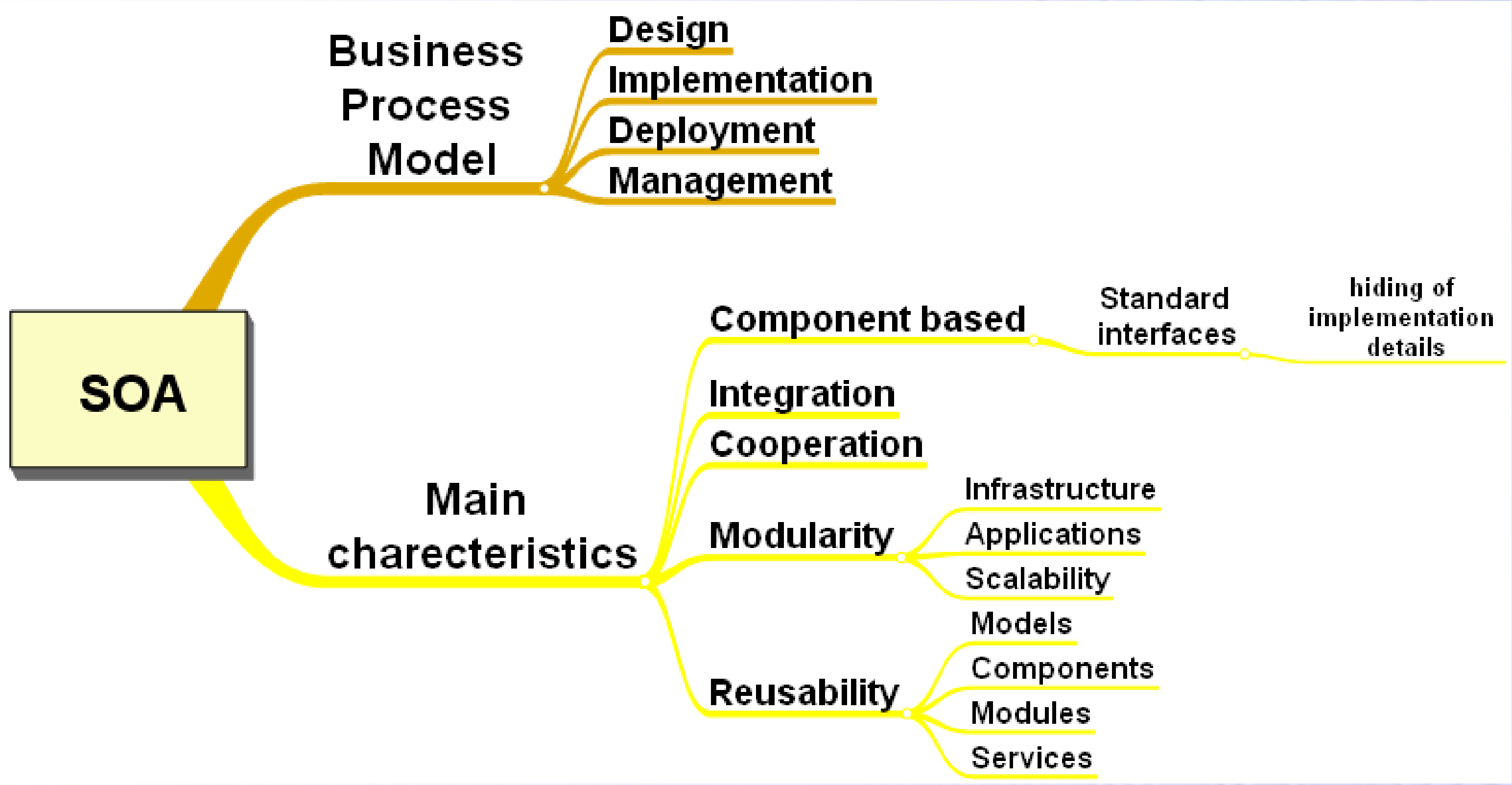
IBM

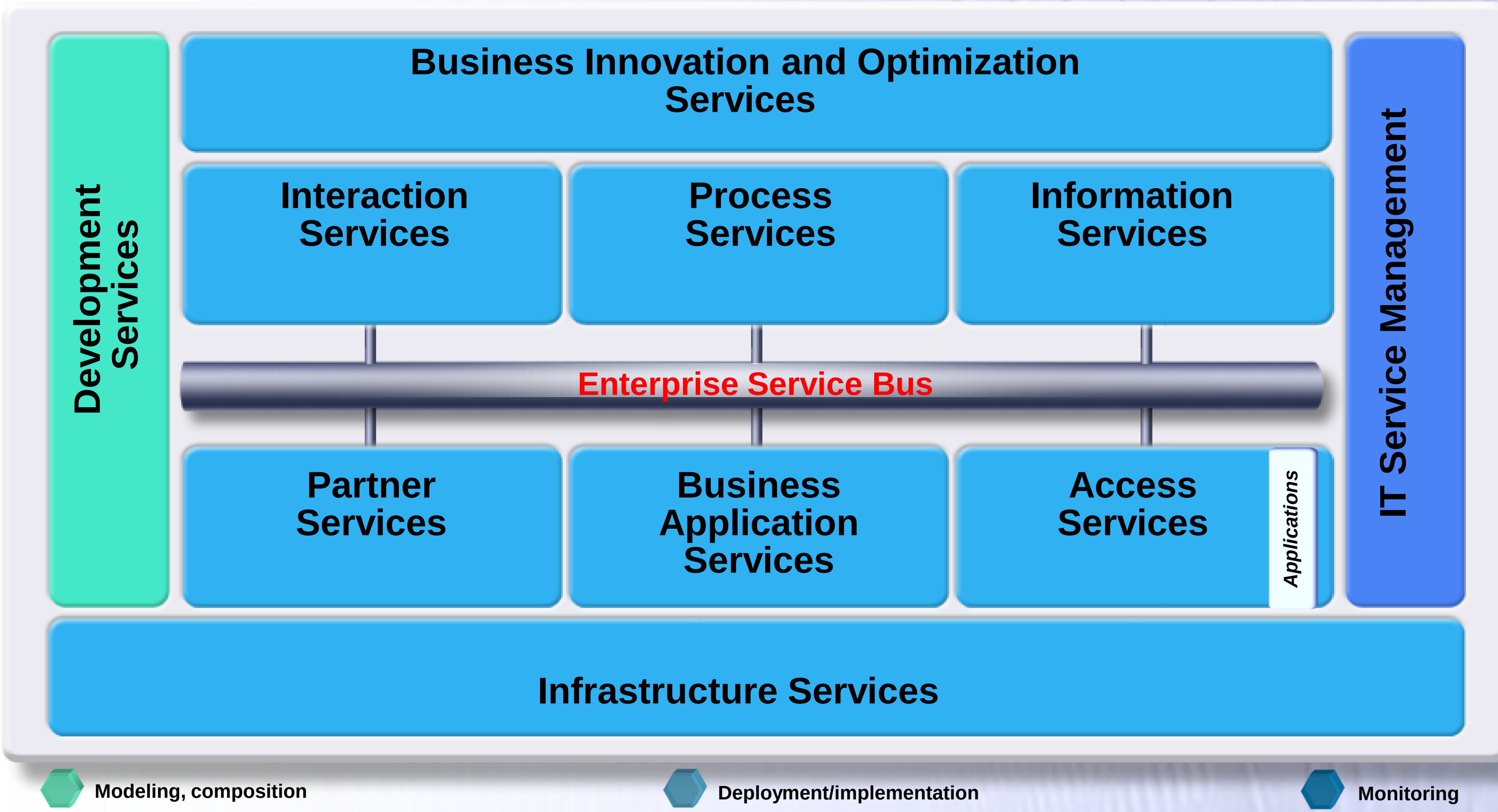
Business Process Transformation

Based on the presentation of András Pataricza,
Budapest University of Technology and Economics
@ IBM Academic Days 2006

