Innovate2011

The Premier Software and Product Delivery Event

Software. Everyware.

How Samsung Applies Model Based Design, Simulation and Verification for Smart Home Appliances

Min Kwang Lee

S/W Engineer, Samsung Electronics

Minkwang.lee@samsung.com

Agenda

Overview of Samsung Electronics

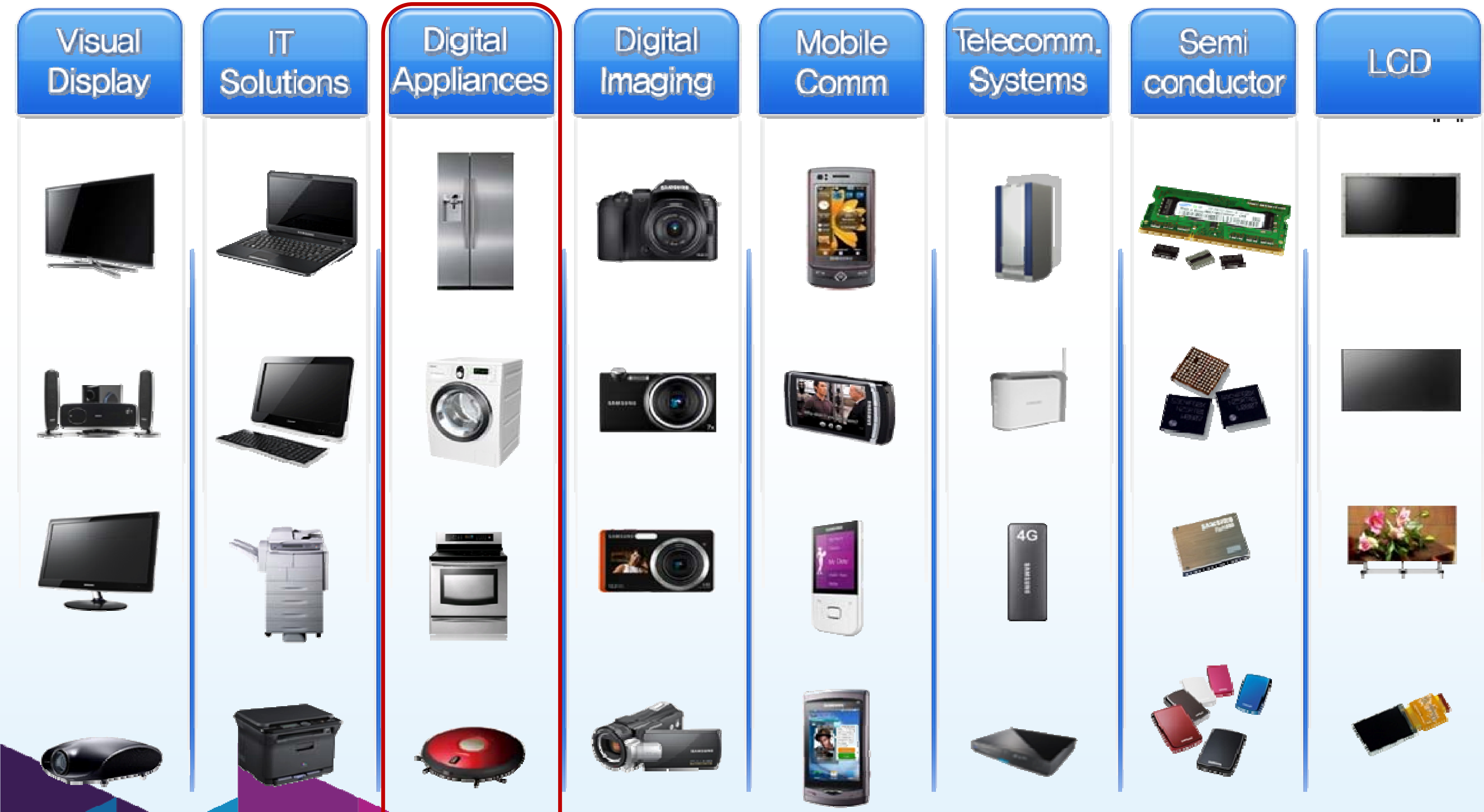
Background

Case Study

Demo

Wrap Up

Samsung Electronics – Product Portfolio



- Visual Display
- IT Solutions
- Digital Appliances
- Digital Imaging
- Mobile Comm
- Telecomm. Systems
- Semi conductor
- LCD

About Samsung Electronics
- Digital Appliances Division

Provide convenient solutions for everyday lives.

