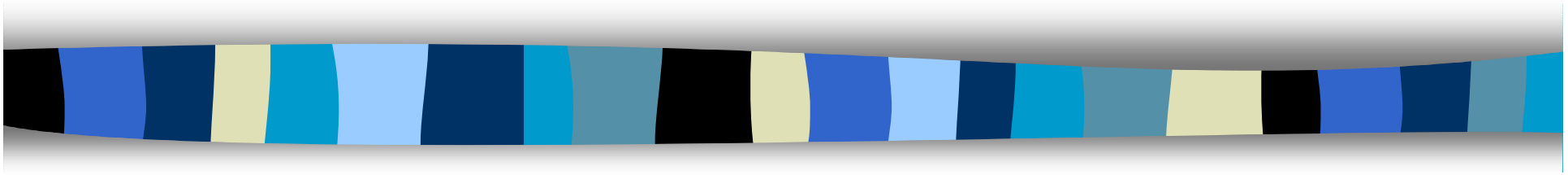
Software Architectures of Dependable Systems:
From Closed to Open Systems



V. Issarny ***et al.***

INRIA

Rocquencourt, France

Architecture-based Development of Complex Software Systems

- **Benefits *wrt* systems robustness**
  - Methods and tools supporting analysis, and the mappings of architectures to their implementations

➢ Focus is on the standard behaviour of the software systems

Supporting the Development of Dependable Systems

- **Crucial to account for the occurrence of failures in architecture-based development**
  - Application-transparent fault tolerance using middleware infrastructures
    - Provide base services for managing failure detection & error recovery
    - ➢ Customized middleware architectures *wrt* composed services

Aiding the Development of Middleware Architectures

- **Middleware infrastructures**
  - Customized composition of services through component-based middleware containers
  - ➢ Still, there is the need of supporting the development of containers
    - ➢ Right composition of services
    - ➢ Achieved quality

Systematic Composition of Middleware Architectures

- **A supporting environment** [CACM 06/02]
  - ADL for modeling middleware architectures
    - Repository of architectural descriptions of middleware infrastructures
  - Automated support for:
    - Composing middleware services
    - Analyzing the quality of composed architectures