Service Oriented Architecture



SOA reference architecture

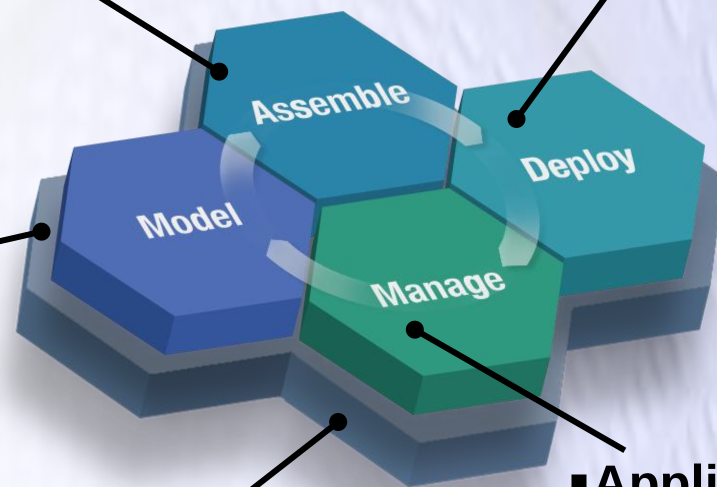


SOA lifecycle

- design
- testing

Integration

- human
- process
- information

