Haifa Verification Conference 2006

Detecting Design Flaws in UML State Charts for Embedded Software

SMARRT:
Static **M**odel checking and **A**nalysis for **R**ose **R**eal**T**ime

An Automated Model Checking Solution for Rose Real Time Models to Detect Design Errors

Rational Software / Haifa Research Lab

extreme**blue**

BizTech

- Rational software, India:
  - Janees Elamkulam
  - Sandeep Kohli
  - Gururaja Kowlali
  - Satish Chandra Gupta
  - Sai Dattathrani
- Global Services, Spain
  - Claudio Paniagua Macia
- Haifa Research Lab
  - Ziv Glazberg
  - Ishai Rabinovitz

Outline

- The problem
  - Testing is not enough
  - Software model checking is (too) hard
- The solution
  - Rational Rose RT
  - RuleBase PE
  - SMARRT - Integration of the two
- Demo

Failure is NOT an Option.

It comes bundled with the Software.

We are in a battle against BUGs

And we are losing

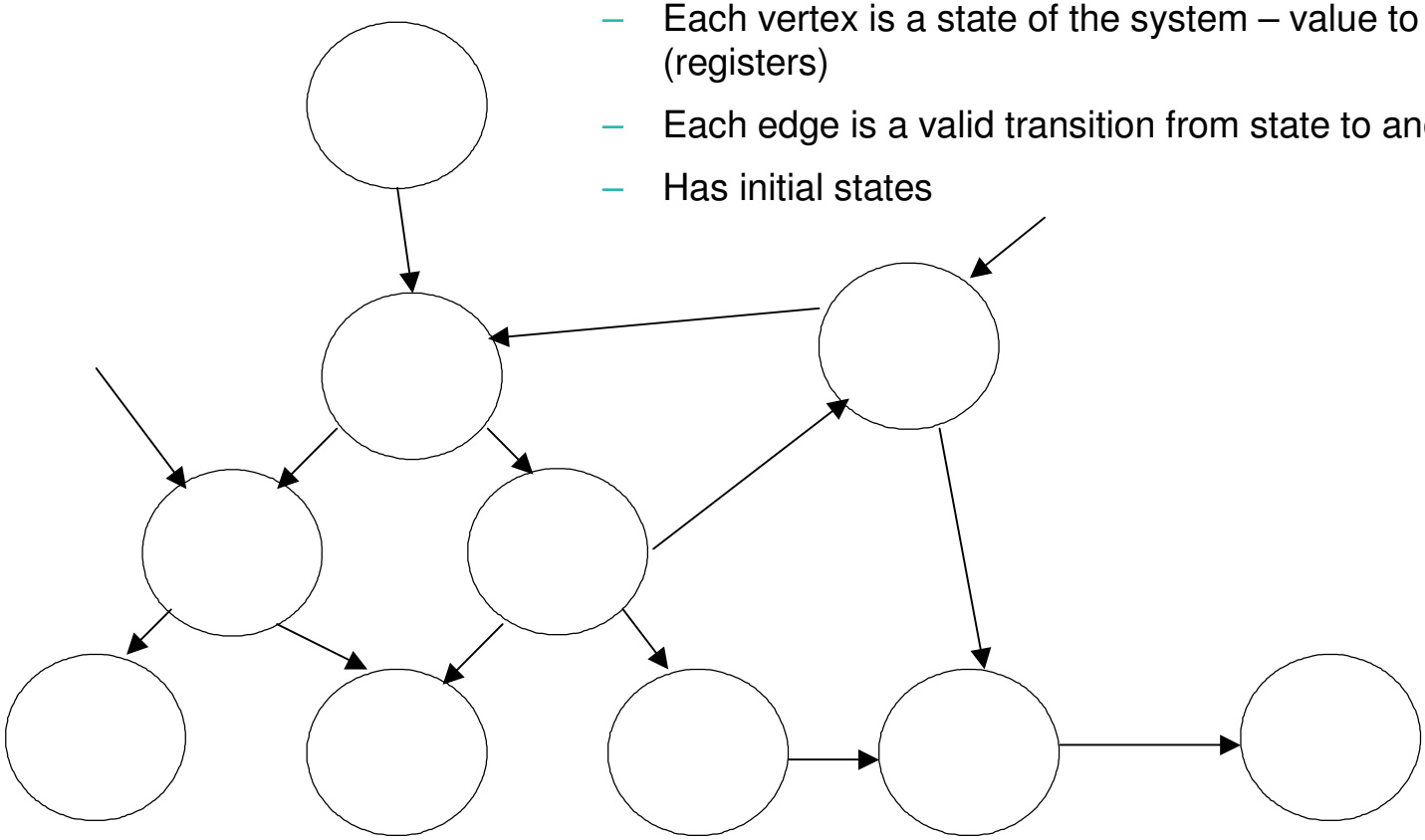
The Problem : Testing is not enough!

- Testing proves presence of a feature, not absence of a defect!
- You could write lot of test cases, still miss out some possibilities
- Even harder for concurrent systems

In embedded systems BUGs are expensive.

