



A General Framework for Formalizing Object-Oriented Modeling Techniques

Betty H. C. Cheng

Software Engineering and Network Systems Laboratory
Department of Computer Science and Engineering

Michigan State University
East Lansing, Michigan 48824

Chengb@cse.msu.edu

www.cse.msu.edu/~chengb

SENS

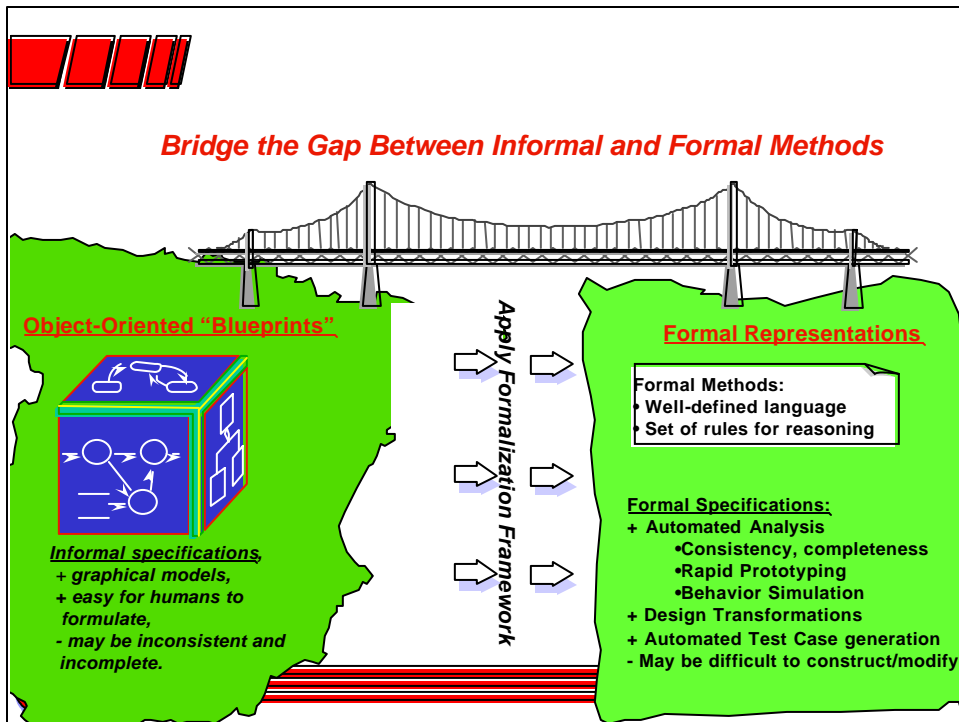


Acknowledgements

- Joint work with the following people:
 - ♦ Robert Bourdeau
 - ♦ Laura Campbell
 - ♦ William McUmbert
 - ♦ Enoch Wang
 - ♦ Ryan Stephenson
- Sponsored in part by:
 - ♦ National Science Foundation Grants:
 - * (CCR-9633391, CCR-9901017, EIA-0000433)
 - ♦ DARPA Grant (EDCS Program): F30602-96-1-0298
 - ♦ Motorola
 - ♦ Eaton Corporation
 - ♦ Siemens Automotive
 - ♦ Detroit Diesel Corporation

SENS





Overview

- Introduction
- Background
- Formalization Framework
- Validation:
 - ◆ Tool Support
 - ◆ Case Study
- Related Work
- Conclusions and Future Investigations

SENS



Objectives and Results

- **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
- **Specific Goals**
 - ◆ Enable use of intuitive diagrammatic notations (UML) for embedded system design
 - ◆ Provide path from UML to existing formal languages
 - * Existing user base
 - * Support Tools
 - ◆ Enable automated analyses of model
 - * Simulation
 - * Model checking

SENS



Domain: Embedded Systems

SENS





Background: Embedded Systems

- Code difficult to design and analyze
 - ◆ Time-dependent
 - ◆ difficult to instrument
 - ◆ often highly concurrent
- High level of robustness required
 - ◆ control real-world, physical processes

