We begin with vanilla software component technology
- marked adoption trend
- .NET, J2EE, custom exemplars

Motivation for adoption include:
- larger, reusable building blocks
- reusable design patterns, rules
- simplified system integration
Prediction Enabled Component Technology

Component model
- required component interfaces and other implementation constraints
- behavior specifications
  - component type-specific, general
  - lifecycle and resource management policies

Component runtime environment
- runtime and deployment services
- execution environment ('container')
- runtime policy enforcement

There are many views on component technology
- our view is representative of theory and practice
Prediction Enabled Component Technology

We extend the base component model:
- these impose additional rules on component/assembly developer.
Prediction Enabled Component Technology

Where useful, and possible, we augment the runtime as well
The added rules and services are not arbitrary, but are imposed by a reasoning framework.
Prediction Enabled Component Technology

Reasoning frameworks provide automated prediction services

**Property theory**
- logic and notation for reasoning about assembly properties

**Decision procedure**
- an automated procedure for computing predicted properties

Reasoning frameworks can be defined for any objectively observable runtime assembly property
A reasoning framework will have assumptions about the systems it models:
- scheduling
- use conditions
- physical environment
- etc.

Our approach makes these assumptions explicit.
The component technology ensures that components & assemblies satisfy reasoning framework assumptions.
Prediction Enabled Component Technology

Tools provided by the component technology do static checking and enforcing of the component model.

- result: assembly behavior is predictable by construction
We use a composition language to specify well-formed assemblies:
- currently is a text-based language
- nearly isomorphic with UML 2.0
A (formal) interpretation maps each well-formed assembly to a unique model in the reasoning framework.

- result: **prediction is fully automated**
A PECT can have N reasoning frameworks, each of which
• will predict some observable assembly runtime property
• will have its own interpretation and decision procedure
• may introduce its own checkable well-formedness
  constraints on components and their assemblies