

Model-Driven Requirements Engineering for High Assurance Embedded Systems

Betty H.C. Cheng

Software Engineering and Network Systems Lab
Michigan State University

This work is supported in part by:

Grants from NSF EIA-0000433, EIA-0130724, CDA-9700732, CCR-9901017, Department of the Navy, Office of Naval Research under Grant No. N00014-01-1-0744, and DARPA grant No. F30602-96-1-0298, managed by Air Force's Rome Laboratories, Siemens Corporate Research, Eaton Corporation, Motorola, and in cooperation with Siemens Automotive and Detroit Diesel Corporation.

Software Engineering and Network Systems Laboratory



Research sponsored by NSF, ONR, DARPA, DOE, NASA, EPA, and several industrial partners

MICHIGAN STATE UNIVERSITY	Outline
S oftware E ngineering & N etwork S ystems Lab	• High Assurance Embedded Systems • Model-driven development • Roundtrip engineering of Requirements – Patterns for requirements models – Simulate and validate models – NL patterns for properties – Analysis of models (model checkers) – Visual feedback of analysis results

3

MICHIGAN STATE UNIVERSITY	Introduction
Overview: Introduction UML Formalization Req. Patterns Process Conclusions Future Work S oftware E ngineering & N etwork S ystems Lab	• Many embedded systems require high assurance (e.g. automotive, medical) • Requirements modeling and analysis – One of the most difficult tasks in software development – Focus on behavioral specification of system activities – Describes a system's modes of operation and events that cause mode changes • Challenges for embedded system development: – Software does not execute in isolation: • Environment (including User) • Hardware – Current technology involves <i>ad hoc</i> techniques from natural language specifications to code • ES community interested in using OO and UML

4

MICHIGAN STATE UNIVERSITY	Problem Statement
Overview: Introduction UML Formalization Req. Patterns Process Conclusions Future Work Software Engineering & Network Systems Lab	- Desirable properties of requirements analysis documents: - Easy to interpret - Structural description of system - Behavioral description of system - Descriptions should be concise and correct - Requirements analyzable for critical properties

5

MICHIGAN STATE UNIVERSITY	Model-driven Development
Software Engineering & Network Systems Lab	- General trend towards MDD - Platform-independent models (PIM) - Platform-specific models (PSM) (Automated) transformations from PIMs to PSMs - Contributing factors - Traceability throughout development process - Separate platform-independent and platform-specific information - Enables domain-specific code generation - Automated test case generation

6

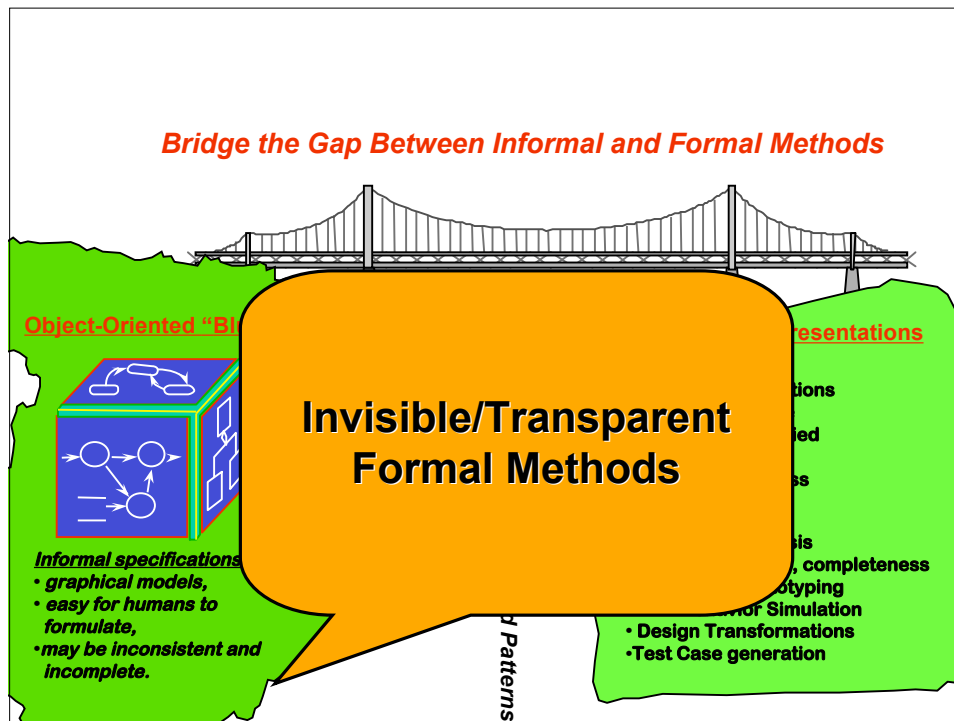
Model-driven Development

Challenges to RE community

- Most efforts in MDD start with design
 - Need to start with requirements-level models
- Leverage previous experience
 - Reuse RE artifacts (e.g., templates, patterns)
- Make formal techniques more accessible
 - Complement current testing techniques
 - Provide assistance in identifying critical properties
- Analyze requirements for critical properties
 - Need analysis feedback loop to focus model refinement

Model-driven Development

- Challenges to Automotive RE community
 - Most efforts in MDD start with design
 - Need to start with requirements-level models
 - Automotive embedded systems can be modeled with MDD diagrams
 - Well-defined domain facilitates automated code generation from models.
 - Leverage previous experience
 - Reuse RE artifacts (e.g., templates, patterns)
 - Well-defined domain can yield generic requirements
 - Make formal techniques more accessible
 - Complement current testing techniques
 - Provide assistance in identifying critical properties
 - Well-defined domain can yield generic properties
 - Analyze requirements for critical properties
 - Need analysis feedback loop to focus model refinement



MICHIGAN STATE UNIVERSITY	General Approach
Overview: - Introduction - UML Formalization - Req. Patterns - Process - Conclusions - Future Work Software Engineering & Network Systems Lab	• Objective: – Easy to use notation and technique for capturing requirements – Notation must be amenable to rigorous analysis • Proposed Solution: – Provide process and <i>requirements-level patterns</i> for constructing UML diagrams – Formalize UML enables <i>automated analysis</i> of UML diagrams – <i>Natural language templates</i> for property specifications – <i>Visualize</i> analysis errors in terms of original UML diagrams • Project Collaborators: – Dr. L. Campbell, Dr. W. McUmber, Dr. E. Wang – R. Bourdeau, G. Coombs, M. Deng, H. Goldsby, S. Konrad – Eaton, Detroit Diesel, Siemens, General Dynamics

MICHIGAN STATE UNIVERSITY

Overview:

- Introduction
- OA Patterns**
- Background
- Description
- Repository
- RT Spec. Patterns
- NL Specification
- Case Study
- Related Work
- Concl. + Fut. Work

Software Engineering & Network Systems Lab

Roundtrip MDD

- Approach :**

A **model-driven software analysis approach** for embedded systems can be made formal and accessible through the use of four key elements:

```

graph TD
    MDOA[Model-driven Object Analysis] --> OAP[Object analysis patterns]
    MDOA --> IS[Informal specification]
    OAP --> UF[UML Formalization]
    IS --> SP[Specification patterns]
    OAP --> IS
    
```

The diagram illustrates the Roundtrip MDD approach. It starts with 'Model-driven Object Analysis' at the bottom, which branches into 'Object analysis patterns' and 'Informal specification'. 'Object analysis patterns' leads to 'UML Formalization' (highlighted with a red box), and 'Informal specification' leads to 'Specification patterns'. There is also a bidirectional arrow between 'Object analysis patterns' and 'Informal specification'.

