1.1.1.3.4 Event Reception name: evDec ..... 8
          - 4.1.1.1.3.5 Event Reception name: evBrake ..... 8
        - 4.1.1.1.4 Statechart information for Class: Car ..... 9

自動產生報表完整目錄

# 範例展示 - 自動產生報表內容

The screenshot shows a Microsoft Word document titled 'report1.doc'. The document content is a report generated from Rational software, detailing statechart information for a class named 'Car'. The report is structured as follows:

- 3.1.1.1.4 Statechart information for Class: **Car**
  - 3.1.1.1.4.1 State information
    - 3.1.1.1.4.1.1 State: ROOT
      - 3.1.1.1.4.1.1.1 Default Transition information for State ROOT
      - 3.1.1.1.4.1.1.2 Incoming Transition information for State ROOT
      - 3.1.1.1.4.1.1.3 Outgoing Transition information for State ROOT
      - 3.1.1.1.4.1.1.4 State information
        - 3.1.1.1.4.1.1.4.1 State: idle
        - 3.1.1.1.4.1.1.4.2 State: EngineOn
          - 3.1.1.1.4.1.1.4.3 State information
            - 3.1.1.1.4.1.1.4.4 State: stationary
            - 3.1.1.1.4.1.1.4.5 State: moving
- 3.1.2 Event information for Package Default:
  - 3.1.2.1 Event name: evTurnOn
  - 3.1.2.2 Event name: evTurnOff
  - 3.1.2.3 Event name: evAcc
  - 3.1.2.4 Event name: evDec
  - 3.1.2.5 Event name: evBrake
- 3.2 Package: PredefinedTypes:
  - 3.2.1 Type information for Package PredefinedTypes:
    - 3.2.1.1 Type name: RhpInteger
    - 3.2.1.2 Type name: RhpCharacter
    - 3.2.1.3 Type name: RhpString
    - 3.2.1.4 Type name: RhpReal
    - 3.2.1.5 Type name: RhpVoid
    - 3.2.1.6 Type name: RhpPositive
    - 3.2.1.7 Type name: RhpAddress
    - 3.2.1.8 Type name: RhpBoolean
    - 3.2.1.9 Type name: RhpEnumeration
  - 3.2.2 Stereotype information for Package PredefinedTypes:
    - 3.2.2.1 Stereotype name: Component
    - 3.2.2.2 Stereotype name: EquipmentConfiguration
    - 3.2.2.3 Stereotype name: Interface
    - 3.2.2.4 Stereotype name: Usage
    - 3.2.2.5 Stereotype name: Send
    - 3.2.2.6 Stereotype name: Resource
    - 3.2.2.7 Stereotype name: Singleton
    - 3.2.2.8 Stereotype name: MessageQueue
    - 3.2.2.9 Stereotype name: Timer
    - 3.2.2.10 Stereotype name: Semaphore
    - 3.2.2.11 Stereotype name: Mutex
    - 3.2.2.12 Stereotype name: EventFlow

The report also includes a statechart diagram for the 'EngineOn' state. The diagram shows the following states and transitions:

- idle** state:
  - Transition to **EngineOn** state triggered by event `evTurnOn/ignitionOn()`.
  - Transition back to **idle** state triggered by event `evTurnOff(ignitionOff)`.
- EngineOn** state (containing sub-states):
  - stationary** state:
    - Transition to **moving** state triggered by event `evAcc/accelerate()`.
  - moving** state:
    - Transition back to **stationary** state triggered by guard `[0==speed]`.

A red callout box with the text '自動產生報表內容與圖表' (Automatically generate report content and diagrams) points to the statechart diagram.

## 參考資料

- 下載Rhapsody產品簡介：Rhapsody brochure.pdf
- 下載Car System範例展示：Demo.zip
- 產品電子型錄：<http://www-142.ibm.com/software/products/tw/zh/ratirhap>
- 試用版：[https://www-01.ibm.com/software/tw/trials/reg\\_rhapsody.html](https://www-01.ibm.com/software/tw/trials/reg_rhapsody.html)





**Learn more at:**

- [IBM Rational software](#)
- [IBM Rational Software Delivery Platform](#)
- [Accelerate change and delivery](#)
- [Deliver enduring quality](#)
- [Enable enterprise modernization](#)
- [Ensure Web site security and compliance](#)
- [Improve project success](#)
- [Manage architecture](#)
- [Manage evolving requirements](#)
- [Small and midsized business](#)
- [Targeted solutions](#)
- [Rational trial downloads](#)
- [developerWorks Rational](#)
- [Leading Innovation](#)
- [IBM Rational TV](#)
- [IBM Business Partners](#)
- [IBM Rational Case Studies](#)

© Copyright IBM Corporation 2008. All rights reserved. The information contained in these materials is provided for informational purposes only, and is provided AS IS without warranty of any kind, express or implied. IBM shall not be responsible for any damages arising out of the use of, or otherwise related to, these materials. Nothing contained in these materials is intended to, nor shall have the effect of, creating any warranties or representations from IBM or its suppliers or licensors, or altering the terms and conditions of the applicable license agreement governing the use of IBM software. References in these materials to IBM products, programs, or services do not imply that they will be available in all countries in which IBM operates. Product release dates and/or capabilities referenced in these materials may change at any time at IBM's sole discretion based on market opportunities or other factors, and are not intended to be a commitment to future product or feature availability in any way. IBM, the IBM logo, Rational, the Rational logo, Telelogic, the Telelogic logo, and other IBM products and services are trademarks of the International Business Machines Corporation, in the United States, other countries or both. Other company, product, or service names may be trademarks or service marks of others.



