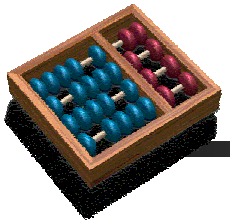


Institute of Computing – UNICAMP - Brazil

A Framework for Analyzing Exception Flow in Software Architectures

Fernando Castor Filho
{fernando}@ic.unicamp.br

Patrick Henrique da S. Brito
{patrick.silva}@ic.unicamp.br

Cecília Mary F. Rubira
{cmrubira}@ic.unicamp.br

IV ICSE Workshop on Architecting Dependable Systems (WADS'2005)
St. Louis, MO, US, May 17th 2005

Exception Handling

- Popular mechanism for structuring forward error recovery in software systems
- Exceptions can be derived incrementally at different phases of development:
  - Requirements – Related to the business logic
  - Architecture – Flow between arch. components
  - Detailed Design – Related to data structures
  - Implementation – Specific language exceptions

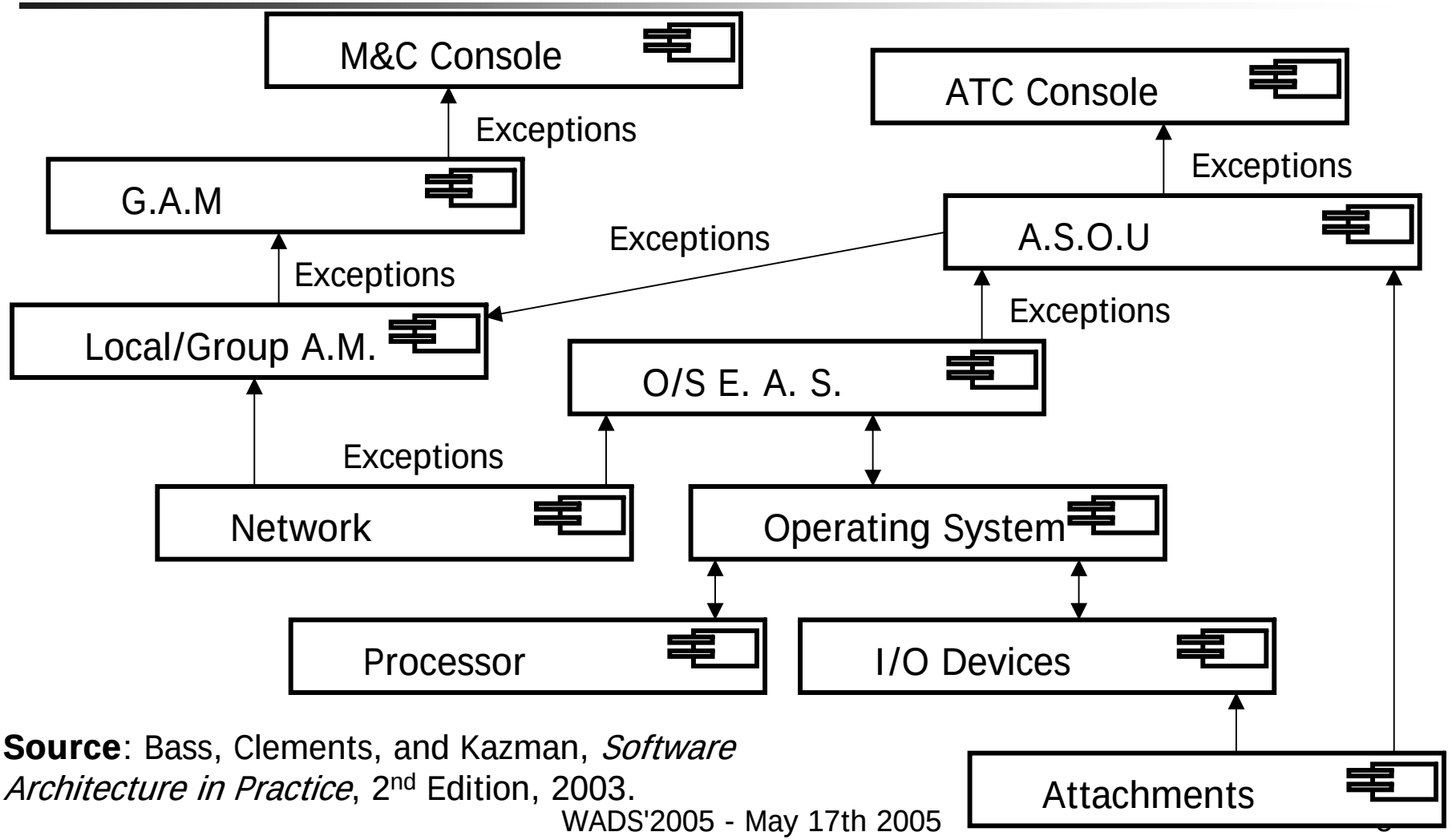
Exception Handling

- Popular mechanism for structuring forward error recovery in software systems
- Exceptions can be derived incrementally at different phases of development:
  - Requirements – Related to the business logic
  - **Architecture – Flow between arch. components**
  - Detailed Design – Related to data structures
  - Implementation – Specific language exceptions

Exceptions at the Architectural Level

- A system's exceptional activity should be addressed since the early phases of development
- In recent years, many approaches combining software architecture and exception handling have been proposed
- There hasn't been much focus on the description of exceptions at the architectural level
  - ✓ This focus may be required for systems with strict dependability requirements such as commercial applications, control systems, and so on.

An Air-Traffic Control System Example



Source: Bass, Clements, and Kazman, *Software Architecture in Practice*, 2nd Edition, 2003.

... Some Interesting questions...

- What does a double-headed arrow mean?
- What are the exceptions that each component signals and handles?
- Are there any relevant cause-effect relationships?
- Is this analyzable?

