Toward a Reasoning Framework for Dependability

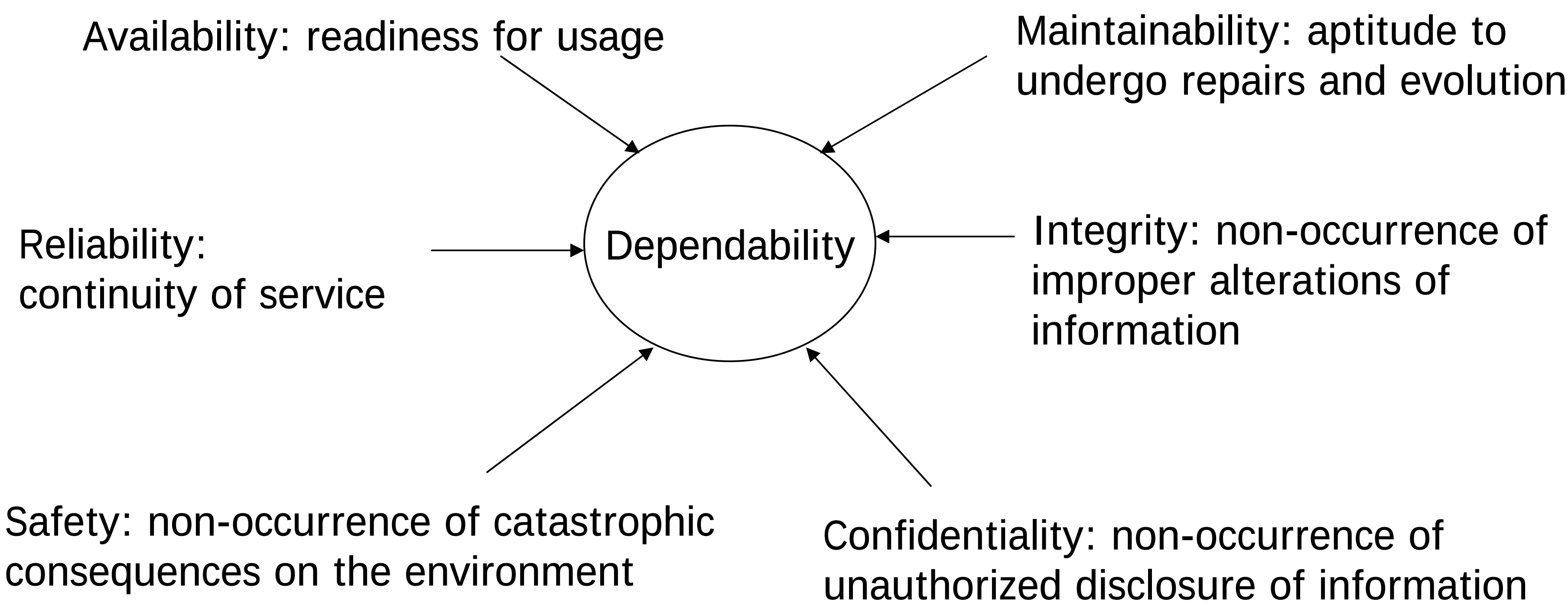
Tacksoo Im and John D. McGregor
{tim,johnmc}@cs.clemson.edu
School of Computing
Clemson University

Problem Statement

- How do we predict and evaluate the dependability of a software intensive system?
- How do we improve the dependability of software systems from the architectural level?
- Is it possible to codify architectural knowledge for dependability in a tool that provides the right information at the right time to the architect?

Definition of Dependability

Dependability is the ability of a system to deliver service that can justifiably be trusted (Avizienis et al., 2004)



Quality Attributes

- Non-functional properties of a software system.
- Difficult to categorize in which quality a certain aspect would belong.
  - “system slowdown” could be related to performance issues or usability
- Can be ambiguous, quality attribute scenarios resolve the ambiguity.
  - an example of a performance scenario: A garage door must detect an obstacle and halt within 0.1 seconds.