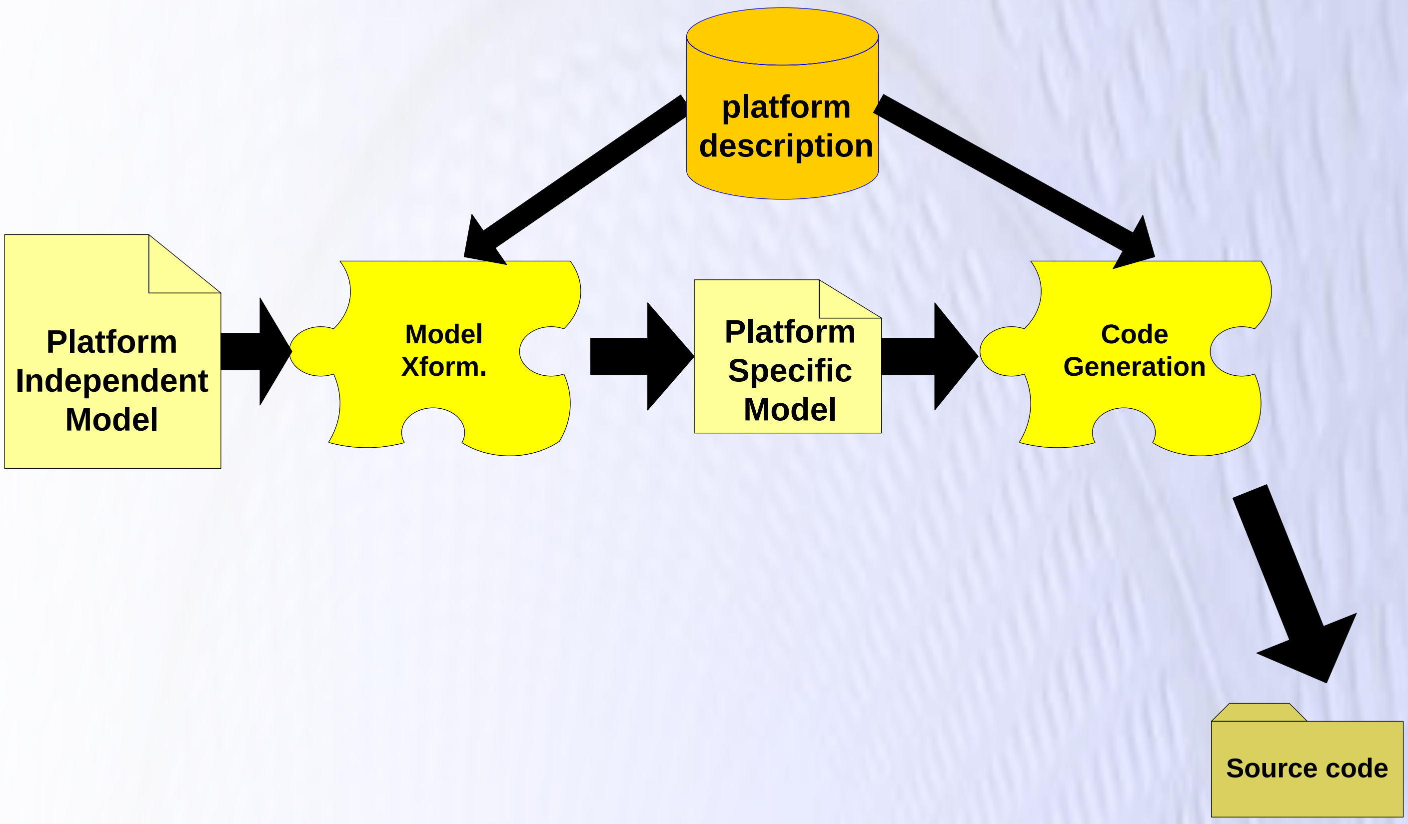


- Requirement analysis
- modeling & simulation
- design

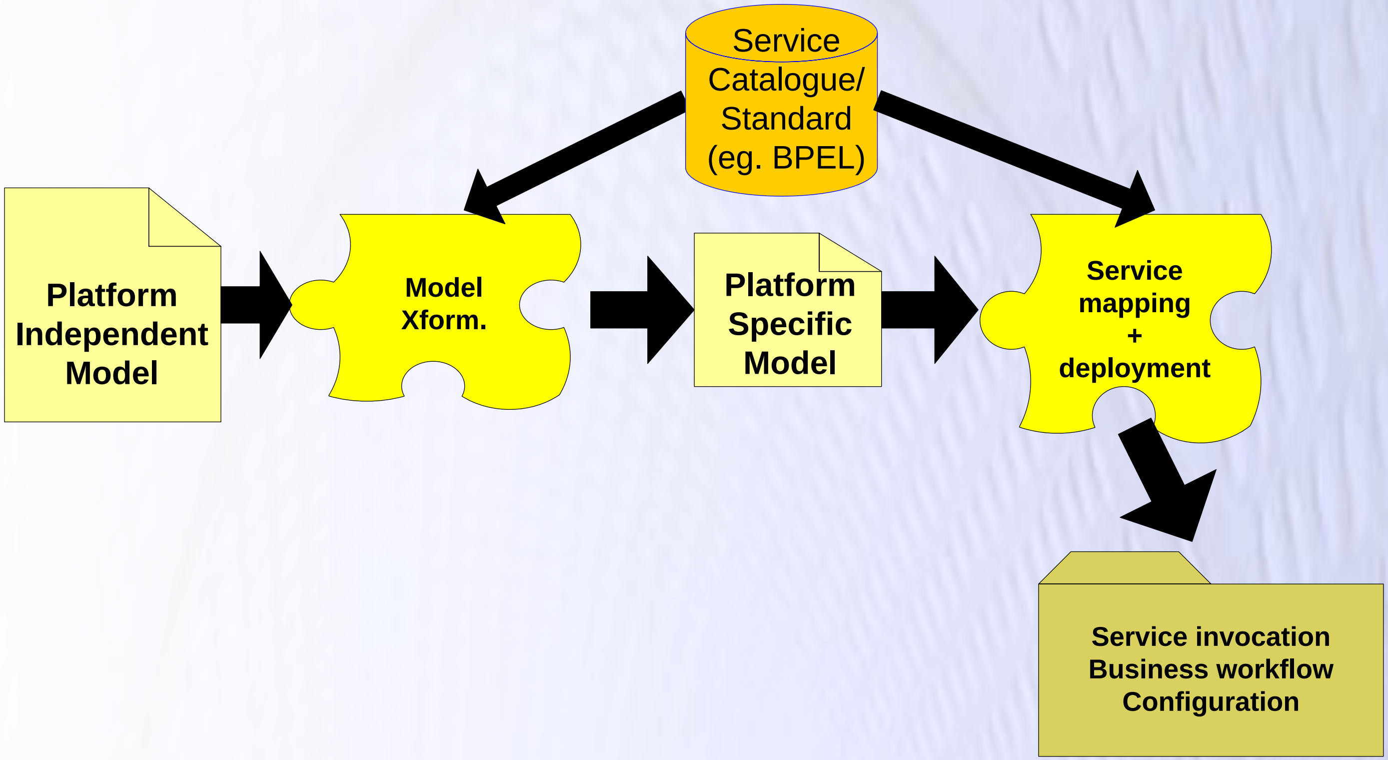
- Application and service monitoring
- Identity management
- Business indicators