Problem

- To describe software architectures so that it is possible to reason about the flow of exceptions at the architectural level

Requirements of the Solution

1. Easy to use (pictorial representation)
2. Integrated with the concept of architectural style
3. Precise (unambiguous)
4. Analyzable
5. Capable of expressing rules of existing exception handling models

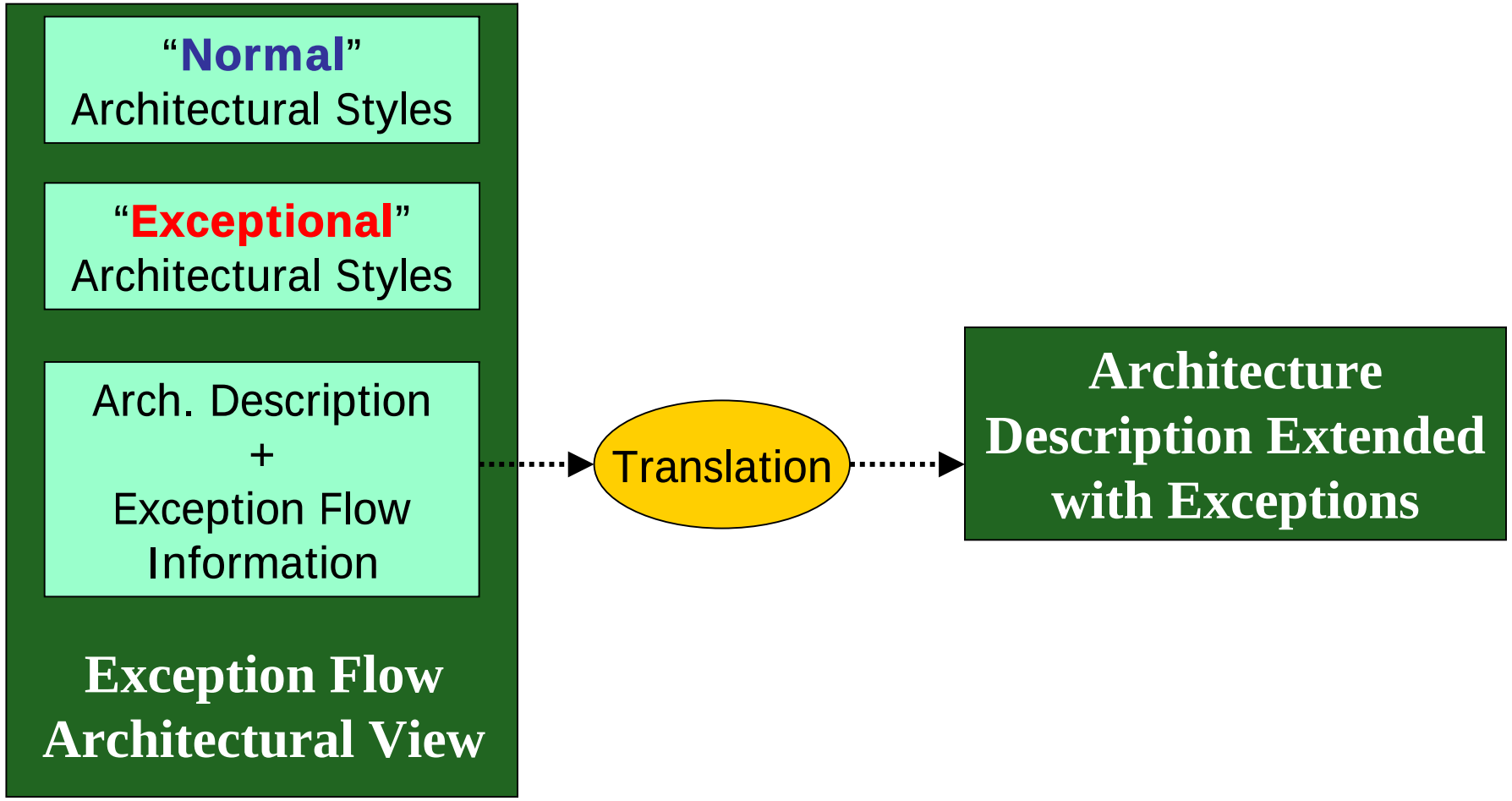
An Architecturel Description Language (ADL)

- ADLs: Notations for describing software architectures
  - Components, connectors, and configurations
- ACME
  - ADL and arch. interchange language
  - Focus on the structure of the system;
  - Constructs for defining architectural styles;
  - Extensible
  - Has mature tool support.
- Requirements (1-4)