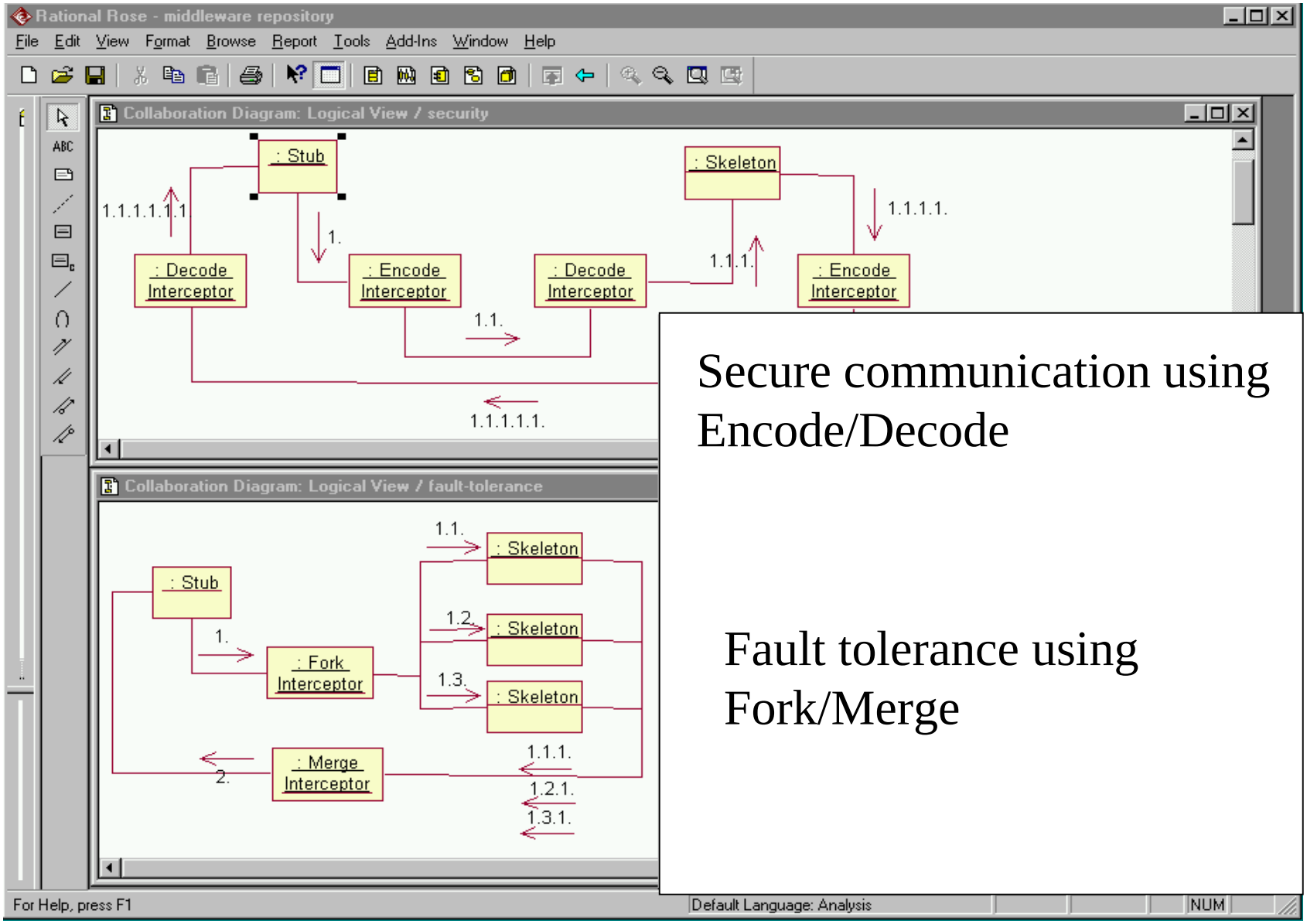
Modeling Middleware Architectures

- **Traditional base modeling elements**
  - Component, connector, configuration
    - Subtypes defining middleware-specific architectural abstractions (stubs, RPC connectors, …)
- **UML-based notation**
  - Component: subsystem
  - Connector: association + refinement
  - Configuration: collaboration

Tool Support

- **Rational Rose tool for the graphical specification of software architectures**
  - Implemented an add-in that eases the specification of architectural descriptions using the stereotypes discussed so far
  - Use of an existing add-in to generate XML textual specs from ADL specs
    - XML specs serve as input to other tools integrated in our environment
  - Implemented in OCAML a verifier of OCL constraints

