MICHIGAN STATE UNIVERSITY

Integrating Informal and Formal Approaches to RE

Betty H.C. Cheng

Software Engineering and Network Systems Lab

Department of Computer Science and Engineering

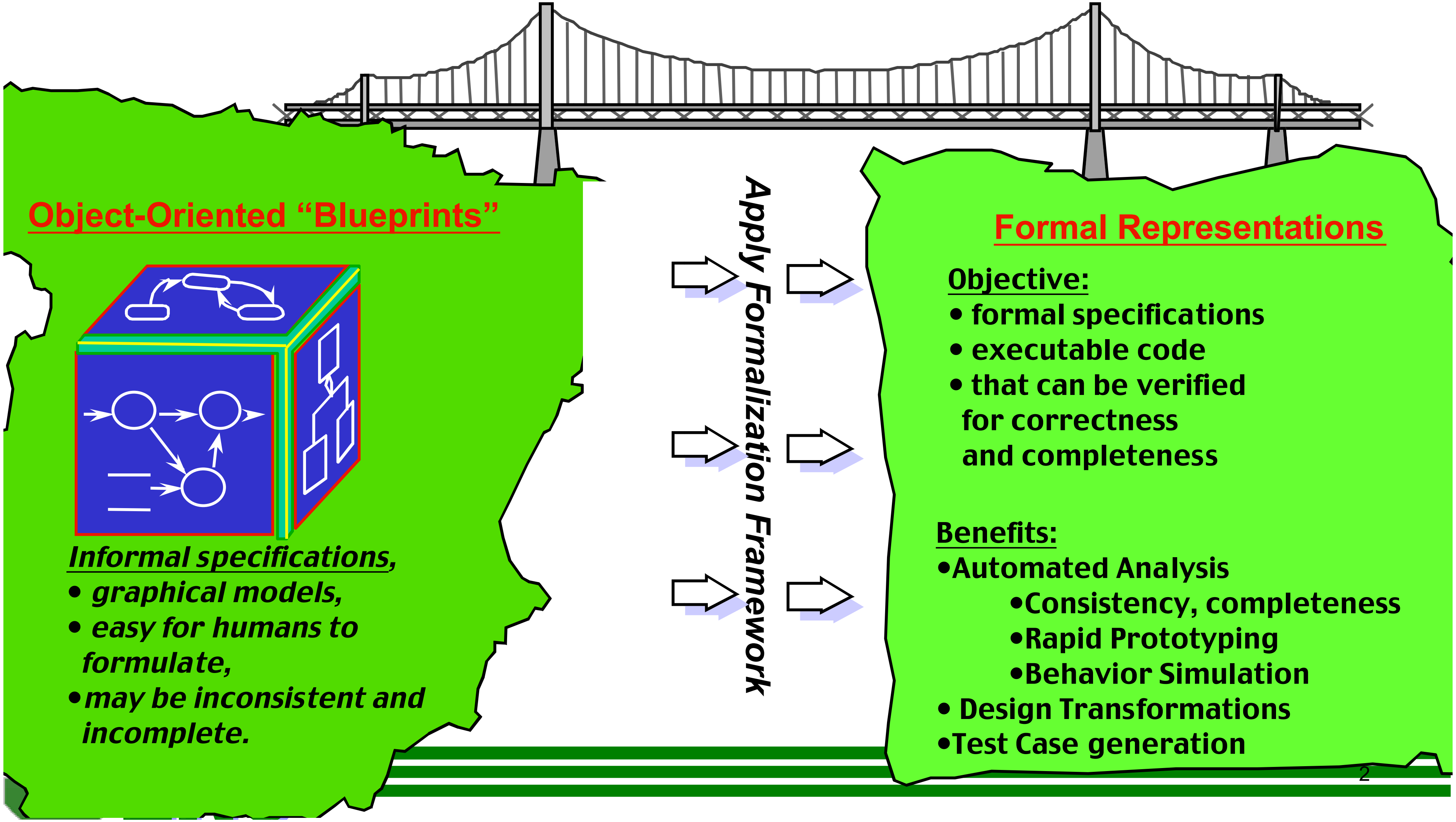
Michigan State University

www.cse.msu.edu/~chengb

SENS

MICHIGAN STATE UNIVERSITY

Bridge the Gap Between Informal and Formal Methods



Object-Oriented "Blueprints"

Informal specifications,
- *graphical models,*
- *easy for humans to formulate,*
- *may be inconsistent and incomplete.*

Apply Formalization Framework

Formal Representations

Objective:
- **formal specifications**
- **executable code**
- **that can be verified for correctness and completeness**