Reasoning Frameworks

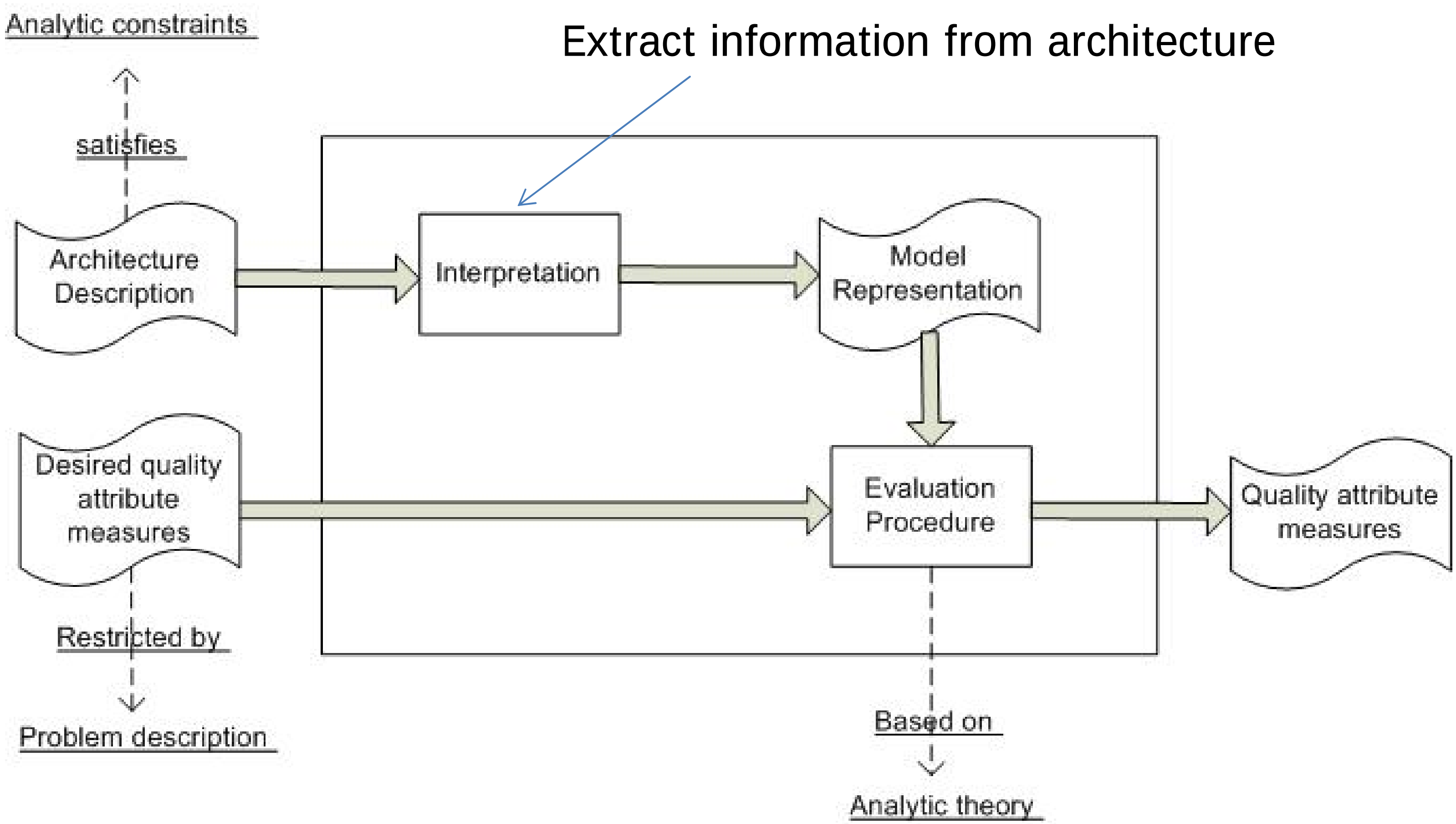
- Reasoning Frameworks are built for the following reasons:
  - Predict behavior before the system is built
  - Understand behavior after it is built
  - Make design decisions while the system is being built and when it evolves
- Each reasoning framework addresses a specific quality attribute.

Reasoning Frameworks (continued)

- Here are the definitions of the six elements in a reasoning framework.
  - Problem Description: the set of quality measures that can be calculated.
  - Analytic Theory: the foundations on which analyses are based.
  - Analytic Constraints: assumptions for using the theory.
  - Model Representation: a model of the architecture that is relevant to the analytic theory and acceptable for the evaluation procedure.
  - Interpretation: a procedure that generates the model from the architectural descriptions.
  - Evaluation Procedure: algorithm or formulae that calculate the specific measures of a quality attribute from a model representation.