12

MICHIGAN STATE UNIVERSITY

Overview:

- Introduction
- UML Formalization**
- Req. Patterns
- Process
- Conclusions
- Future Work

Software Engineering & Network Systems Lab

UML Formalization

- Automate translation of diagrams into a formal language**
 - OMT Formalization
 - [TSE95, ICSE97, J. SEKE00, TSE02, IWSSD00, DSN00]
 - UML Formalization [HASE99, ICSE01]
 - General framework for mapping diagrams to multiple formal languages
 - Embedded systems domain
 - Currently targets Promela
 - Hydra
 - Mapping from UML to the target language (such as Promela, VHDL)
 - Enables execution through simulation and analysis through model checking

13

MICHIGAN STATE UNIVERSITY

Overview:

- Introduction
- OA Patterns**
- Background
- Description
- Repository
- RT Spec. Patterns
- NL Specification
- Case Study
- Related Work
- Concl. + Fut. Work

Software Engineering & Network Systems Lab

Roundtrip MDD

• Approach :

A **model-driven software analysis approach** for embedded systems can be made formal and accessible through the use of four key elements:

```

graph TD
    MOA[Model-driven Object Analysis] --> OA[Object analysis patterns]
    OA --> UML[UML Formalization]
    OA --> IS[Informal specification]
    IS --> SP[Specification patterns]
    SP --> OA
  
```

14

MICHIGAN STATE UNIVERSITY

Overview:

- Introduction
- OA Patterns
- Background**
- Description
- Repository
- RT Spec. Patterns
- NL Specification

Network Systems Lab

Background: Object Analysis

• Development processes focused on embedded systems, (such as COMET [Gomaa 00] and ROPES [Douglass 99]) **advocate analysis phase** (i.e., late/detailed requirements)

- Produces **conceptual (analysis) object model**
- **Key objects** and their interactions
- Focus on **functional requirements**, e.g., no implementation constraints

```

graph LR
    SR[System Requirements] --> Analysis
    subgraph Analysis
        RA[Requirements Analysis] --> SOA[Structural Object Analysis]
        SA[Systems Analysis] --> BOA[Behavioral Object Analysis]
        SOA --> OA[Object Analysis]
        BOA --> OA
    end
    AP[Analysis Patterns] --> Analysis
    OA --> AOM[Analysis Object Model]
    AOM --> Design
    DP[Design Patterns] --> Design
    Design --> Ellipsis[...]
  
```

15

MICHIGAN STATE UNIVERSITY

Overview:

- Introduction
- UML Formalization
- Req. Patterns**
- Process
- Conclusions
- Future Work

Software Engineering & Network Systems Lab

Object Analysis Patterns Template

Design Pattern Template	Object Analysis Pattern Template
- Pattern Name and Classification - Intent - Also Known As - Motivation - Applicability - Structure - Participants - Collaborations - Consequences Implementation Sample Code - Known Uses - Related Patterns	- Pattern Name and Classification - Intent - Motivation (<i>incl. use cases</i>) Constraints - Applicability - Structure (<i>class diagram</i>) Behavior (sequence, state) - Participants - Collaborations - Consequences Design Patterns - Also Known As - Related Patterns

16

MICHIGAN STATE UNIVERSITY

Object Analysis Pattern Overview

```

graph TD
    OA[OA Pattern] --> S[Structural]
    OA --> B[Behavioral]
    S --> AS[Actuator-Sensor]
    S --> CD[Controller Decompose]
    S --> UI[User Interface]
    B --> CL[Communication Link]
    B --> DC[Detector-Corrector]
    B --> CC[Computing Component]
    B --> FH[Fault Handler]
        
```

- Structural patterns: Capture structural properties
- Behavioral patterns: Refine behavior
 - Object identification
 - Define essential behavior of reactive objects
 - Object abstraction
 - Construct mechanisms of object correlation

Network Systems Lab

17

MICHIGAN STATE UNIVERSITY

Overview:

- Introduction
- OA Patterns**
- Background
- Description
- Repository
- RT Spec. Patterns
- NL Specification
- Case Study
- Related Work
- Concl. + Fut. Work

Software Engineering & Network Systems Lab

Roundtrip MDD

• Approach :

A **model-driven software analysis approach** for embedded systems can be made formal and accessible through the use of several key elements:

```

graph TD
    MDOA[Model-driven Object Analysis] --> OAP[Object analysis patterns]
    MDOA --> IS[Informal specification]
    OAP --> UF[UML Formalization]
    OAP --> IS
    IS --> SP[Specification patterns]
    
```

18

MICHIGAN STATE UNIVERSITY

Overview:

- Introduction
- OA Patterns
- RT Spec. Patterns
- Description**
- Background
- Pattern Template
- NL Specification
- Case Study
- Related Work
- Concl. + Fut. Work

Software Engineering & Network Systems Lab

Problem Description

- Formal specification languages rarely used:
 - Perceived as difficult: **cryptic, declarative**
 - **Extensive training** in formal methods needed
- Specification pattern systems:
 - Collection of generalized descriptions of **commonly occurring requirements**
 - Increase **usability and applicability**
 - Focus on **qualitative properties**
 - Specified properties may still be **hard to comprehend**

$$\Box(((Q \wedge \neg R \wedge \Diamond R) \rightarrow (P \rightarrow (\neg R \cup (S \wedge \neg R))) \cup R)$$

Specification pattern [Dwyer 99]:
“Between Q and R, S responds to P.”