SENS



Informal Modeling Notation

SENS



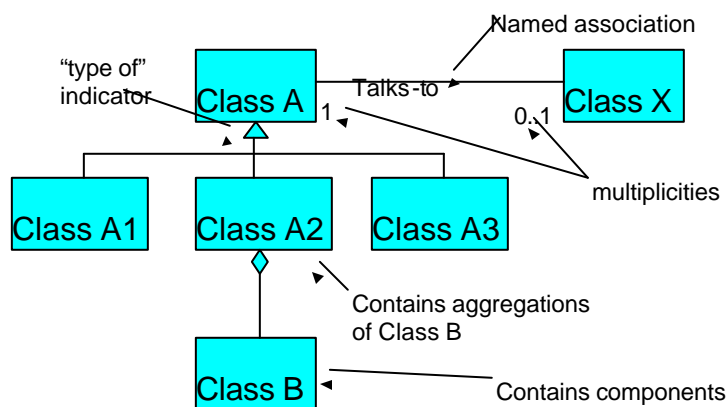
Background: UML

Unified Modeling Language

- “General-purpose” visual modeling language
 - ◆ *de facto* Standard
- (At least) nine different diagrams
 - ◆ use case, class, state, interaction (2), implementation (2), etc
- Diagrams described by metamodels:
 - ◆ A graphical model that describes syntax of model
- Therefore, nine different metamodels

SENS

UML Class Diagram



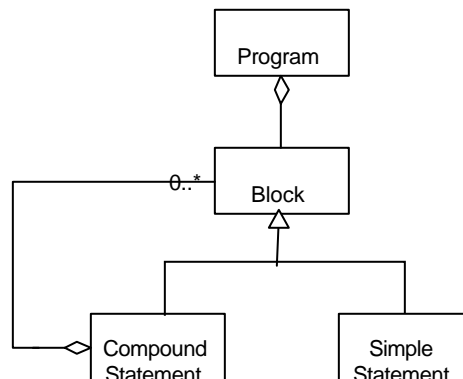
SENS

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

SENS

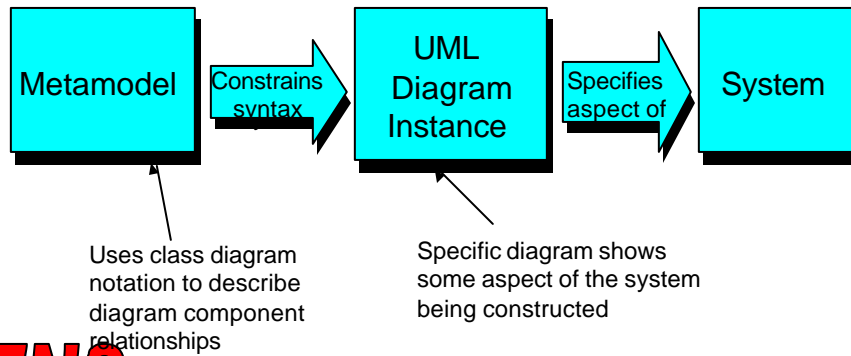
Example Metamodel



SENS



Metamodel - Diagram - System Relationship



SENS



Target Formalization Languages

SENS

Background: VHDL

- IEEE standard language
- Intended for abstract description of hardware
- Uses multiple, concurrent, communicating processes
 - ◆ Communication through “signals”
- Syntax is Ada-like, procedural in nature
- Models can be “executed” in simulation.

SENS



Background: Promela (SPIN)

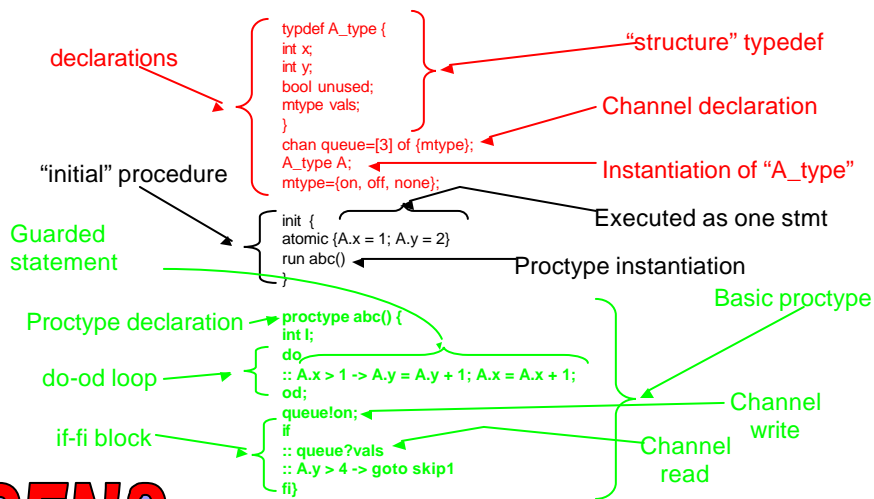
- Promela is language for SPIN model checker
 - ◆ Simulation and model checking of concurrent systems
- SPIN: commonly used in telecommunication domain
 - ◆ Developed by Bell Labs (now Lucent part)
 - ◆ Protocol verification
- Guarded Command Language + CSP + C
- Collection of processes, channels, and variables

