

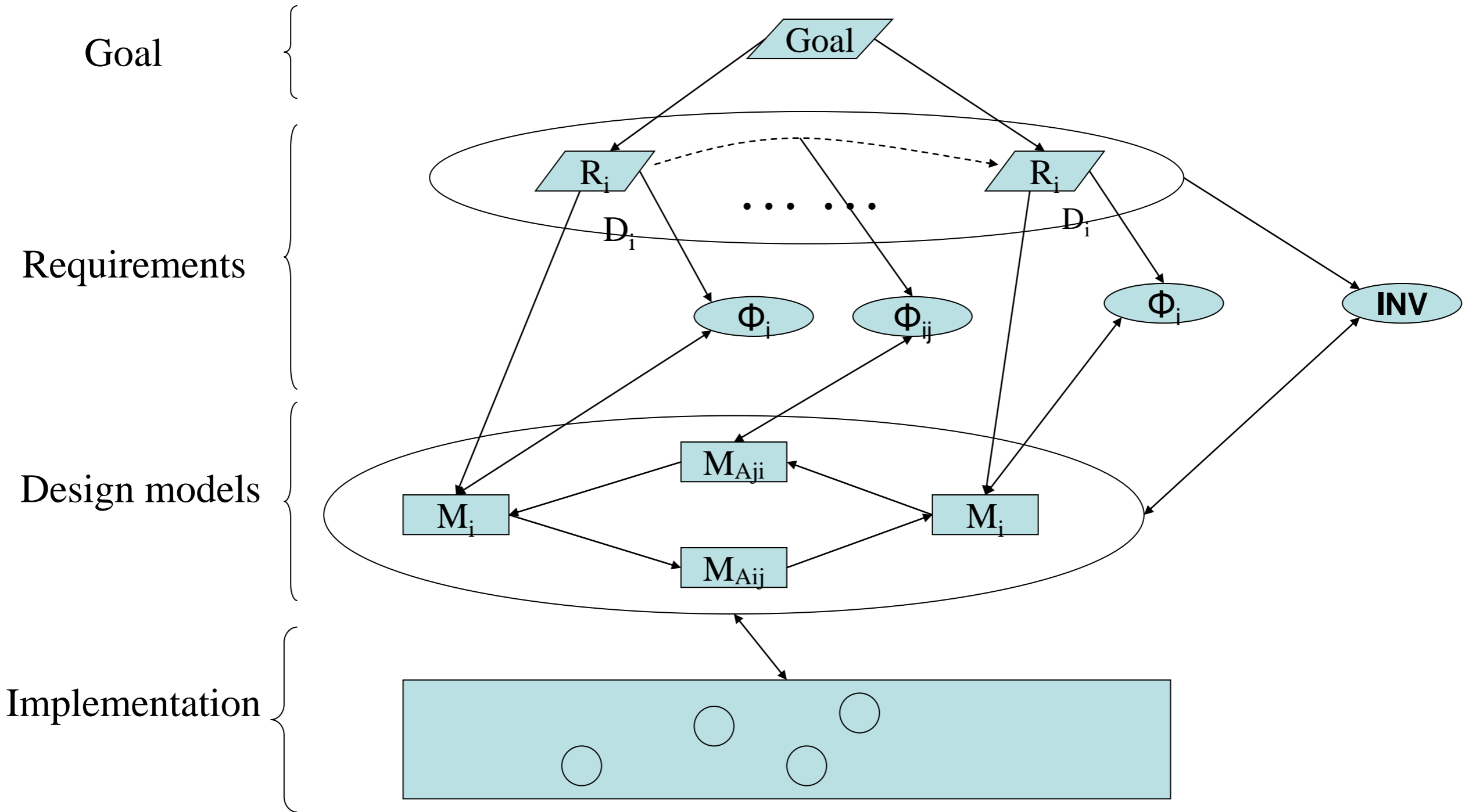
Assurance for Dynamically Adaptive Systems: Dealing with Uncertainty at the Requirements Level