- Financial
- Business/IT harmonization
- Process control

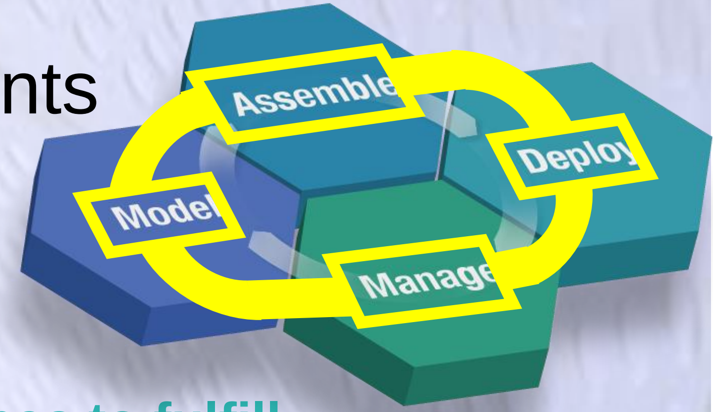
Model-Driven Architecture for Classical Approaches



Model-Driven Architecture for SOA

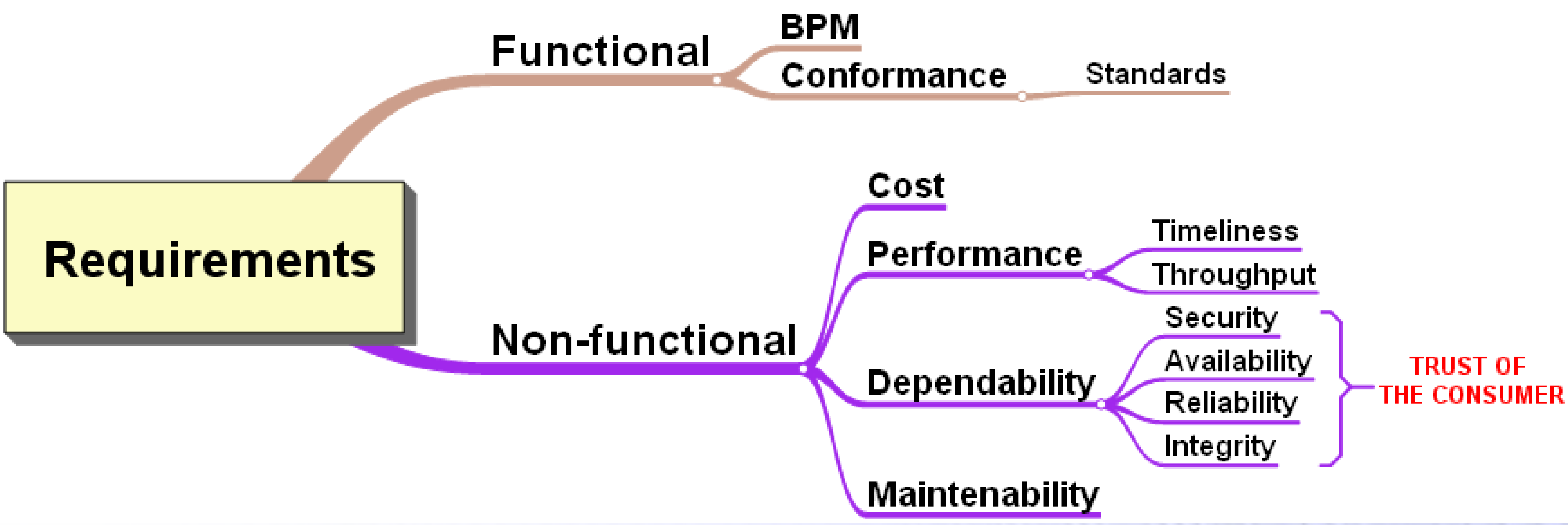


Componentization and requirements

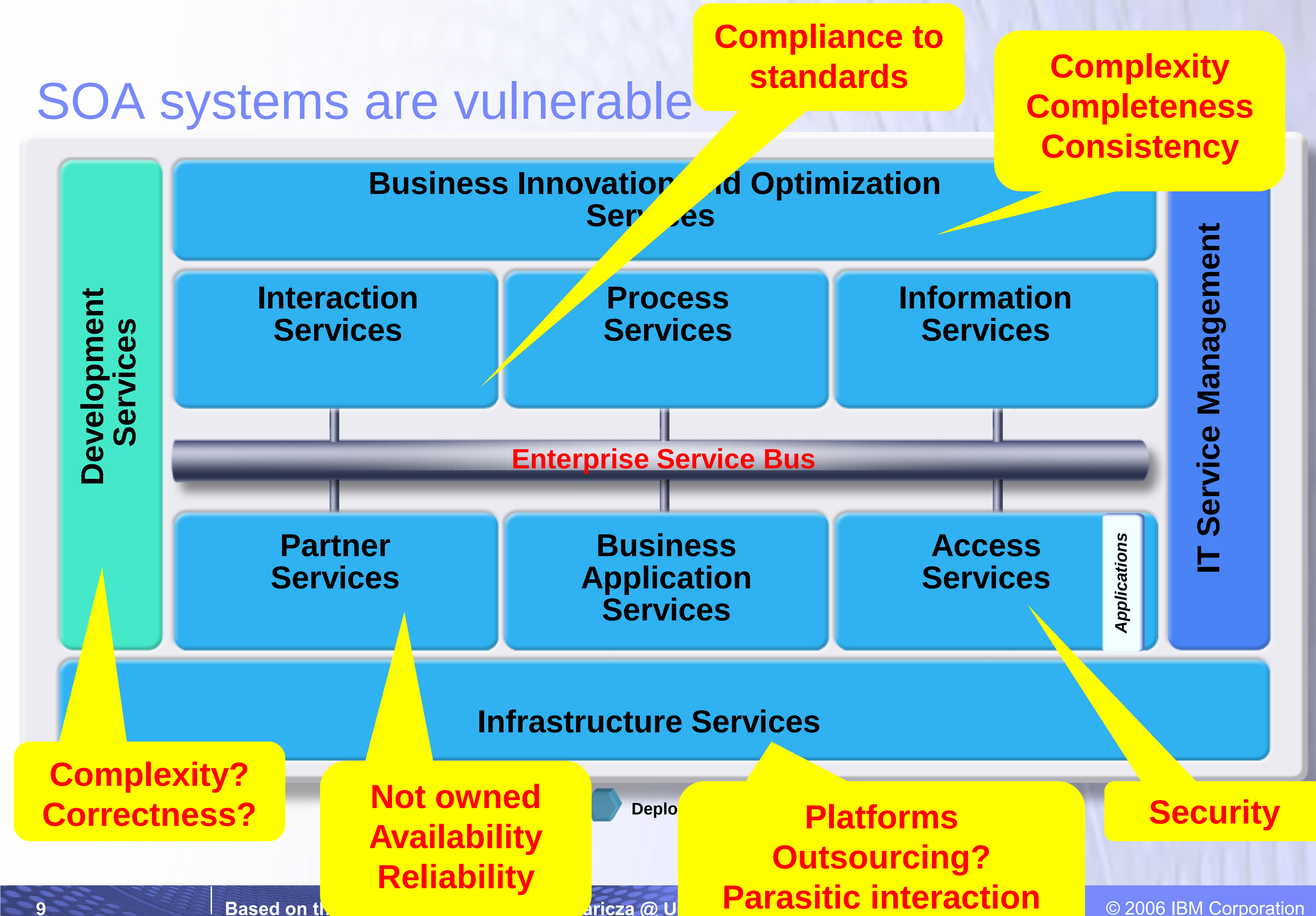


A functionally correct system has to fulfill additional non-functional requirements, as well