19

MICHIGAN STATE UNIVERSITY

Overview:

- Introduction
- OA Patterns
- RT Spec. Patterns
- Description**
- Background
- Pattern Template
- NL Specification
- Case Study
- Related Work
- Concl. + Fut. Work

Software Engineering & Network Systems Lab

Real-time Specification Patterns

Two-pronged approach:

1. **Templates for real-time temporal logic specifications of software properties**
 1. Assess 9 existing requirements documents for real-time properties (some of them with UML models)
 2. Create taxonomy of real-time properties
 3. Develop representations in three real-time temporal logics (MTL, TCTL, RTGIL)
2. **Natural language handles**
 - Develop structured English handles to specification patterns

20

MICHIGAN STATE UNIVERSITY

Overview:

- Introduction
- OA Patterns
- RT Spec. Patterns
- Description
- Background
- Pattern Template**
- NL Specification
- Case Study
- Related Work
- Concl. + Fut. Work

Software Engineering & Network Systems Lab

RT Specification Pattern Template

Real-time Specification Pattern Template

- • **Pattern Name and Classification**
- • **Structured English Specification**
 - Pattern Intent
- • **Real-time Temporal Logic Mappings**
 - Examples and Known Uses
 - Relationships

- Variation of the specification pattern template by Dwyer et al. [Dwyer 99]
- Most notable change is the addition of the “**Structured English Specification**” field

22

MICHIGAN STATE UNIVERSITY

Overview:

- Introduction
- OA Patterns
- RT Spec. Patterns
- Description
- Background
- Pattern Template**
- NL Specification
- Case Study
- Related Work
- Concl. + Fut. Work

Software Engineering & Network Systems Lab

RT Specification Pattern Template

Real-time Specification Pattern Template

- • **Pattern Name and Classification**
- • Structured English Specification
- • Pattern Intent
- • Real-time Temporal Logic Mappings
- Examples and Known Uses
- Relationships

- Variation of the specification pattern template by Dwyer et al. [Dwyer 99]
- Most notable change is the addition of the “Structured English Specification” field

23

MICHIGAN STATE UNIVERSITY

Overview:

- Introduction
- OA Patterns
- RT Spec. Patterns
- Description
- Background
- Pattern Template**
- NL Specification
- Case Study
- Related Work
- Concl. + Fut. Work

Software Engineering & Network Systems Lab

Classification and Repository

- Collected 80 RT requirements from 9 requirements documents for industrial applications
- Classified into three broad categories:
 1. **Duration:** Properties to place bounds on the duration of an occurrence
 1. Minimum Duration
 2. Maximum Duration
 2. **Periodic:** Properties that address periodic occurrences
 1. Bounded Recurrence
 3. **Real-time Order:** Captures properties that put real-time bounds on the order of multiple occurrences
 1. Bounded Response
 2. Bounded Invariance

Complementary to Dwyer

24

MICHIGAN STATE UNIVERSITY

Overview:

- Introduction
- OA Patterns
- RT Spec. Patterns
- Description
- Background
- Pattern Template**
- NL Specification
- Case Study
- Related Work
- Concl. + Fut. Work

Software Engineering & Network Systems Lab

RT Specification Pattern Template

Real-time Specification Pattern Template

- Pattern Name and Classification
- **Structured English Specification**
- Pattern Intent
- Real-time Temporal Logic Mappings
- Examples and Known Uses
- Relationships

- Variation of the specification pattern template by Dwyer et al. [Dwy99]
- Most notable change is the addition of the “Structured English Specification” field

25

MICHIGAN STATE UNIVERSITY

Overview:

- Introduction
- OA Patterns
- RT Spec. Patterns
- Description
- Background
- Pattern Template**
- NL Specification
- Case Study
- Related Work
- Concl. + Fut. Work

Software Engineering & Network Systems Lab

Structured English Grammar: Usage

Assuming user needs to specify a RT cause-effect relationship in MTL

- Identify *Bounded Response* pattern:
“Globally, it is always the case that if **P** holds, then **S** holds after at most **c** time unit(s).”
- Replace placeholders (**P**, **S**, and **c**) with concrete elements:
“Globally, it is always the case that if **(x = 0)** holds, then **(y = 1)** holds after at most **6** time unit(s).”
- Specify the property using the specification pattern temporal logic template:

$$\Box((x = 0) \rightarrow \Diamond_{\leq 6}(y = 1))$$

26

MICHIGAN STATE UNIVERSITY

Overview:

- Introduction
- OA Patterns
- RT Spec. Patterns
- Description
- Background
- Pattern Template**
- NL Specification
- Case Study
- Related Work
- Concl. + Fut. Work

Software Engineering & Network Systems Lab

RT Specification Pattern Template

Real-time Specification Pattern Template

- • Pattern Name and Classification
- • Structured English Specification
- • Pattern Intent
- • **Real-time Temporal Logic Mappings**
- Examples and Known Uses
- Relationships

- Variation of the specification pattern template by Dwyer et al. [Dwy99]
- Most notable change is the addition of the “Structured English Specification” field

27

MICHIGAN STATE UNIVERSITY

Overview:

- Introduction
- OA Patterns
- RT Spec. Patterns
- Description
- Background
- Pattern Template**
- NL Specification
- Case Study
- Related Work
- Concl. + Fut. Work

Software Engineering & Network Systems Lab

The Real-time Temporal Logics

- Rationale for choosing MTL, TCTL, and RTGIL:
 1. Related to temporal logics used in Dwyer et al.’s patterns
 - LTL, CTL, and GIL
 2. Tool support
 - Run-time verifier, model checker, theorem prover
 3. Syntax and semantics
 - MTL and TCTL cover linear and branching time
 - RTGIL covers linear time and provides graphical notation

28

Overview:

- Introduction
- OA Patterns
- RT Spec. Patterns
- Description
- Background
- Pattern Template
- NL Specification
- Case Study
- Related Work
- Concl. + Fut. Work

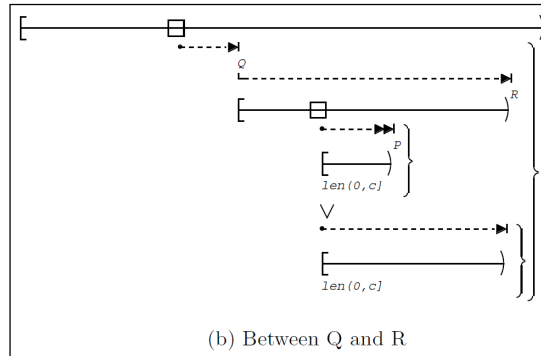
Bounded Recurrence Pattern

“Between Q and R , it is always the case that P holds at least every c time unit(s).”

MTL: $\Box((Q \wedge \neg R \wedge \Diamond R) \rightarrow ((\Diamond_{\leq c}(P \vee R)) \mathcal{U} R))$

TCTL: $AG((Q \wedge \neg R) \rightarrow A[(\neg AF_{\leq c}(P \vee R)) \vee AG(\neg R)] \vee R)$

RTGIL:



29

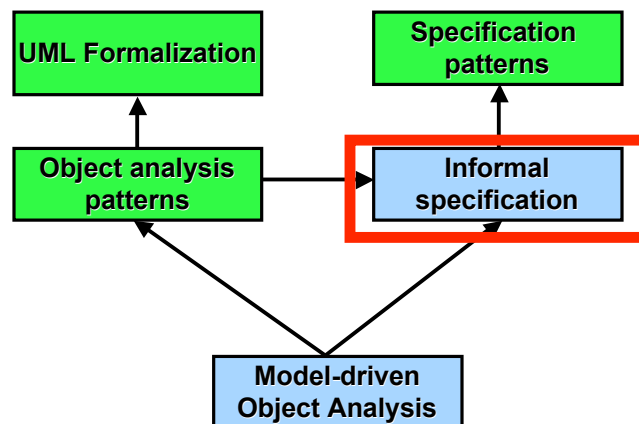
Overview:

- Introduction
- OA Patterns
- Background
- Description
- Repository
- RT Spec. Patterns
- NL Specification
- Case Study
- Related Work
- Concl. + Fut. Work

Roundtrip MDD

• Approach :

A **model-driven software analysis approach** for embedded systems can be made formal and accessible through the use of four key elements:



30

MICHIGAN STATE UNIVERSITY

Overview:

- Introduction
- OA Patterns
- RT Spec. Patterns
- NL Specification
- Description**
- 1. Derivation
- 2. Instantiation
- 3. Mapping
- Case Study
- Related Work
- Concl. + Fut. Work

Software Engineering & Network Systems Lab

Structured English Grammar

- Generalize NL handles into grammar that can be used to derive properties
- Non-recursive** structured English grammar **w/o repetitions**
- Sentences in the **finite language** of the grammar are formally analyzable qualitative and real-time properties
- Advantages to using a grammar:
 1. Identifies the **subset of natural language**
 2. **Highlights similarities** between NL sentences capturing different patterns
 3. Supports **syntax-guided derivation and instantiation process**

31

MICHIGAN STATE UNIVERSITY

Overview:

- Introduction
- OA Patterns
- RT Spec. Patterns
- NL Specification
- Description**
- 1. Derivation
- 2. Instantiation
- 3. Mapping
- Case Study
- Related Work
- Concl. + Fut. Work

Software Engineering & Network Systems Lab

NL Specification

3 step process:

1. **Derivation**
 - Derive a NL property template from a structured NL grammar
2. **Instantiation**
 - Instantiate the NL property template with model-specific elements
3. **Mapping**
 - Map the NL property to a formal specification understood by the targeted analysis tool and execute the analysis

SPIDER (Specification Pattern Instantiation and Derivation Environment)

Prototype implementation of the specification process

Customizable through the use of XML and plug-ins

32

MICHIGAN STATE UNIVERSITY

Overview:

- Introduction
- OA Patterns
- RT Spec. Patterns
- NL Specification
- Description
- 1. Derivation**
- 2. Instantiation
- 3. Mapping
- Case Study
- Related Work
- Concl. + Fut. Work

Software Engineering & Network Systems Lab

1. Property Deriver Component

- Used to **derive NL property templates**
- Syntax-directed editor**
 - Guided by structured natural language grammar
 - Offers additional descriptive information
- User interface**

33

MICHIGAN STATE UNIVERSITY

Software Engineering & Network Systems Lab

1. Property Deriver Component

34

1. Property Deriver Component

Choices

- 1: 'Global' scope
- 2: 'Before R' scope
- 3: 'After Q' scope
- 4: 'Between Q and R' scope
- 5: 'After Q until R' scope

Property Deriver

hit(s).

property:

that a property has to hold over

that a property only holds up to

that a property has to hold after

denotes that the property only

which Q holds and a state in which

which R holds after a state in which

quired to be satisfied after Q.

specifies that a property has to hold

until a state in which Q holds. The

Q never holds after a state in which

Derivation History

- property
 - specification (1)
 - specificationChoice (2)
 - realtimeType (4)
 - realtimeTypeChoice (5)
 - periodicCategory (6)
 - boundedRecurrencePattern (7)
 - scope (3)
 - scopeChoice (8)
 - Non-Terminal: scopeChoice (8)

Undo

Instantiate

SE Engineering & Network Systems Lab

35

1. Property Deriver Component

Property Deriver

Grammar Options Help

NL Property Being Derived

Globally, it is always the case

Choices

- 1: 'Global' scope
- 2: 'Before R' scope
- 3: 'After Q' scope
- 4: 'Between Q and R' scope
- 5: 'After Q until R' scope

Select Cancel

Description

Specifies the scope of the property:

- 1: The 'Global' scope denotes that a property has to hold over the whole execution.
- 2: The 'Before R' scope denotes that a property only holds up to state in which R holds.
- 3: The 'After Q' scope determines that a property has to hold after a state at which Q holds.
- 4: The 'Between Q and R' scope denotes that the property only has to hold between a state in which Q holds and a state in which R holds. If there is no state in which R holds after a state in which Q holds, the property is not required to be satisfied after Q.
- 5: The 'After Q until R' scope specifies that a property has to hold after a state in which R holds until a state in which Q holds. The property might hold infinitely if Q never holds after a state in which R holds.

SE Software Engineering & Network Systems Lab

36

MICHIGAN STATE UNIVERSITY

Overview:

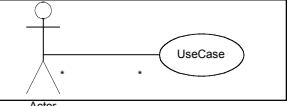
- Introduction
- OA Patterns
- Background**
- Description
- Repository
- RT Spec. Patterns
- NL Specification
- Case Study
- Related Work
- Concl. + Fut. Work

Software Engineering & Network Systems Lab

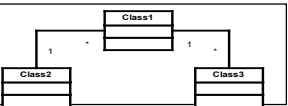
Background: UML

- Unified Modeling Language:
 - Object modeling and specification language
 - Object Management Group standard
- Three types of models:

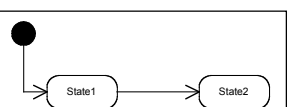
1. Functional model



2. Object model



3. Dynamic model

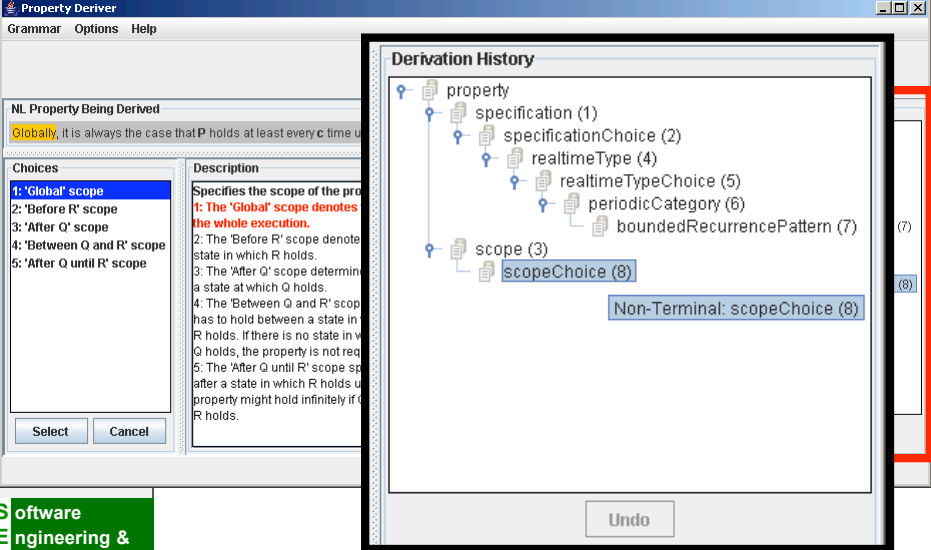


37

MICHIGAN STATE UNIVERSITY

Software Engineering & Network Systems Lab

1. Property Deriver Component



38

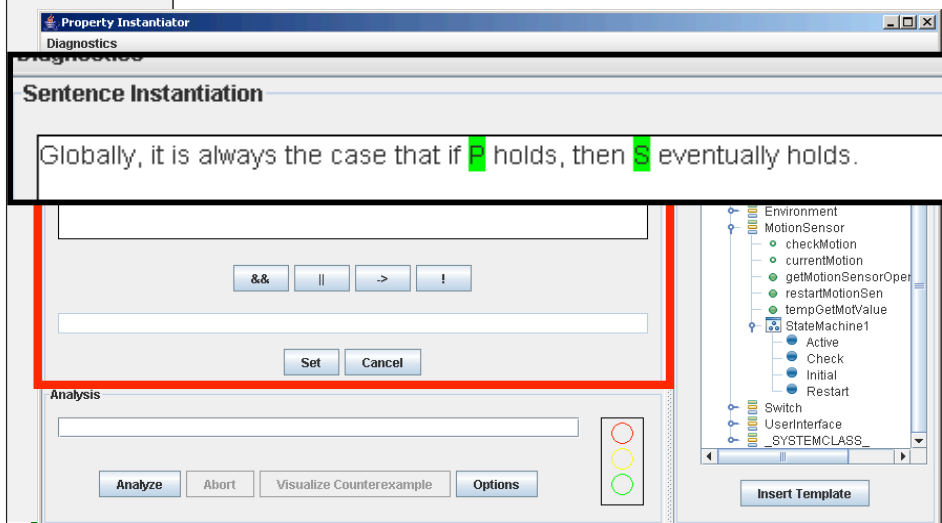
2. Property Instantiator Component

Overview:

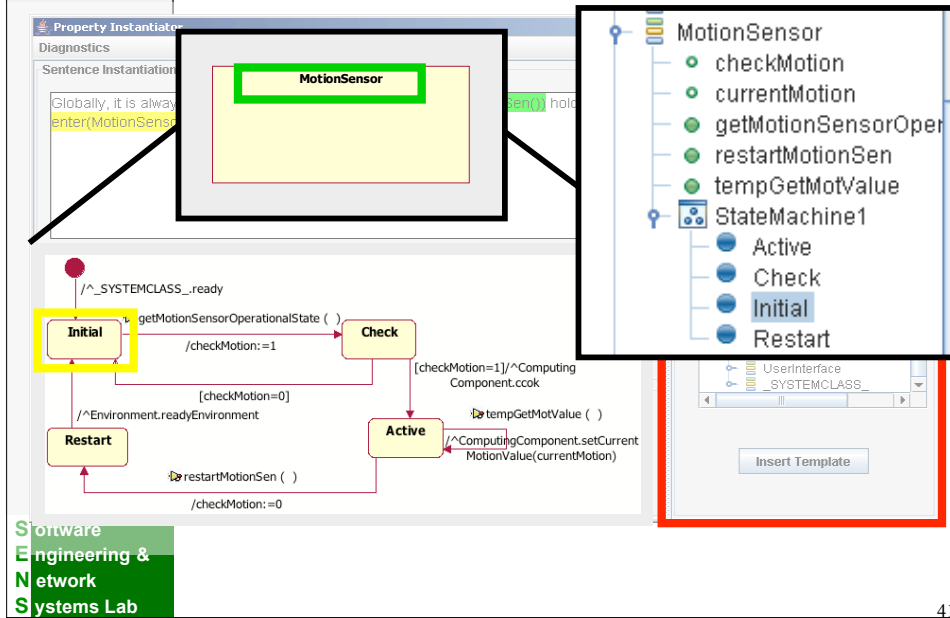
- Introduction
- OA Patterns
- RT Spec. Patterns
- NL Specification
- Description
- 1. Derivation
- 2. Instantiation**
- 3. Mapping
- Case Study
- Related Work
- Concl. + Fut. Work

- Used to **instantiate placeholders** in NL property template
- Aids in constructing **syntactically correct statements**
 - Provides templates
 - Provides special predicates
 - `enter(...)`
 - `call(...)`
 - Performs syntax checks of statements
- **User interface**

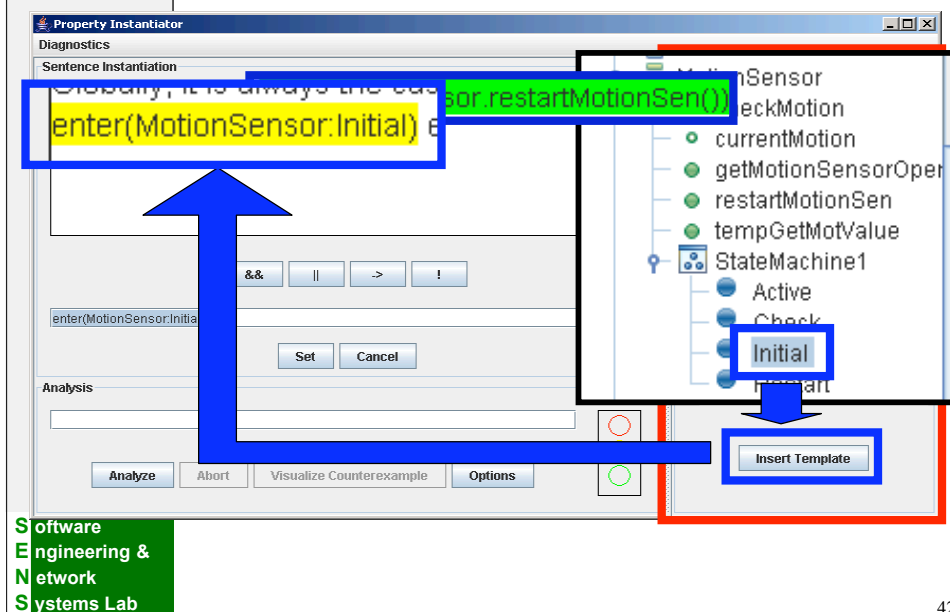
2. Property Instantiator Component



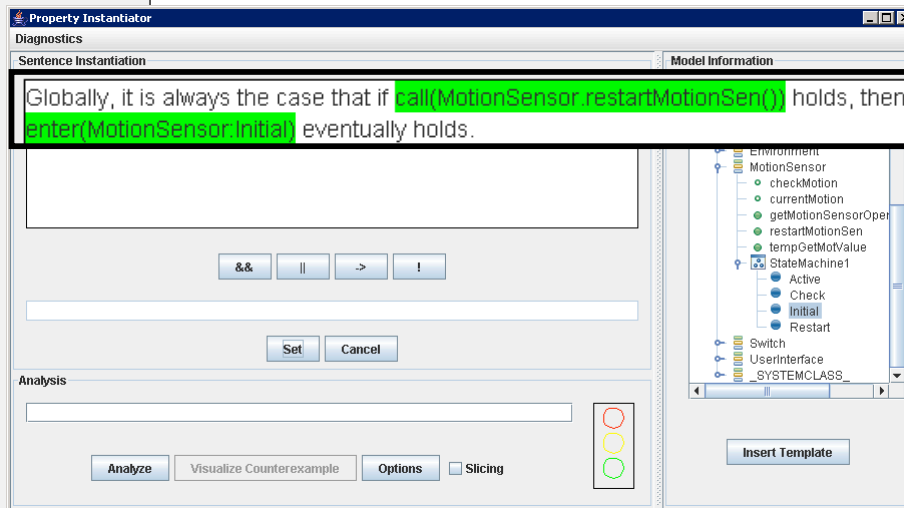
2. Property Instantiator Component



2. Property Instantiator Component



2. Property Instantiator Component



3. Formal Analysis Tool Connector

Overview:

- Introduction
- OA Patterns
- RT Spec. Patterns
- NL Specification
- Description
- 1. Derivation
- 2. Instantiation
- 3. Mapping
- Case Study
- Related Work
- Concl. + Fut. Work

Specific to formal modeling language and analysis tool used

- Plug-in to SPIDER
- Consists of two components:
 1. **Formal Model Interpreter:**
 - Extracts information from the formal model
 2. **Property Analyzer:**
 - Translates the NL property to formal property
 - Performs optimizations
 - Invokes the analysis tool, interprets analysis results, and provides visualization

MICHIGAN STATE UNIVERSITY

Overview:

- Introduction
- OA Patterns
- RT Spec. Patterns
- NL Specification
- Description
- 1. Derivation
- 2. Instantiation
- 3. Mapping
- Case Study
- Related Work
- Concl. + Fut. Work

Software Engineering & Network Systems Lab

3. Formal Analysis Tool Connector

Specific to **formal modeling language** and **analysis tool** used

- Plug-in** to SPIDER
- Consists of **two components**:
 - 1. Formal Model Interpreter:**
 - Extracts information from the formal model
 - 2. Property Analyzer:**
 - Translates the NL property to formal property
 - Performs optimizations
 - Invokes the analysis tool, interprets analysis results, and provides visualization

45

MICHIGAN STATE UNIVERSITY

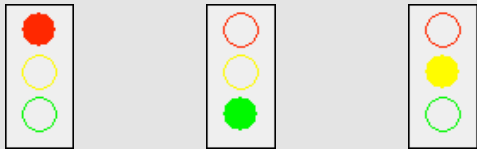
Overview:

- Introduction
- OA Patterns
- RT Spec. Patterns
- NL Specification
- Description
- 1. Derivation
- 2. Instantiation
- 3. Mapping
- Case Study
- Related Work
- Concl. + Fut. Work

Software Engineering & Network Systems Lab

3. Formal Analysis Tool Connector

- Analysis Results:**
 - Formal analysis tool connector **parses analysis tool output**
 - Displayed in **textual form**:
 - "Property successfully verified"
 - "Counterexample found"
 - "Analysis error"
 - Displayed in **terms of traffic light icon**:



Counterexample found

Property verified

Analysis error

46

MICHIGAN STATE UNIVERSITY

Overview:

- Introduction
- Motivation
- Problem Desc.**
- Proposed Solution
- Background
- Process
- Pro. Trace Data
- Architecture
- Related Work
- Case Study
- Conclusion

Software Engineering & Network Systems Lab

Problem Description

- Analysis tool: Model Checkers
 - Less expertise in formal methods
 - Automated
 - Provide counterexamples
- Model Checkers shortcomings:
 - Complex specification language
 - Intricate representation of the generated output

47

MICHIGAN STATE UNIVERSITY

Overview:

- Introduction
- Motivation
- Problem Desc.
- Proposed Solution**
- Background
- Process
- Pro. Trace Data
- Architecture
- Related Work
- Case Study
- Conclusion

Software Engineering & Network Systems Lab

Proposed Solution

- Intuitive understanding of analysis results:
 - Parse and process the output
 - Elicit necessary information
 - Visual feedback in
 - Original UML state diagram
 - New generated UML sequence diagram

THESEUS
Prototype implementation of the developed approach

48

MICHIGAN STATE UNIVERSITY

Overview:

- Motivation
- RT Spec. Patterns
- NL Specification
- OA Patterns
- Related Work
- Remaining Invest.

Software Engineering & Network Systems Lab

Proposed Solution

- Approach:

A **model-driven software analysis approach** for embedded systems can be made formal and accessible through the use of several key elements:

Specification patterns

←

Real-time specification patterns [ICSE05,TSE06]

Integration formal-informal

←

Syntax-guided specification [RE05,LNCS06]

UML model patterns

←

Object analysis patterns [RE02,TSE04]

Iter.,incremental modeling and analysis

←

Current investigations

52

MICHIGAN STATE UNIVERSITY

Overview:

- Motivation
- RT Spec. Patterns
- NL Specification
- OA Patterns
- Related Work
- Remaining Invest.

Software Engineering & Network Systems Lab

Current Investigations

- i2MAP (**Incremental, Iterative Modeling and Analysis Process**):
 - **Incremental, iterative model development** guided by goals
 - Each Model increment must adhere to properties associated with increment and all previously specified properties
 - Related to **test-driven development**
 - User specifies and associates properties associated with each increment before modeling
 - Incorporates some **XP principles**
 - Incremental changes
 - Rapid feedback

53

Overview:

Motivation

OA Patterns

RT Spec. Patterns

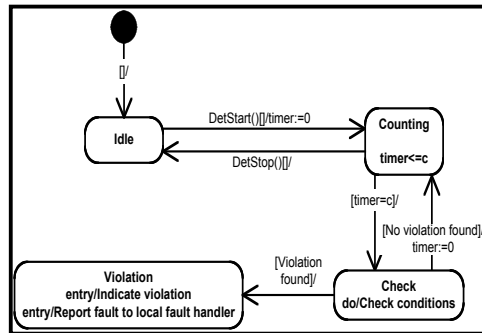
NL Specification

Related Work

Remaining Invest.

Current Investigations

- Extensions to existing components:
 - Real-time extension of object analysis patterns



After the start of a detector until the detector is stopped, it is always the case that a constraint violation check occurs at least every c time units.

54

Overview:

Motivation

OA Patterns

RT Spec. Patterns

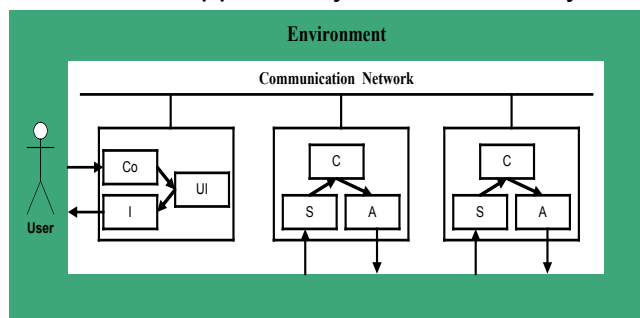
NL Specification

Related Work

Remaining Invest.