BETTY H.C. CHENG

Software Engineering and Network Systems Laboratory
BEACON: NSF Center for Evolution in Action
Department of Computer Science and Engineering
Michigan State University

This work has been supported in part by NSF Cooperative Agreement No. DBI-0939454, and NSF grants CCF-0541131, IIP-070329, CCF-0750787, CCF-0820220, CNS-0854931, CNS-0751155, and CNS-0915855. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation. This work has also been supported in part by Army Research Office grant W911NF-08-1-0495, Ford Motor Company, and a Quality Fund Program grant from Michigan State University

Model-Driven Process



Implementation

Goal

Goal

Requirements

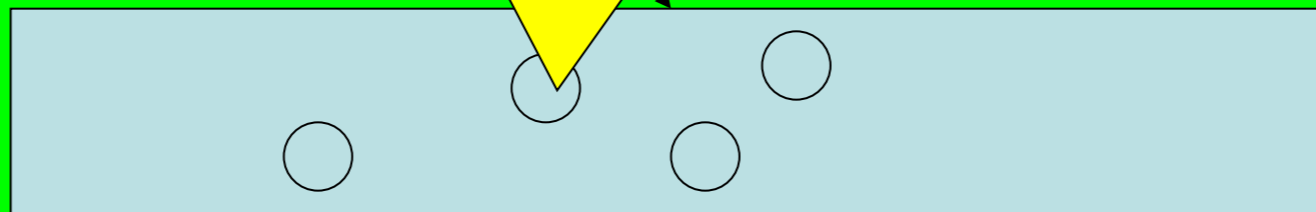
- Introduce adaptation capabilities into *legacy systems* (non-invasively) [**WOSS02**, DOA04, DEAS05, IEEEComputer04]
- Still need to satisfy functional/non-functional properties [MiSE07, M@RT07]
- How do you achieve **adaptation safely**? [WADS05, JSS06]