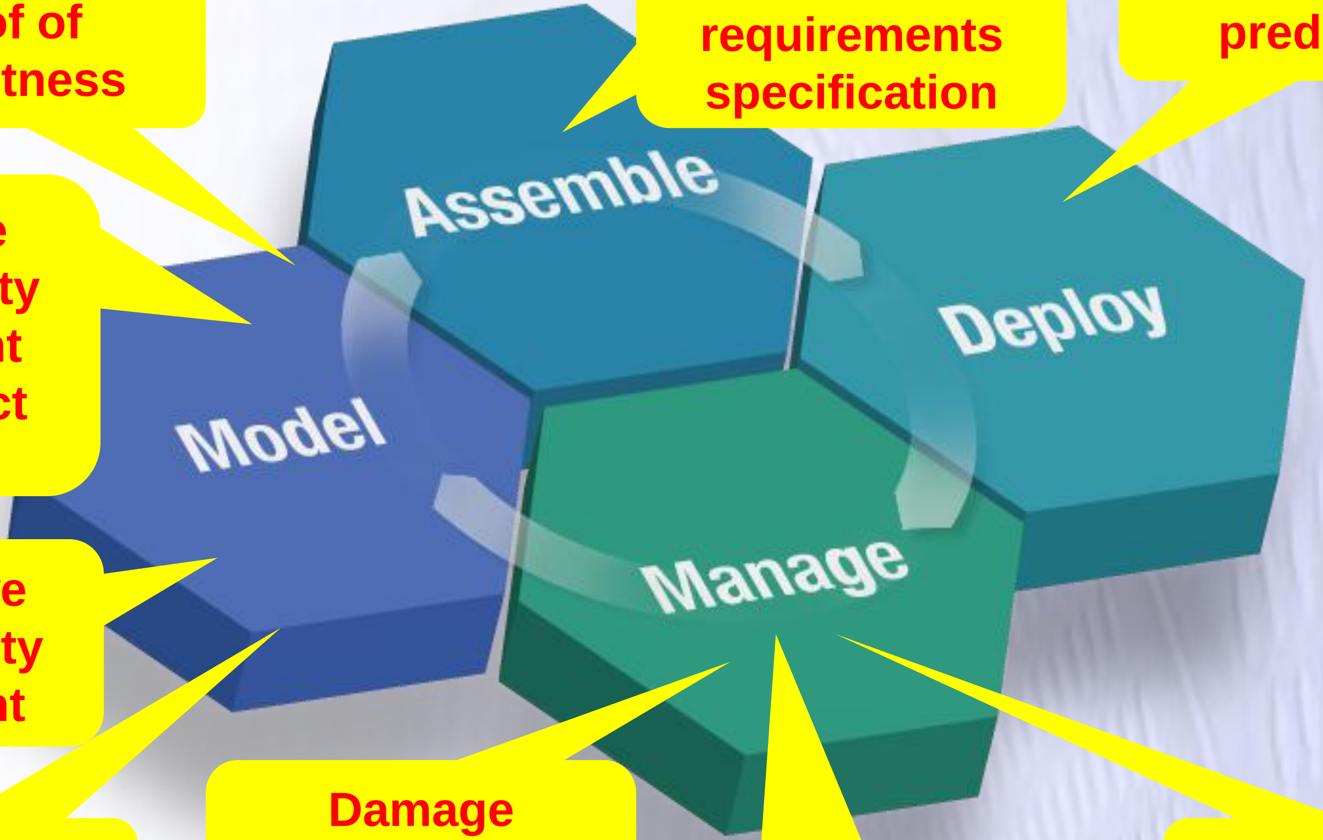
Requirement classes



SOA systems are vulnerable



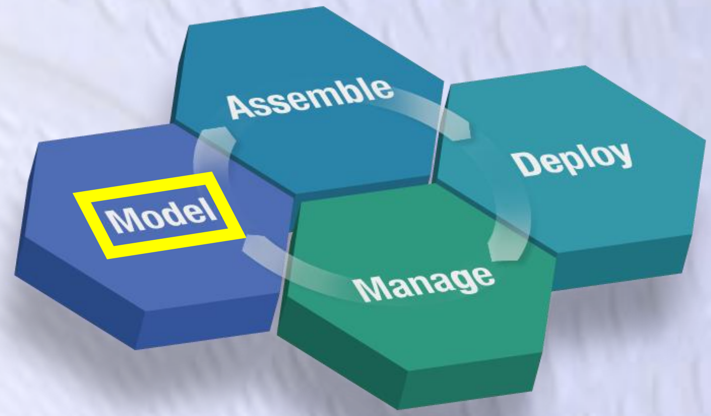
V&V problems in non-functional design



- Proof of correctness
- SLA requirements specification
- Availability … prediction
- Qualitative dependability assessment (fault impact analysis)
- Quantitative dependability assessment
- Performance prediction
- Damage confinement and recovery
- Error detection
- Performability control

IBM

Core: Mathematical Analysis



Huge complexity ->
Importance of mathematical analysis integrated into the design workflow