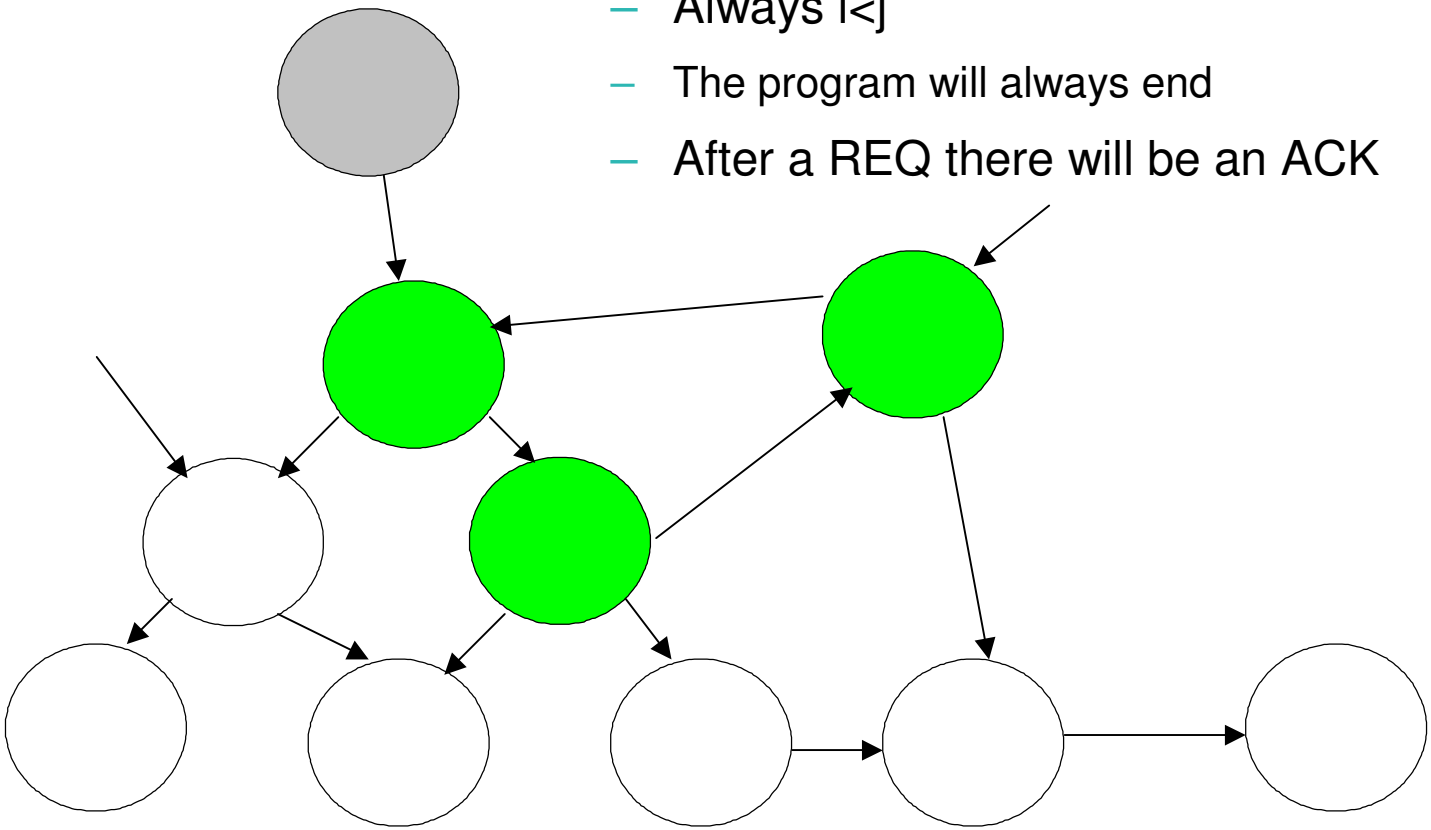
Model checking

- Build a model
- A model can be represented as a graph
  - Each vertex is a state of the system – value to all the variables (registers)
  - Each edge is a valid transition from state to another state
  - Has initial states



Model checking (specifications)

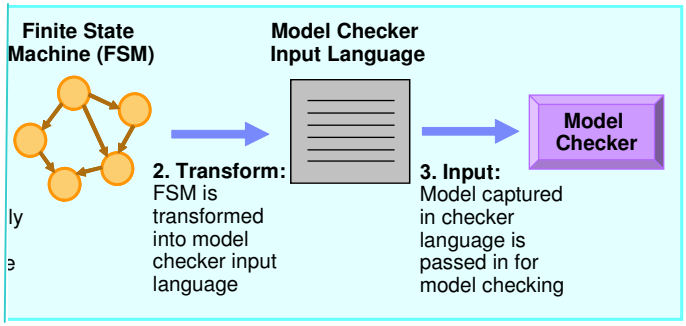
- We can check specifications like:
  - Always i<j
  - The program will always end
  - After a REQ there will be an ACK



Model Checking can help !!

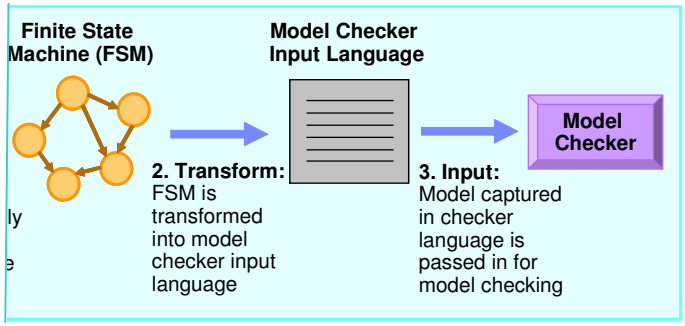
- Algorithmically verify finite state systems formally
- Orthogonal to testing
  - Checks all possible executions on a (limited) spec
  - Finds other (hard) bugs than testing
- Can check liveness properties
  - A certain property will finally hold (e.g., There is no live-lock)

- Hardware Model Checkers have been around for a long time.
  - SMV, BMC, IBM Rule Base PE
- Extensive research about Software model checking
  - SPIN, Zing, CBMC, Java Path Finder
  - HRL: Wolf, TCBMC, ExpliSAT



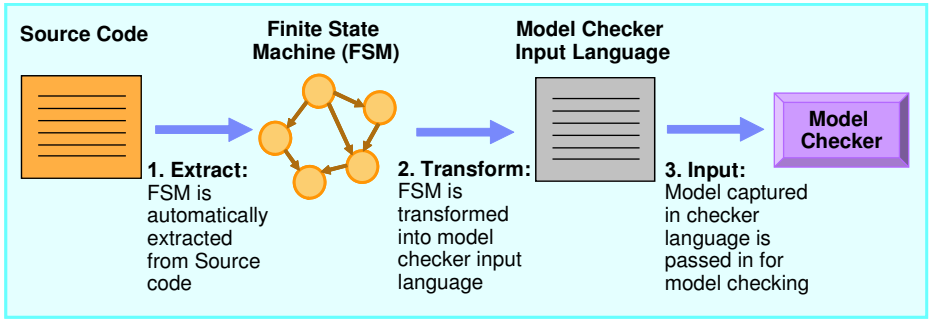
Model Checking is (too?) hard

- State explosion problem
- Experts
  - Create a model
  - Write specs
  - Understand results
- Software Model Checkers requires
  - Either an abstract model definition in a propriety language
  - Or extract model from source code



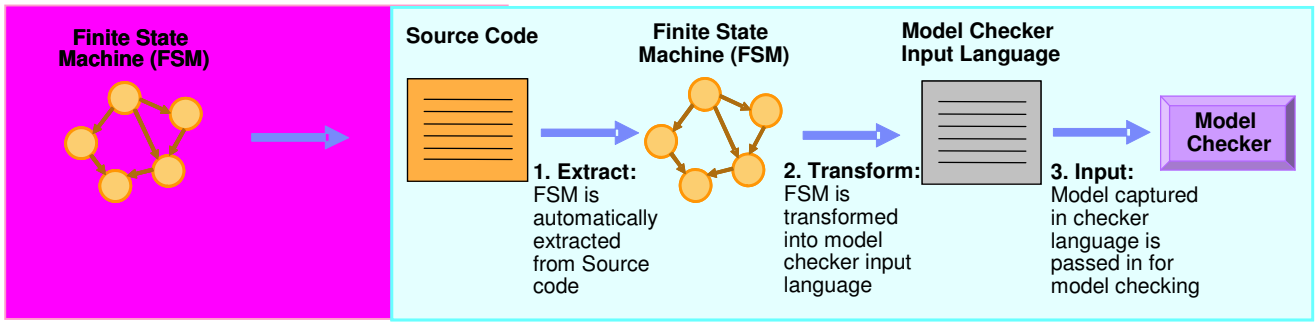
Model Checking is (too?) hard

- An abstract model definition in a propriety language
  - **Abstract model definition** could be outdated and inconsistent with the source code
  - **Expertise for developing model** in the language is very hard to find
  - Developing system model, just to check errors, is **time consuming**



Automatic translation

- Solves some of the problems
  - No need for an expert to build the model
  - Model is always up to date
- But introduces other problems
  - Creates big models
    - Source code is optimized to be executed not to be verified
    - Code breaks abstract ideas into too small parts,
    - uses (unnecessary) pointers
    - It is impossible to automatically reverse engineer the source
  - Same abstract component need to be rechecked each time