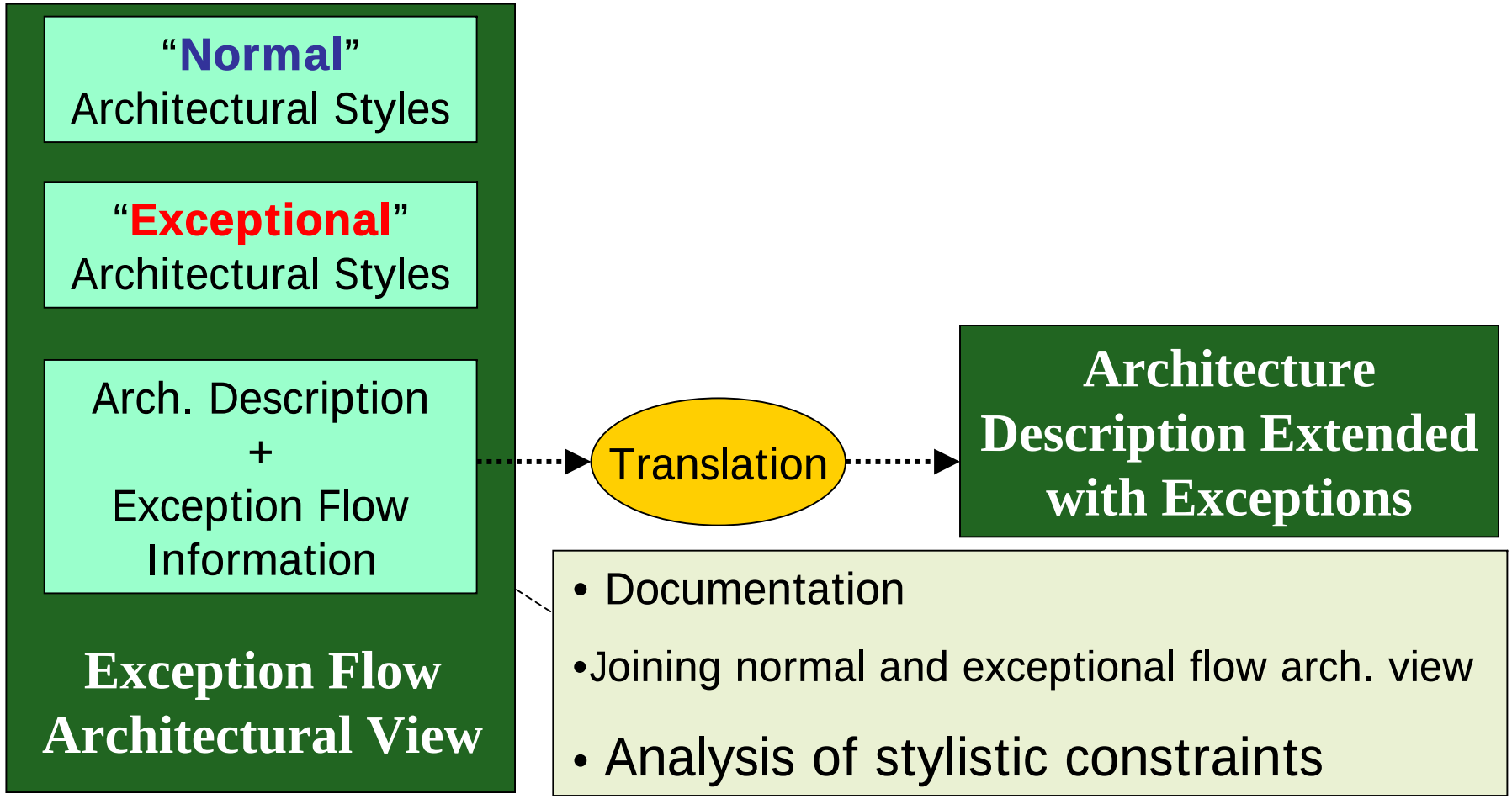
A Lightweight Formal Method

- Easy to use
- Support complex data structures
- Alloy design language
  - ✓ Similar to Z (less expressive but supports automated analysis)
  - ✓ Alloy constraint analyzer (AA)
- Requirements 3-5