Benefits:
- **Automated Analysis**
  - **Consistency, completeness**
  - **Rapid Prototyping**
  - **Behavior Simulation**
- **Design Transformations**
- **Test Case generation**

General RE Issues

- Modeling for RE should support:
  - Decomposition
  - Domain-specific/independent abstractions
  - Tool support, including traceability mechanisms
- Analysis for RE must support:
  - Tool support
  - Ability to check for inconsistencies (local and global)
  - Validation capabilities (e.g., simulation)

Objectives of Integration Project

- **Overarching goals**:
  - *Broaden base of developers who can use rigorous software engineering techniques*
  - *Provide palatable path to more rigorous SE techniques*
  - *Leverage existing expertise and technology*
- Enable use of intuitive diagrammatic notations (UML)
- Provide path from UML to existing formal languages
  - Existing user base
  - Support Tools
- Enable automated analyses of model
  - Simulation
  - Model checking

Current Results

- General Framework for Formalizing UML diagrams
- Provide precise semantics for diagrams and their integration
- Establish consistency of mapping rules
- Allow choice of formalization language

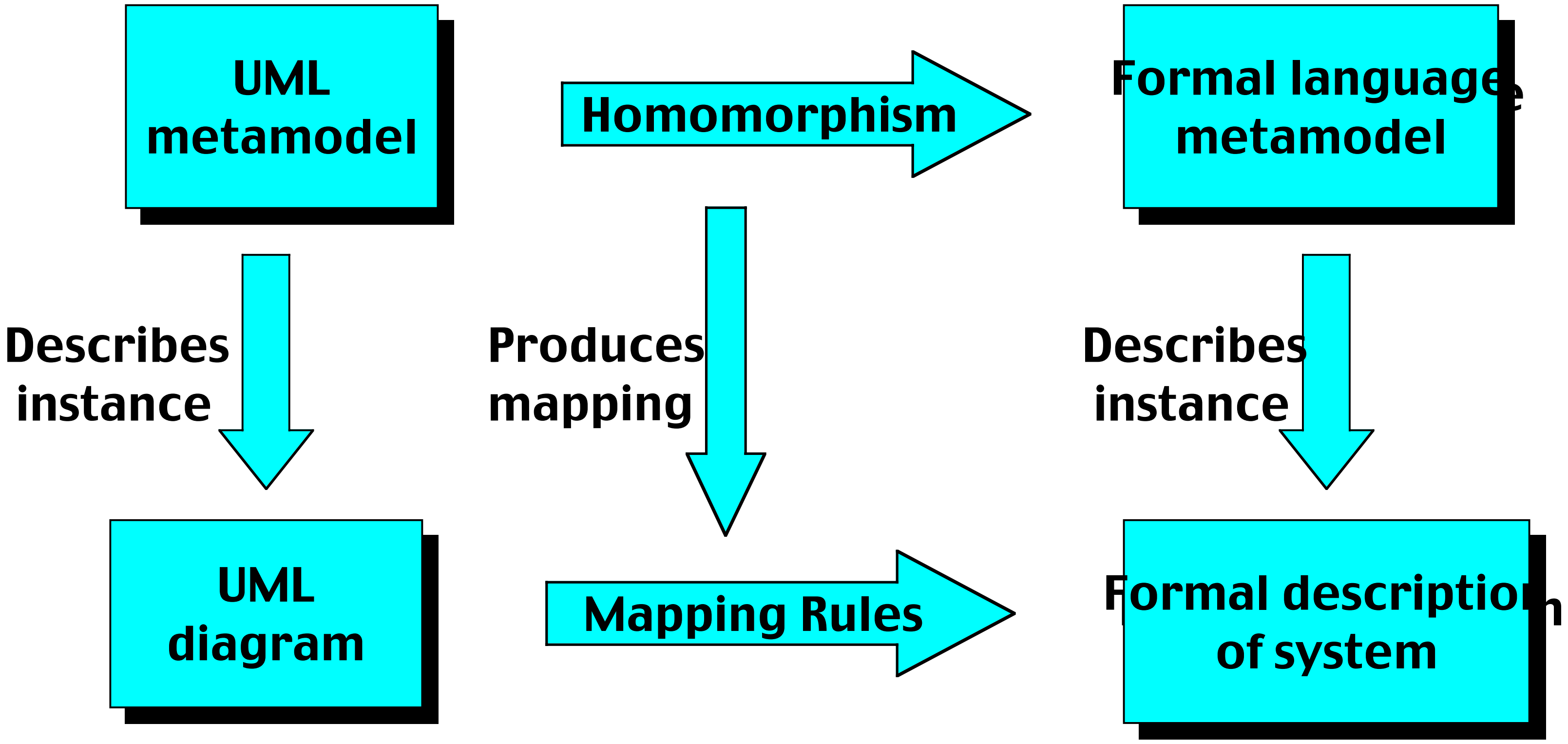
Background: UML

- “General-purpose” visual modeling language
  - *de facto* Standard
- (At least) nine different diagrams
- Diagrams described by metamodels:
  - A graphical model that describes syntax of model
- Therefore, nine different metamodels