- Refrigerator
- Washing Machine
- Air-Conditioner
- Cooking Appliances
- Vacuum Cleaner

Highly popular in U.S. market

- Ranked No.1 in market share of French door refrigerator
  - 36.7% (2010), *Source: NPD 2010
- Ranked No.1 in market share of Drum W/M
  - 19.3% (2H, 2010), *Source: NPD 2010
- Top brand power of home appliances
  *Source: J.D Power 2010

Agenda

Overview of Samsung Electronics

Background

Case Study

Demo

Wrap Up

The Problem Domain – Drum Washing Machine

Small rom size

Two Microcontrollers (Main controller + UI Panel)

Single task(No operating system)

C language

Complicated requirements

- Based on progress table for each course
- Dynamic response to changes(ex. Temperature, water level)
- Need to handle to unpredictable events(ex. Key press, door open, power down)
- But, Suitable for state behavior modeling

Frequent change of requirements

Lots of derived models

Background – S/W Design Phase Challenges

Hard to maintain consistency between design and source-code

Need a way to validate s/w design after frequent change

Target test takes too long

Hard to set test condition for test case

Application Layer

- Low reusability of source code
- Need structural design based on commonality and variability

Use several software design tools

Lack of unified representation

Need a paradigm shift from conventional code-centric to model driven development

Agenda

Overview of Samsung Electronics

Background

Case Study

Demo

Wrap Up

Samsung’s Approach

Objective

- *Productivity & Quality Improvement by Validating Model and Using Code Generation*

Select application layer as modeling scope

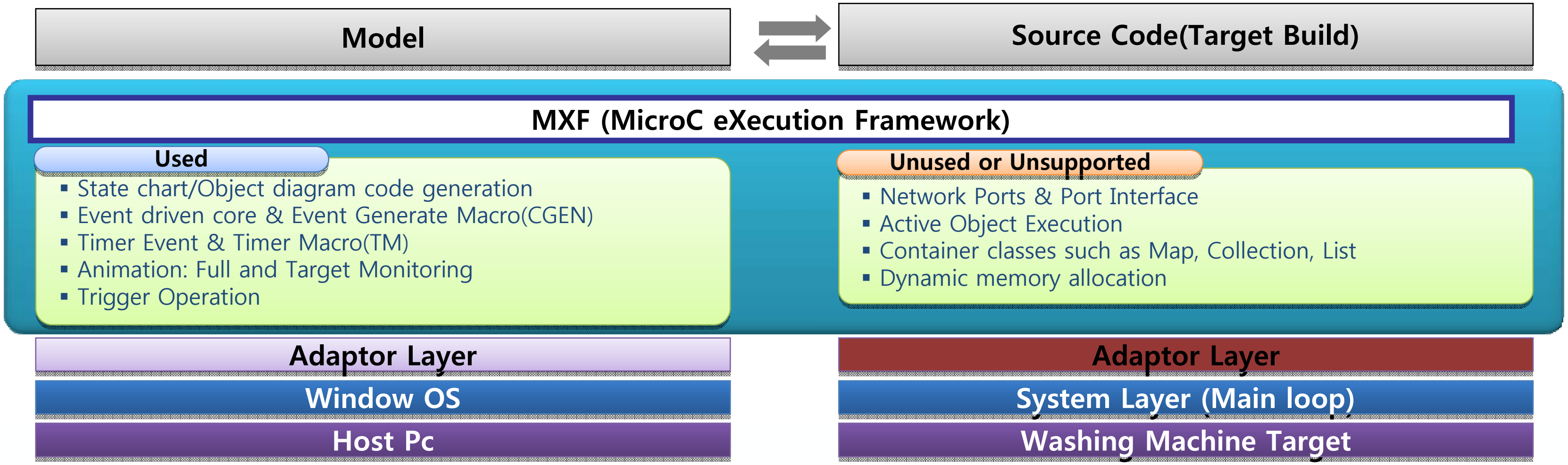
Identify commonality and variability based on feature modeling