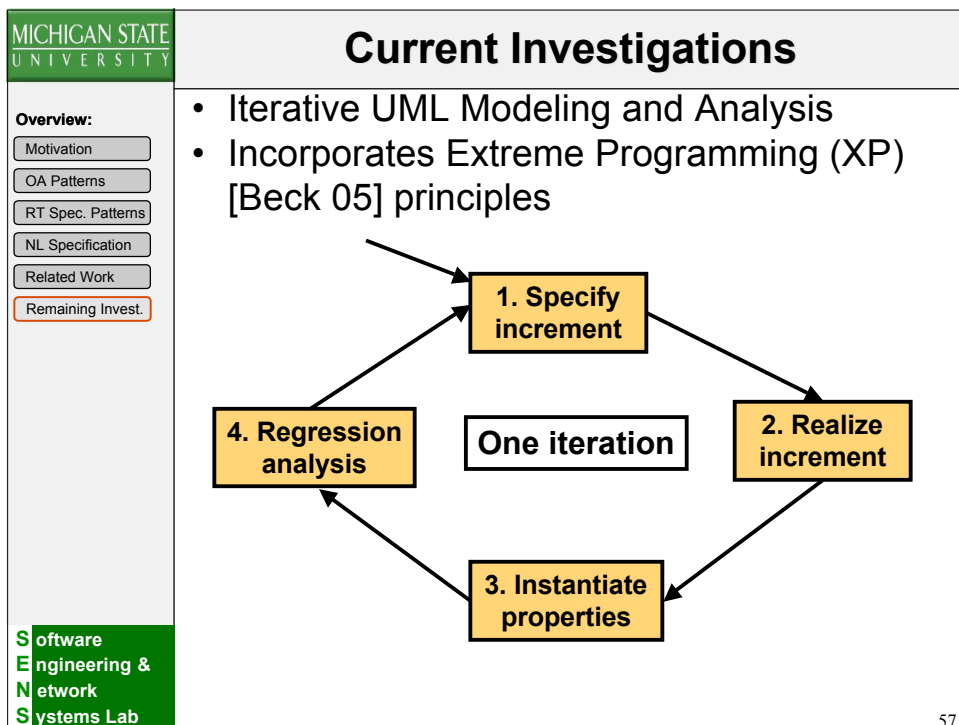
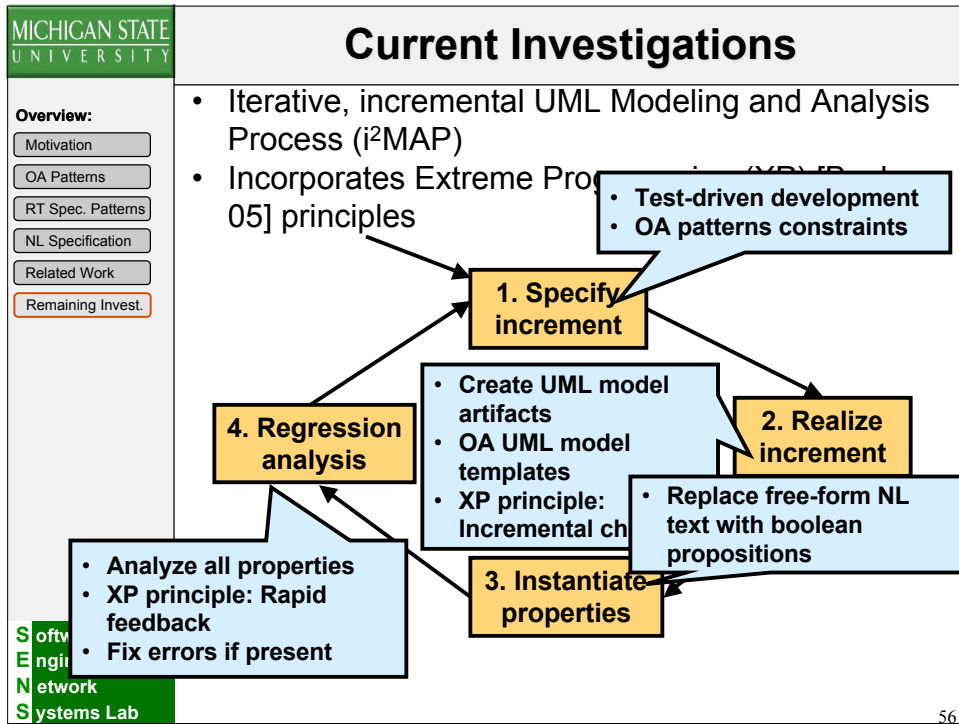
Current Investigations

- Changes to existing components:
 - Enable applicability to distributed systems



- Address distributed real-time embedded systems (DREs)

55



| | |
|--|--|
| MICHIGAN STATE
UNIVERSITY | <h2 style="text-align: center;">Summary</h2> |
| <div style="display: flex; align-items: center;"> <div style="writing-mode: vertical-rl; transform: rotate(180deg);"> Software
Engineering &
Network
Systems Lab </div> </div> | <ul style="list-style-type: none"> • Address RE challenges <ul style="list-style-type: none"> – Leverage what has been done in MDD <ul style="list-style-type: none"> • Expand definition of modeling – Capture and leverage reusable RE artifacts – Insist on assurance for Automotive ES • Make Software a first class citizen <ul style="list-style-type: none"> – Software has requirements/contraints/assumptions too |

58

| | |
|---|--|
| MICHIGAN STATE
UNIVERSITY | <h2 style="text-align: center;">Future Work</h2> |
| <div style="display: flex; align-items: center;"> <div style="writing-mode: vertical-rl; transform: rotate(180deg);"> Software
Engineering &
Network
Systems Lab </div> <div style="margin-left: 10px;"> <p>Overview:</p> <div style="border: 1px solid #ccc; padding: 2px; margin-bottom: 2px;">Introduction</div> <div style="border: 1px solid #ccc; padding: 2px; margin-bottom: 2px;">Background</div> <div style="border: 1px solid #ccc; padding: 2px; margin-bottom: 2px;">Process</div> <div style="border: 1px solid #ccc; padding: 2px; margin-bottom: 2px;">Conclusions</div> <div style="border: 1px solid #ccc; padding: 2px; margin-bottom: 2px; border-color: #f00;">Future Work</div> </div> </div> | <ul style="list-style-type: none"> • Current and future work: <ul style="list-style-type: none"> – Extend our tools and patterns to support discrete timing aspects [ASE-04] – Extend our pattern repository to address security [RHAS03] – Explore integration between goal modeling and UML modeling. – Examine how to abstract model specifications – HPCC for performing model checking (10 fold speedup) • Other projects: <ul style="list-style-type: none"> – RAPIDware (ONR adaptive middleware project) – Safeness and Correctness of adaptations – Feature Interactions – Use AOP to weave adaptability – Code generation for adaptations. |

