Model-Driven Development of High-Assurance Dynamically Adaptive Systems

Betty H.C. Cheng,

Software Engineering and Network Systems Lab
Digital Evolution Laboratory
BEACON: NSF Center for Evolution in Action
Department of Computer Science and Engineering
Michigan State University
chengb at cse dot msu dot edu
http://www.cse.msu.edu/~chengb

Acknowledgments

Current and Former Students:
- Andres Ramirez
- Adam Jensen
- Chad Byers
- Ji Zhang
- Masoud Sadjadi
- Heather Goldsby
- Ben Beckmann
- David Knoester

Sponsors
- U.S. Army Research Office
- U.S. Department of the Navy
- National Science Foundation
- Michigan State University

Additional Collaborators
- Philip McKinley (CSE)
- Charles Ofria (CSE)
- Xiaobo Tan (ECE)
- Raza Haque (Medicine)

Industrial Collaborators:
- Ford Motor Research
- General Motors
- HP
- Siemens
- Motorola
- Continental Automotive

Sponsors
High-Assurance Autonomic Computing

- **Autonomic computing**: Promises self-managed and long-running systems with limited human guidance.
- Systems must continue to **operate correctly** during exceptional situations, upgrades, and evolution under **uncertain** conditions.
- Need for assurance
  - hardware component failures
  - network outages
  - software faults
  - security attacks

*ONR CIP RAPIDware project revealed many open challenges for current and future systems*

RAPIDware Project

- Funded by U.S Office of Naval Research
  - Adaptable Software / Critical Infrastructure Protection Program
  - 2001-2007
- High-assurance software adaptation technologies for:
  - Detecting and responding to environmental changes
  - Strengthening self-auditing capabilities of “always-on” systems
- Autonomic cyberinfrastructure for:
  - crisis management and public safety systems
  - military command and control
  - environmental monitoring
  - management of large industrial installations
- Enable systems to operate **through** failures and attacks
RAPIDware Focus Areas

- **Mechanisms for software adaptation**
  - Language extensions, middleware tools
    [Wear02, IEEC04, ICAC04, DOA04, DARE04, DEAS05, SPE06,]
  - Decision-making [ICDL04, TKDE07]
  - Migration path for existing software
    [WOSS02, Auton06, MiSE07]

- **Assurance in adaptation**
  - Requirements engineering
    [REFSQ05, SEAMS06, REV07]
  - Formal Analysis [WADS05, JSS06, ISSE07]
  - Design technologies [ICSE06]
  - Run-time support [WADS04, M@RT07, MiSE07]

- **Adaptive cyberinfrastructure**
  - Adaptive group security
  - Extension to sensor networks
  - "Sensor to server" coordination

RAPIDware Approach

- Gain assurance in adaptive software through
  - A systematic software development process
  - Application of formal methods at every stage

  - **Requirements**
    Specify adaptation properties, including local, global, and transitional properties

  - **Design**
    Create design models and model check against adaptation requirements

  - **Implementation**
    Develop adaptive system that ensures consistency before, during, and after adaptation
System Goal

Goal

Requirements

Design models

Implementation

Goal modeling of systems (e.g., KAOS, i*, use cases)

Goal Refinement

Goal

Requirements

Design models

Implementation

- Refine goals to system requirements
- Different requirements for different execution environments
- e.g.: [REFSQ05, SEAMS06, REV07]
Each system has *functional/non-functional properties*:
- Properties *during adaptation*
- Properties that should *always* hold

- e.g.: [WADS04, ADS05, EAS06, ISSE07]

---

What *system designs* can be used to realize the system requirements:
- Still need to satisfy functional/non-functional properties
- Separate *functional logic* from *adaptive logic*
- e.g.: [ICSE06, ModVerif-AOSD09]
Implementation

Goal

- Introduce adaptation capabilities into legacy systems (non-invasively)
- Still need to satisfy functional/non-functional properties
- How do you achieve adaptation safely?
  - e.g.: [WADS05, JSS06, MiSE07, M@RT07]