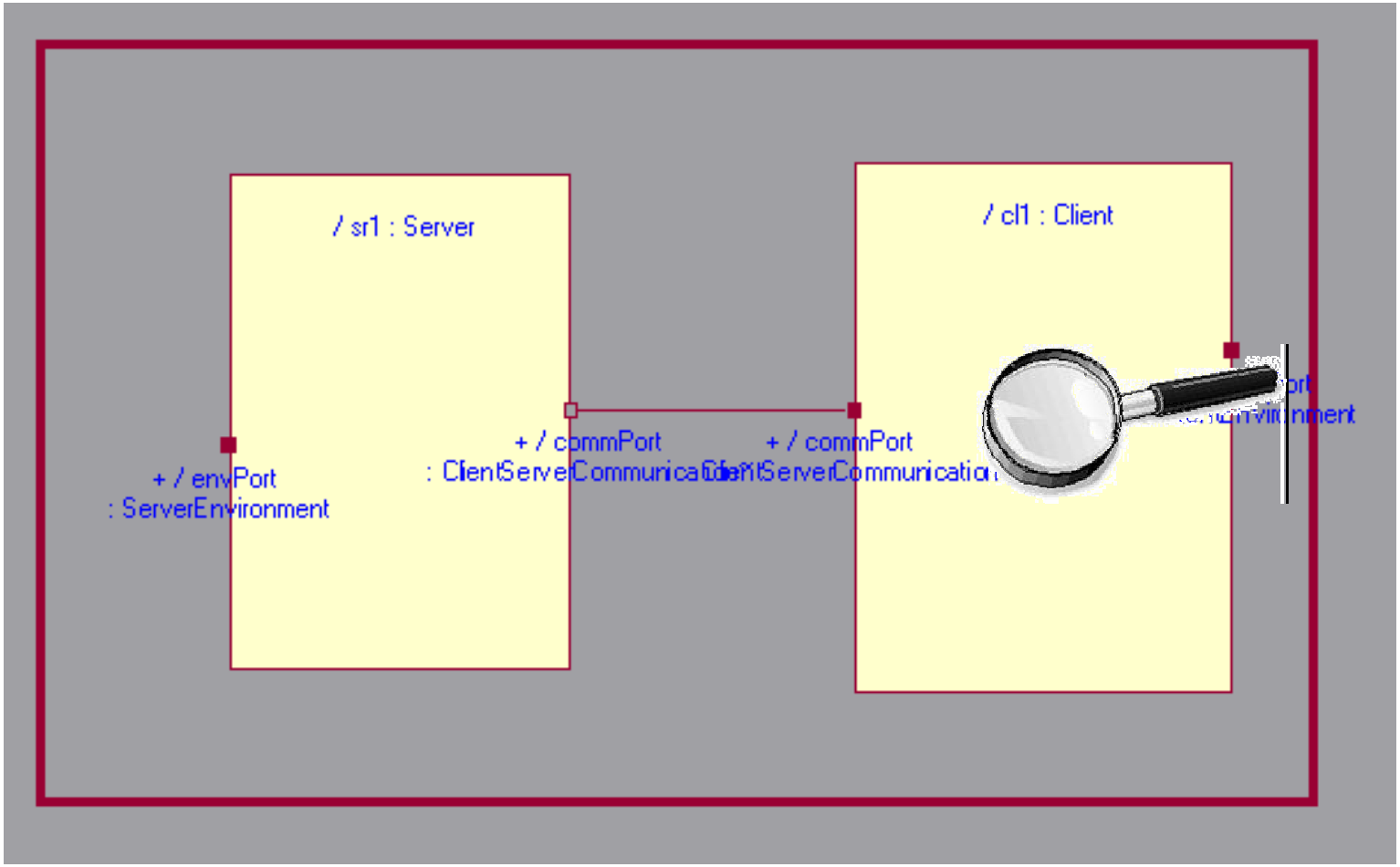


The solution

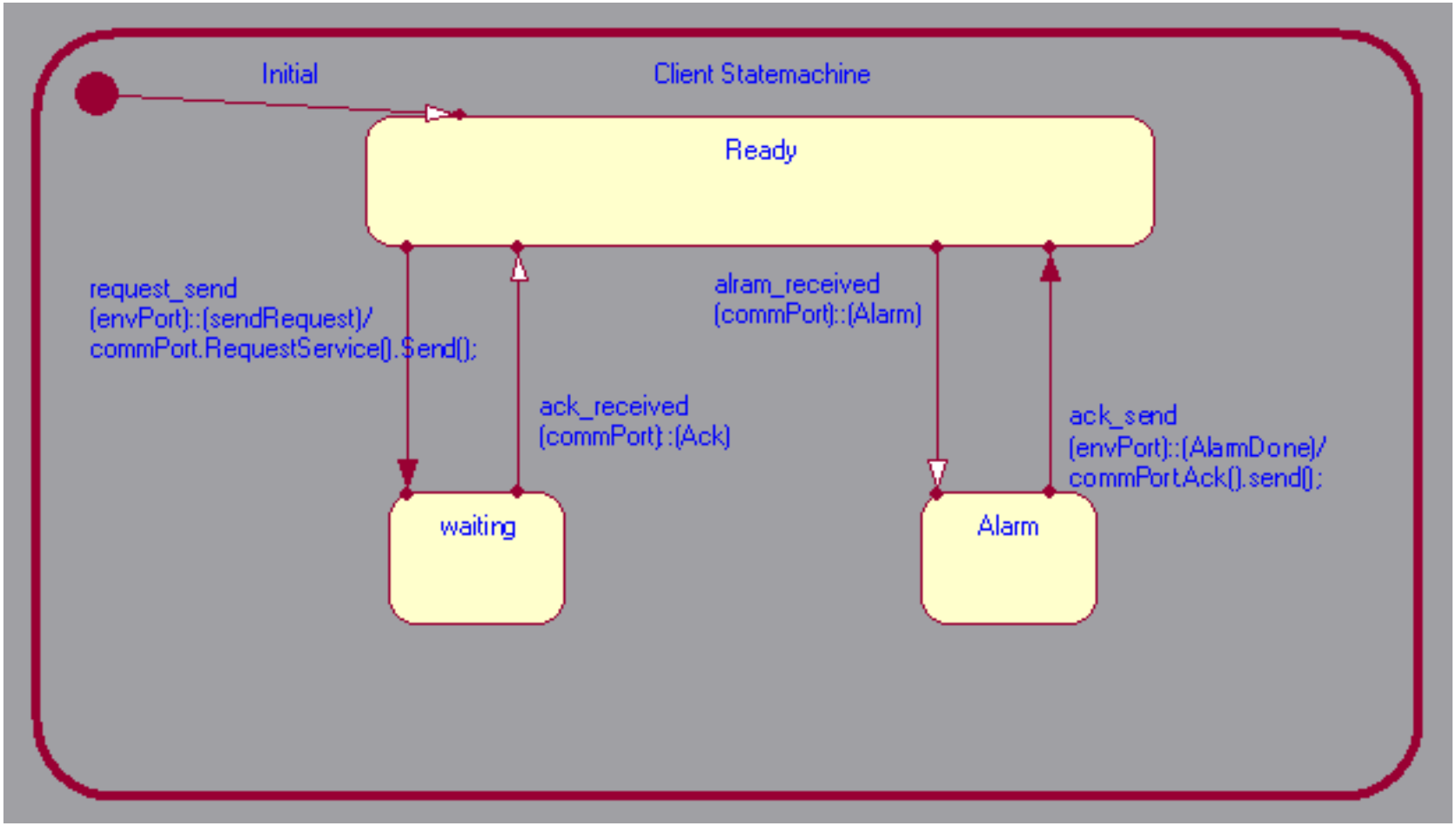
Integrate Rose-RT and RuleBase PE

extremeblue

Introduction to UML – Structure diagram



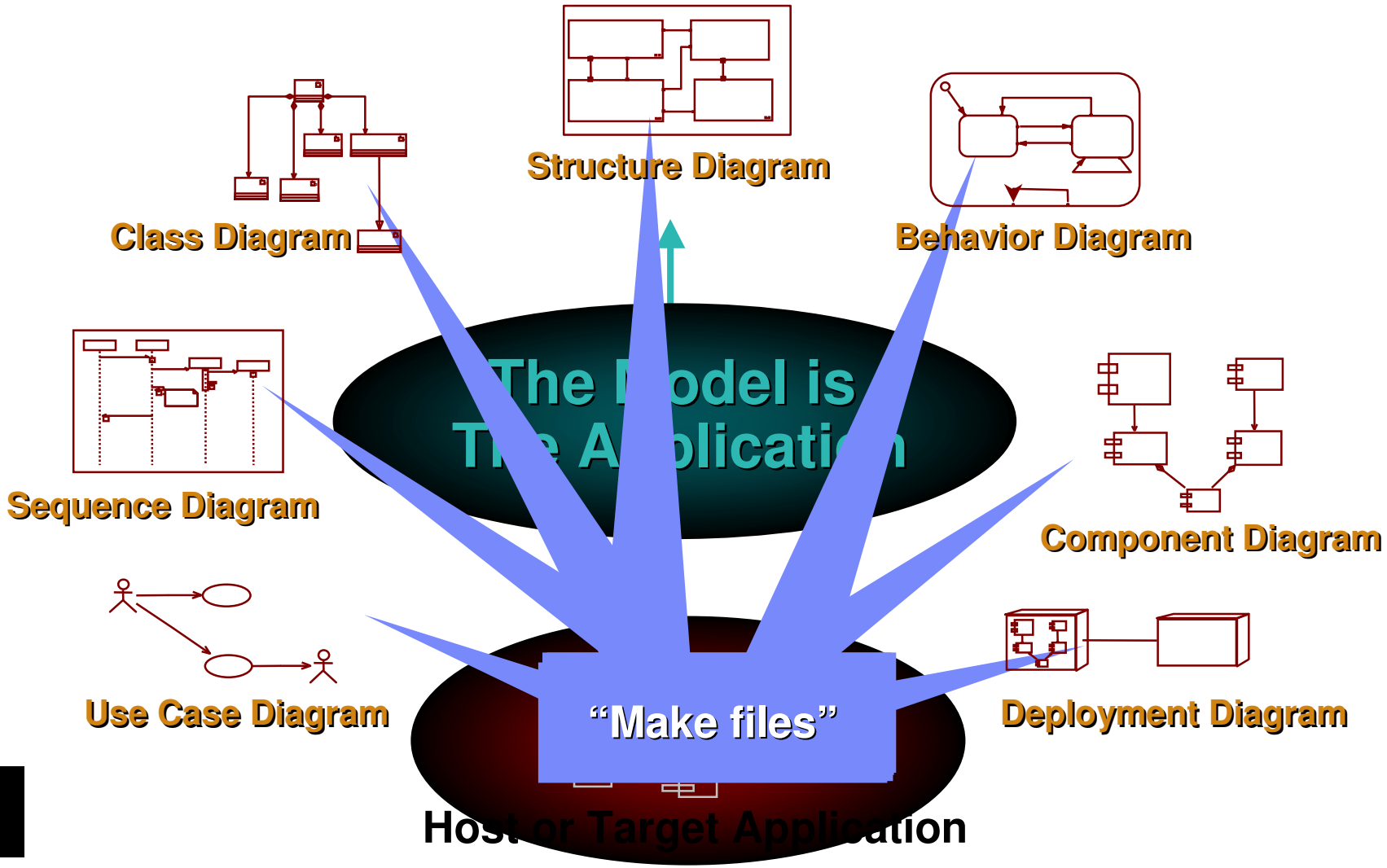
Introduction to UML – State diagram



IBM Rational Rose Real-Time

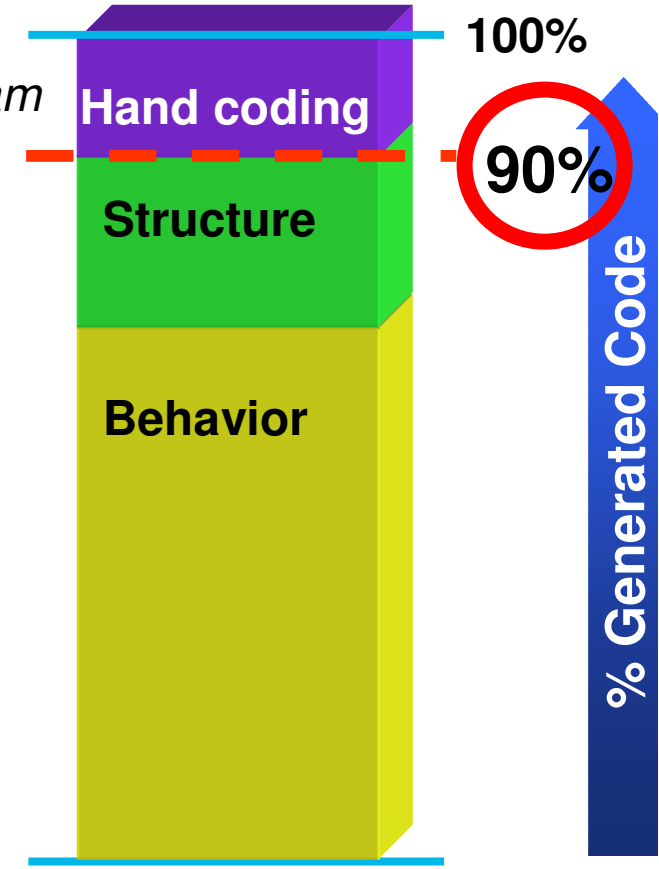
- Complete lifecycle UML development environment for real-time embedded software
- Leading Model-Driven Development tool for embedded systems
- UML model compiler generates complete C, C++, Java applications for host, RTOS and no-RTOS targets.
- Validate and debug host/target application with UML model debugger
- **Model is the Code !!**
- Successfully used in the industry

Model-driven development in Rational Rose RealTime



IBM Rational Rose Real-Time

- High level of Abstraction
  - Capture Behavior through *State Diagram*
  - System Interconnection through *Structure Diagram*
  - User works and debug application at model level
  - Model is the Code !!



RuleBase PE

- A platform for model checking and bug hunting.
- Has different engines:
  - BDD based