SENS





Background: Promela Example



SENS



General Formalization Framework

SENS



Homomorphisms

Preserve operations, hence structure and semantics

$$h(a \oplus b) = h(a) \otimes h(b)$$

With the mapped objects

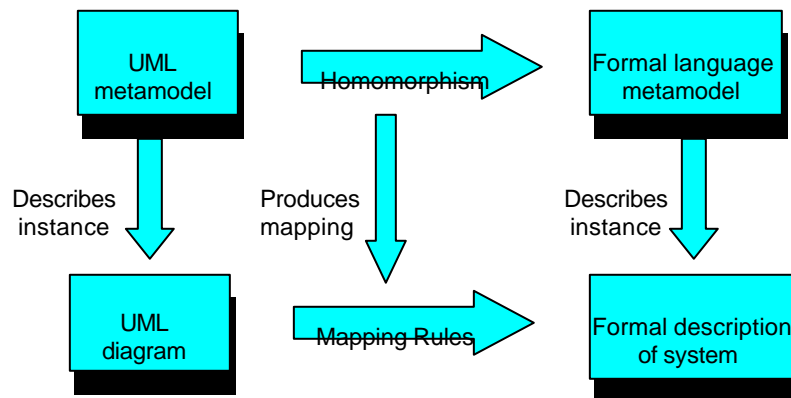
This operation in this system with these objects (a & b)