Design models

M_i

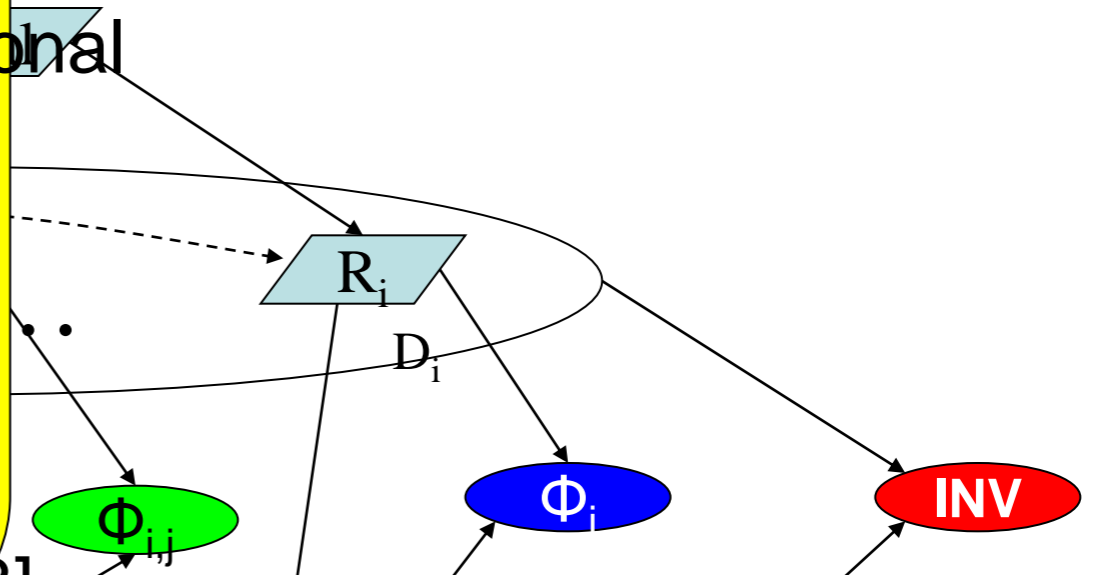
M_j

Implementation

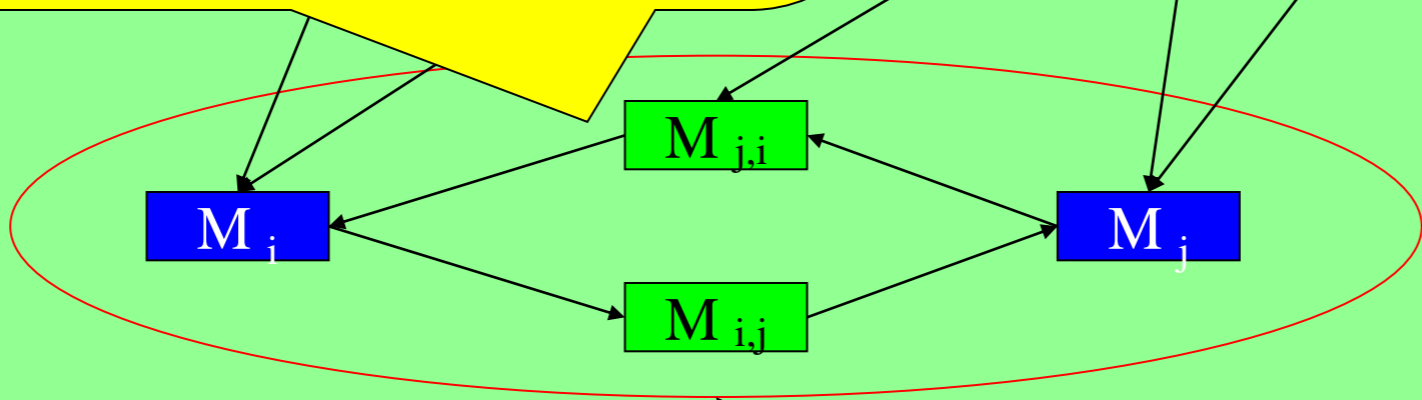


and Analysis

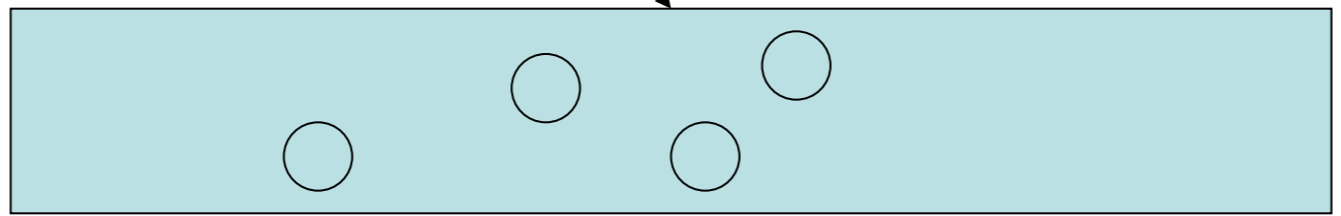
- What *system designs* can be used to realize the system requirements
- Still need to satisfy functional/non-functional
- **properties** [ModularVerification-AOSD09]
- Separate *functional logic* from *adaptive logic* [ICSE06]
- Automatically generate different system Configurations satisfying properties [SEAMS07, ICAC08, ICAC09, MODELS08]