  - SAT based
  - Localization abstraction
  - Other specialized engines
- Success story – widely used

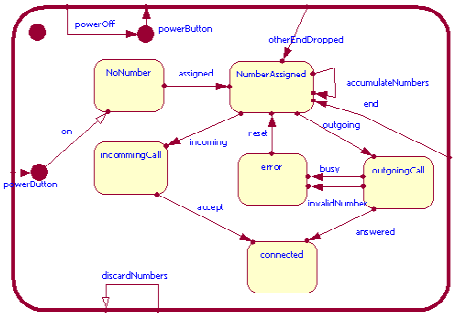
- IBM Products
  - Gigahertz Processor
  - Main Frame line (S/390)
  - Mid-range line (AS/400)
  - Workstation line (RS/6000)
  - PC line (Netfinity)
  - Super Computers (ASCI)
  - ASIC/OEM business
- Licensees
  - Marvell (Comm. Chips)
  - Zoran (Video Codec chips)
  - ST Microelectronics (Emb. CPU line)
  - Mellanox (Infiniband Chipset)
  - Analog Devices (DSPs)

Model checking and Rose-RT synergy

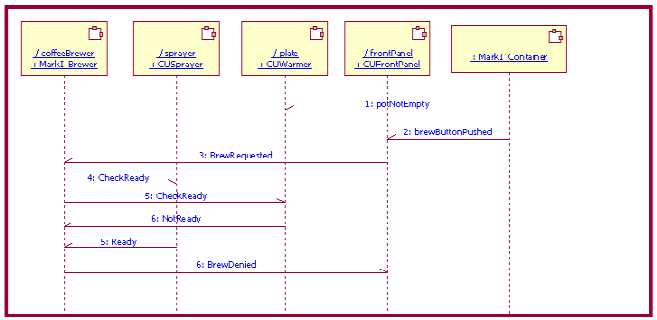
- The customers of RoseRT want to verify their models:
  - Bugs are expensive
- RoseRT models can be used in model checking:
  - simple translation of a RoseRT model to a GDL model
  - No disparity between model and system implementation
  - RoseRT models contain information that can be exploited for creating abstractions.
- The customers of RoseRT understand models
- It is possible to extend RoseRT for this purpose:
  - Define the specifications (PSL)
  - Present counter-examples from RulseBase PE graphically