Does the "same thing" as this operation in this system

SENS



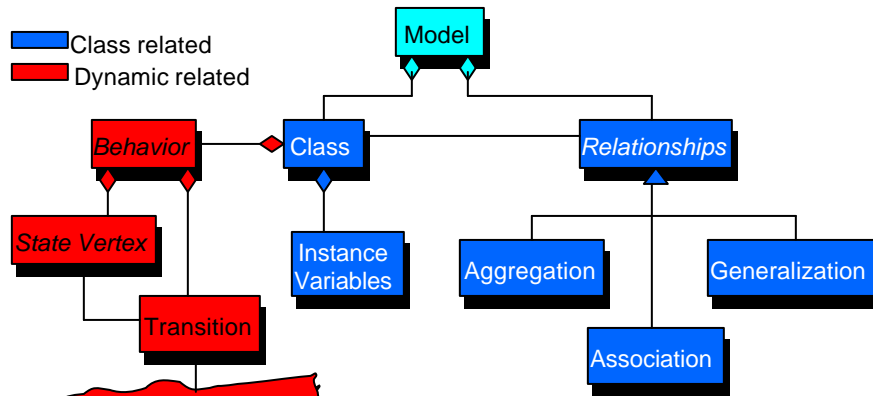
Metamodel mapping



SENS



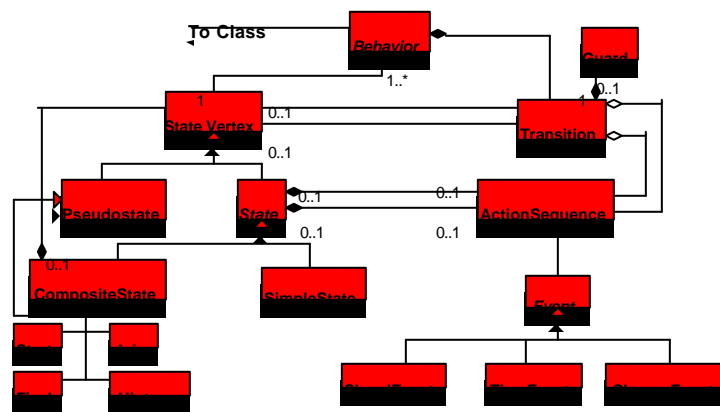
Unified Class/Dynamic Metamodel



SENS



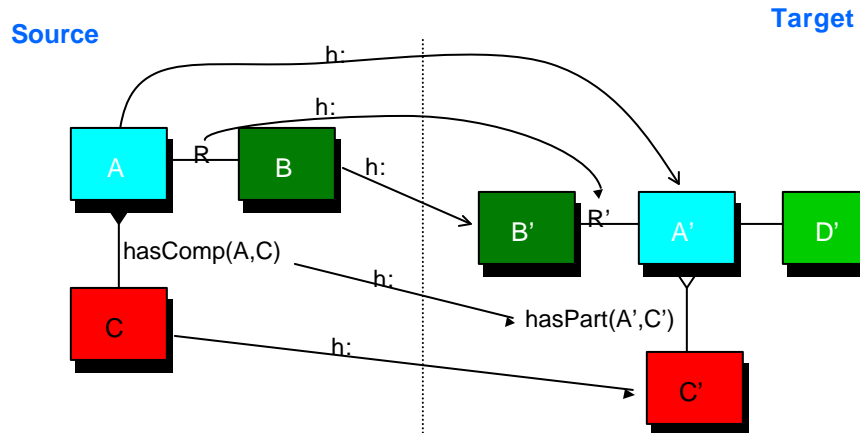
Dynamic Model Portion of Unified Metamodel



SENS



Example Metamodel Mapping



SENS



Introduction to Mapping Rules

- VHDL used for embedded systems
 - ◆ VHDL contains timing notations
 - ◆ Many commercial tools available
 - ◆ Comprehensive simulation capability
- SPIN used in industry
 - ◆ Spin provides model simulation and checking
- Concurrency is a feature of both

SENS



Promela Class Diagram Mapping Rules

- **Classes** (objects) map to **proctypes**.
- **Relationships** map to **channels**.
- **Instance variables** map to global **typedef** structures.

SENS



Promela Dynamic Model Mapping Rules

- **Simple states** map to blocks of Promela statements.
- **Transitions** map to **goto** and **run()**
- **Composite states** map to **proctypes**
- **Events** map to channel **writes/receives**
- **Pseudo-states** map to blocks of various Promela statements

SENS

SPIN Analyses

- Random simulation
- Exhaustive search of states
 - ◆ State transition system checked by temporal logic assertions
 - ◆ Often provides counter-examples (path to problem state)
 - ◆ “Easier” than theorem proving
- Better than simulation when precise timing not required

SENS

Summary of Mappings

<u>VHDL</u>		<u>Structure</u>		<u>Promela</u>
Ent/Arch	←	Class	→	proctype
Port signature	←	Relationship	→	channels
procedure	←	State	→	Labeled block of statements
Ent/Arch	←	Composite State	→	proctype
Write to signal	←	Event	→	Channel assignment