Requirements

Design models

Implementation

Emerging Area

- SEI conducted study on the need for research and technology for Ultra-Large Scale software systems
  (report by SEI, 2006: L. Northrup et al)
  http://www.sei.cmu.edu/uls/
- Objective: Support Future Information Processing
  - Superior collection, fusion, analysis, and use of information to meet future computational objectives
- Requires: increasingly complex systems
  - Thousands of platforms, sensors, decision nodes, complex systems connected through heterogeneous wired and wireless networks.
- New scale:
  - Data stored, accessed, manipulated, and refined
  - Number of connections/interdependencies among components
  - Number of hardware elements
New Scale
Ultra-Large Scale SW-Intensive Systems

Healthcare Infrastructure

New Scale
High-Assurance Cyberphysical Systems

Intelligent Transportation and Vehicle Systems
High-level Objective:
– How to design a safe adaptive system with incomplete information and evolving environmental conditions

Challenges:
• Execution environment
  – How to model environment
  – How to effectively monitor changing conditions
  – Adaptive monitoring
• Decision-making for dynamic adaptation
  – Decentralized control
  – Assurance guarantees (functional and non-functional constraints)
• Adaptation mechanisms:
  – Application level
  – Middleware level

The ULS Ecosystem
• Key elements:
  – Computing devices
  – Business and organizational policies
  – Environment (including people)
• Forces:
  – Competition for resources
  – Unexpected environmental changes
  – Decentralized control
  – Demand for assurance
uncertainty is an inherent problem with ULS systems...

Dynamic Adaptation and Autonomic Computing are enabling technologies to help systems evolve to respond to changing conditions...

But what (combinations of) conditions should be considered?

Motivation

• To develop an autonomic system:
  ▶ Identify target systems
    - Non-adaptive systems
    - i.e., Business logic
  ▶ Adaptations among steady-state systems
    - Adapt from source to target system

Viable Target System
Motivation

- To develop an autonomic system:
  - Identify target systems
    - Non-adaptive systems
    - i.e., Business logic
  - Adaptations among steady-state systems
    - Adapt from source to target system
  - Model multiple viable target systems using UML (defacto standard)

UML Class & State Diagrams

Use UML to Model Structure

- UML Class Diagram
  - Identifies the system elements (classes)
    - E.g., ProcessingComponent, TempSensor, Environment
  - Describes the relationships between elements (classes)
    - E.g., ProcessingComponent reads info from TempSensor and TempSensor monitors Environment

Class Diagram
Use UML to Model Behavior

- UML State Diagram
  - Describes the behavior of an element (class)
  - One state diagram per element (class) on the class diagram
    - E.g., 3 state diagrams -- one for ProcessingComponent, one for TempSensor, and one for Environment

Objective: Automatic Generation of Behavior Models

- UML State Diagram
  - States:
    - E.g., Idle, RequestTemperatureData, ProcessTemperatureData
  - Transitions - triggers [guards] / actions
    - E.g., checkTemp() / TempSensor.getTempData(), setTempData(temp_data) /
Motivation

• To develop an autonomic system:
  ▶ Identify target systems
    - Non-adaptive systems
    - i.e., Business logic
  ▶ Adaptations among steady-state systems
    - Adapt from source to target system
  ▶ Model multiple viable target systems using UML (*defacto standard*)
  ▶ Leverage Model-Driven Engineering (MDE)

**Challenge:** Construct viable target system models that address uncertainty in environment...
Now for something outside the box …

Inspiration from Nature

- Living organisms have amazing ability to adapt to changing environments
  - Short term: phenotypic plasticity
    - (e.g., snails change tooth shape depending on environment)
  - Long term: genetic evolution
    - (e.g., leaf mantis)
- Most complex organisms exhibit traits desirable in self-managing systems
- Examples:
  - System monitoring (senses, awareness)
  - Reconfiguration (muscle growth, calluses)
  - Self repair (blood clotting, tissue healing)
  - Intrusion detection/elimination (immune systems)

Can evolution be leveraged to identify viable target systems?
Bio-Inspired Work

• Biomimetic systems
  – *Mimic* behavior of natural systems (e.g., swarm UAVs)