Reasoning Frameworks (continued)

Extract information from architecture



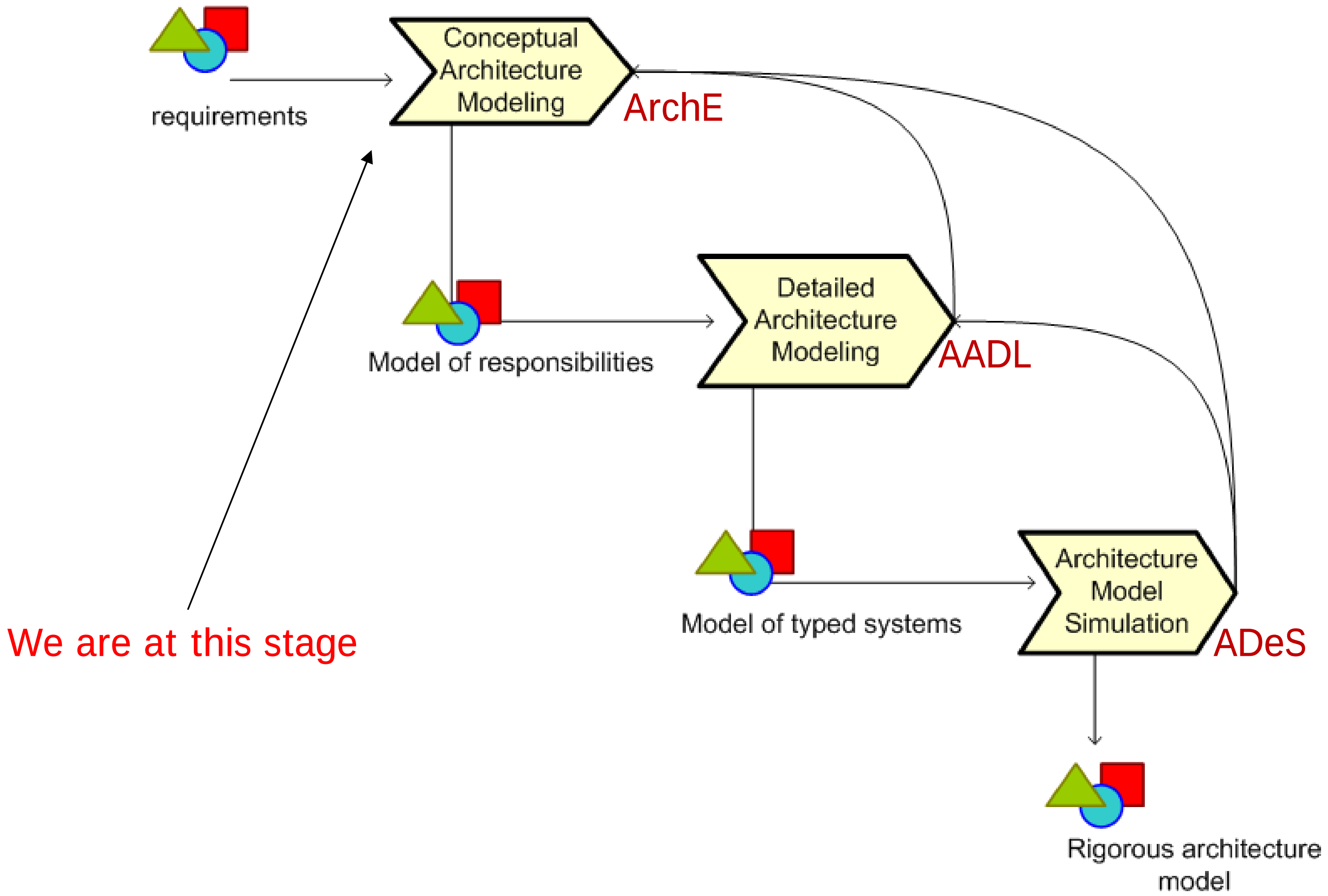
Reasoning Framework Diagram

Image from *Reasoning Frameworks (cmu/sei-2005-tr-007)* by Len Bass et al.