SENS

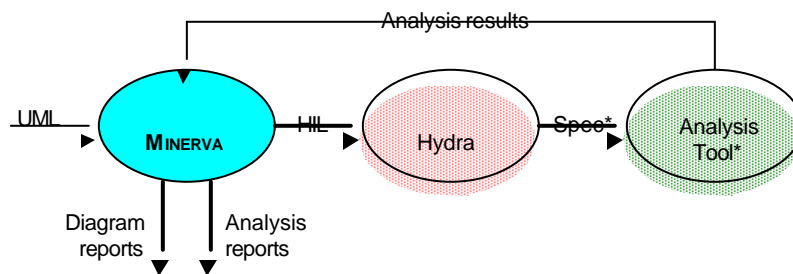


Tool Support

SENS

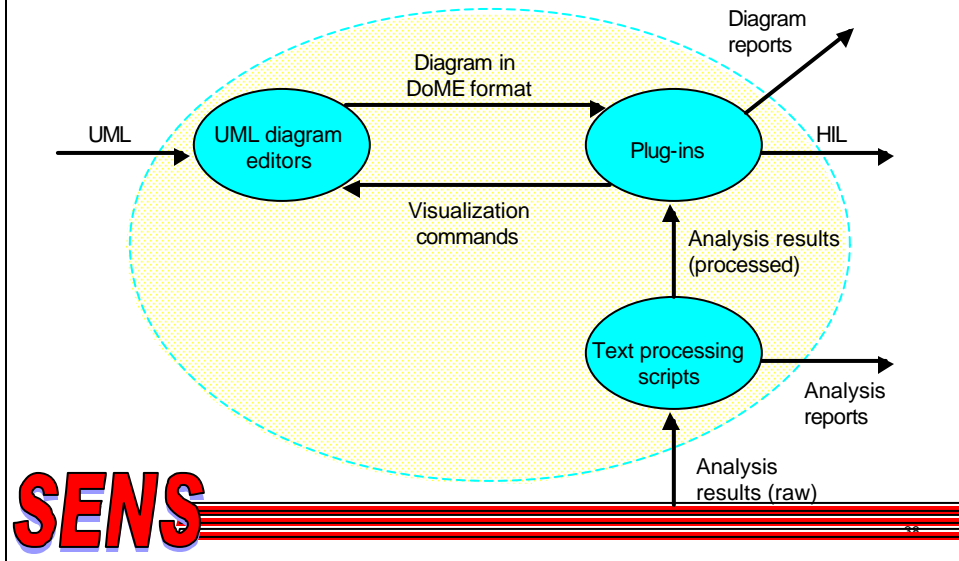


Tool Support

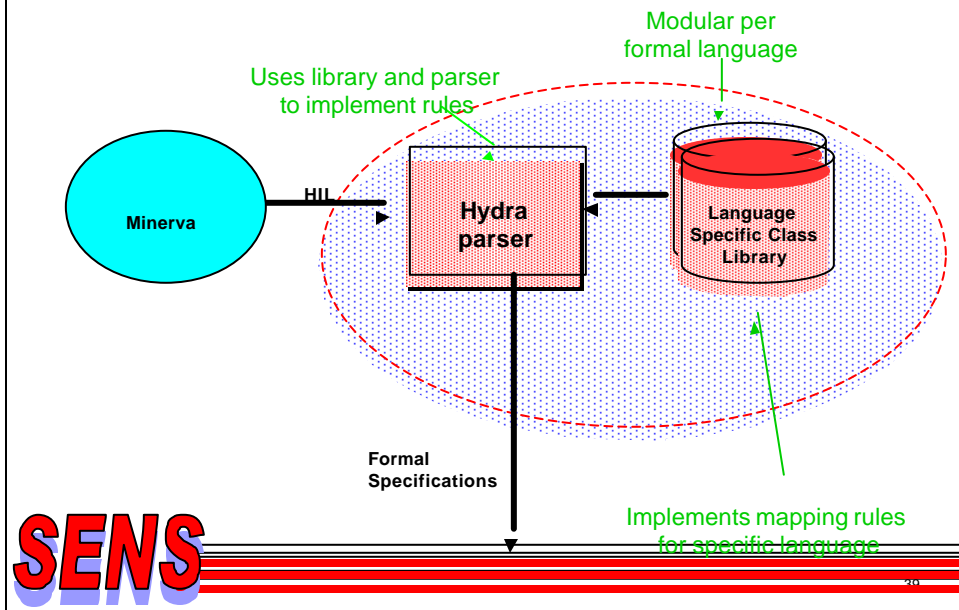


SENS

Architecture of Minerva



Hydra Translation Tool

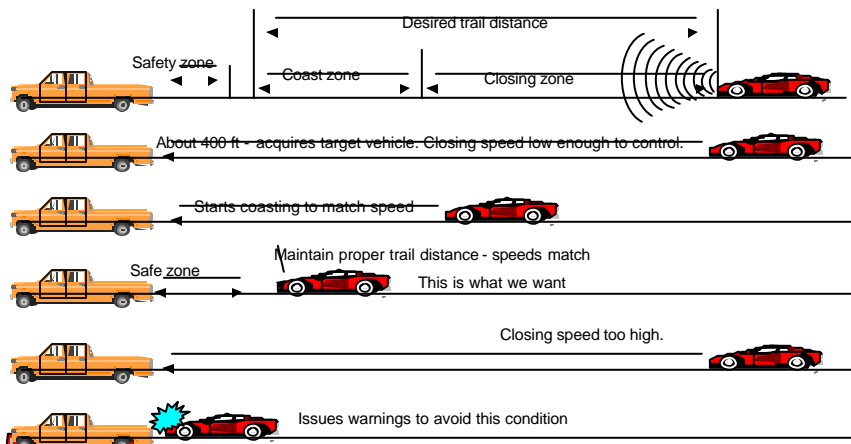




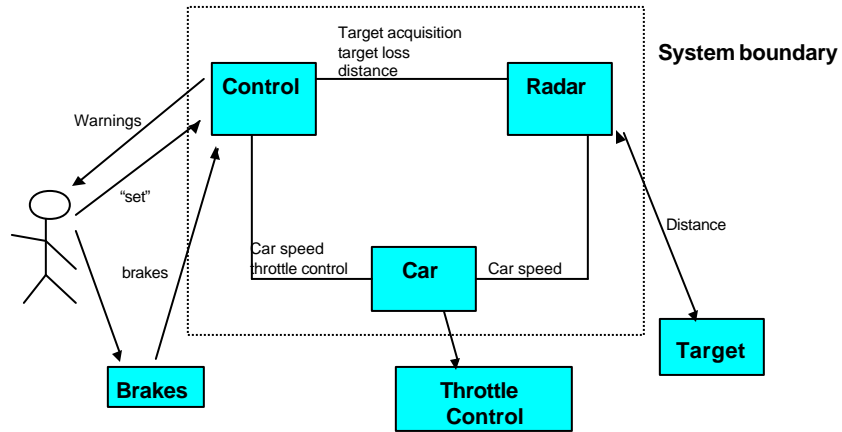
Industrial Case Study



Smart Cruise Requirements

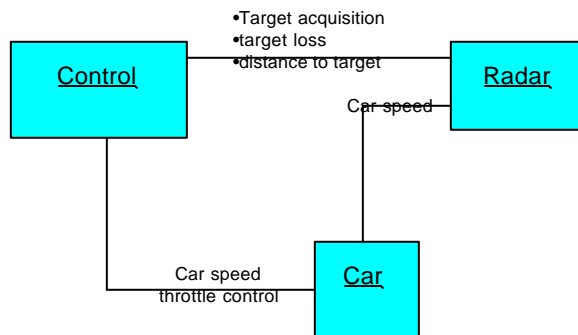


Class - Context Diagram



SENS

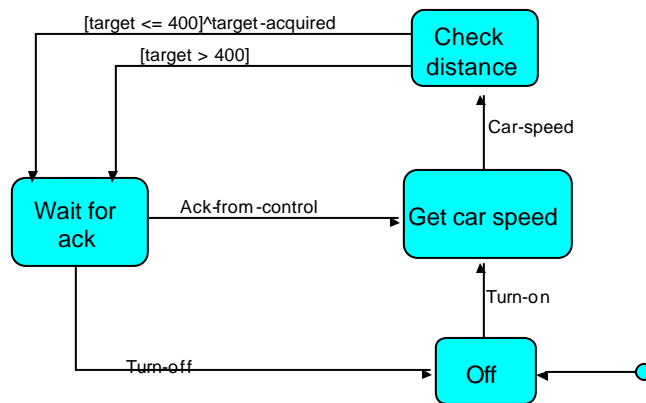