• Evolutionary computation
  – Must: *mutate, replicate,* and *compete*

• 3 types of evolutionary computation
  – **Genetic Algorithms:** Individuals encode a solution
  – **Genetic Programming:** Individuals are programs

    • Individuals are programs
    • Individuals are self-replicating
    • Receives more resources to replicate faster by performing *tasks*
    • No explicit fitness function -- evolution is more open-ended
    • Individuals can interact with each other

Approach: Digital Evolution

• **Avida**
  - Digital evolution platform
  - Used to conduct pioneering research in evolution of biocomplexity
Approach: Digital Evolution

- Organism = computer program

An Organism

- Genome (instruction) mutations

An Organism
Approach: Digital Evolution

- Organism = computer program
  - Genome (instruction) mutations
  - Exist in a computational environment

- Compete for resources (e.g., CPU)
Designing Software with Digital Evolution


Energy Management [ECAL07b,SASO-07]

- Energy management is critical issue in mobile devices
- Use Avida to investigate the evolution of energy conservation
- Organisms can conserve energy by executing sleep instructions
- Organisms that manage energy efficiently will proliferate

“Early to bed, early to rise...”
Designing Software with Digital Evolution

ORCHID: Harnessing Digital Evolution to Develop High-Assurance Dynamically Adaptive Systems
Objective: Generate behavioral models

- Resilient behavior
- Potentially innovative

E.g., Globally, it is always the case that eventually the robot’s current position will be its destination.
Objective: Generate behavioral models
• Resilient behavior
• Potentially innovative

An execution path that satisfies the property.

Technique: Brute Force Generation
• Purpose: generate all possible models to find solution
• Input: Alphabet, properties
• Output: Multiple behavioral models
• Characteristics:
  – Alphabet: defined by triggers, guards and actions
  – Generates every possible model
  – Discovers ALL models that satisfy properties
  – Not actually feasible with limited resources (e.g., time, CPU)
Synthesis Techniques

• **Purpose:** Generate a behavioral model for analysis & code generation

• **Input:** Basic scenarios or properties

• **Output:** One behavioral model

*Synthesis techniques based on specific design strategy or algorithm – limited by what is known at development time*

Avida-MDE

[ICAC08, GECCO-08, MODELS08]

• Start with a population of organisms that generate models
Avida-MDE [ICAC08, GECCO-08]

- Start with a population of organisms that generate models
- Organisms replicate and are mutated
  - Different models are produced

• Arrows indicate parent-child relationship
• Direction indicates “better/worse” model

Avida-MDE [ICAC08, GECCO-08]

- Start with a population of organisms that generate models
- Organisms replicate and are mutated
  - Different models are produced
- Selection pressures (tasks) drive evolution:
  • Model density increases closer to the solution space
Avida-MDE [ICAC08, GECCO-08]

- Start with a population of organisms that generate models.
- Organisms replicate and are mutated.
  - Different models are produced.
- Selection pressures (tasks) drive evolution:
  - Poor performing organisms die.
  - Better organisms have more children.
  - E.g., scenarios, properties, software engineering metrics.
  - Deterministic states, fewer transitions.

Avida-MDE Organism

- "Instinctual knowledge" generates state diagrams.
- Evaluated by SE tools (Hydra & Spin).
- Merit determined by interactive state diagrams.
**Avida-MDE Organism**

- **Instinctual knowledge** - Information embedded in an organism at birth
  - Class diagram information *(alphabet)*
    - triggers (methods)
    - guards (boolean expressions of attributes)
    - actions (methods of associated classes)
  - Existing state diagrams (if any)

**How does an organism use its instinctual knowledge to generate state diagrams?**

---

**Avida-MDE Organism**

- Possible instructions:
  - **Select** instinctual knowledge elements
  - **Create** transitions

**Interactive State Diagrams**

- Avida-MDE genomes:
  - Blank to start
  - Mutations
    - Insert/remove/swap instructions
    - Offspring state diagram ≠