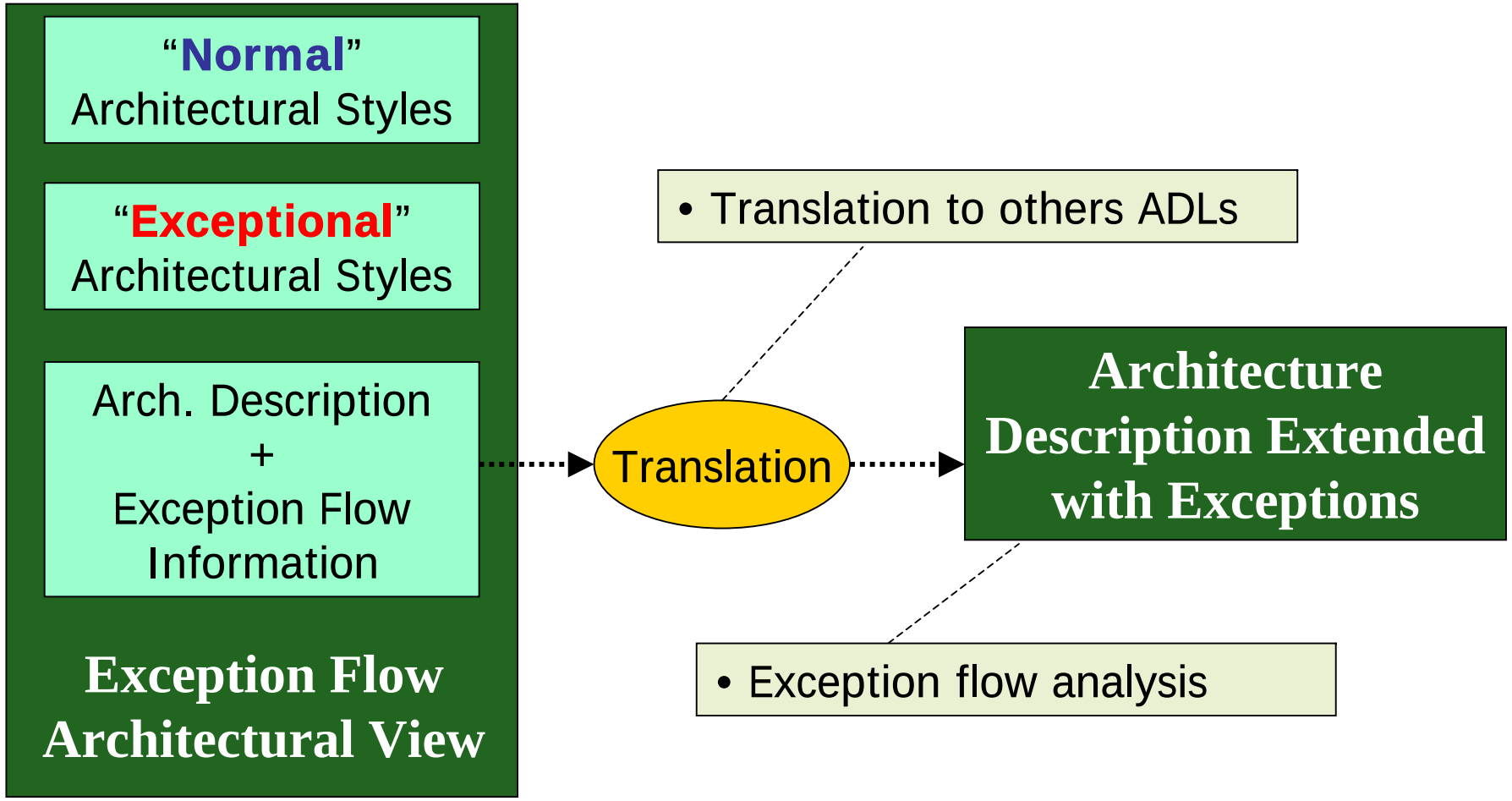
Proposed Framework: Aereal



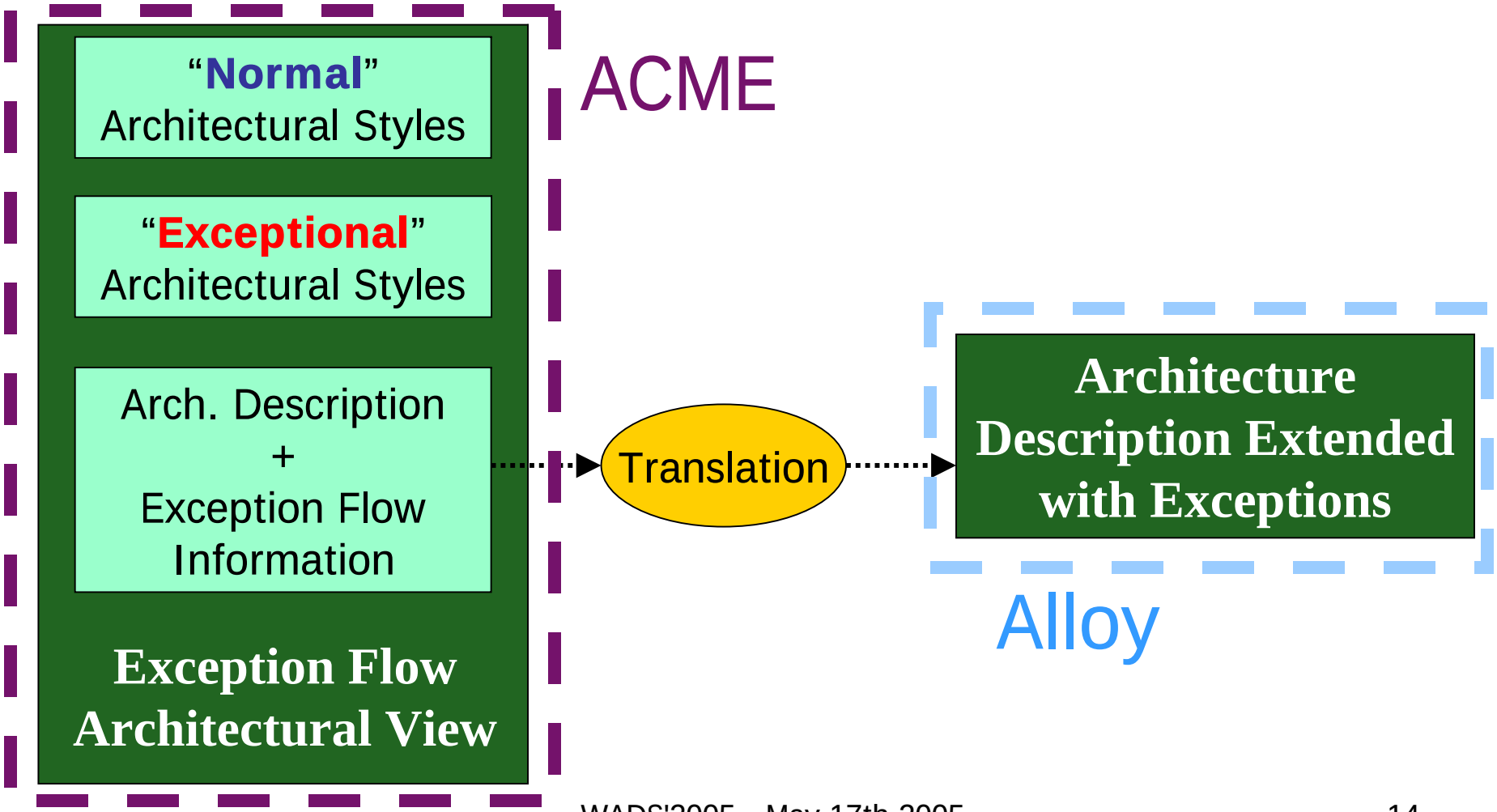
Proposed Framework: Aereal



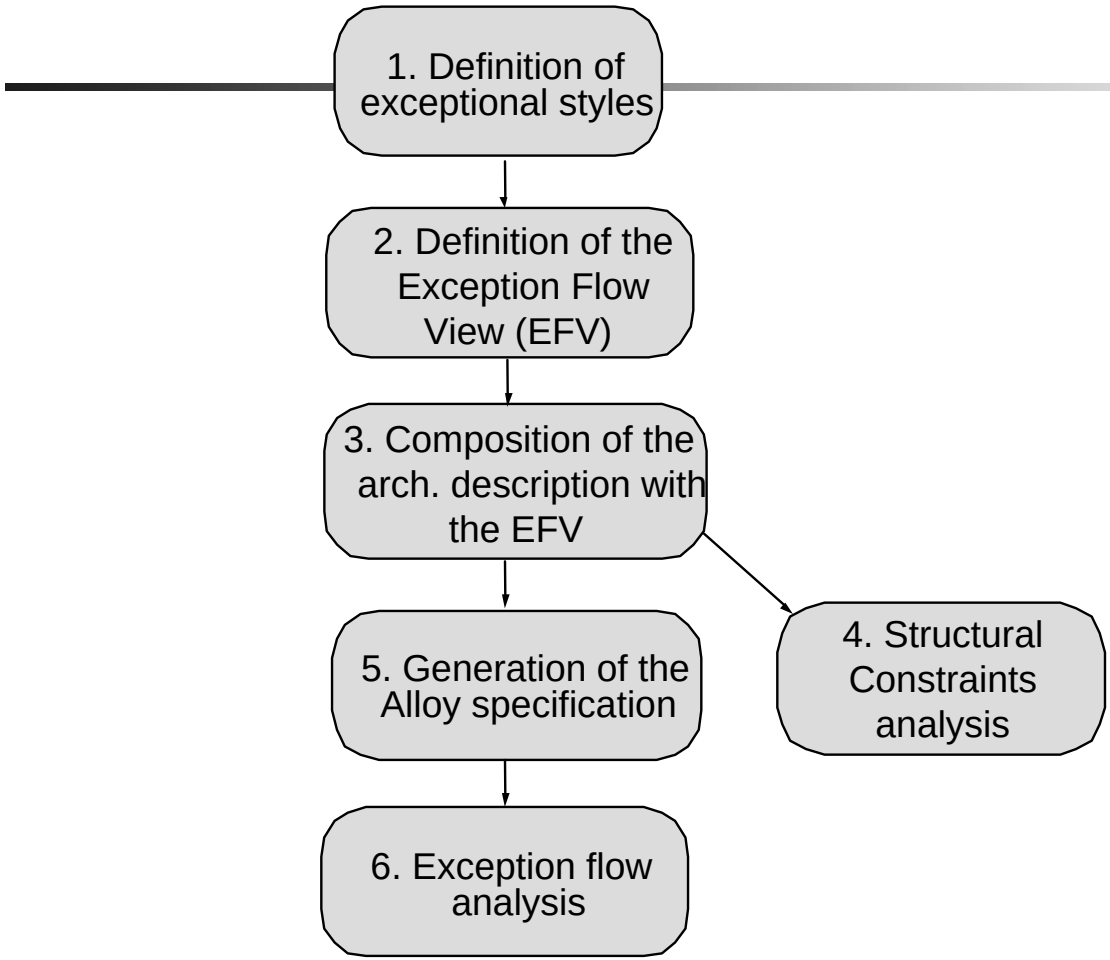
Proposed Framework: Aereal



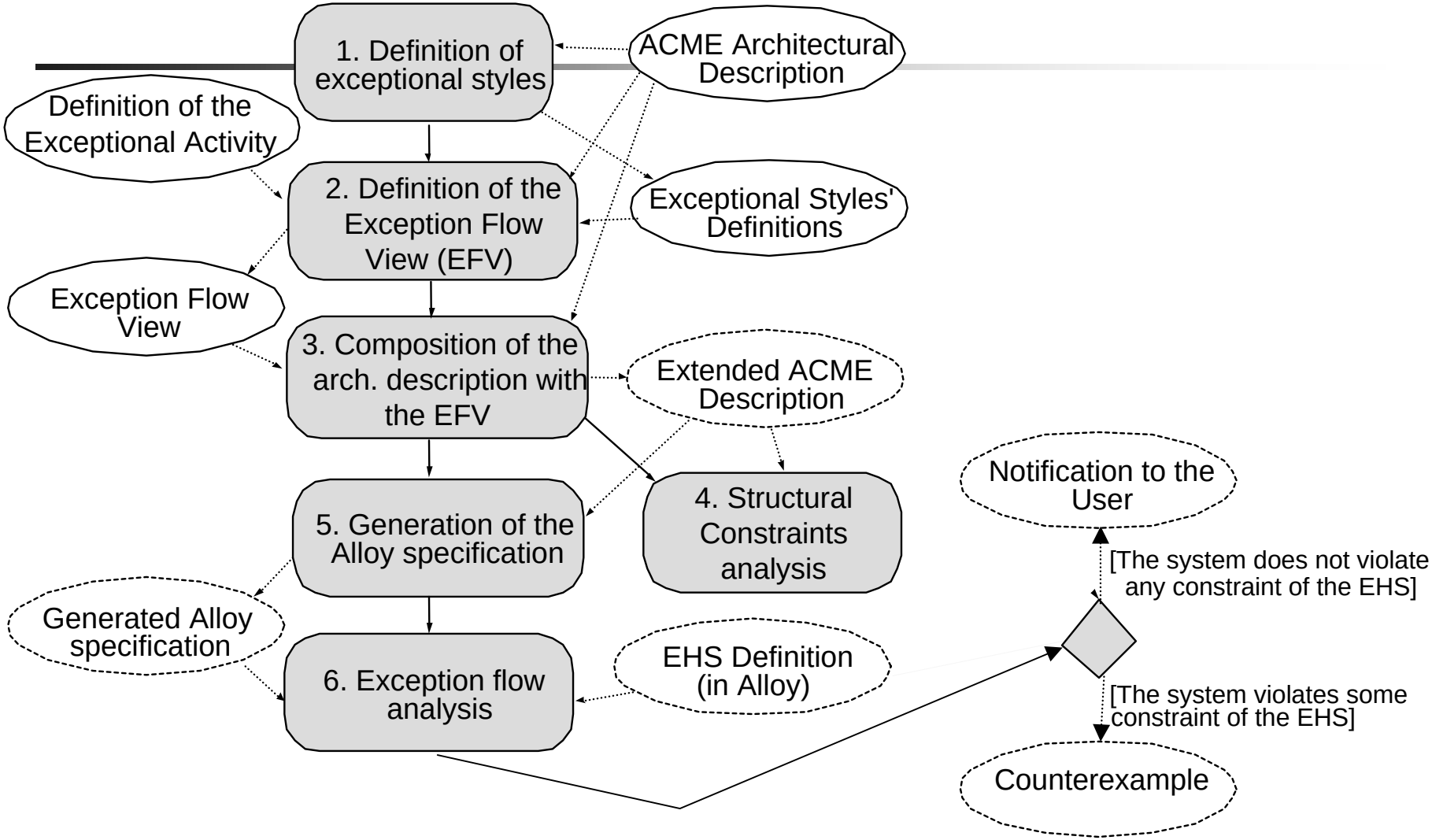
Proposed Framework: Aereal



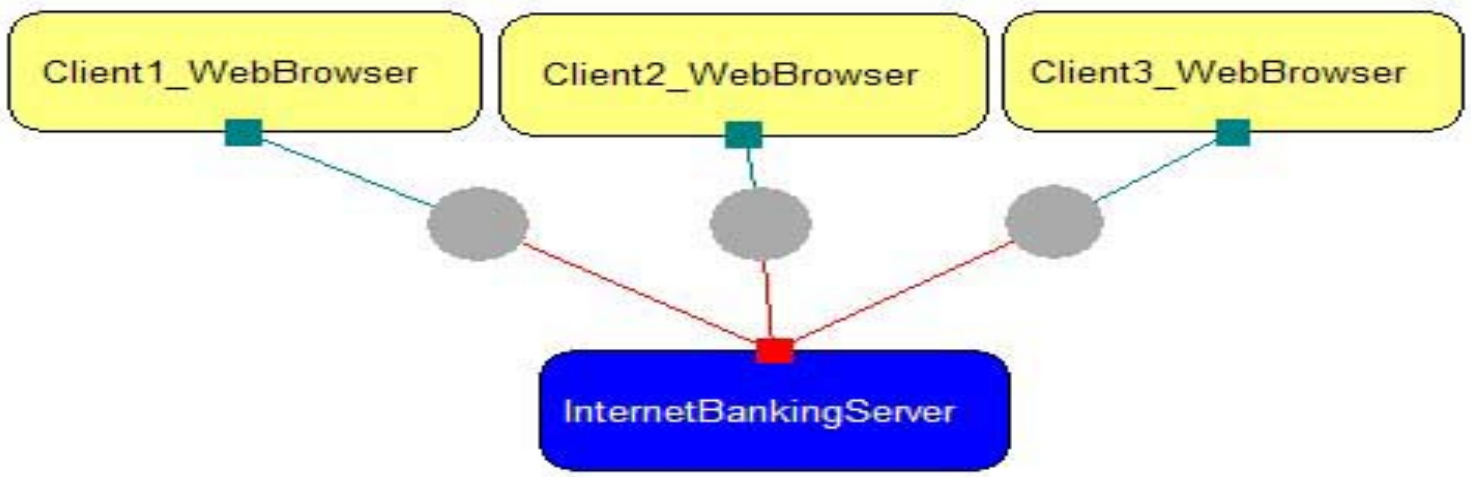
Process Supported by Aereal



Process Supported by Aereal



An Example: A Simple Internet Banking System



- Described in ACME:

```
import families\ClientAndServerFam.acme;
System ExtendedNetbanking : ClientAndServerFam =  new ClientAndServerFam extended with {
   Attachment Client1_WebBrowser.sendRequest to conn.clientSide;
   Attachment InternetBankingServer.receiveRequest to conn.serverSide;
   Component InternetBankingServer : ServerT =  new ServerT extended with {
      Port receiveRequest : ServerPortT =  new ServerPortT extended with {