Smart Cruise Class Model



SENS



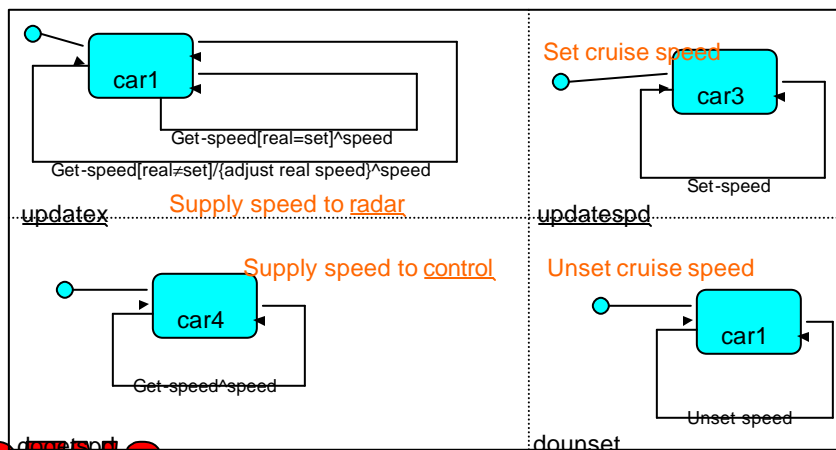
High Level Radar Dynamic Model



SENS



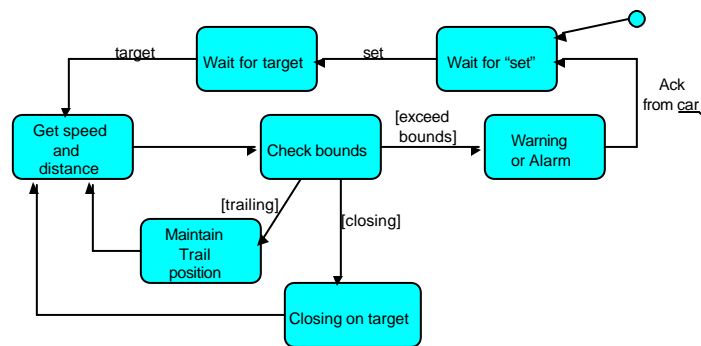
Car Dynamic Model



SENS



High Level Control Dynamic Model



SENS



SPIN Analyses Performed

- Random simulation
- State reachability
- State reachability with assertions
- Progress loop analysis (cycle checks)
- Model checking with temporal claim
- Model checking with temporal claim and non-deterministic execution paths.