SMARRT Solution

Rose RealTime Tool Set



State Diagram : Capture Behavior of the System in States Chart



Sequence Diagram : Shows a sequence of Events

Transform to PSL: State Diagram along with Design Constraints to target model checker input language

Design Constraints
Rose RT Model Extensions (Sequence Diagram)

Transform: Results into Rose RealTime Sequence Diagram showing The sequence of events that result in an error.

```
var Client_envPort_in : { Client_envPort_Empty,
Client_envPort_sendRequest,
Client_envPort_AlarmDone};
var Server_envPort_in : { Server_envPort_Empty,
Server_envPort_sendAlarm,
Server_envPort_requestDone};
assign Client_envPort_in := {
Client_envPort_Empty,
Client_envPort_sendRequest,
Client_envPort_AlarmDone};
assign Server_envPort_in := {
Server_envPort_Empty, Server_envPort_sendAlarm,
Server_envPort_requestDone};

var Server_port_2_handle : {commPort, envPort};
```

Pass to Model Checker

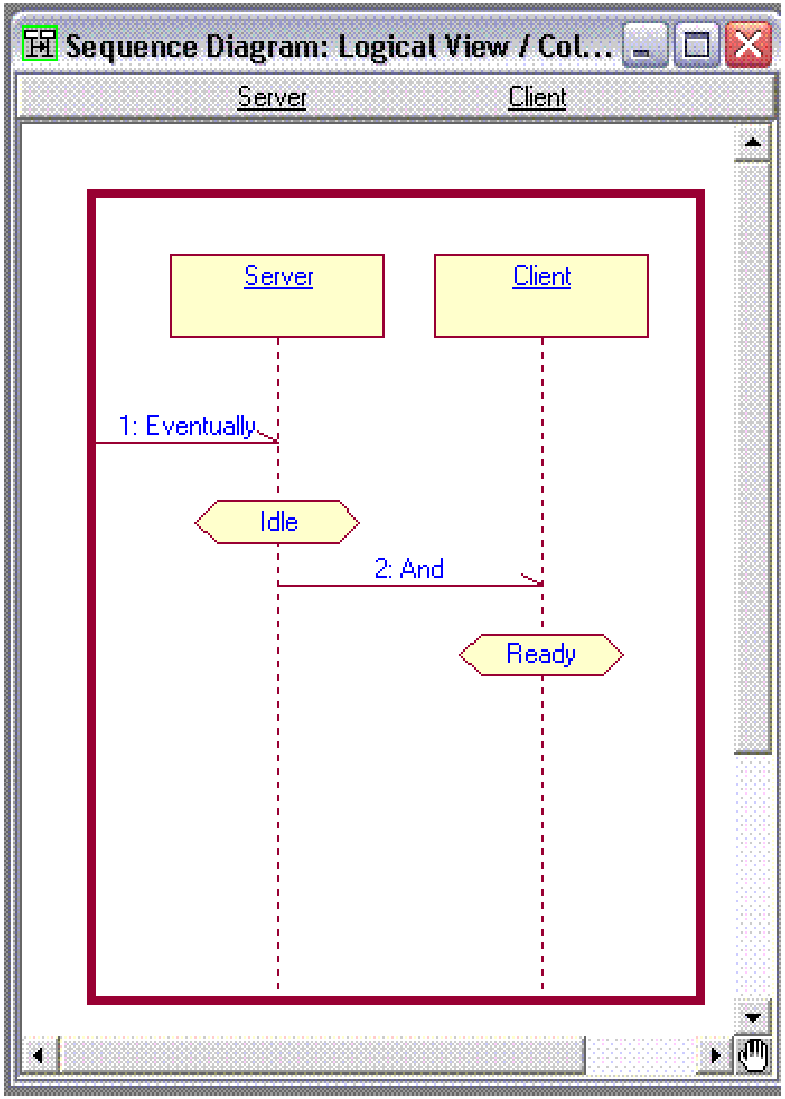
IBM Rule Base (Model Checker)

Store the Results

Results

Writing specifications

- Allow users to specify constraints graphically
- User-friendly interface – an extension of the known sequence diagram
- Sequence diagrams are able to depict:
  - Interaction between objects in terms of states and events
  - Timeline