59

| | |
|--|---|
| MICHIGAN STATE
UNIVERSITY | <h2>Acknowledgements</h2> |
| <div> <div>Overview:</div> <div>Introduction</div> <div>Background</div> <div>Process</div> <div>Conclusions</div> <div>Future Work</div> </div> <div> Software
Engineering &
Network
Systems Lab </div> | <ul style="list-style-type: none"> Software Engineering and Networking Systems Faculty/Students This work has been supported in part by <ul style="list-style-type: none"> NSF grants EIA-0000433, EIA-0130724, CDA-9700732, CCR-9901017, Department of the Navy, Office of Naval Research under Grant No. N00014-01-1-0744, and DARPA grant No. F30602-96-1-0298, managed by Air Force's Rome Laboratories Eaton Corporation, Siemens Corporate Research, a Motorola doctoral fellowship, and in cooperation with Siemens Automotive and Detroit Diesel Corporation |

60

| | |
|---|---|
| MICHIGAN STATE
UNIVERSITY | <h2>References</h2> |
| <div>[Gebhard]</div> <div>[Broy]</div> <div>[Glinz]</div> <div>[Dwyer]</div> <div>[Gamma]</div> <div> Software
Engineering &
Network
Systems Lab </div> | <p>Bernd Gehhard, Martin Rappl, <i>Requirements Management for Automotive Systems Development</i>. SAE World Congress, 2000</p> <p>Manfred Broy, <i>Requirements Engineering for Embedded Systems</i>. Workshop on Formal Design of Safety Critical Embedded Systems, 1997</p> <p>Martin Glinz, <i>Problems and Deficiencies of UML as a Requirements Specification Language</i>. Proceedings of the Tenth International Workshop on Software Specification and Design, San Diego, 11-22, 2000</p> <p>M. B. Dwyer, G. S. Avrunin, J. C. Corbett, <i>Patterns in Property Specifications for Finite-State Verification</i>. UM-CS-1998-035, 1998</p> <p>Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides, <i>Design Patterns: Abstraction and Reuse of Object-Oriented Design</i>. Lecture Notes in Computer Science, vol. 707, p. 406 – 431, 1993</p> |

61

| MICHIGAN STATE UNIVERSITY | Relevant Publications |
|--|--|
| [TSE95] | "A Formal Semantics of Object Models" R.H. Bourdeau and B. Cheng, <i>IEEE Trans. on Software Engineering</i> , Vol. 21, No. 10, pp. 799--821, October 1995. |
| [IWSSD-10] | "Object-Oriented Modeling and Automated Analysis of a Telemedicine Application," L. Campbell and B. Cheng, <i>IEEE International Workshop on Software Specification and Design</i> , November 2000. |
| [DSN00] | "Enabling Automated Analysis through the Formalization of Object-Oriented Modeling Diagrams," L. Campbell, B. Cheng, and E. Wang, <i>IEEE Dependable Systems and Networks</i> , June 2000. |
| [IJSEKE00] | "Formalizing the Functional Model within Object-oriented Design," E. Wang and B. Cheng, <i>International Journal on Software Engineering and Knowledge Engineering</i> , Vol 10, No. 1, February 2000. |
| [ICSE01] | "A General Framework for Formalizing UML with Formal Language," William E. McUmber, Betty H.C. Cheng, <i>Proceedings of IEEE International Conference on Software Engineering</i> , Toronto, 2001 |
| [RE01] | "Integrating Informal and Formal Approaches to Requirements Modeling and Analysis," L. Campbell and B. Cheng, <i>IEEE Requirements Engineering</i> , Poster Workshop, August 2001. |
| Software Engineering & Network Systems Lab | 62 |

| MICHIGAN STATE UNIVERSITY | Relevant Publications |
|--|--|
| [RHAS02] | "Adding Formal Specifications to Requirements Patterns," Betty H.C. Cheng, Laura A. Campbell, and Sascha Konrad, <i>International Workshop on Requirements for High Assurance Systems</i> , Essen, September 2002 |
| [RE02] | "Requirements Patterns for Embedded Systems," Sascha Konrad and Betty H.C. Cheng, <i>Proc. Of IEEE 10th International Requirements Engineering Conference</i> , Essen, September 2002 |
| [REJ02] | "Automatically detecting and visualizing errors in UML diagrams," Laura A. Campbell, Betty H. C. Cheng, William E. McUmber, and R. E. K. Stirewalt, <i>Requirements Engineering Journal</i> , 7(4):264-287, 2002. |
| [TSE02] | "Formalizing and Integrating the Dynamic Model for Object-Oriented Modeling," B. Cheng and E. Wang, <i>IEEE Transactions on Software Engineering</i> , Vol 28, No. 8, August, 2002. |
| [SPIN03] | "A Requirements Pattern-Driven Approach to Specify Systems and Check Properties" S. Konrad, L. Campbell, B. Cheng, M. Deng, <i>SPIN 2003</i> , May 2003 (Co-located with ICSE03.) |
| [RHAS03] | "Using Security Patterns to Model and Analyze Security Properties," S. Konrad, B. Cheng, L. Campbell, R. Wassermann, <i>IEEE Workshop on Requirements for High Assurance Systems</i> , September 2002. (Co-located with RE02.) |
| Software Engineering & Network Systems Lab | 63 |

| MICHIGAN STATE UNIVERSITY | Relevant Publications |
|---|---|
| <div data-bbox="354 375 440 401">[ASE04]</div> <div data-bbox="354 499 428 525">[Trace]</div> <div data-bbox="354 623 451 648">[Patterns]</div> <div data-bbox="354 711 423 737">[RE05]</div> | <p>``Automated Analysis of Timing Information in UML Diagrams," Sascha Konrad, Laura Campbell, and Betty H.C. Cheng), <i>Proc. of IEEE International Conference on Automated Software Engineering</i> (to appear), September 2004, Linz Austria.</p> <p>``Retrieval-By-Construction: A Traceability Technique to Support Verification and Validation of UML Formalizations," M. Deng, R.E.K. Stirewalt, and B. Cheng accepted to appear in <i>International Journal on Software Engineering and Knowledge Engineering</i>, Special issue on Traceability, October 2005.</p> <p>``Object Analysis Patterns for Embedded Systems," S. Konrad, L. Campbell, and B. Cheng, in <i>IEEE Transactions on Software Engineering</i>, vol. 30, no. 12, pp. 970-992, December 2004.</p> <p>``Facilitating the Construction of Specification Patterns-based Properties" S.~Konrad and B.H.C.~Cheng,13th IEEE International Requirements Engineering Conference (RE05),Paris, France, August-September 2005, (accepted to appear).</p> |
| <div data-bbox="342 890 479 989">Software Engineering & Network Systems Lab</div> | 64 |

| MICHIGAN STATE UNIVERSITY | Questions/Discussion |
|---|--|
| <div data-bbox="342 1209 418 1230">Overview:</div> <div data-bbox="342 1241 479 1262">Introduction</div> <div data-bbox="342 1272 479 1293">Background</div> <div data-bbox="342 1304 479 1325">Process</div> <div data-bbox="342 1335 479 1356">Conclusions</div> <div data-bbox="342 1367 479 1388">Future Work</div> |  |
| <div data-bbox="342 1730 479 1824">Software Engineering & Network Systems Lab</div> | 65 |