SENS

Use of Simulation

- Check that model runs (does not deadlock)
- Model appears to achieve basic requirements
- Model not erratic (simulation is random)
- Exercise common paths
- Explore extremes for initial proper behavior
- *Basically, high level debugging strategy*

SENS



State Reachability Analysis

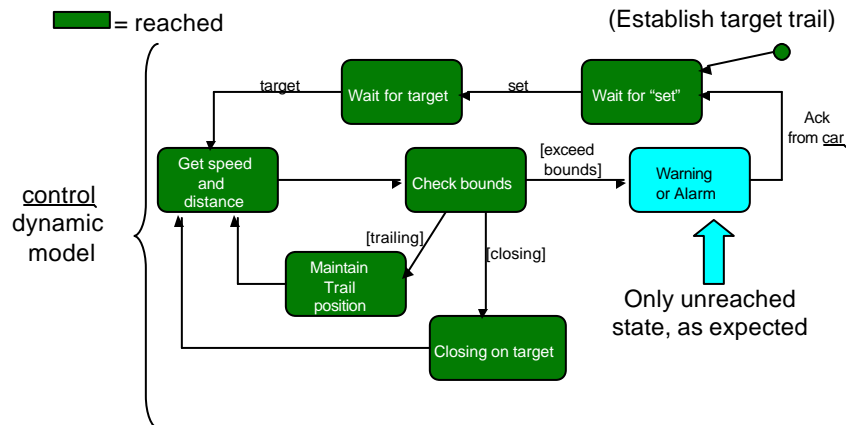
- Reachability is exhaustive (unlike simulation)
- For common scenarios,
 - ◆ ensure set of states is correct and
 - ◆ exception states not entered
- For exception scenarios,
 - ◆ ensure exception states entered

SENS





State Reachability for Normal Scenario



SENS



SPIN Progress Loop Analysis

- Ensures no cycles of only unmarked states.
- Reports cycles unless state(s) are marked.
 - ◆ If nothing marked, reports cycles
 - ◆ If known cycles are marked, reports unexpected cycles

SENS



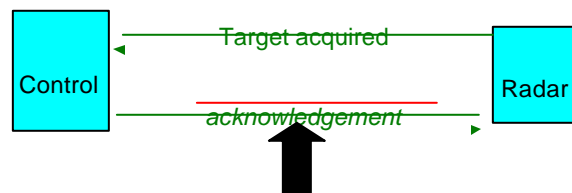
Model Checking Tests

- Car achieves trail position, and stays there.
Three checks:
 - ◆ Once in *idle*, model never comes back
 - ◆ when *target* sent, *ack* replied
 - ◆ Remove *ack* to demonstrate check works
- Brake application leads to return to idle state.
 - ◆ Revealed missed an event on transition

SENS



Ensure Target is Never Missed Demonstrate Check Works



Remove this message to force
claim to fail

This check failed (as expected)


SENS



Related Work

- Object-Orientation and Embedded Systems
- Formalization of UML
- Formalization of OO Modeling Techniques

SENS



Embedded System Methodologies

- Ad Hoc (frequently used in industry)
- Structured methods - RTSA
 - ◆ [Ward & Mellor, Hatley & Pirbhai]
 - ◆ RTSA models semi-formal, uses top-down
- Hybrid OO -- RTOOSA [Ellis]
 - ◆ Still structured, semi-formal - little object use
- OO, non-UML (ROOM) [Selic, Gullekson]
 - ◆ Formal, but unusual OO model
- OO, UML based [Douglass]
 - ◆ Semi-formal. No behavior verification

SENS





Formalization of UML

Precise UML (pUML) based UML on Z

- * [Evans, Clarks, Bruel, France, Lano]
- ◆ Attempts to provide direct manipulations of diagrams
- ◆ But no dynamic behavior mapping
- ◆ No way to verify behavior or properties, other than potential theorem prover
- Latella *et al*
 - ◆ Formalized UML state diagram in terms of hybrid automata

SENS



Other OO Formalizations

- OCL shown to have problems
 - ◆ [Mandel & Cengarle]
- Fusion well-defined process, but informal semantics [Coleman, *et al*]
- TROLL formally defined, but no checkers or simulation capability
 - ◆ [Jungclaus, Saake, Hartman, Sernadas]
- Formalized OMT with rules but no general mapping framework [(Wang & Cheng), (Bourdeau & Cheng)]
 - ◆ Rules specific to LOTOS

SENS



Overview of Contributions

- General framework for providing semantics.
- Unified UML Class/Dynamic metamodel.
- Mapping to VHDL and Promela.
- Means to perform simulation and model checking from semi-formal diagrams.
- Systematic process for developing OO graphical models for embedded systems.