Example



Secure communication using Encode/Decode

Fault tolerance using Fork/Merge

Composing Middleware Services

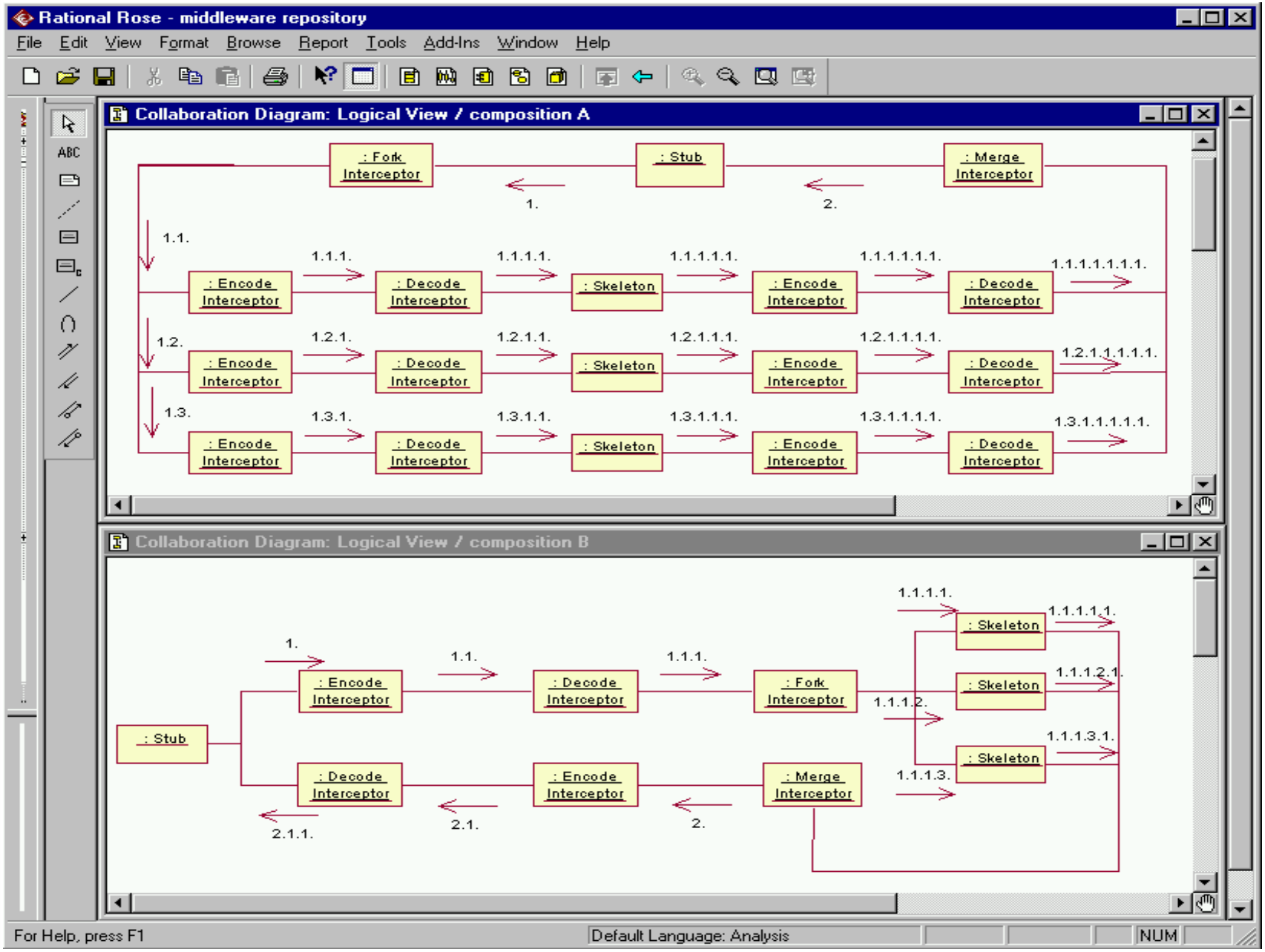
- **Approaches to architecture composition**
  - Horizontal = parallel composition [Qian *et al.,* 95]
    - Secure communication // multi-cast communication
  - Serial composition for linear architectures [Steffen & Beec 97]
    - FT architecture is not linear
  - Explicit interposition [Spitznagel & Garlan, 01]

➢ Need for an automatic solution to identify valid interpositions of components

Automating Composition

- **Solution** [WICSA'01]
  - Composition through model checking
  - Constrain composition through structure
- **Additional benefits**
  - Allows identifying unexpected compositions
  - Allows understanding interaction of qualities