Port MicroC framework

Use IBM Rational Rhapsody for application layer modeling

- Focus on state chart modeling of file class
- Set up development environment to validate models on host pc and target
- Use auto generated code without any further modification
- Conduct feasibility study first
- Minimize reverse engineering of legacy code

IBM Rational Rhapsody Framework



Model

Source Code(Target Build)

MXF (MicroC eXecution Framework)

Used

- State chart/Object diagram code generation
- Event driven core & Event Generate Macro(CGEN)
- Timer Event & Timer Macro(TM)
- Animation: Full and Target Monitoring
- Trigger Operation

Unused or Unsupported

- Network Ports & Port Interface
- Active Object Execution
- Container classes such as Map, Collection, List
- Dynamic memory allocation

Adaptor Layer

Window OS

Host Pc

Adaptor Layer

System Layer (Main loop)

Washing Machine Target

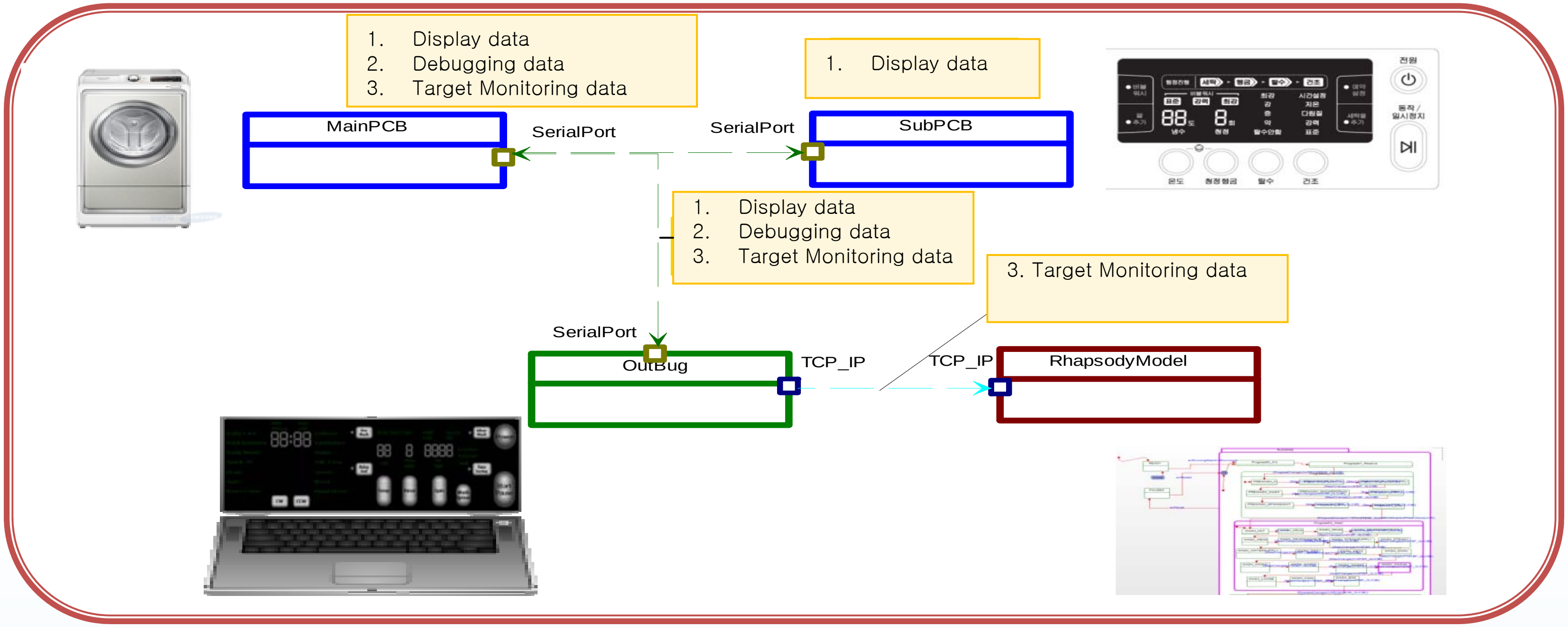
MicroC eXecution Framework (MXF)