SENS



Where does this all fit in Big Picture?



SENS



Meridian: Automating Development of IDAs

PIs: B. Cheng, L. Dillon, P. McKinley, K. Stirewalt

- Interactive Distributed Applications (IDAs)

- Examples:

- ◆ On-board driver/pilot navigation systems.
- ◆ Computer-supported collaborative work environments.
- ◆ Distributed interactive simulation.



Increasing interest fueled by:

- The World-Wide Web.
- Middleware technology (e.g., CORBA, DCOM, JavaBeans).
- New network services and protocols.

SENS



Meridian Research goals

- Improve quality of IDAs.
 - ◆ Better IDAs (reliable, maintainable, extensible).
 - ◆ Better development (faster, cheaper).
- Advance state of automated software-engineering (ASE) practice.
 - ◆ Incorporate ASE techniques into mainstream development.
 - ◆ Apply various formal methods in a new domain.
- Identify end-to-end automation techniques that take advantage of multiple phases of development.

SENS



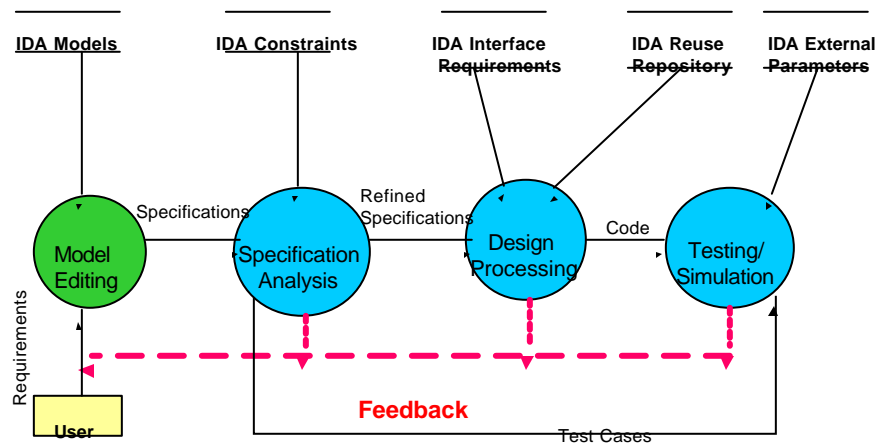
Meridian Practical goals

- To have techniques adopted in practice:
 - ◆ Must complement existing design methods and notations.
 - ◆ Otherwise, acceptance must overcome stiff economic hurdles.
- Implications:
 - ◆ Designers should not reformulate designs in a formal notation.
 - ◆ Designers should not have to view the output of a formal analysis tool.
- We chose (UML) for representing IDA designs.

SENS



Meridian Vision



SENS

Summary of Contributions

- General framework for constructing mappings of diagrams to formal target languages
 - ◆ Framework enables use of rigorous techniques to establish completeness, consistency, and correctness of mapping rules.
- A set of rules for generating VHDL and Promela specifications from UML
- Enable behavior simulation and analysis on informal diagrams via their formal specifications
- Systematic process for developing OO graphical models for embedded systems

SENS



Current and Future Research

- Consider other UML Diagrams:
 - ◆ Use Case: provide high-level user goals
 - ◆ Interaction Diagrams (Sequence and Collaboration): model behavior of specific scenarios
- Add temporal and real-time constraints
- Explore modified UML semantics
 - ◆ Adapt semantics to application?

SENS





Current/Future Research

- Mapping to SMV
 - ◆ Different temporal logic (CTL)
 - ◆ Different analysis capabilities (e.g., fairness)
- Explore the use of specification patterns to guide analysis capabilities
- Domain-specific refinement of UML diagrams
 - ◆ Move closer towards implementation
 - ◆ Use of Design Patterns and Frameworks

SENS



Discussion...

SENS