Mechanized approach

Design

Engineering model

Implementation

Code gener

Autom

m

Mathematical analysis

Mathematical model

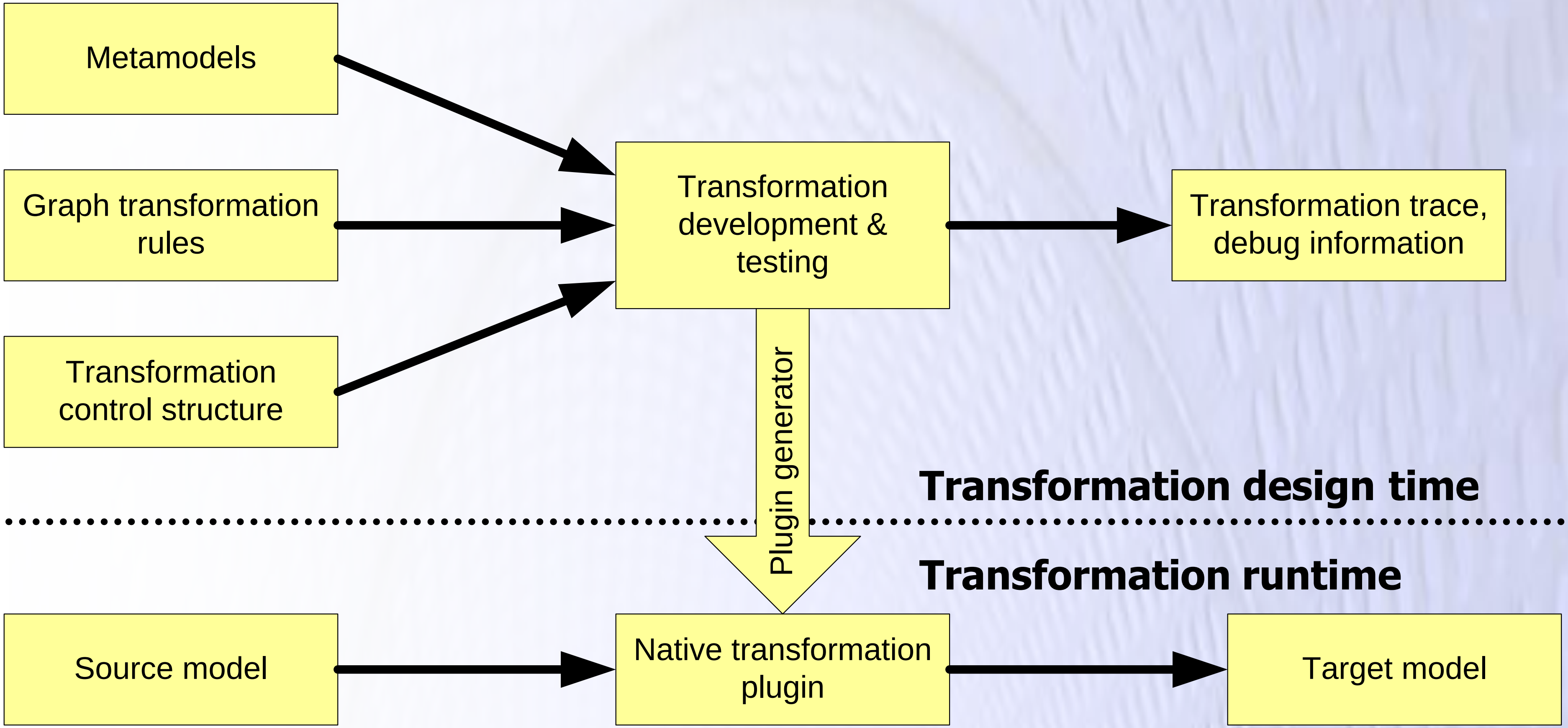
Analysis tool

HIDE

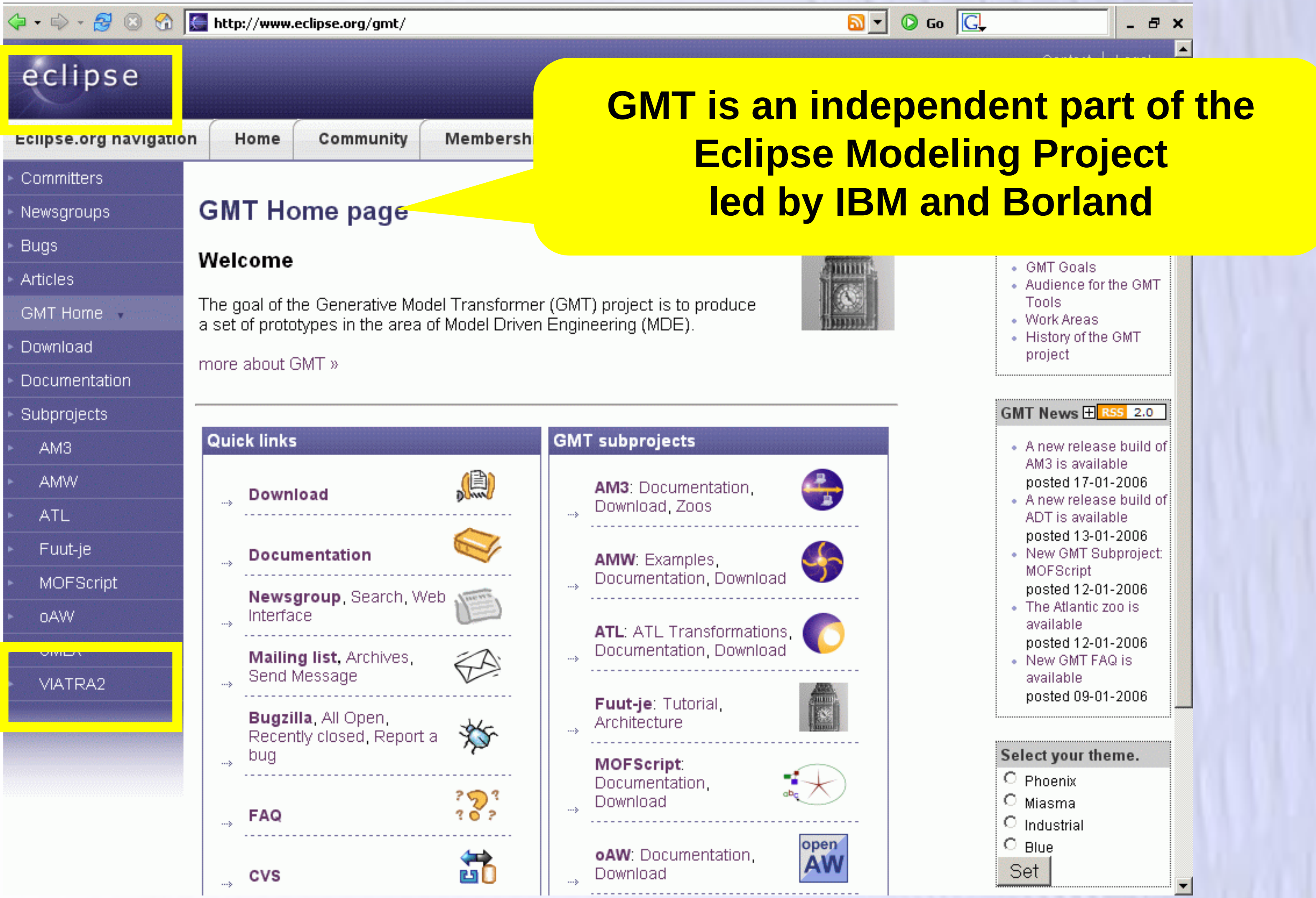
ESPRIT Open LTR 27439

High–level Integrated Design Environment for Dependability