- Optimized for embedded systems
- No-OS (mainloop) adaptors

Issues Encountered during the framework porting phase

- Rhapsody task should be included in Main task of washing machine system
  - Had to eliminate “while(1)” in RiCOSMainTask mainloop
- Had to minimize Framework size (11 K)

Environment for Target Monitoring



1. Display data
2. Debugging data
3. Target Monitoring data

MainPCB

SerialPort

SerialPort

SubPCB

1. Display data

1. Display data
2. Debugging data
3. Target Monitoring data

3. Target Monitoring data

SerialPort

OutBug

TCP_IP

TCP_IP

RhapsodyModel

Issue

- Limited number of serial port
  - Share the port by protocol definition (header + length + data + checksum)
  - Retransmit target monitoring data from Host Pc using TCP/IP

Structural modeling

Feature Modeling

- Analysis Mandatory/Alternative/Optional Feature
- Variation point

Create feature table

Create variation point table

Object identification

Object interaction

State Chart Modeling – Washing Machine Behavior

More than 180 states indentified for WM Behavior

- Status(Ready, Running, Pause, Etc)
- All of the WM's progresses and steps

Issues Encountered during state chart modeling phase

- Guard condition is checked when event occurs only
- Need to check a guard condition more frequently or at a more regular interval than whenever an event occurs
- Had to create a polling mechanism

State Chart Modeling – Example

Example of the software requirement specification

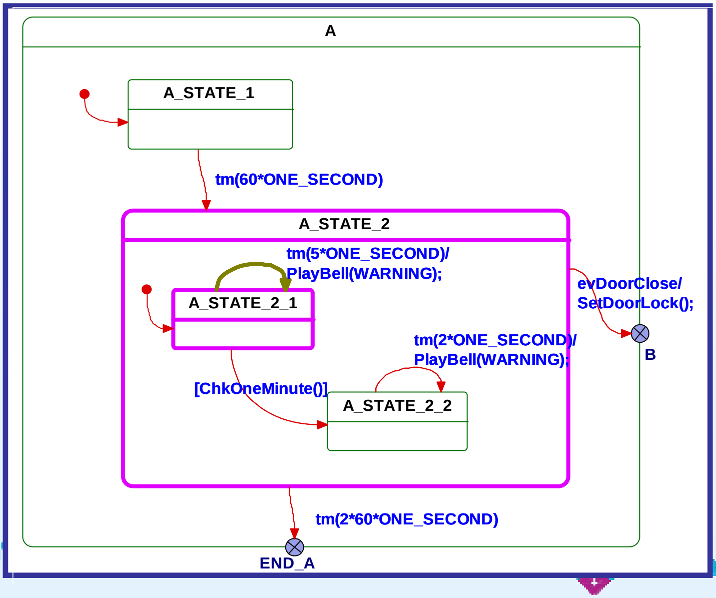
- 3.1.01 …
- 3.1.32. If door is still in open condition after one minute has elapsed since A mode entry, warning beep would be played at every five seconds for following one minute, then at every 2 seconds for next 1 minute. If door is closed during 2 minutes, door lock would be set and B mode would be started
- *There are lots of statements excluding progress table*

From conventional code-centric to state chart modeling

```
…
if(…)
          if(…)
          else(…)
else(…)
           if(timeflag)
          timeflagcount++
switch(…)
          case :
…
```