Example


Rational Rose - middleware repository
File Edit View Format Browse Report Tools Add-Ins Window Help
Collaboration Diagram: Logical View / composition A
: Fork Interceptor
: Stub
: Merge Interceptor
1.
2.
1.1.
: Encode Interceptor
: Decode Interceptor
: Skeleton
1.1.1.
1.1.1.1.
1.1.1.1.1.
1.1.1.1.1.1.
1.2.
1.2.1.
1.2.1.1.
1.2.1.1.1.
1.2.1.1.1.1.
1.3.
1.3.1.
1.3.1.1.
1.3.1.1.1.
1.3.1.1.1.1.
Collaboration Diagram: Logical View / composition B
: Stub
: Encode Interceptor
: Decode Interceptor
: Fork Interceptor
: Skeleton
: Merge Interceptor
1.
1.1.
1.1.1.
1.1.1.1.
1.1.1.2.
1.1.1.2.1.
1.1.1.3.
1.1.1.3.1.
2.1.1.
2.1.
2.
For Help, press F1
Default Language: Analysis
NUM

Analyzing the Quality of Middleware Architectures

- **Base solution**
  - ATAM: Architecture Tradeoff Analysis Method [Kazman *et al*., 00]
    - Attribute-based architectural styles combined with scenarios
    - 25% of ATAM spent for building quality attribute models

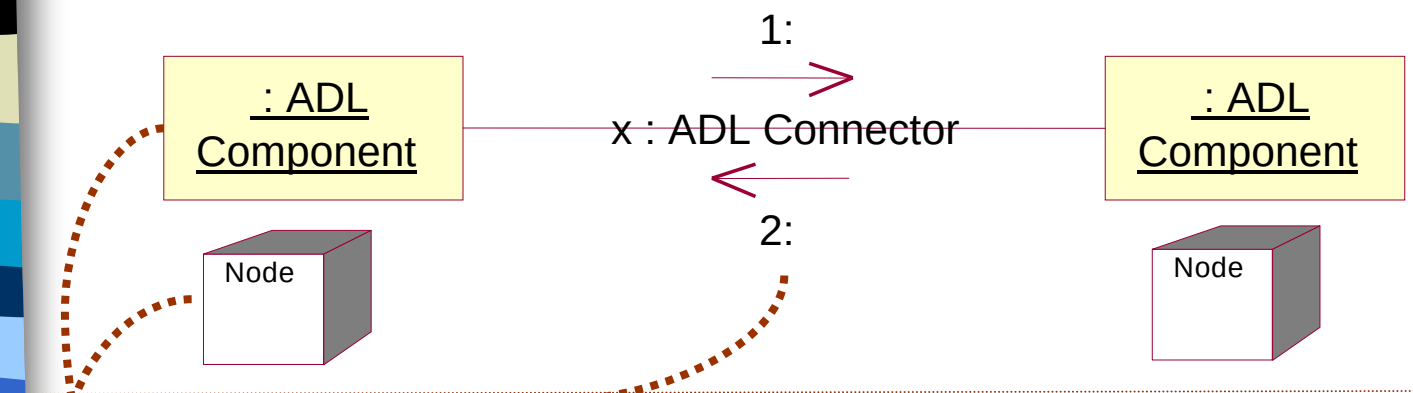
➢ Need for automated procedures for the generation of quality models from ADL specifications

Automating Quality Analysis

- **Modeling support**
  - Scenarios are specified as UML collaboration diagrams
  - Scenarios are associated with quality measures
  - Components/Connectors/Nodes are associated with properties characterizing various quality stimuli and parameters
    - The values of those properties are used to customize the generation of the traditional quality models.
- **Tool support**
  - Performance: QNAP-2 (SIMULOG)
  - Reliability: SURE-ASSIST (NASA)
  - ➢ Procedures mapping scenarios into models for QNAP and SURE

Reliability Analysis

UML Collaboration + Deployment

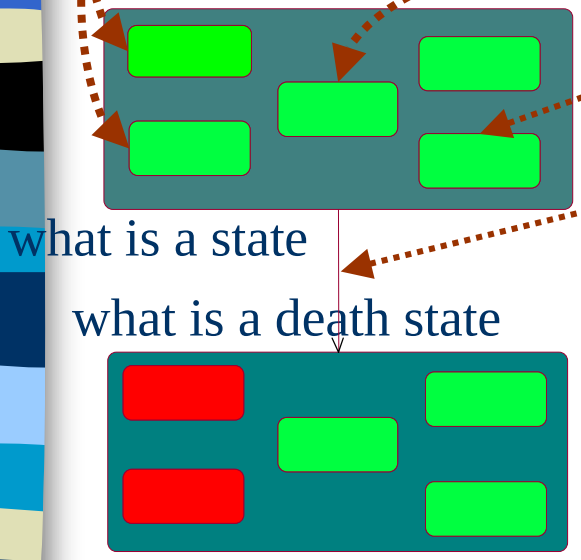


range =

f (kind of faults, redundancy)