**How does Avida-MDE assess the generated state diagrams?**
Avida-MDE Organism

• Avida-MDE Tasks (Selection Pressures):
  ‣ checkScenarios
  ‣ checkProperty
  ‣ checkSEMetrics

"Avida-MDE Organism"

Interactive State Diagrams

2

evaluated by

SE tools

Hydra & Spin

genertes

An Organism

Avida-MDE Organism

• Avida-MDE Tasks:
  ‣ checkScenarios
    - The specific scenario is supported by the state diagrams
    - Compatible with properties
  ‣ checkProperty
  ‣ checkSEMetrics

Interactive State Diagrams

2

evaluated by

SE tools

Hydra & Spin

genertes

An Organism

25
Avida-MDE Organism

- Avida-MDE Tasks:
  - checkScenarios
    - The specific scenario is supported by the state diagrams
    - Compatible with properties
  - checkProperty
    - The state diagrams adhere to the temporal logic property
    - Uses Hydra [ICSE01] & Spin [Holzmann]
  - checkSEMétrics
    - Minimum number of transitions used
    - Percentage of states that are deterministic
Avida-MDE Uses in Software Development

**Initial development**
- Limited information is available
- Developer provides:
  - Class diagram
  - Scenarios
- Avida-MDE generates:
  - A set of state diagrams potentially innovative initial target system

**New Class extension**
- A new class is added to an existing system
- Developer provides:
  - Class diagram
  - Some state diagrams
  - Scenarios
  - Properties
- Avida-MDE generates:
  - A new state diagram for the new class
  - Extend existing state diagrams

**New behavior for existing class**
- Existing classes with new functionality
- Developer provides:
  - Class diagram
  - State diagrams
  - Scenarios
  - Properties
- Avida-MDE generates:
  - Extend existing state diagrams to support new behavior

Case Study: Robot Navigation System

- **T-Rot** - Intelligent robot for elderly [Park06ICSE]
  - Autonomously navigate to a destination
  - Avoid obstacles (*New Class Extension*)

We describe the desired behavior using scenarios, properties, and software engineering metrics...
Property Tasks

1. Normal Behavior:

_Globally, it is always the case that eventually the robot’s current position will be its destination._

• New Behavior:

_Globally, it is always the case that if the ObstacleAvoidanceTimer detects an obstacle, then eventually the WheelActuatorInterface will stop the wheels and the NavigationControl will be suspended._

Scenario Tasks

Two Sequence Diagrams
Scenario Tasks

1. When an obstacle is detected, the ObstacleAvoidance timer stops the wheels and suspends the NavigationControl.

Results: organisms generated 7 compliant behavioral models.
Case Study: Robot Control System

Generated Behavioral Models

<table>
<thead>
<tr>
<th>NavigationControl</th>
<th>Obstacle Avoidance Timer</th>
<th>Sensor Interface</th>
<th>NavigationControl</th>
<th>Obstacle Avoidance Timer</th>
<th>Sensor Interface</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

There is structural variation among the generated models...
Generated Behavioral Models

- fewer states
- increased fault tolerant
- fewer transitions
- increased safety
- increased readability

NavigationControlState Diagram

- Extended with 4 additional transitions
SensorInterface State Diagram

- Extended with 3 additional transitions

ObstacleAvoidanceTimer State Diagram

- Developed from scratch
- 10 transitions
Conclusions

• **Summary:**
  
  - Harnessing Digital Evolution to automatically generate interesting models of target systems [ICAC08, *Best Paper*]
  
  - Suite of solutions help address uncertainty in execution environment [MODELS08, *Distinguished Paper*]
    
    • Use non-functional criteria to distinguish models

• **Current and Future work** *(industrial applications):*
  
  - **Plato-RE:** Use EC for run-time adaptive monitoring [RE@RT2010]
  
  - **Plato:** Run-time generation of new SW configurations, multi-objective tradeoffs [ICAC09, *Best Paper*]
  
  - **Hermes:** Generation of Adaptive Logic [ICAC10]
  
  - **Loki:** Harness EC for generating interesting environmental conditions (e.g., failures, combinations, attacks, etc.) [ASE11]

