Augmenting Architectural Modeling to Cope with Uncertainty

Orieta Celiku, David Garlan, and Bradley Schmerl
Carnegie Mellon University

5 November 2007
Motivation: Relating Uncertainty to Architecture

- Coping with uncertainty at design time and a high level
  - understanding *expected* behavior even when we cannot know exact outcomes
- Working with architectural models
- Adopting existing specification and analysis approaches to specifying and analyzing architectural models
Today’s Architecture Models

- Primarily focus on architectural properties that can be precisely defined
  - Assume complete information (at some level) about architectural behavior
- May use probabilistic uncertainty for very specific kinds of analysis
  - Limited to specific architectures and domains
  - E.g., queuing-theoretic performance analysis
Our Approach

1. Augment property specification in ADLs with the ability to incorporate uncertainty in those properties
2. Make use of such properties for analysis such as architecture-based simulation and run-time detection of behavioral drift
3. Augment behavior descriptions to explicitly account for probabilistic behavior
Architecture Properties as Distributions

Example:

Property type
NormalDistribution = Record [  
  mean : float;  
  stddev : float;  
]

...  

property
arrival-rate : NormalDistribution =  
  [mean=100; stddev=10;];

Such properties can be represented and visualized using standard mechanisms
Using Distributions in Analyses

- Properties specified as distributions can be used
  - as the basis of simulations
    - e.g., performance simulation and security simulation
  - to compare observed values with predicted distributions
    - e.g., to determine whether changes should be made to executing systems
Using Distributions in Analyses: (Monte-Carlo) Performance Simulation
Augmenting Architectural Behavior with Probabilities

- Incorporate quantifiable uncertainty in behavioral descriptions of architectures
  - uncertainty expressible in terms of probabilistic distributions

- Enable incremental refinements of models:
  - making information more precise as we learn more about behaviors
  - extending specifications without invalidating previous analyses

- Give access to explicit probabilistic choices (in addition to standard choices in behavior)
  - e.g., coin := heads $\frac{2}{3}$ ⊕ coin := tails
  - represent the frequency with which branches of a choice are executed
  - give rise to probabilistic assurances about properties (as opposed to absolute certainty)

- Semantics extends standard assertion-based reasoning with probabilistic analysis over a suitable probabilistic space on computation paths
  - can reason about probability of a property holding, or more generally
  - expected value of some random variable of interest
Modeling with Probabilistic Action Systems: A Client-Server System

- Action Systems Client, Broker, ServerA, and ServerB execute in parallel
- Each action system is a collection of guarded actions executing in parallel
Explicit Probabilistic Choices: Servers

ServerA ==

\textbf{var} \textit{sa}: \{\textit{wait}, \textit{serving}, \textit{served}\}

\textbf{initially} \hspace{0.5em} \textit{sa} := \textit{wait}

\textit{serviceArequest}: \ (\textit{sa} = \textit{wait}) \hspace{0.5em} \rightarrow \hspace{0.5em} \textit{sa} := \textit{serving}

\textit{serviceA}: \hspace{0.5em} (\textit{sa} = \textit{serving}) \hspace{0.5em} \rightarrow \hspace{0.5em} \textit{sa} := \textit{served} \ p_{A} \oplus \textit{skip}

\textit{serviceAreceive}: \ (\textit{sa} = \textit{served}) \hspace{0.5em} \rightarrow \hspace{0.5em} \textit{sa} := \textit{wait}

- Server A chooses to delay serving with probability 1\(\cdot\)p_{A}
- ServerB is specified analogously and chooses to delay serving with probability 1\(\cdot\)p_{B}
Probabilistic Properties

Properties we can reason about using AS semantics:

- The probability of establishing a postcondition
  - e.g. “For pA and pB bounded away from 0, the Client will eventually be served with probability 1”

- Probabilistic versions of temporal properties,
  - guarantees expressed as “the least probability with which the property holds”
Nondeterministic Choices: Broker

Broker ==

\[
\text{var } r: \{\text{listen, serve, serving, served}\}
\]

\text{initially } r := \text{listen}

- request: \((r = \text{listen}) \rightarrow r := \text{serve}\)
- serviceArequest: \((r = \text{serve}) \rightarrow r := \text{serving}\)
- serviceBrequest: \((r = \text{serve}) \rightarrow r := \text{serving}\)
- serviceAreceive: \((r = \text{serving}) \rightarrow r := \text{served}\)
- serviceBreceive: \((r = \text{serving}) \rightarrow r := \text{served}\)
- receive: \((r = \text{served}) \rightarrow r := \text{listen}\)

Two or more enabled actions give rise to (standard) nondeterministic behavior
Nondeterministic Choices: Room for Refinement

Broker ==

... 

serviceArequest: (r = serve) → r := serving
serviceBrequest: (r = serve) → r := serving

... 

When both servers are ready, we have no guarantees about which one will be chosen

- the best we can do is assume that “the worse” server is chosen
- we can only guarantee service within the first “tick” with probability \( \min(p_A, p_B) \)
Nondeterministic Choices: Refinement

- We could do better if we flipped a coin instead

\[ \text{Broker} == \]

\[ \ldots \]

\[ \text{whichserver:} \quad (r = \text{serve}) \quad \rightarrow \text{which} = A \quad \frac{1}{2} \oplus \text{which} = B \]

\[ \text{serviceArequest:} \quad (\text{which} = A) \quad \rightarrow r := \text{serving} \]

\[ \text{serviceBrequest:} \quad (\text{which} = B) \quad \rightarrow r := \text{serving} \]

\[ \ldots \]

- A valid implementation of arbitrary choice is one in which a coin is used to decide which branch to execute
  - probabilistic choices “average” results
- We could postpone the decision until we know how pA and pB are related
  - use a biased coin favoring the faster server
On Going Research

- Tailoring the described formalism to better express architectural notions and concerns
  - e.g. connectors, hierarchy, ports/roles
- Case studies
- Tool support for analyses