         Property vis-order : int = 3;
      };
      Property multiThreaded : boolean = true << default : boolean = false;  >>;
      Property max-concurrent-requests : int;
   };
   Connector conn : CSConnT =  new CSConnT extended with {
      Role clientSide : clientSideRoleT =  new clientSideRoleT; extended with {
         Property vis-y : float = 90.0;
         Property vis-x : float = 176.0;
      };
    ...
```

Defining Exceptional Styles (1)

- An exceptional style constrains the ways in which exceptions flow between architectural components in a given architectural style
  - ✓ Exceptional styles extend **SingleExceptionFam**, an ACME family provided by Aereal
- Developers can define more than one exceptional style for the same normal style

Defining Exceptional Styles (2)

- Aereal uses **Exception Ducts** to model exception flow between components
  - ✓ Point-to-point links
  - ✓ Only for exception flow
  - ✓ Orthogonal to “regular” connectors

Defining Exceptional Styles in the Internet Banking System

- Uses a single architectural style: Client and Server

```
import families\SingleExceptionFam.acme;
import families\ClientAndServerFam.acme;
family ExceptionalClientAndServerFam extends SingleExceptionFam,
       ClienAndServerFam with {
   Component Type ExceptionalClientT extends ExceptionalComponent with {
      Port catchesPort : CatcherPortT =  new CatcherPortT;
      invariant(self.ports == 1);
   }
   Component Type ExceptionalServerT extends ExceptionalComponent with {
      Port signalsPort : SignalerPortT =  new SignalerPortT;
      invariant(self.ports == 1);
   }
   Connector Type ExceptionalCSConnT extends ExceptionalConnector with {...}
}
```