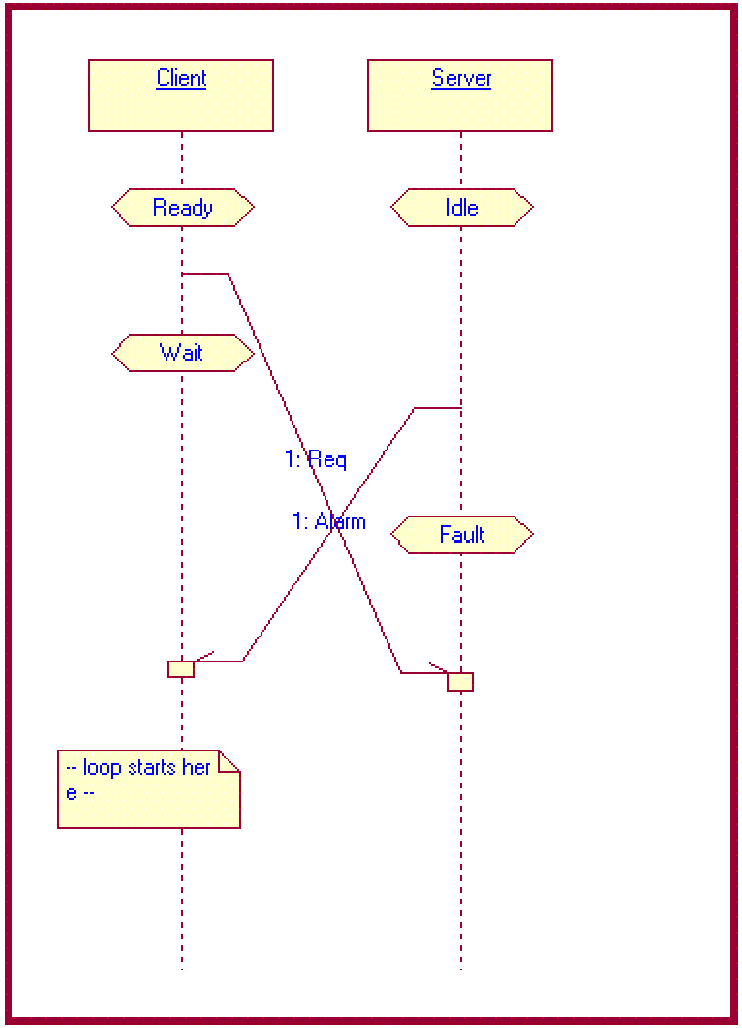
Extracting a model

The RoseRT model is translated into RuleBase PE model

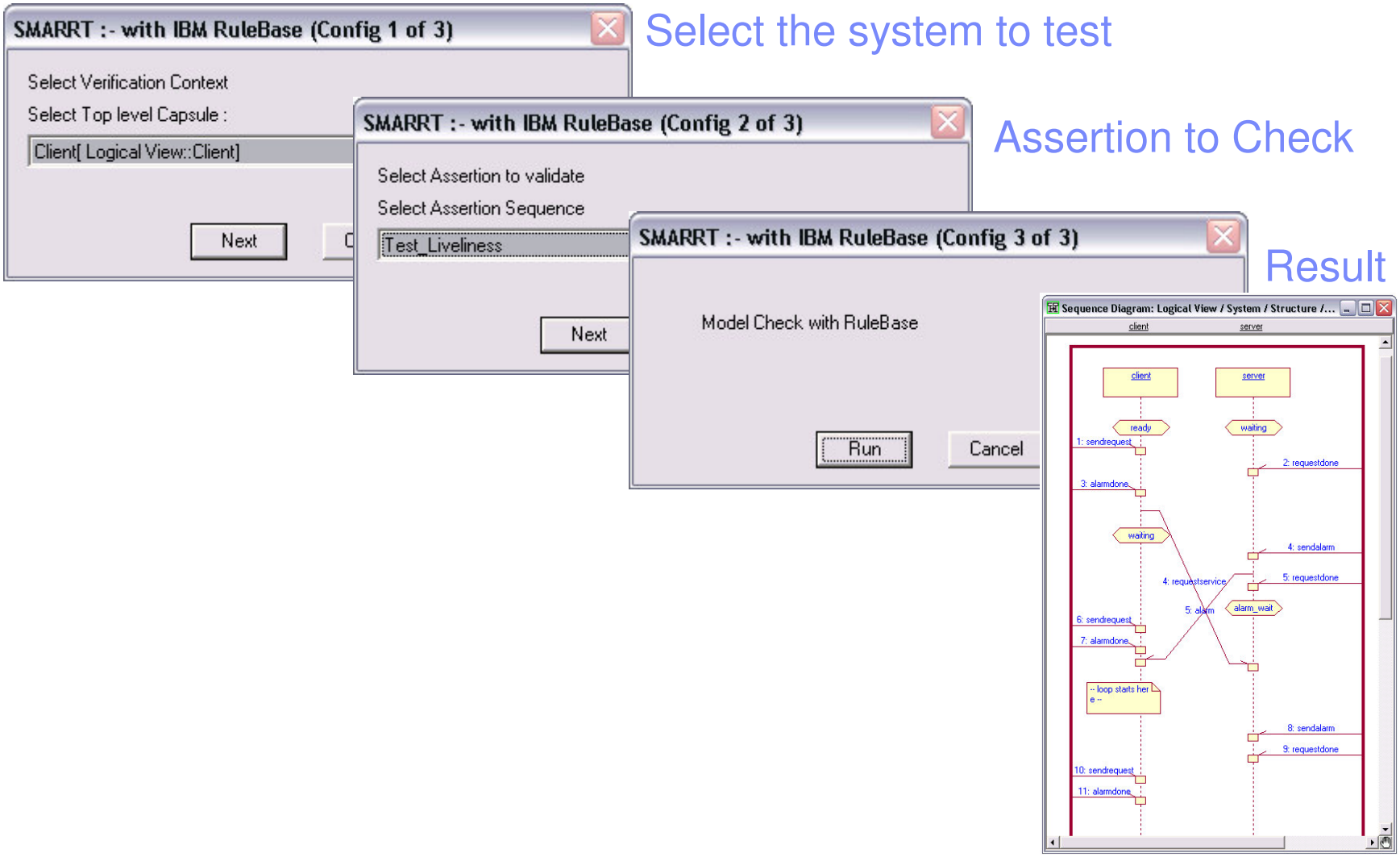
- Efficient template model for every building block
  - Message queue
  - Priority message queues
  - Environment port
- Automatic translation of the state diagrams:
  - A variable records the state of each capsule
  - The capsule's state determines behavior
  - Introducing nondeterministic:
    - Environment
    - Concurrency

Presenting the Results

- Converts Rule-Base Text trace file to UML Sequence Diagram
- Maps the PSL state variables back to Rose Real Time States
- Shows the States and event that triggered the transition from one state to another
- Shows message send and receive time.
- Easy to understand problems