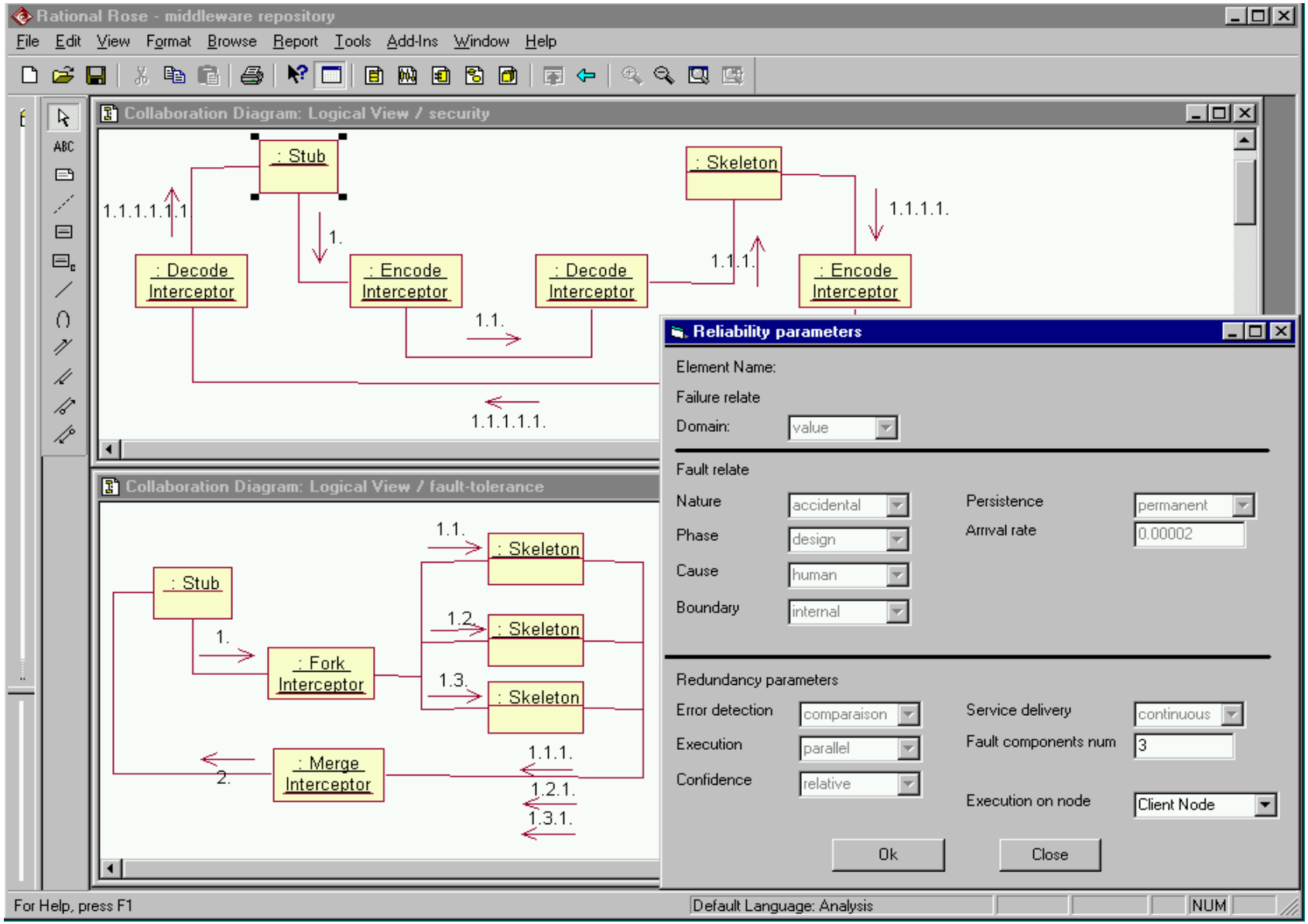
Generic transition rules for Components/Connectors/Nodes