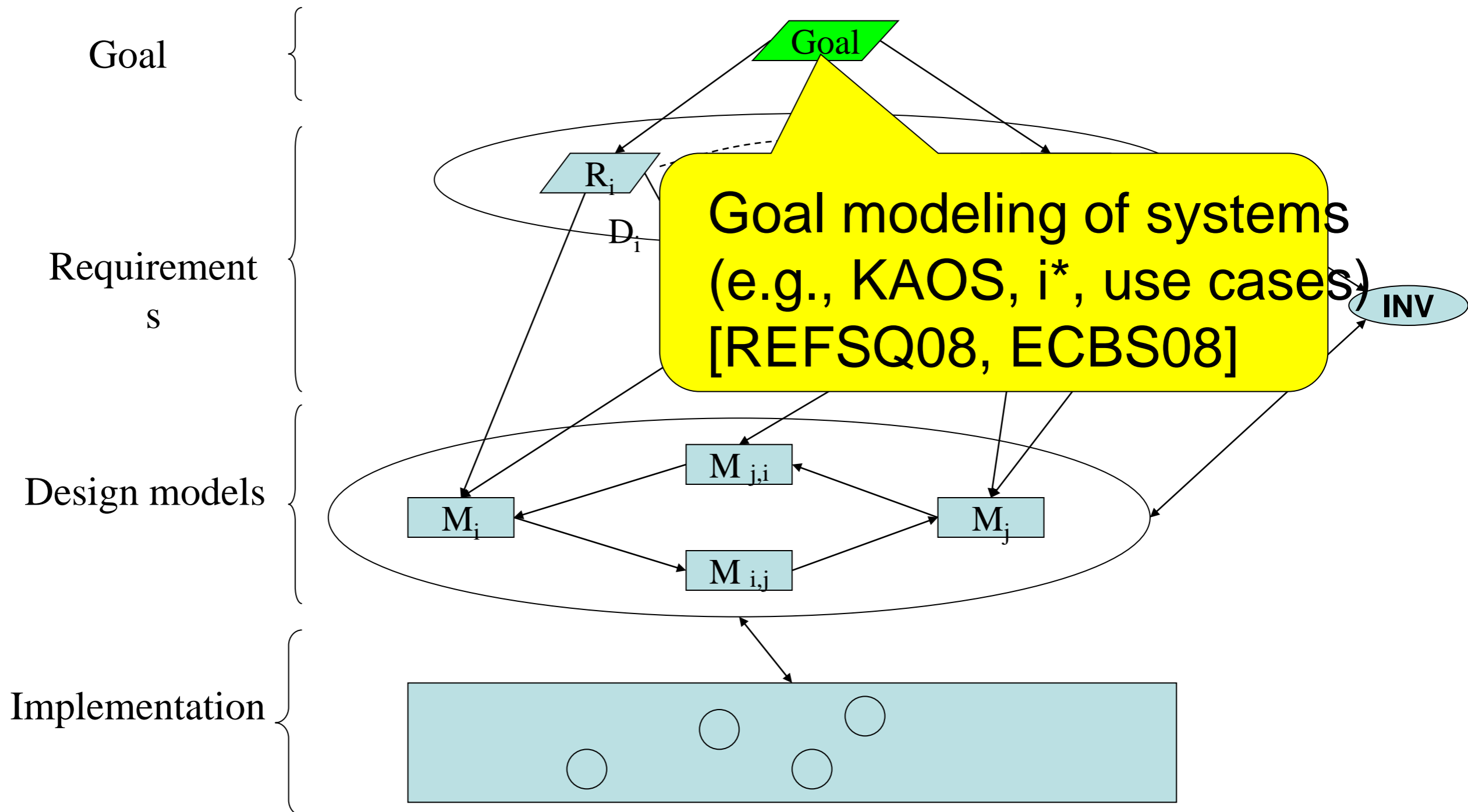
Design models



Implementation



System Goal



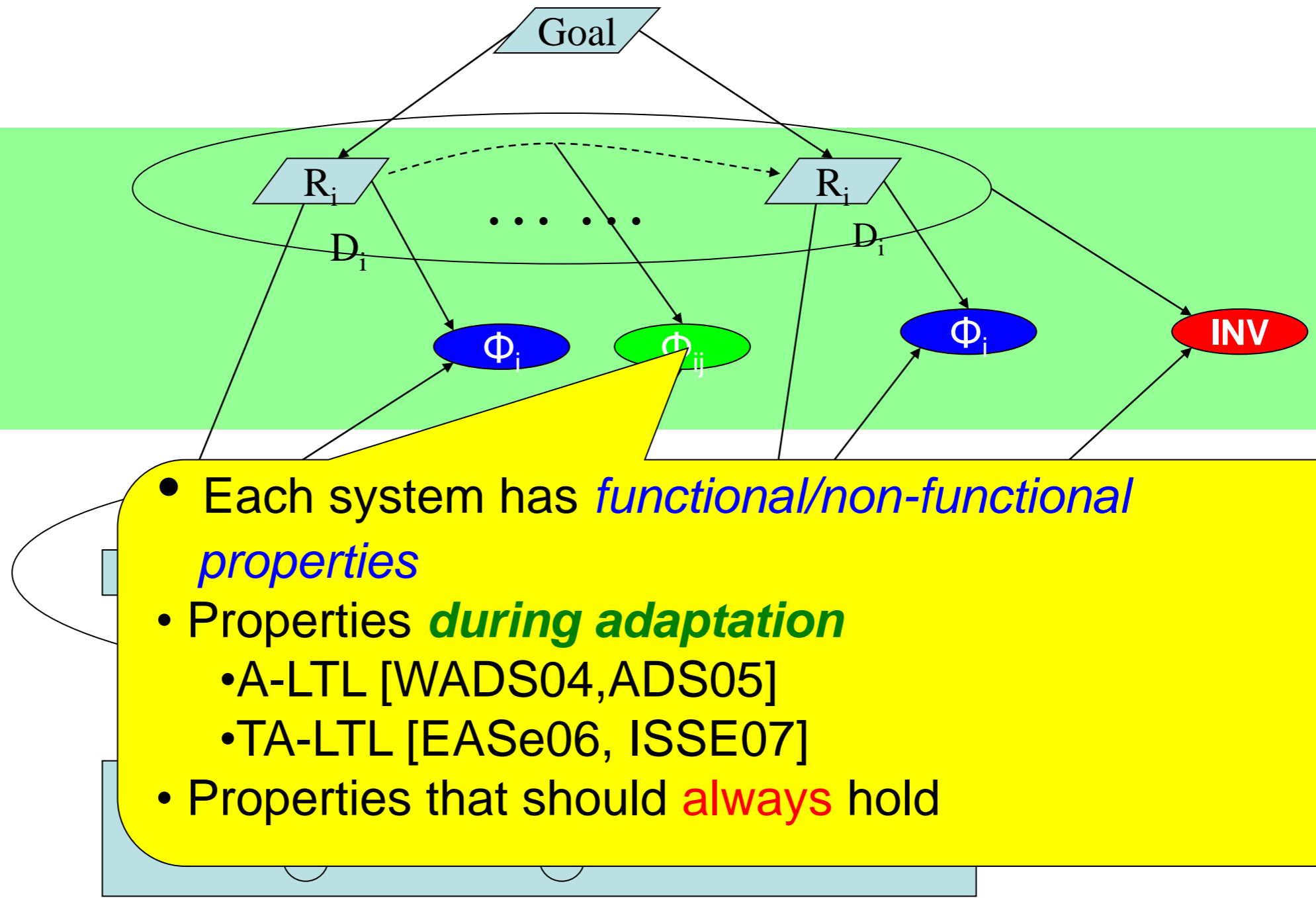
Requirements

Goal

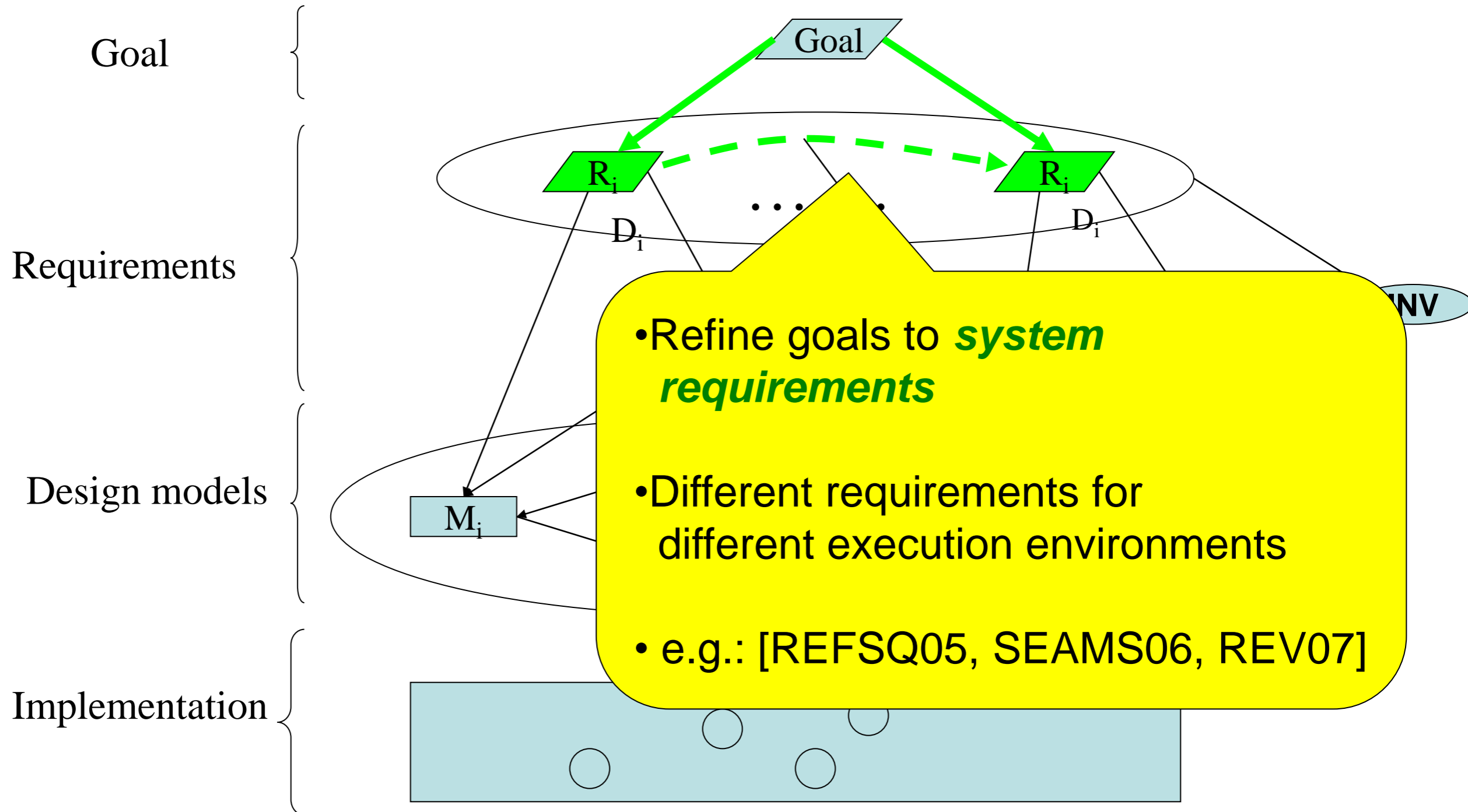
Requirements

Design models

Implementation



Goal Refinement



One Key Lesson Learned

- A dynamically adaptive system (DAS) must:
 - *monitor* itself and its execution environment.
 - decide *if, when, and how* to apply a reconfiguration.
 - *safely* reconfigure itself.
- Decision-making depends on how well a DAS is able to monitor itself and its environment.
 - unreliable monitoring information may cause a DAS to:
 - ▶ apply the incorrect reconfiguration strategy.
 - ▶ apply a reconfiguration when none is required.

Uncertainty in a DAS

- The monitoring capabilities of a DAS may be affected by uncertainty from:
 - sensor failures, variance in sensor accuracy / precision, unexpected environmental conditions, humans, etc



- It is practically infeasible to know the effects of uncertainty on monitoring requirements.
 - A DAS must cope with uncertainty

Uncertainty and Requirements

- **At development time:**
 - ▶ Specification of reqts, domain assumptions
 - ▶ Identify sources of uncertainty
 - ▶ Impact of uncertainty
 - ▶ Adaptive response
- **At run time**
 - ▶ Monitoring of requirements
 - ▶ Adaptations that satisfy/satisfice requirements

Development Time: Impact of Uncertainty

- Identifying *sources* and *impacts* of uncertainty:
 - Probability-based framework for augmenting goal refinement models with a probabilistic analysis of partial goal satisfaction [Letier04].
 - RELAX modeling language explicitly captures sources of uncertainty, how uncertainty affects goals, and how to mitigate such impacts [Whittle-RE09, Cheng-MODELS09].