No longer struggle with convoluted if-else statement and time flags

Easy, fast and fun



Model Validation

Model should be validated

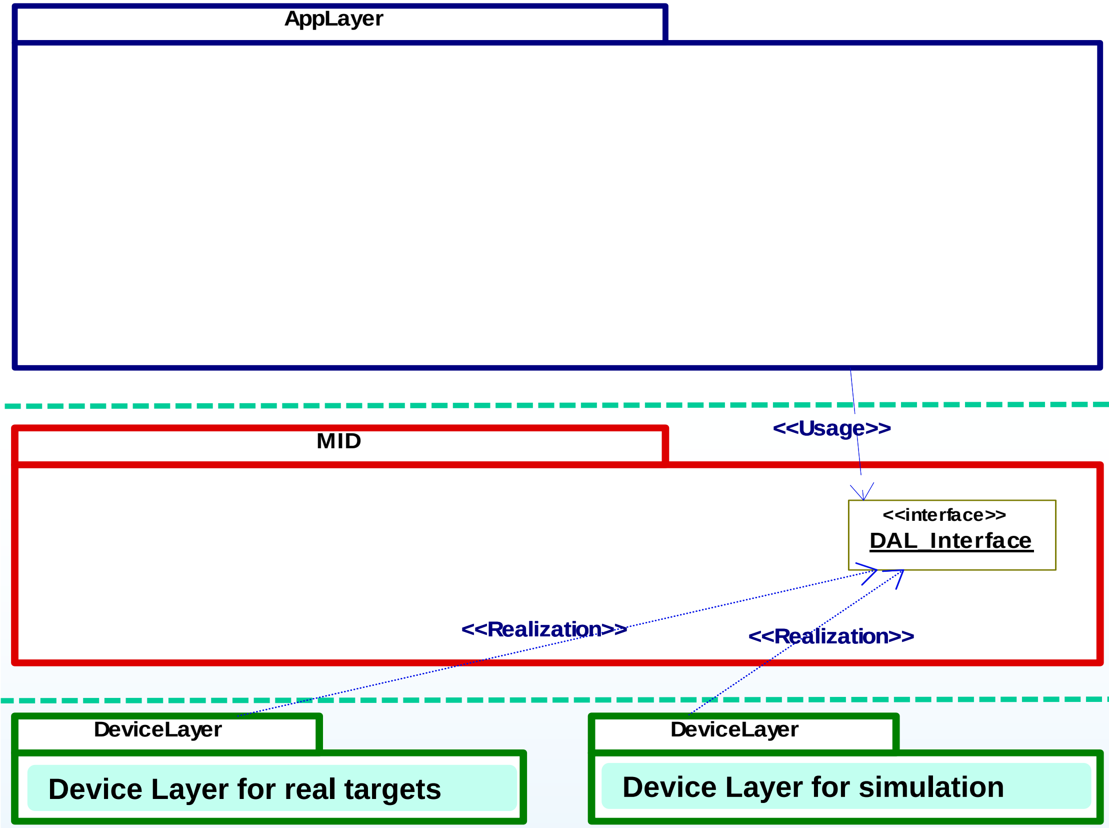
Conditions for model validation

- Model should be built
- Model should be executable
- Event could be generated
- Virtual device could be controlled

How to validate the Washing Machine Model

- Device layer for simulation
- UI prototype using IBM Rational Rhapsody’s UI panel diagram