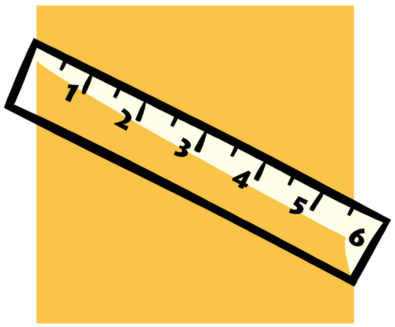
ArchE

- ArchE (Architecture Expert Design Assistant) is a tool for analyzing architectures using reasoning frameworks.
- The three core concepts of ArchE are:
  - Quality Attribute Scenarios: concrete scenario is a instance of a general scenario.
  - Reasoning Frameworks: converts scenario into quality-attribute specific model for analysis.
  - Responsibilities driven design: describes the role of a modules in a system and guides the reasoning framework to produce an architecture that satisfies the quality requirements.

Architecture Definition Process

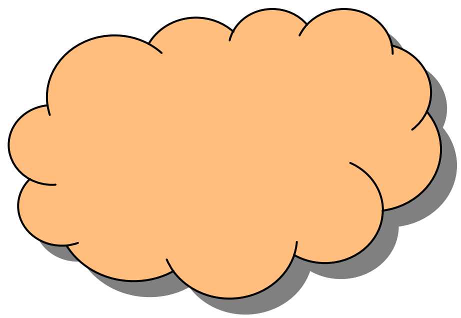


Quantitative vs. Qualitative Reasoning



Quantitative Attributes
Interval Scale
Analytic Theory
0.5 < 0.7

Reliability
Availability
Maintainability



Qualitative Attributes
Ordinal Scale
Non-Analytic Theory
Secret < Top Secret

Confidentiality
Integrity

Qualitative Attributes
Unordered Scale
Non-Analytic Theory
Case by Case

Safety

Qualitative Reasoning

- Qualitative Reasoning is reasoning with imprecise data.
- Often used to model tacit (implicit) knowledge.
- Certain attributes of software architectures are often hard to quantify.
  - Adding a “User Verification Module” increases confidentiality, but by how much?
  - What does it mean to satisfy a quality attribute scenario when there is no quantitative metric for a quality attribute?

Quantitative Reasoning Frameworks

- Quantitative Reasoning Frameworks are based on models that produce quantitative results based on well established analytic theories.
- Example analytic theory for each quantitative quality attribute.
  - Reliability: execution path based analysis.
  - Availability: structure of performance task architecture based analysis.
  - Maintainability: cost model based analysis.
- The models used by the analytic theories for each quantitative reasoning framework is limited by the scope of model.

Reliability Reasoning Framework

- Reliability
  - Measure of the probability of failure-free operation for a specified time.
  - Represented in terms of failures per hour (failure intensity).
  - Perceived reliability and an actual reliability
  - Can be modeled with reliability growth models or software architecture based reliability analysis models.
- In this work, we are calculating the perceived reliability of the system using software architecture based reliability analysis by Gokhale et al.

S. Gokhale, W.E. Wong, K. Trivedi, and JR Horgan. An analytical approach to architecture based software reliability prediction. Proceedings of IEEE International Computer Performance and Dependability Symposium (IPDS), 1998..

Reliability Reasoning Framework (continued)

- Problem Description: the estimation of reliability for a reliability scenario and the overall reliability based on the operational profile
- Analytic Theory: software architecture based reliability analysis.
- Analytic constraints: the responsibilities of the modeled software architecture are the components of the system.
- Model Representation: Nodes represent components and the arcs represent a dependency, sequence, or containment.
- Interpretation – the components in the model are generalized into responsibilities.
- Evaluation Procedure – consider the relationships between the responsibilities and the operational profile to calculate the reliability of the scenario with the formulas from the Gokhale model.