UML Metamodel

- Metamodel defines UML syntax using class diagram notation.
- Semantics not defined by metamodel
- *Note:* Any language or diagram syntax can be defined with a metamodel

Metamodel mapping



UML metamodel

Homomorphism

Formal language metamodel

Describes instance

Produces mapping

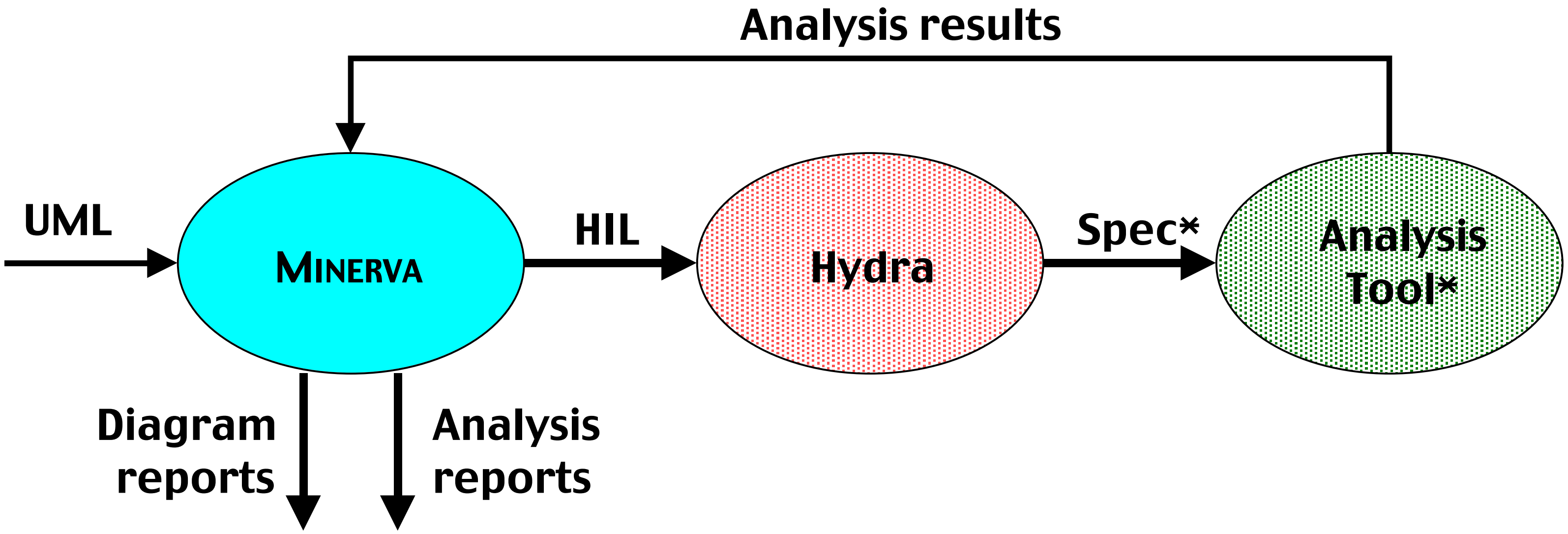
Describes instance

UML diagram

Mapping Rules

Formal description of system

Tool Support



Analyses Supported

- **Structural**
  - well-formedness
  - within and between diagrams
  - Tool support:
    - **MINERVA and Hydra**
- **Behavioral**
  - simulation
  - model checking
  - Tool support:
    - existing analysis tools (SPIN)

Visualization Support

- Within the original UML diagrams:
  - Highlights **structural anomalies** and **inconsistencies**
  - Quick and easier detection of errors
- Trace data visualization
  - Obtained from simulations or counterexamples
  - Animate existing **state diagrams**.
  - Explore how to automatically generate
    - **collaboration** and **sequence diagrams** from trace data
      - augment the playback of state diagram execution.

Discussion

- How do we incorporate more information obtained from other RE tasks/approaches:
  - Elicitation process
  - What’s the bridge between Natural Language and graphical models for RE purposes?
  - Should we identify/develop “requirements patterns” for a given domain?
  - How can problem frames help with abstraction and decomposition?