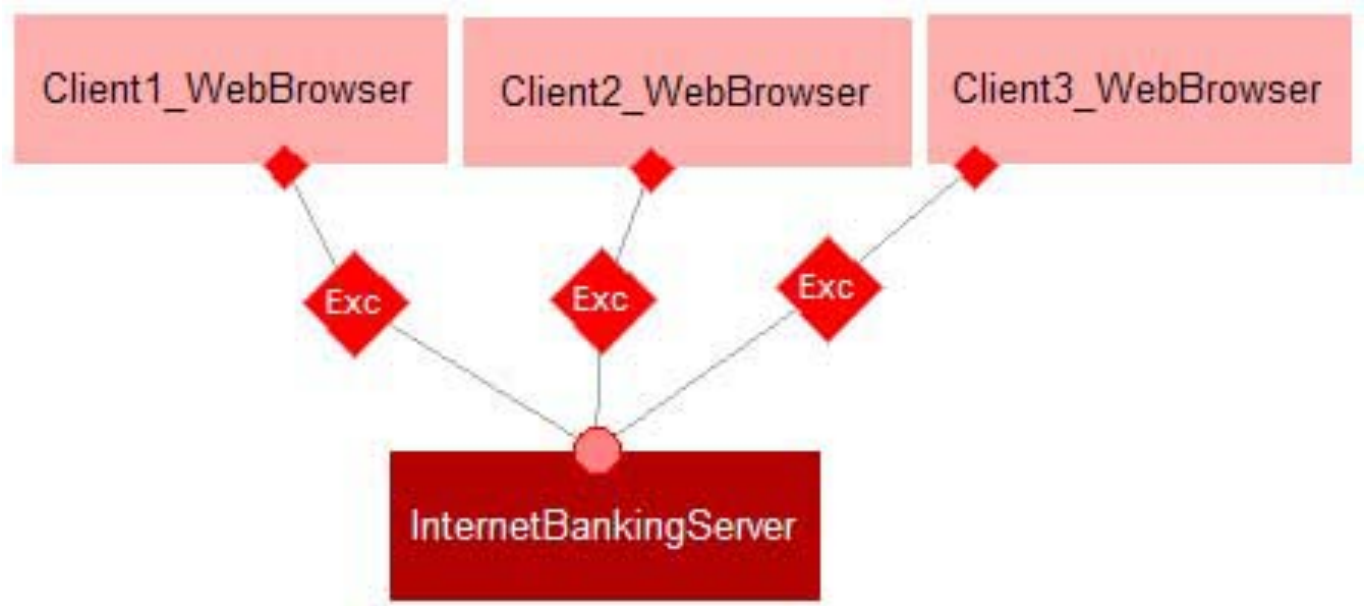
Specifying the Exception Flow View (1)

- The exception flow view is a *Components-and-Connectors* view that represents exceptions at the architectural level
  - A component/exception duct can **raise**, **signal**, **catch**, **handle**, and **propagate** exceptions
  - This is represented by assigning values to ACME properties

Specifying the Exception Flow View (2)

- The exception flow view uses one or more exceptional styles
- If exceptions flow between two arch. components, an exception duct is introduced between these components
  - The type of exception duct depends on the styles to which the components adhere

Exception Flow View of the Internet Banking System



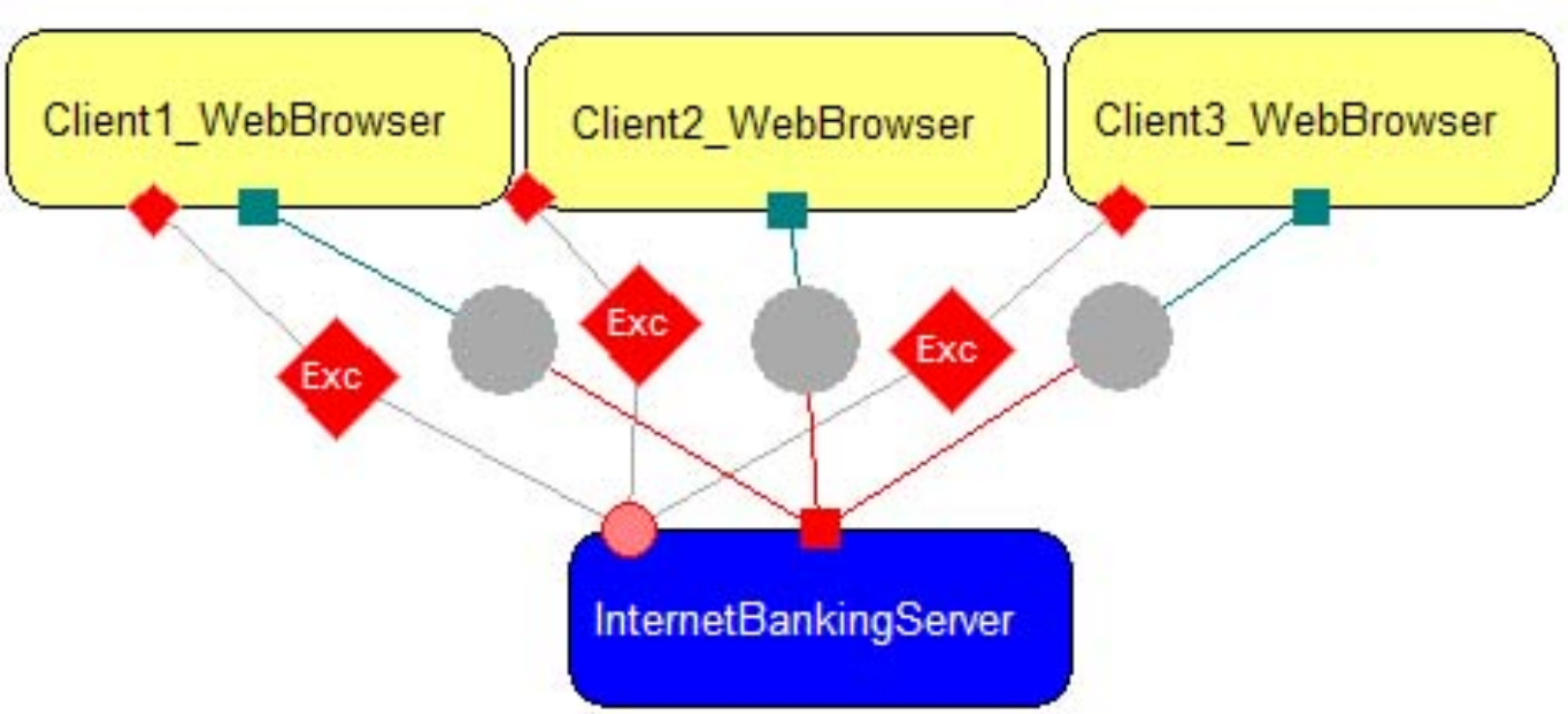
Exception Flow View of the Internet Banking System

```
import families\ExceptionalClientAndServerFam.acme;
System ExceptionalNetbanking:ExceptionalClientAndServerFam=
     new ExceptionalClientAndServerFam extended with {

  Component InternetBankingServer : ExceptionalServerT =
         new ExceptionalServerT extended with {
     Port signalsPort : SignalerPortT =  new SignalerPortT extended with {
       Property raises : Set{} = {RequestNotProcessedException};
       Property signals : Set{} = {RequestNotProcessedException};
     };
  };
  Connector ExceptionalCSConnT0 : ExceptionalCSConnT =
         new ExceptionalCSConnT extended with {
     Property catches : Set{} = {RequestNotProcessedException};
     Property signals : Set{} = {RemoteException};
     Property exceptionMappingFrom : Sequence<> = < RequestNotProcessedException >;
     Property exceptionMappingTo : Sequence<> = < RemoteException >;
  };
   ...
};
```