Reliability Reasoning Framework (continued)

- The analytic theory for reliability will the software architecture based reliability analysis which uses a state-based analysis model expressed as a DTMC (Discrete Time Markov Chain).
- The reliability of a component will be expressed as:

Number of times passed through a component

Average cumulative failures at time point t

$$R_i = e^{-\int_0^{V_i t_i} \lambda_i(t)dt} \approx e^{-a_i c_i(t_i)}$$

Time-dependent failure intensity

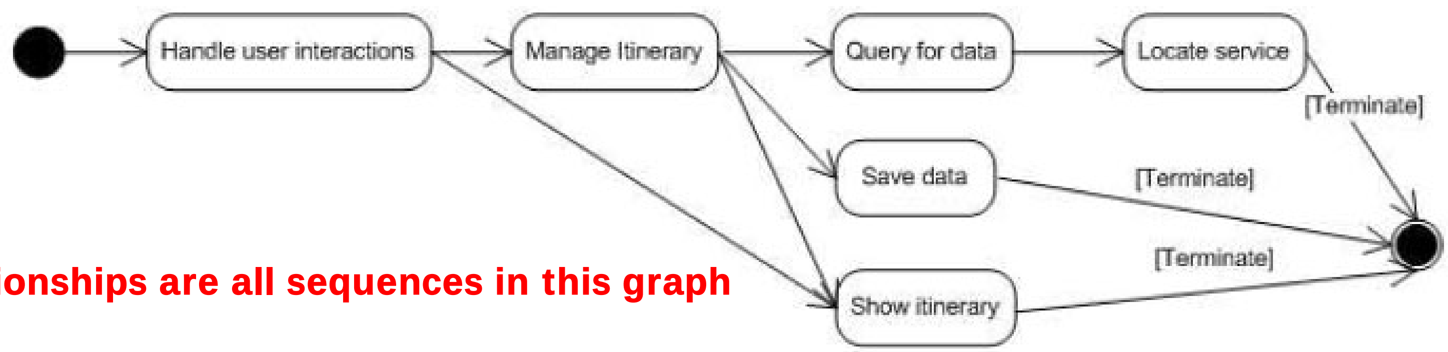
- The component reliability value is calculated by the user.

Reliability Reasoning Framework (continued)

- A reliability scenario: "When a user requests a new itinerary, the system shall compute it with a reliability of 0.95"
- The reliability scenario closely mirrors an execution path(s) through a software system.
- The components in the reliability model are the responsibilities and the execution paths are expressed with responsibilities and the relationships among them.
- The user of the reliability reasoning framework must provide the reliability value of each responsibility and the relationships among them.

Reliability Reasoning Framework (continued)

- There are three types of relationships between two responsibilities when computing reliability.
  - Contains: the reliability of the child node determines the reliability of the parent node
  - Dependency: the overall reliability of the two nodes is the product of the reliabilities of the two nodes.
  - Sequence: computed just like a dependency but shows a sequential relationship.
- The graph shows the relationships in the previous scenario.



*** Relationships are all sequences in this graph**

Reliability Reasoning Framework (continued)

- The reliability of each scenario can be calculated by taking the product of the reliability of each possible path that can be taken to fulfill the scenario.
- Calculate the reliability of the system by taking the product of the reliabilities of the scenarios.
- The reliabilities of the scenarios are also multiplied with the probability of operating that scenario.
- The perceived reliability of the system is described with the following equation:

$$R = \prod_{i=1}^{n} f R_i$$

Qualitative Reasoning Frameworks

- Qualitative Reasoning Frameworks are based on models that produce qualitative results.
- Quality Attributes such as safety, confidentiality and integrity do not have analytic theories that produce an output based on numeric parameters.
- Qualitative models can be used to reasoning about qualitative attributes
  - Confidentiality: model based on threats and its response.
  - Integrity: model based on threats and its response.
  - Safety: model based on failures and its response.

