New Target for Code Generation

- The user of code generation usually must model everything (no runtime services).

- Application servers provide reusable services (don’t need to model these every time).

- Examples: transactionality, fault tolerance, concurrency, and distribution, and built-in performance optimization.

- Implements the distinction between analysis and design.
New Issues

- Integrating server functionality transparently to the model and methodology.

- Abstractly specifying some server functionality in the model.

- Specifying the rest of server functionality apart from the model.

- Coding in the model independently of the server.

- Integrating external systems into the model.
## Transparent Server Functionality

### Transactionality

- Transaction around state actions.
- Rolled back if B fails.
- Rolled back if D fails.
- New transaction if asynchronous.
- Guaranteed delivery in distributed environment.

<table>
<thead>
<tr>
<th>Object O</th>
<th>Attribute</th>
<th>Operation ()</th>
</tr>
</thead>
</table>

State S

entry / action A, B, C, D
Concurrent Server Functionality

- **Concurrency**

  Queued requests

- **Fault Tolerance**
Abstract Server Functionality

- Modeled as UML tagged values.
- Operations invoked on process state change (initialization, recovery, termination, aborting).
- Non-distributable objects.
- Maintaining a class’ extent.
- Operations invoked on association and attribute changes, object creation and deletion, and refresh from external storage.
Choose how objects or attributes are stored, e.g., which are persistently stored.

Choose how often to refresh an object from external storage to internal cache.

Compile to a specific target machine.

Distribute packages of objects to processes and machines.

Record the above in separate design or deployment models, multiple D&D models per analysis model.
**Model-level Coding**

- **Server-independent**: operate at level of UML models (objects, attributes, states, etc), use on any server.

- **Syntax independent**: use your favorite coding language.

- **Actions**: are the part of UML for specifying methods, state machine reactions, activity graph steps, collaboration interactions.

- Much more completely specified in the upcoming Action Semantics submission and UML 2.0 OCL submission.
Models of Code

- Objects store the semantics of your code.
- Vendor syntax hides the coding objects.
- Same technique will apply to OCL in UML 2.0.
Model-level Adapters

- External systems appear as objects in the UML model.

- Integrate external systems with model through attributes and relations.

- Surface external behavior through operations and receptions.

- Requires object triggers to map model actions to external system, and access to the model actions from external systems.
Example

- Mediation between communication networks and business systems.
- Modeled as messages processed step-by-step from one side to the other.
Example

- Steps in flow modeled as abstract classes with virtual operations.

- Application manages flow by passing messages along links between steps.

- Server handles concurrency, distribution, transactionality, fault tolerance, adapters.
For More Information

- conrad.bock@kabira.com
- www.kabira.com
- www.umlactionsemantics.org
- www.celigent.com/omg/adptf/wgs/uml2wg.htm