The primary objective is to specify how uncertainty may impact the satisfaction of goals.

Impact of Uncertainty

- Automated testing of prototypes
 - Simulation-based validation of semi-formal requirements models [Seybold05]
 - ▶ records and replays previous user interactions with the application.
 - Automatically generate environmental conditions to test autonomous behavior [Nguyen09].
 - ▶ leverages evolutionary computation to find test inputs that violates requirements.

Discover inconsistencies between requirements, goal models, and prototypes during early stages of RE

Loki

- Automatically explores the effects of environmental uncertainty upon requirements satisfaction.
 - Use Genetic Algorithm to generate combinations of environmental conditions
 - specifies how sensors are fuzzed at run time (i.e. duration/severity of noise)
 - Apply environmental conditions in simulation of system
 - utility functions capture how the DAS satisfied (satisficed) requirements in the presence of uncertainty
 - Codify requirements in form of utility functions [Garlan06, Valetto09]
 - novelty search rewards solutions that produced behaviors different from those already encountered

Challenges in Requirements

Monitoring

- Monitoring is often computationally expensive, intrusive, and difficult to design [Garlan01].
 - i.e., continuously retrieving and analyzing real-time data.
- Monitoring involves tradeoffs [Prieto06, Wang07].
 - costs (i.e., overhead, energy consumed)
 - accuracy (i.e., coverage, coherence of gathered data).
- Many parameters to configure:
 - what data should be gathered?
 - how often should data be gathered?
 - how to evaluate gathered data?

How do these parameters affect a DAS?

Requirements Monitoring

- Static Approaches (i.e., Flea [Fickas95], ReqMon [Robinson])
 - Developers, at design time:
 - model requirements, and identify assumptions.
 - The system, at run time:
 - analyzes violations and selects appropriate responses.

Does not support adaptive monitoring.

Adaptive Monitoring

- Adaptive Sampling [Jain04, Zhou06]
 - to conserve energy, data gathering rates are changed in response to environmental conditions.
 - resource-constrained domains (i.e., wireless sensor networks)

Monitoring data is not mapped back to satisficement of requirements.

Adaptive Monitoring

- Policy-based [Talwar06].
 - event - condition - action engine.
- Statistical correlation models [Munawar06].
 - statistical model dictates which metrics to collect.
- Performance-based monitoring **Requires foresight to anticipate events and corrective actions**
 - configure filters to control distribution of monitoring data [Prieto06].
 - fine-tune the processing of log data by leveraging propositional encodings [Mylopoulos07].

Plato-RE

- Dynamically reconfigure monitoring infrastructure in response to current conditions
 - Inputs:
 - monitoring data
 - utility functions (for evaluating requirements satisfaction and satisficement)
 - fitness functions (for evaluating candidate monitoring configurations)
 - monitoring preferences (i.e., cost versus accuracy)
 - Outputs:
 - monitoring configuration that specifies, for each sensor, how often to gather environmental data

Plato-RE

- Enables a DAS to automatically balance monitoring concerns
 - increases monitoring activity in response to changing environments and requirements
 - satisfaction/satisficement
 - ▬ provides better monitoring coverage and accuracy at the expense of higher monitoring overhead
 - decreases monitoring activity in response to static environments
 - ▬ conserves energy

Take Home Message

- Need light weight techniques to monitor requirements satisfaction/satisficement
- Uncertainty needs to be addressed explicitly
 - Development time
 - Run time
- Make requirements as robust as possible at development time
- Think outside the box
 - Look for the right tool for the task
 - Look to other disciplines for useful tools (e.g., control theory, economic models, probabilistic models, evolutionary computation, etc.)