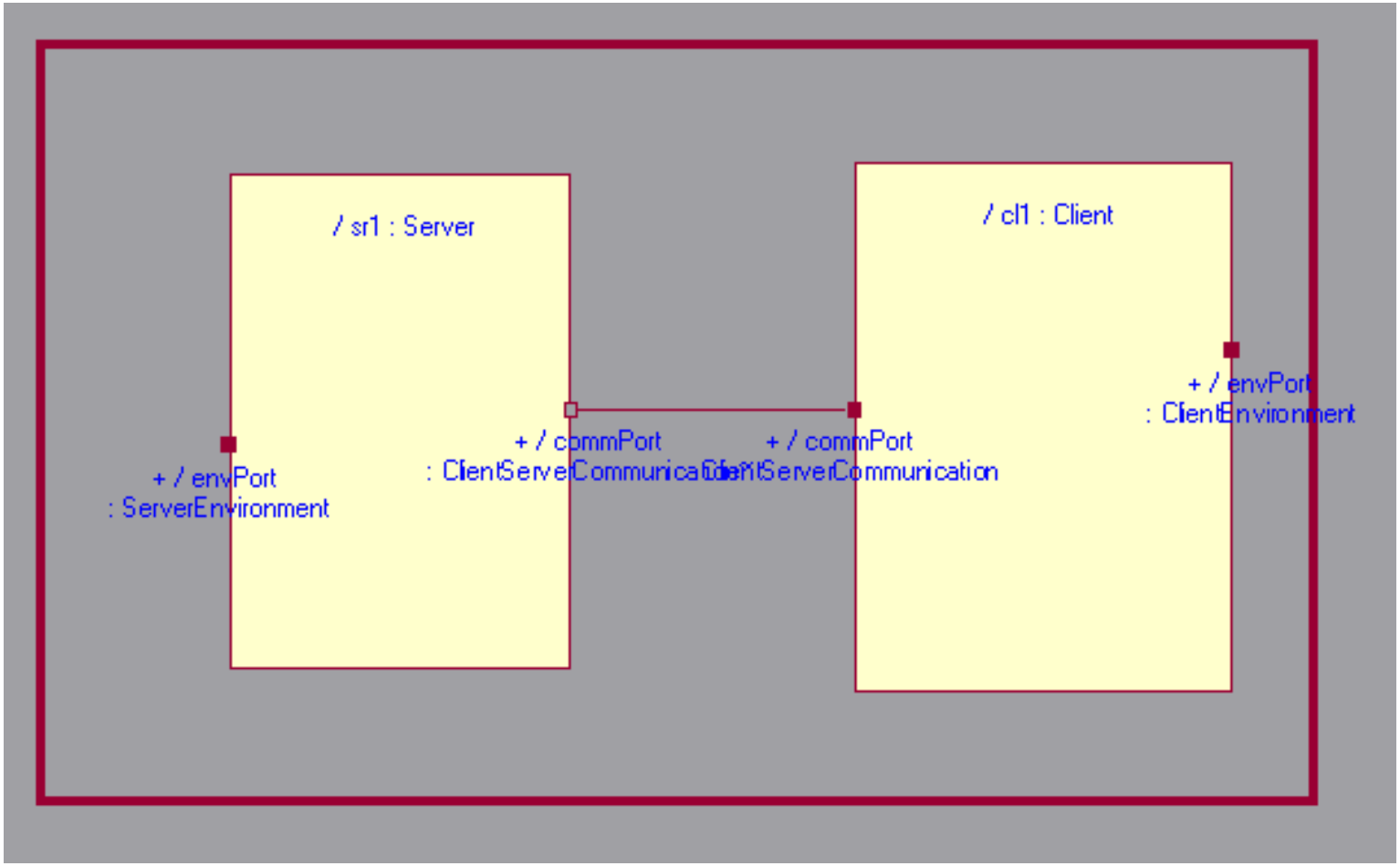


Model Checking made easy : Screen Snapshots

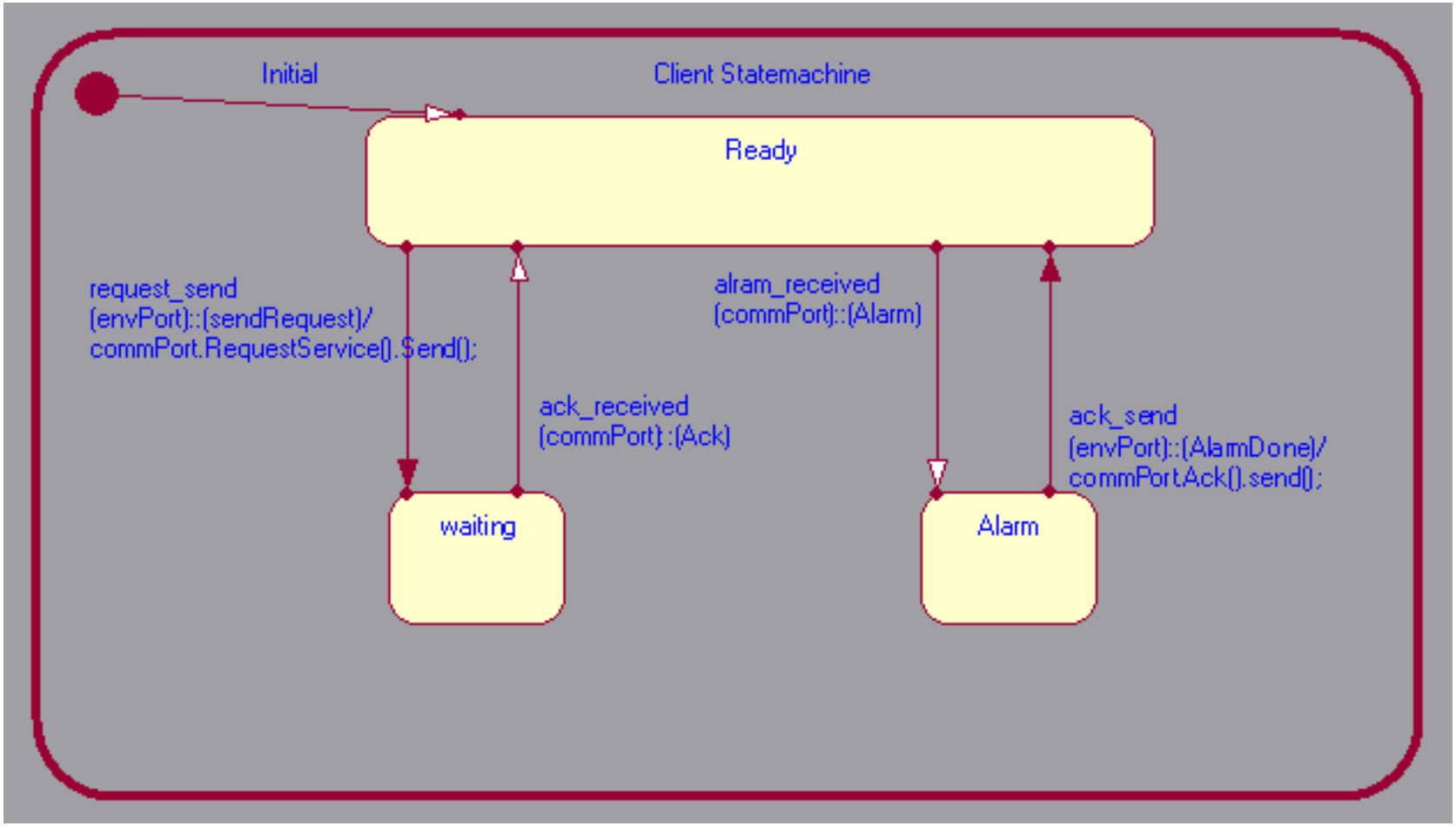


Demo – Client Server Example

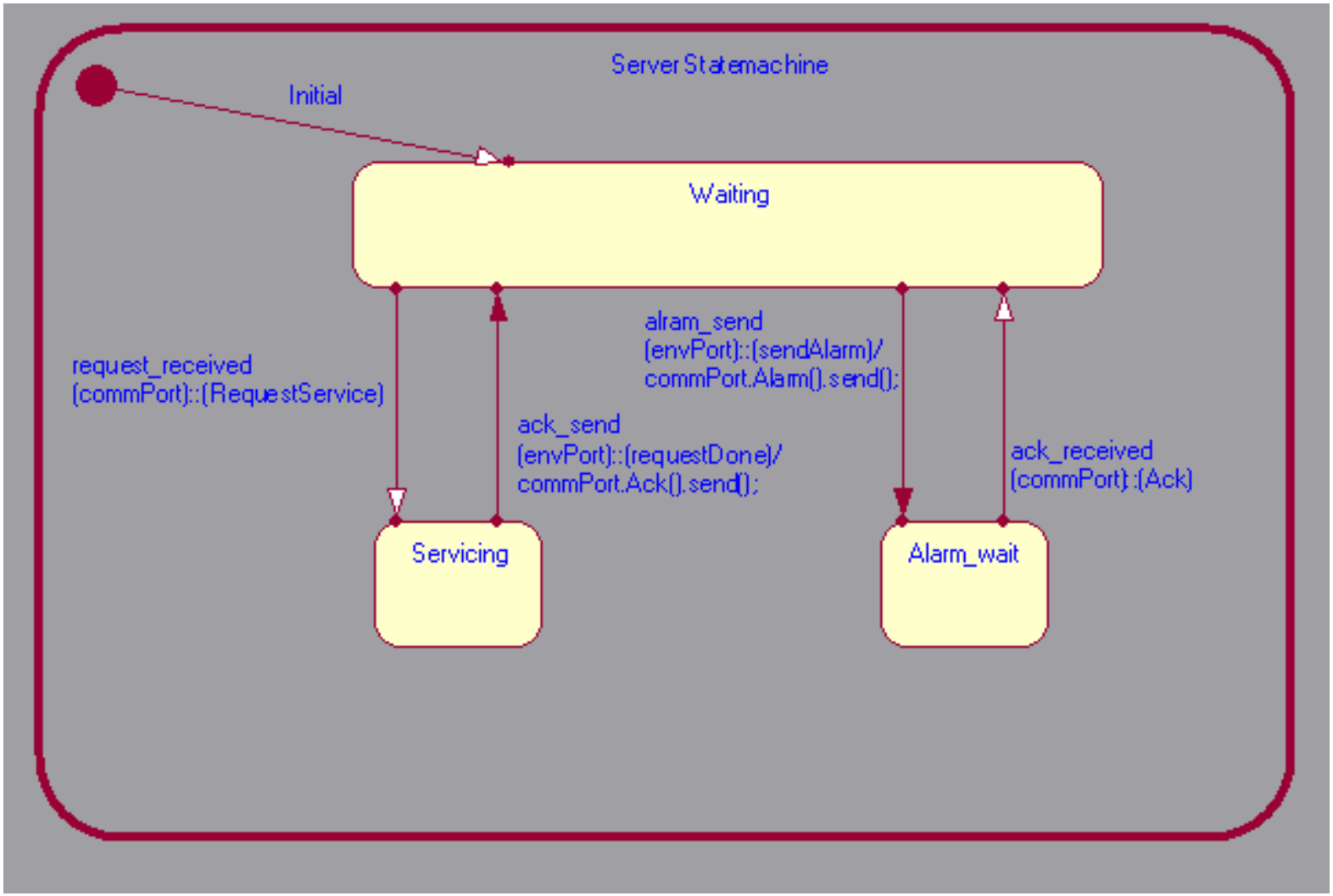
- Shows a simple Client-Server example

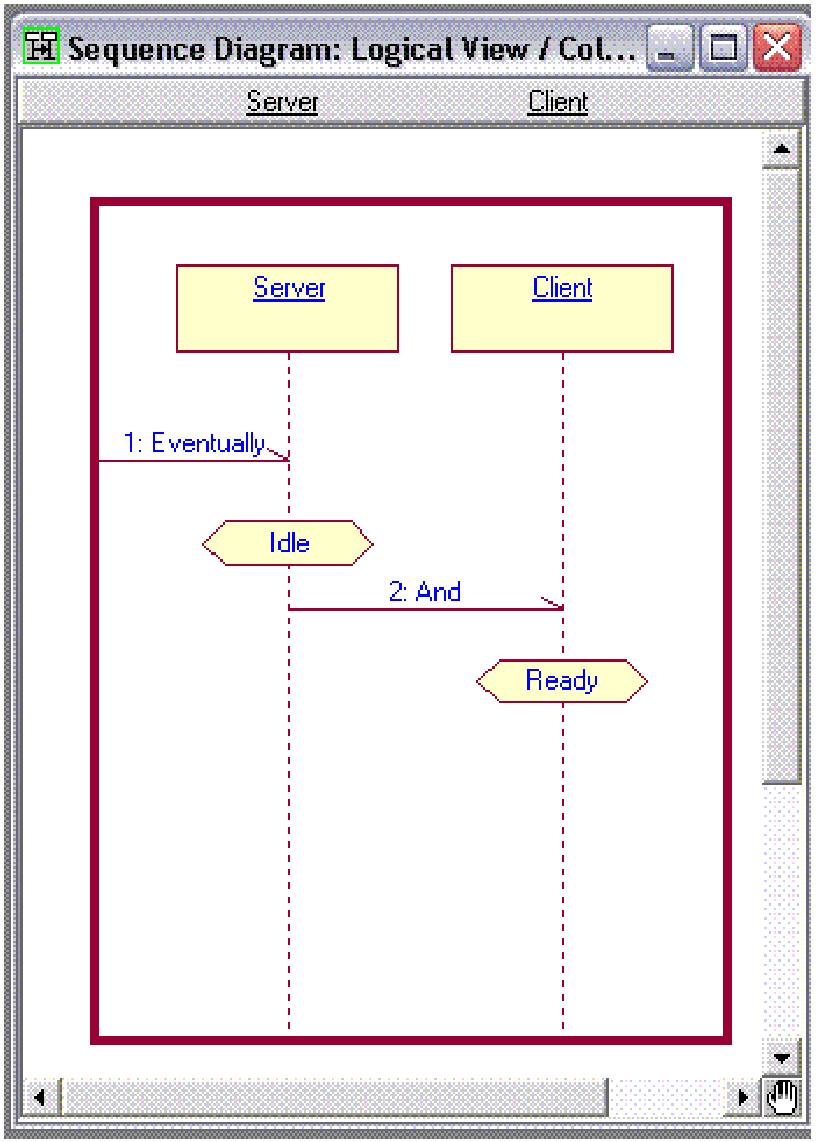


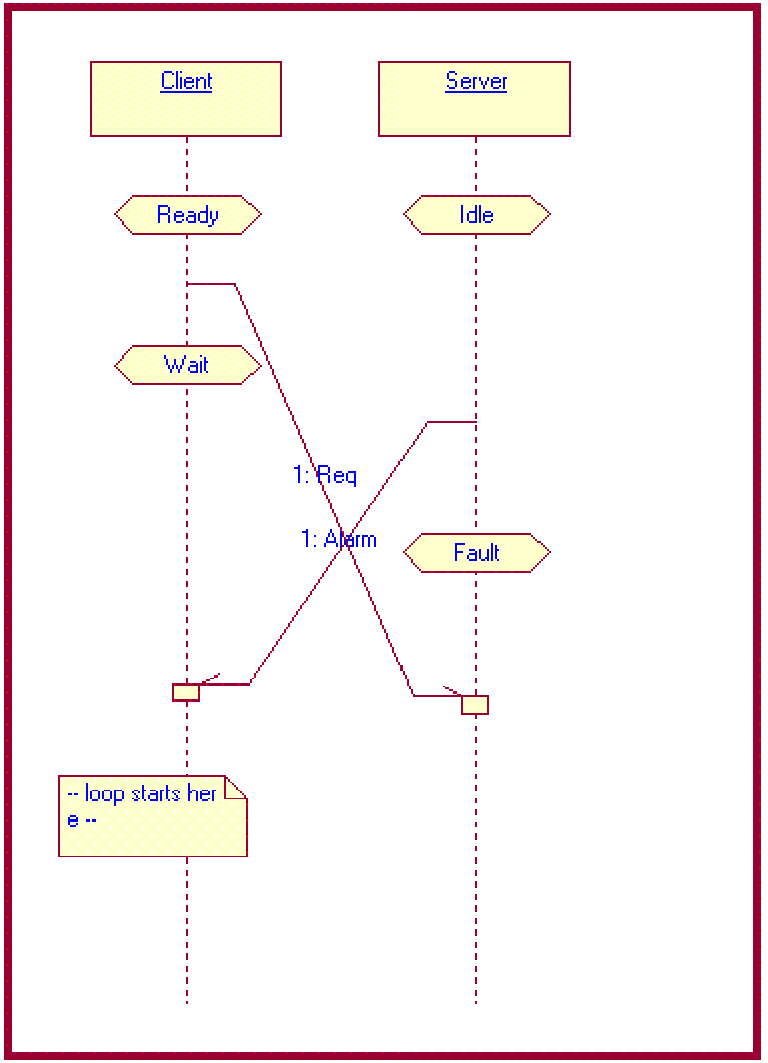
Demo – Client Server Example



Demo – Client Server Example





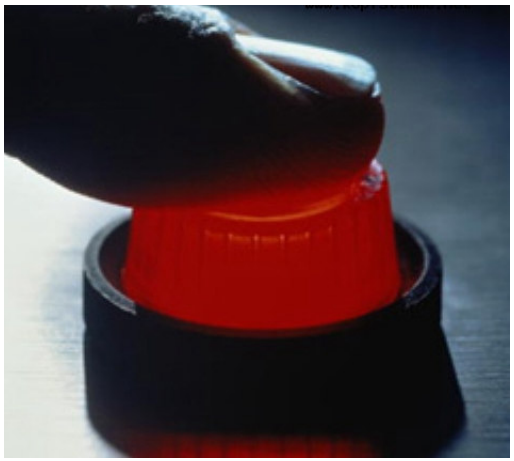


Benefits of SMARRT

- Rose RT (using SMARRT) can detect
  - Race Condition
  - Dead Lock Detection
  - Live Lock Detection
  - Termination Problems
  - State Reachability (finding States that are never reached).
  - Design/Safety Constraints
    - When object A is in Run State, object B should not be in Run State. Any violation ?
  - Adherence to System Criteria
    - Does there exist a case where the flight control system does not deploy the breaks while landing ?

Benefits of SMARRT

- Model Checking at a click on a button tool
  - No explicit model checker tool expertise required.



Model checking results on benchmarks

Benchmark	UML Model Complexity				Model Checking Complexity				
	State m/c's	States	Trans-itions	Trigg-ers	BDD Size	State Space	Nodes Allocated	Memory (MB)	User Time(s)
Client Server	2	6	8	4	67	53056	5657	38	0.85
ATM Machine	2	10	15	9	27	800	2778	38	0.54
ATM Error	2	10	15	9	28	672	4752	38	0.77
Railroad 1	4	24	31	11	673	$9.47*10^{10}$	727518	52	13.21
Railroad 2	4	24	31	11	673	$9.47*10^{10}$	727153	52	24.89

Thank You