Model Validation – Device layer for Simulation



Application Layer

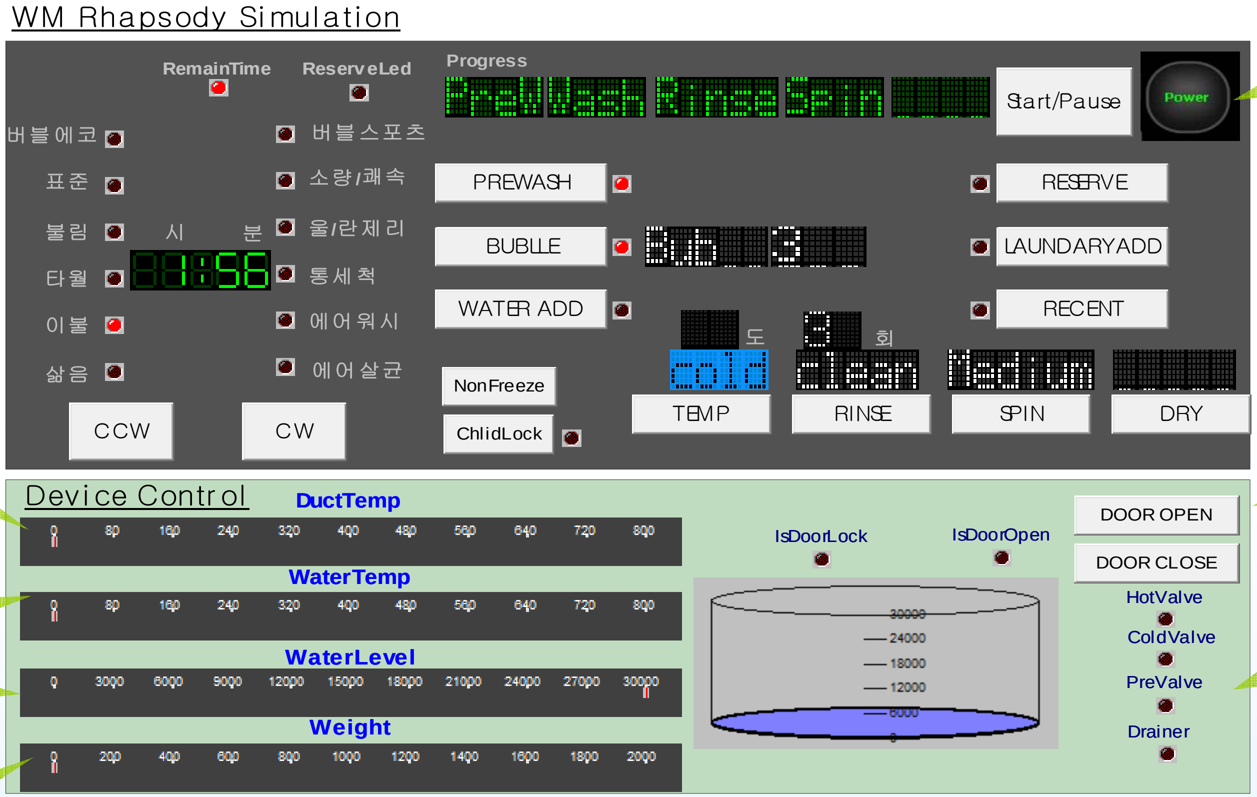
Device Abstraction Layer hides the implementation and device dependent details by providing an abstract interface