Beyond Orchid

• **Marple:** Exploring latent properties of software systems
  
  - Discover implicit (latent) properties [MODELS2010]
  
  - Detect unwanted properties with Industrial Models (e.g., feature interactions) [MODELS2011, *Best Paper nomination*]

• **RELAX:** a new specification language to explicitly incorporate uncertainty in requirements [RE09, *One of Best Papers*, REJ10, MODELS09]

• **Medical Applications:**
  
  - **Polypharmacy:** drug bloat problem for the elderly

• **Automotive Applications:**
  
  - **Recent:** Onboard electronics (smart cruise, lane keeping, electronic assisted steering)
  
  - **Current:** Autonomous vehicles (prognostics, self-healing, adaptive, etc.)
BEACON
An NSF Center for the Study of Evolution in Action

Crosscutting Themes:
★ Biological Evolution
★ Digital Evolution
★ Evolutionary Applications

Transformative: $25 million over 5 years to identify and pursue multi-disciplinary opportunities for research, education, and applications

Unifying: Exchange information, hypotheses, and capabilities:
★ 3 thrust groups—evolution of genomes, behavior, communities
★ 5 partners → chosen for their
  – crosscutting strengths in evolution
  – Recruiting Industrial Collaborators for
    Challenge problems

Transformative:  $25 million over 5 years to identify and pursue multi-disciplinary opportunities for research, education, and applications

Unifying:  Exchange information, hypotheses, and capabilities:
★ 3 thrust groups—evolution of genomes, behavior, communities
★ 5 partners → chosen for their
  – crosscutting strengths in evolution
  – Recruiting Industrial Collaborators for
    Challenge problems

Selected RAPIDware Publications (1)


• “Applying Design Patterns to an Adaptive News Server,” (Andres J. Ramirez and Betty H.C. Cheng) Sixth IEEE International Conference on Autonomic Computing (ICAC09), Barcelona, Spain, June 2009, short paper.

• “Modular Verification of Dynamically Adaptive Systems” (Ji Zhang, Heather Goldsby, and Betty H.C. Cheng), the Proceedings of Eighth International Conference on Aspect-Oriented Software Development (AOSD09), Charlottesville, Virginia, March 2009.


Selected RAPIDware Publications (2)

- “Visualizing the Analysis of Dynamically Adaptive Systems Using i* and DSLs*,” (Pete Sawyer, Nelly Bencomo, Danny Hughes, Paul Grace, Heather J. Goldsby, and Betty H.C. Cheng), Proceedings of Second International Workshop on Requirements Engineering Visualization (REV07), New Delhi, India, October 2007.
- "Model-Based Development of Dynamically Adaptive Software" (Ji Zhang and Betty H.C. Cheng), in Proceedings of IEEE International Conference on Software Engineering (ICSE06), Shanghai, China, May 2006. (9% acceptance rate) (Received Distinguished Paper Award), pp. 371–380.

Selected RAPIDware Publications (3)

Selected RAPIDware Publications (4)


Selected RAPIDware Publications (5)

Selected EC Publications (1)


Selected EC Publications (2)


Selected EC Publications (3)

- "Automatically Generating Behavioral Models of Adaptive Systems to Address Uncertainty" (Heather Goldsby and Betty H.C. Cheng), the Proceedings of the ACM/IEEE International Conference on Model Driven Engineering Languages and Systems (MoDELS 2008), Toulouse, France, October 2008 (Selected as one of the Best Papers).

Selected EC Publications (4)

Model-Based Approach to High-Assurance Dynamically Adaptive Systems

Acknowledgements:

Research Team: Heather Goldsby, Philip McKinley, Dave Knoester, Charles Ofria, Xiaobo Tan, Andres Ramirez, Chad Byers, Adam Jensen, Ben Beckmann, Erik Fredericks, and many others from SENS Lab, DevoLab, and BEACON Center

Current Funding:
NSF Grants: CCF-0820220, CNS-0751155, IIP-0700329, CNS-0854931, Army Research Grant W911NF-08-1-0495, Ford