Model–Driven Software Engineering — Promises and Challenges

Zhiming Liu

Center for Software Engineering
Faculty of Technology, Engineering and Environment
Birmingham City University

zhiming.liu@bcu.ac.uk

Acknowledgements:

Jonathan Bowen, He Jifeng, Esther Palomar
Software Engineering & Crisis

• Inherent Complexity of Software - no silver bullet [Fred Brooks 1986]

  1) Application domains are complex [Requirement Analysis]
  2) Software offers much flexibility [Design]
  3) The development process is still changing [Management]
  4) The behavior of a software system is hard to understand [V&V]

• Complexity is caused by the growing power of machines

  “The major cause of the software crisis is that the machines have become several orders of magnitude more powerful! …”

  --- Edsger Dijkstra
Complex Systems Prone to Failures

Nuisances caused by software bugs
Ariane-5 Explosion in 1996

Caused by an exception in software
Cost about 300 million Euro
WASHINGTON - After studying Mars four times as long as originally planned, NASA's Mars Global Surveyor orbiter appears to have succumbed to battery failure caused by a complex sequence of events involving the onboard computer memory and ground commands. ….
Chinese High Speed Train Accident 2011

Signal Flow Fault
Large Scale Software Projects Fail

"...software debacles are routine. And the more ambitious the project, the higher the odds of disappointment.” [Software Hell, Carr 05]

FBI
- Since 2001: a database on suspected terrorists
- Jan. 2005: $M170, “not even close to having a working system”

Ford Motors
- Since 2000, project “Everest”: buying supplies, replacing legacy
- Aug. 2004: $M200 over budget, abandoned (β-version slower than legacy)

McDonald’s
- Since 1999, project “Innovate”, budget $M1,000
- Killed in 2002, writing off $M170
Software Engineering Today

- **Increasing complexity of modern software**
  1) Software are running every every, and any time
  2) Models of different views of system data and services (model transformations)
  3) Integration of services to support collaborative workflows
  4) More and more software becomes *safety & security critical*

- **Formal methods are essential for**
  1) **handling complexity** through abstraction, separation of concerns and divide and conquer, as well as for
  2) **provably correct system design**
A Scenario in Internet of Things (1)

• **Street lightening**
  1) City council’s view: lighting of streets
  2) Electricity company’s view: readings on meters and/or bills
  3) Police’s view: in relation with crimes

• **Design to meet the requirements of all stakeholders?**

  Separation and integration of different views
A Scenario in Internet of Things (2)

• **Design challenges**

1) How to derive an engineering/system model from the different views of users and vice versa?

2) How to reason or validate the system model against requirements stated in the users’ view models?

3) How to design a system to support dynamic addition of support to users with different views?

**Model integration and transformation**
Future Electronic Health Records

Source: Electronic Health Records Overview, @MITR 2006
rCOS

Problems

• Dynamic integration of new components and legacy components – plug & play
• Interfaces for interoperable integration of heterogeneous components
• Specification purpose of integration – models of workflows

Objectives

• Unifying semantics & theories of programming (UTP)
• Models, analysis, verification and simulation
• System architecture modeling, composition, refinement, evolution
• Language and tool support for design and integration

Putting theories, methods and tools together in evolution Processes!
rCOS – Integrating Models

Build *system models* to gain confidence in requirements and designs

- **Use abstraction** for information hiding
  - well-focused
  - problem-oriented

- **Use decomposition and separation of concerns**
  - divide and conquer
  - incremental development

- **use rigor/formalization**
  - repeatable process
  - analyzable artifacts

Basis for Tool Support
Architectural Components

• Services, including devices realizing functions are components
  
  **Component M** {
    Z d;
    provided interface MIF {
      W(Z v) { d:=v }; R(;v) { v:=d }; }
  }

• A process that controls and coordinate interactions
  
  **Component C** {
    Bool w = 1;
    Provided interface CIF {
      W(){(w:= not w)}; R(){not w&(skip) // guarded actions }
    }
  }

  **Component C1** {
    Bool w =1;
    Provided InterfaceC1IF {
      W(){w&(w:=not w)}; R(){not w&(w:=not w)}
    }
  }
More General Component

```plaintext
component fM {
    Z d;
    provided interface MIF {
        W(Z v) { d:=v };
        R(;v) { v:=d };
        protocol { ?W({?W,?R}) // protocol of C, realized by guards }
    }
    actions { //fault modeling corruption
        fault {true|- d’ < > d }
    }
}

fMi=fM[fMi.W/W,fMi.R/R], i=1,2,3, renaming as a built-in connector
```
Component A { /* an appliance 
  rate: [Time \rightarrow Real];
  status: \{on, off\};
  provided interface {
    rate {* signal: given by manufacture};
    switch() {* operation: switch A on and off}
  }
}

Component M { /* meter 
  val: [time \rightarrow Real];
  provided interface {
    read(;r){true |- r'=val};
  }
  required interface rate { /* signal 
    val= energy(rate)
  }
}

Composition: H = A||M
System Evolution for Home Automation (a)

- Add provided signal ‘val’ to M
- Add a control pad P that requires signal ‘val’, provides `set()`, and calls A.switch, etc. 
  \[ M' = P \parallel M, \quad H' = M' \parallel A \]
- Refine P with planning with daily budget, and schedule functionality 
  \[ M'' = P' \parallel M, \quad H'' = M'' \parallel A \]
System Evolution for Home Automation (b)

\[ A_i = A[\text{switch}_i/\text{switch}, \text{rate}_i/\text{rate}], \ A = A_1 || ... || A_m \]
\[ M_i = M[\text{read}_i/\text{read}, \text{val}_i/\text{val}], \ M = M_1 || ... || M_m \]
\[ P_i = P[...], \ P = P_1 || ... || P_m \]
• Add a global controller for planning and schedule
  \( H = G \| P \| M \| A \)
• Control C with mobile phone from car or office
Network Evolution

- Consider $k$ households $H_j$, each with its own budget
  \[ H = H_1 \parallel H_2 \parallel ... \parallel H_k \]

- Consider Coordinator, interacting the households to coordinate their budgets, based on the interaction with one utility company Utility
  \[ N = H \parallel Coordinator \parallel Utility \]

- Evolve to a network $U$ of utility companies
  \[ N = H \parallel Coordinator \parallel U \]

- Implement $H \parallel Coordinator$ by distributing the coordination activities among the households themselves

Note individual households evolve in parallel, and in parallel with the network with more households and companies.
Conclusions (1)

- FMDE promises to deal with system complexity;
- Interface models are essential for
  - integration of heterogeneous components, and
  - the domain linkages in interdisciplinary collaboration;
Conclusions (2)

• Architecture models are essential for
  ▪ correct and security by design,
  ▪ Analysis and checking of safety vulnerabilities and security threats, as to,
  ▪ make architecture decisions for different concerns

• rCOS provides a method of modeling and design
  – with a tool support driven by model transformations,
  – Underpinned by a unified semantic theory,
  – but only for digital component systems.
To Do ...

• rCOS needs to
  – include CPS Component and interfaces, and
  – the challenge is the unified semantic theory with tools support.

• How to combine the “Four Variable Models” and “Problem Frames Model”, systematically into the continuous evolutionary integration system development process?

• Model-driven approach is again promising