e.g. if the **collaboration** is in a state where a **node n** is **operational**, then it may get into a state where **n** is **failed** and **all components** deployed on top of it are **failed**.

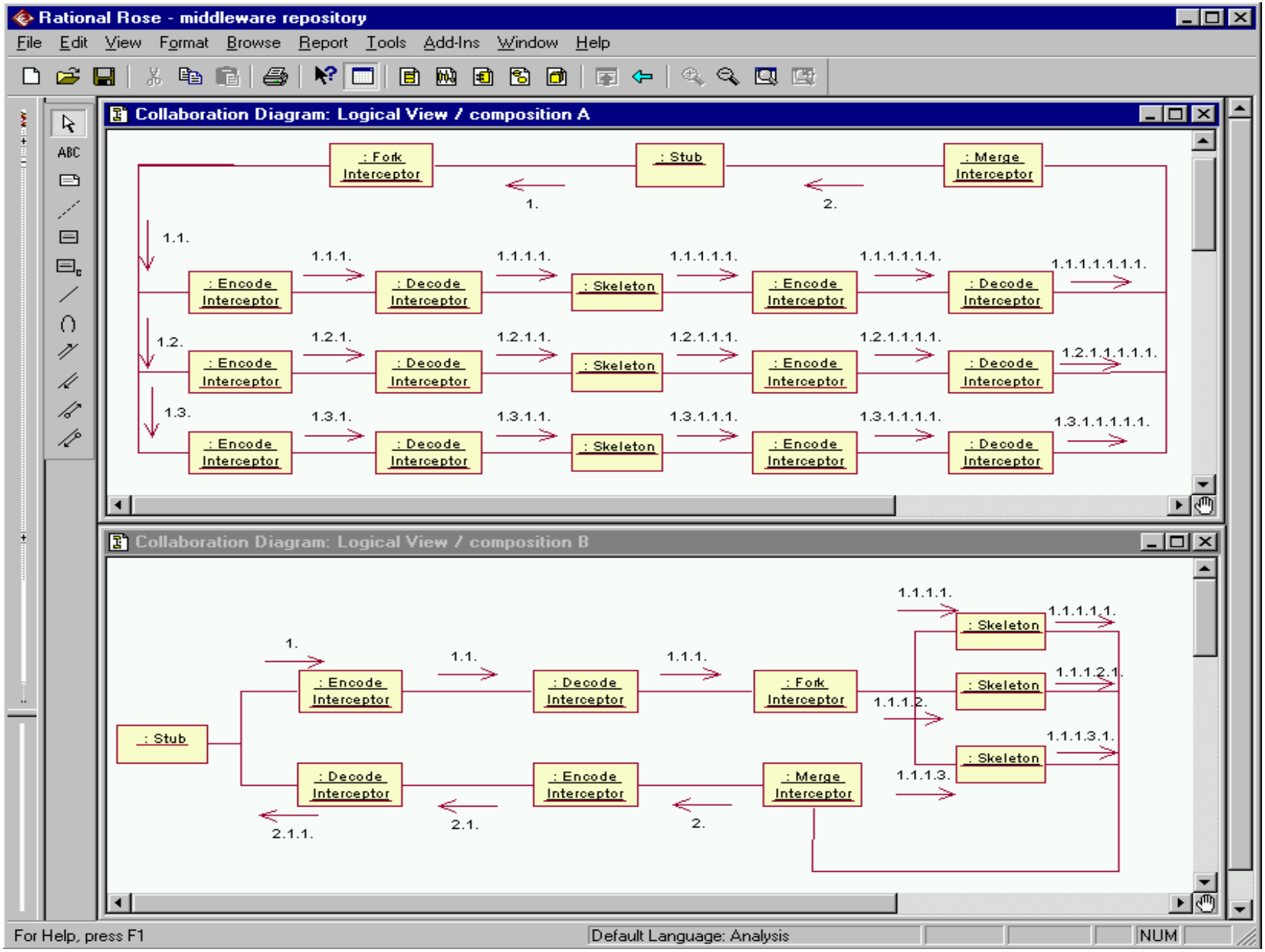


State Space Model

Example - Specification



Example – Composition



Example – Analysis results

Cases	*#transitions*	*Reliability (upper bound)*	*Reliability (lower bound)*
Composition A (Single version security service)	24	0.74	0.72
Composition A (n-version security service)	48	0.80	0.79
Composition B	12	0.70	0.67

Assessment

- **Making systems dependable is eased by middleware infrastructures**
  - Infrastructures offer base supporting services
  - Service composition may be automated
- **But…**
  - Allows only for backward error recovery and cannot cope with all failures

➢ Need for complementary application-specific forward error recovery

➢ Exception handling as it is the most general mechanism

Architecture-based Exception Handling

- **Exception handling mechanisms**
  - Serves implementing the system's exceptional specification (definition of exceptions & handlers)
  - Relies on some model (e.g., termination, resumption)
- **Existing mechanisms are for handling exceptions within components**

➢ What about exception handling requiring changes to the architecture [HICSS'01]

Base Solutions to Architectural Exception Handling

- **Exception handling within ADL**
  - Limited to the specification of signalled/handled exceptions within the definition of component/connector interfaces
  - Behavioural specification would further improve correctness checking
    - Pre/post as supported by Inscape [Perry, 89]
    - ➢ Issue of taking into account the exception handling model

Base Solutions to Architectural Exception Handling (Cont'd)

- **Dynamic reconfiguration**
  - Determined at runtime
    - Reconfiguration manager
    - Possibly constrained based on invariant on the system structure
  - Fixed at design time
    - Specified in the architecture description (e.g., Durra [Barbacci *et al*., 93])