Composing Exception Flow View and Architectural Description

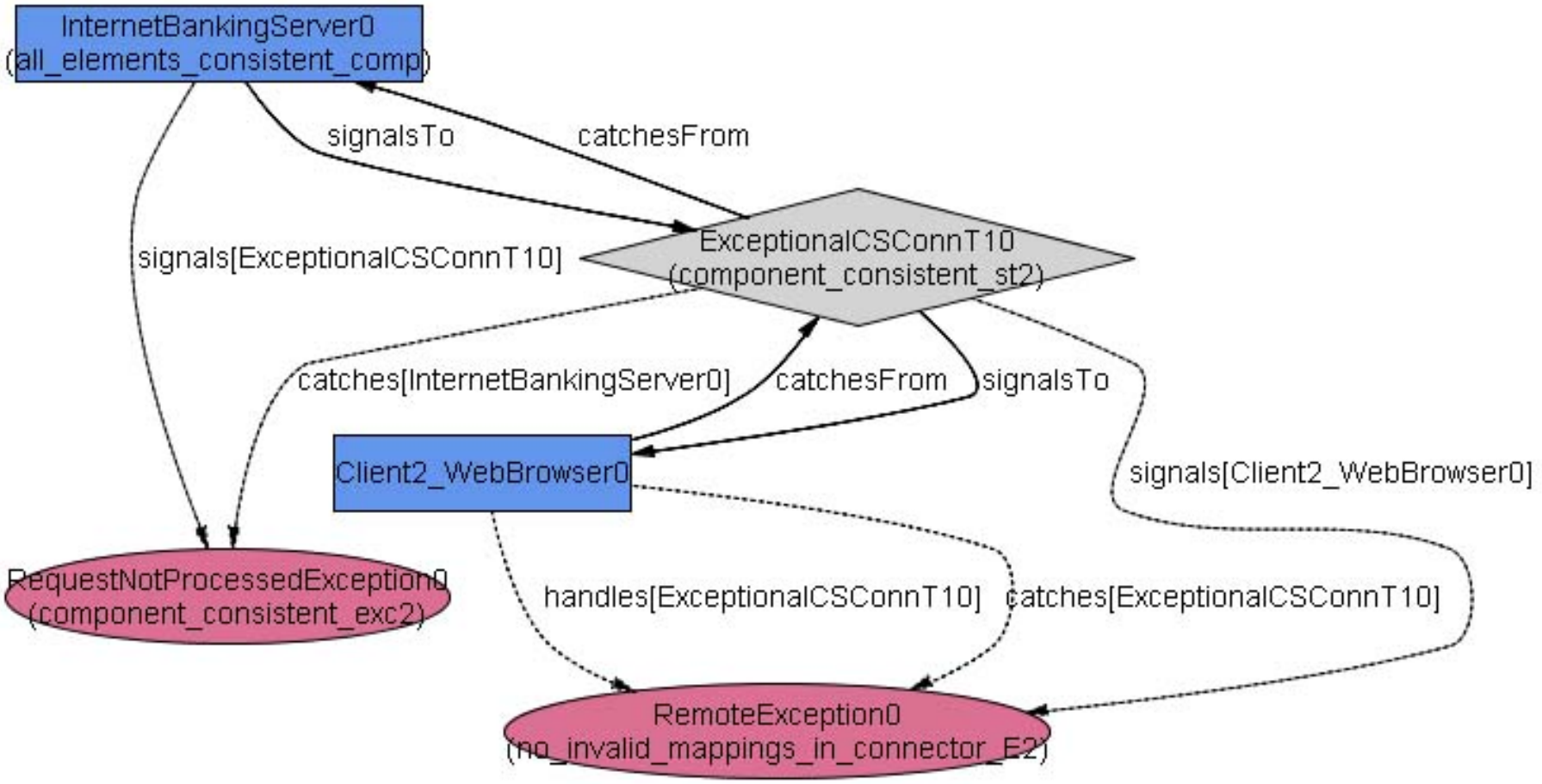
- Aereal performs the composition automatically



Analyzing Structural Constraints and Exception Flow

- The Armani constraint solver is used to check for violations of structural (exceptional style-related) constraints
  - Violations result in error messages
- Extended arch. descriptions are translated to Alloy and the Alloy Analyzer is used to analyze exception flow
  - User can specify rules of the assumed exception handling model
  - Violations of these rules result in counter-examples

An Example Counter-Example



Future Directions

- Assessment of Aereal:
  - ✓ Representing different modeling approaches involving multiple architectural styles
  - ✓ Describing rules of existing exception handling systems, some of which were already specified:
    - explicit exception propagation;
    - detection of exception subsumption;
- Extend the implementation of Aereal in order to automatically compute the sets of exceptions that are **caught** and **signaled**

Thank You!

Contact information:

Patrick Henrique da S. Brito – {patrick.silva}@ic.unicamp.br