Device layer implementation for real targets or UI prototype

Real SETs

UI Prototype

Model Validation – UI Prototype



UI PANEL

Control internal temperature

Control water temperature

Control water level

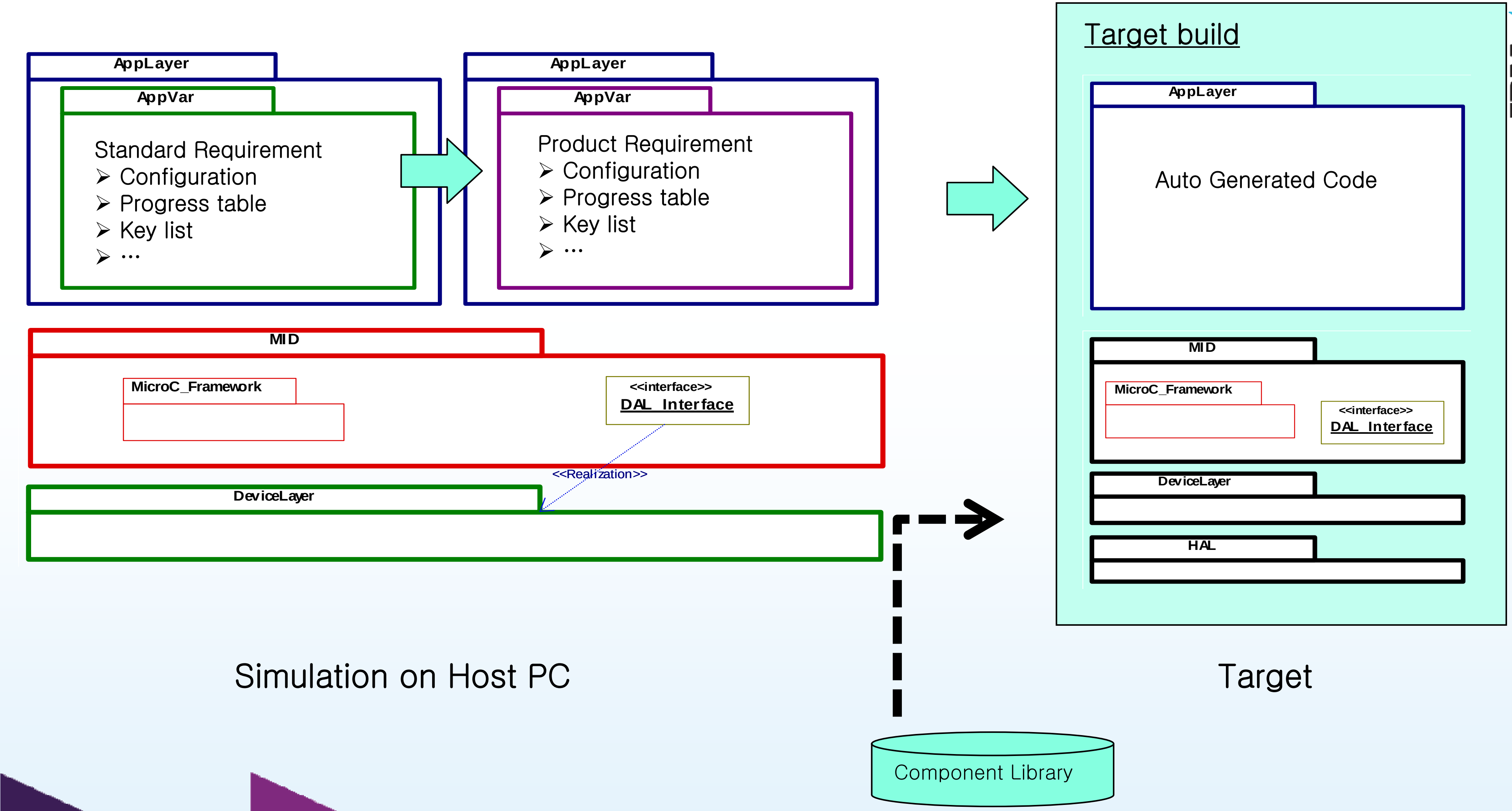
Control weight

Control door

Display valve and drainer condition

UI for user panel and device control(including sensors) for simulation

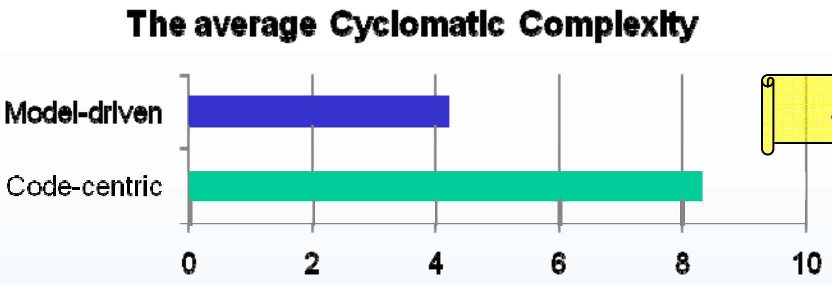
Transform Model to Target



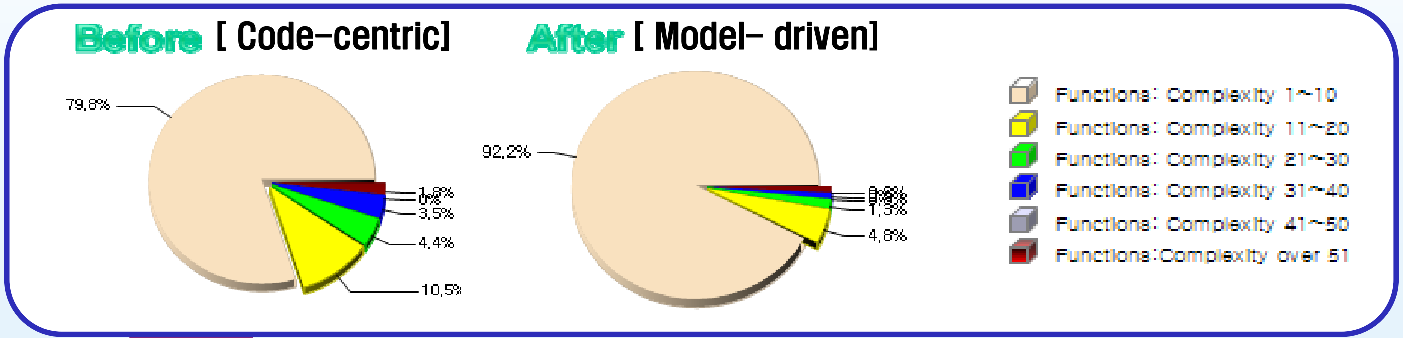
Comparison Auto generated code with legacy code