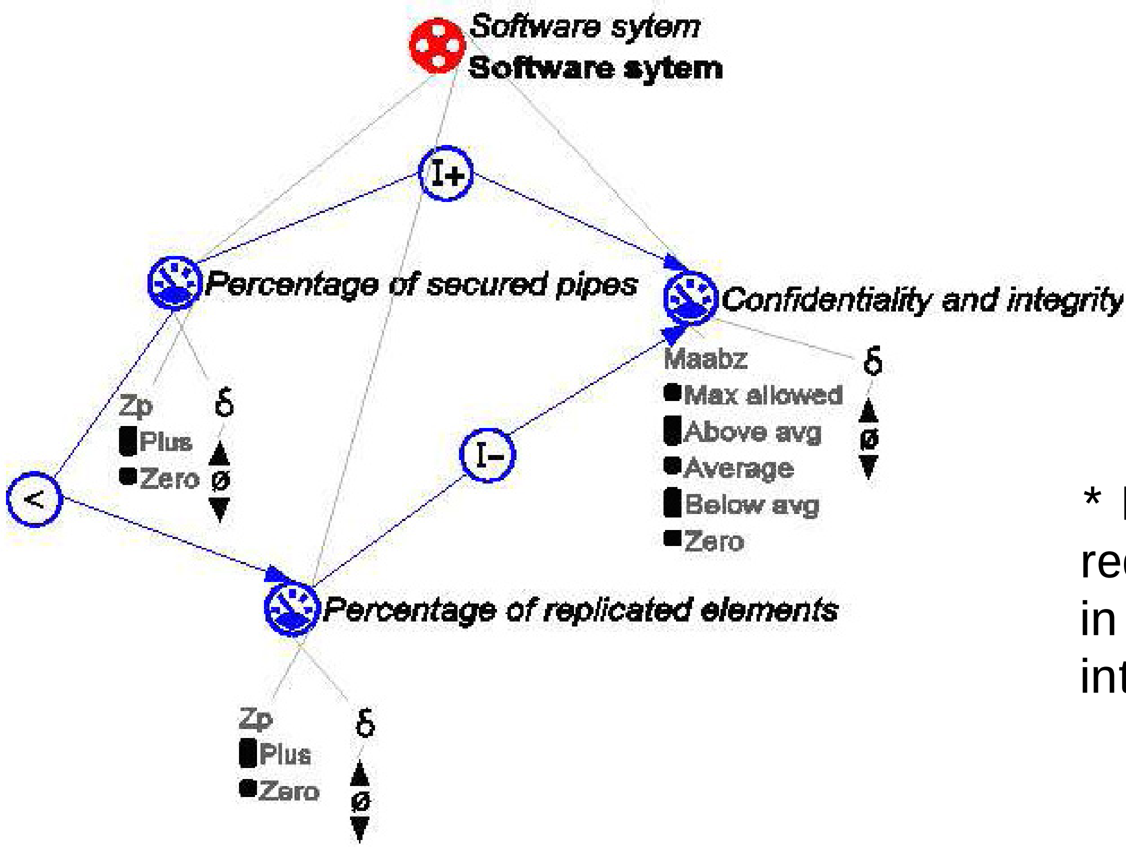
Security Reasoning Framework

- Problem description: to determine whether the security scenario is satisficed based on the design tactics and the security threat present.
- Analytic theory: qualitative reasoning based on trade-offs and causality.
- Analytic constraints: satisficing security requirements requires the modeling of causal relationships of design tactics and security threats.
- Model representation: model fragments that show how the influences of design tactics and security threats.
- Interpretation: the causality of each design tactic and security threat determines the satisficing of a security scenario.
- Evaluation procedure: the causality of each design tactic and security threat is traced to see if it leads to the satisficing of the security scenario.

Satisficing a scenario means to derive a calculation from the model and see if the result of the calculation is within a range of values.

Security Reasoning Framework (continued)

- Model Fragment from a QR model for



* Model fragments may be required to be programmed in Java that can be plugged into ArchE.

Security Reasoning Framework (continued)

	Availability	Confidentiality	Integrity
Replacing an insecure pipe	No change	++	++
Implementing an intercepting validator	++	++	++
Replication of modules	++	--	--

- The symbols indicate positive/negative satisficing influences.
- Tactics will be expressed as effects on the quality attributes.
- The effect of each tactic will be used to derive the response to the qualitative quality scenario.
- Multiple effects might need to be considered.

Assembling the Reasoning Frameworks

Attribute	Sub-Attribute	Value	Standard Scale
Performance			Meets goal
Dependability			Meets goal
	confidentiality	Required controls in place	Meets goal
	integrity	Required controls in place	Meets goal
	reliability	97%	Meets goal
	availability	92%	Meets goal
	maintainability	5 man-days	Does not meet goal
	safety	Required controls in place	Meets goal
Accessibility			Exceeds goal

- A quality profile that shows the state (as shown by the response) of the architecture given the scenarios that are under considerations.
- The quality profile may be interpreted into a single dependability measure.
- The tradeoffs among the attributes must be considered.

Conclusions

- The goal is to provide a reasoning framework that combines the quantitative and qualitative attributes of dependability.
- A new approach for reasoning about qualitative attributes was presented.
- A method of blending the quantitative and qualitative attributes of dependability into a single metric that can be used to measure the dependability of a software system.