    - Independent of exception handling

Exception Handling Model

- **Exception handling within components and connectors**
  - Let exceptions flow among the architectural elements according to the embedding architectural style
- **Exception handling at the architecture level**
  - To enable changing the running configuration

Impact on Architecture Description

- **Support for internal exception handling**
  - Specification of exceptions raised/handled by the elements
- **Support for architectural exception handling**
  - Definition of configuration exceptions and associated handlers using the ADL
    - Keep abstract the description of architectures for the sake of analysis and synthesis
  - Mapping to implementation using a service for dynamic reconfiguration

Assessment

- **Architecture-based development can aid in the construction of dependable systems**
  - Application-transparent fault tolerance: systematic aid in the design of customized middleware architectures
  - Application-specific fault tolerance: support for exception handling at the architectural level
- **But…**
  - Existing support is mainly aimed at closed systems

➢ Need solutions for open systems

Towards Dependable Open Systems

- **Issues in the development of open systems**
  - Composition of autonomous systems
  - Highly dynamic systems
    - Mobility,
    - Evolution,
    - …

Towards Dependable Open Systems

- **Ongoing work**
  - Architecting open systems with mobile nodes
    - Design and analysis of dynamically composed systems
    - Supporting middleware infrastructure
  - Fault-tolerance mechanisms for autonomous systems
    - “Dependability in the Web Services Architecture” – Ferda Tartanoglu *et al*.

For more information…

- Web page of the ARLES group at INRIA-Rocquencourt
  - http://www-rocq.inria.fr/arles/
- Work as part of the following projects
  - DSoS:

    http://www.newcastle.research.ec.org/dsos/
  - OZONE:

    http://www.extra.research.philips.com/euprojects/ozone/