Cyclomatic Complexity

- Measure the control flow complexity of a program
- Cyclomatic Complexity is related to understandability and maintainability.
- Recommended average value of cyclo. complexities : Less than or equal to 5

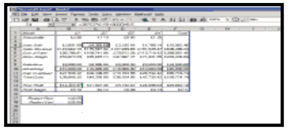


49% improvement

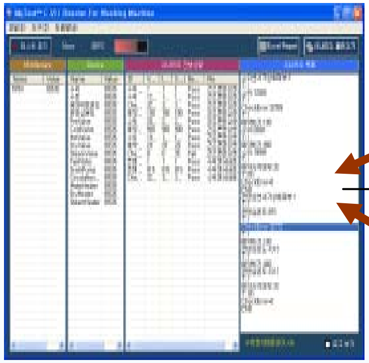


Environment for Automated Test

Excel File–test case

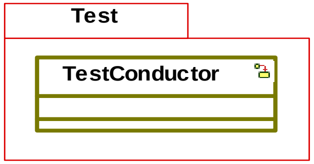


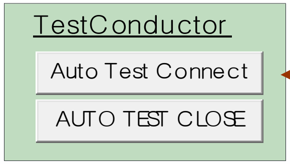
Autotest Director



Internal TCP/IP

Test conductor

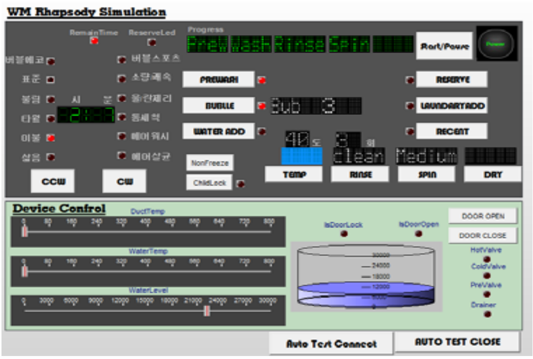




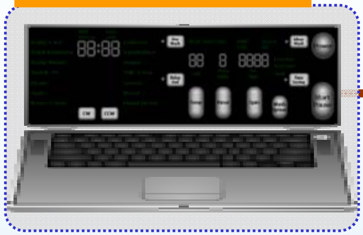
Dependency – Event Generate

Dependency – UI Result

Simulator



Outbug



Serial TX/RX

Real Target

Excel File – Test Report

Agenda

Overview of Samsung Electronics

Background

Case Study

Demo

Wrap Up

www.ibm/software/rational

Agenda

Overview of Samsung Electronics

Background

Case Study

Demo

Wrap Up

Results & Beneficial Effects

Set up IBM Rational Rhapsody development environment for washing machine software

- Framework porting
- Target monitoring
- UI prototype and simulation
- Automated test

Models which meet requirements have been designed, validated on simulation mode and real target

IBM Rational Rhapsody generated 70% of the total code

Consistency between design and implementation

Rhapsody has significantly improved productivity

Results & Beneficial Effects (cont.)

Applications on IBM Rational Rhapsody MicroC Framework

- Reduce complexity
  - We replaced flag based complex conventional timer with framework ‘s simple tm() macro
  - We substituted complex if-else statements with event-driven state charts which enhanced readability and visibility

Application Layer Simulation with fast prototyping

- Animated debugging
- Ability to run the model on the host PC, then test it and debug it logically without real target
- Simulated time with ONE_SECOND macro could make simulation faster than real environments
- Test condition could be set very easily and fast

Lessons Learned

Minimize reverse engineering

Play ping pong first

- Port the framework to your real target

Need a deep understanding regarding many properties of rhapsody

- particularly properties related to code generation

Rhapsody is not a magic

Training is essential

QUESTIONS

www.ibm/software/rational

www.ibm/software/rational

Software. Everyware.