Transformation development



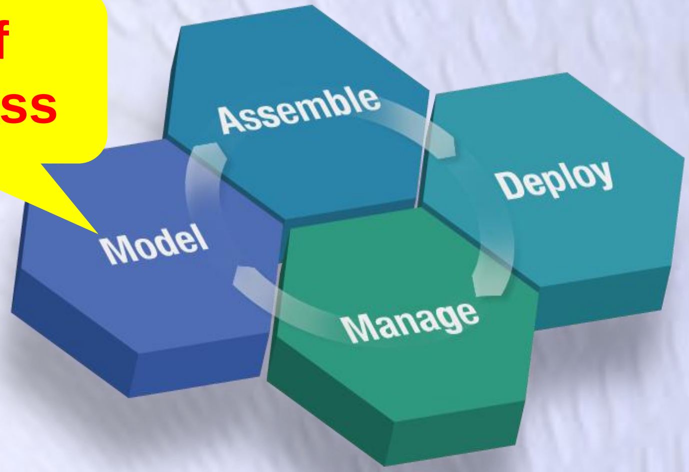
VIATRA 2 as Eclipse Generative Model Transformer



IBM

Design verification

Proof of correctness

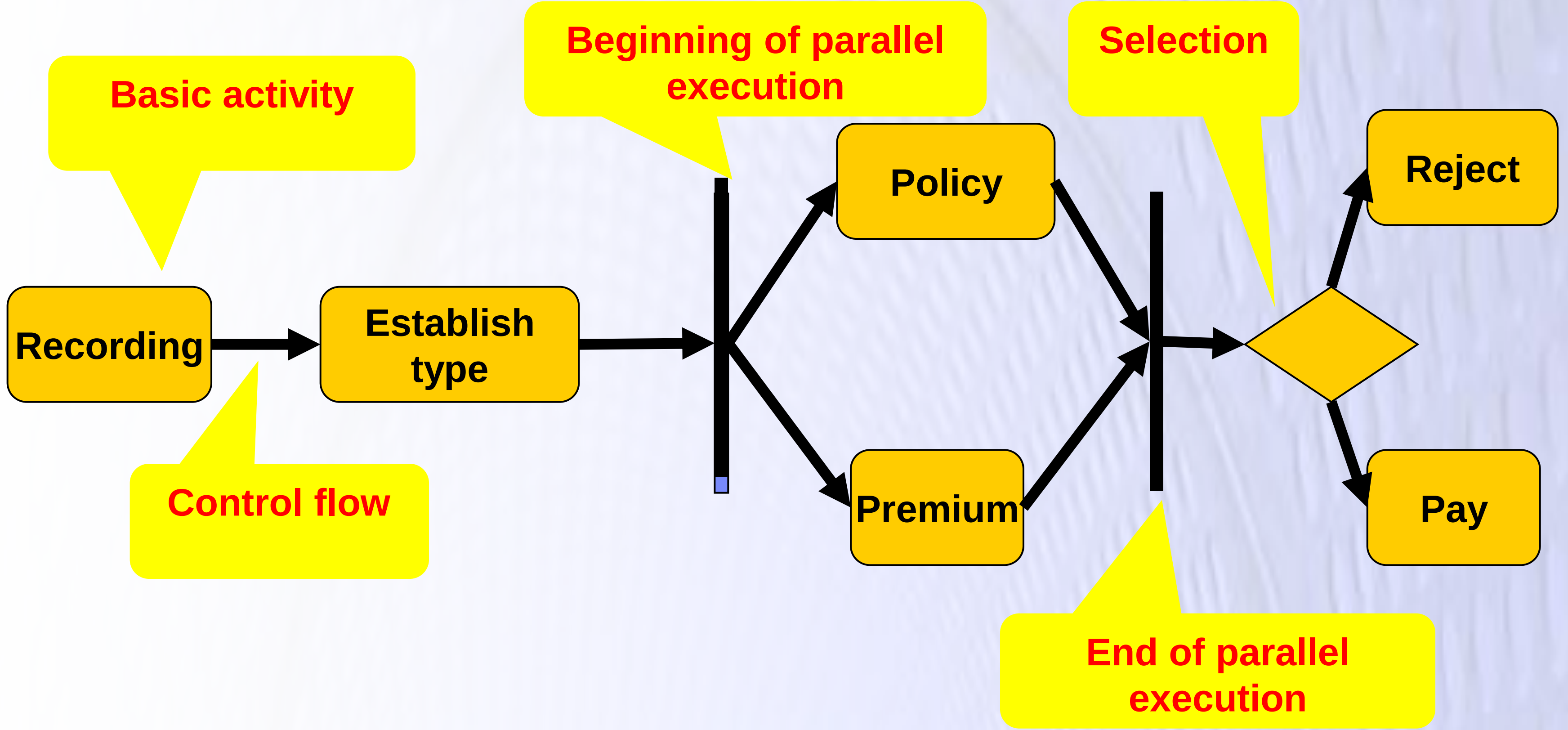


Complex, critical business processes require a proof of correctness covering ALL the cases of operations

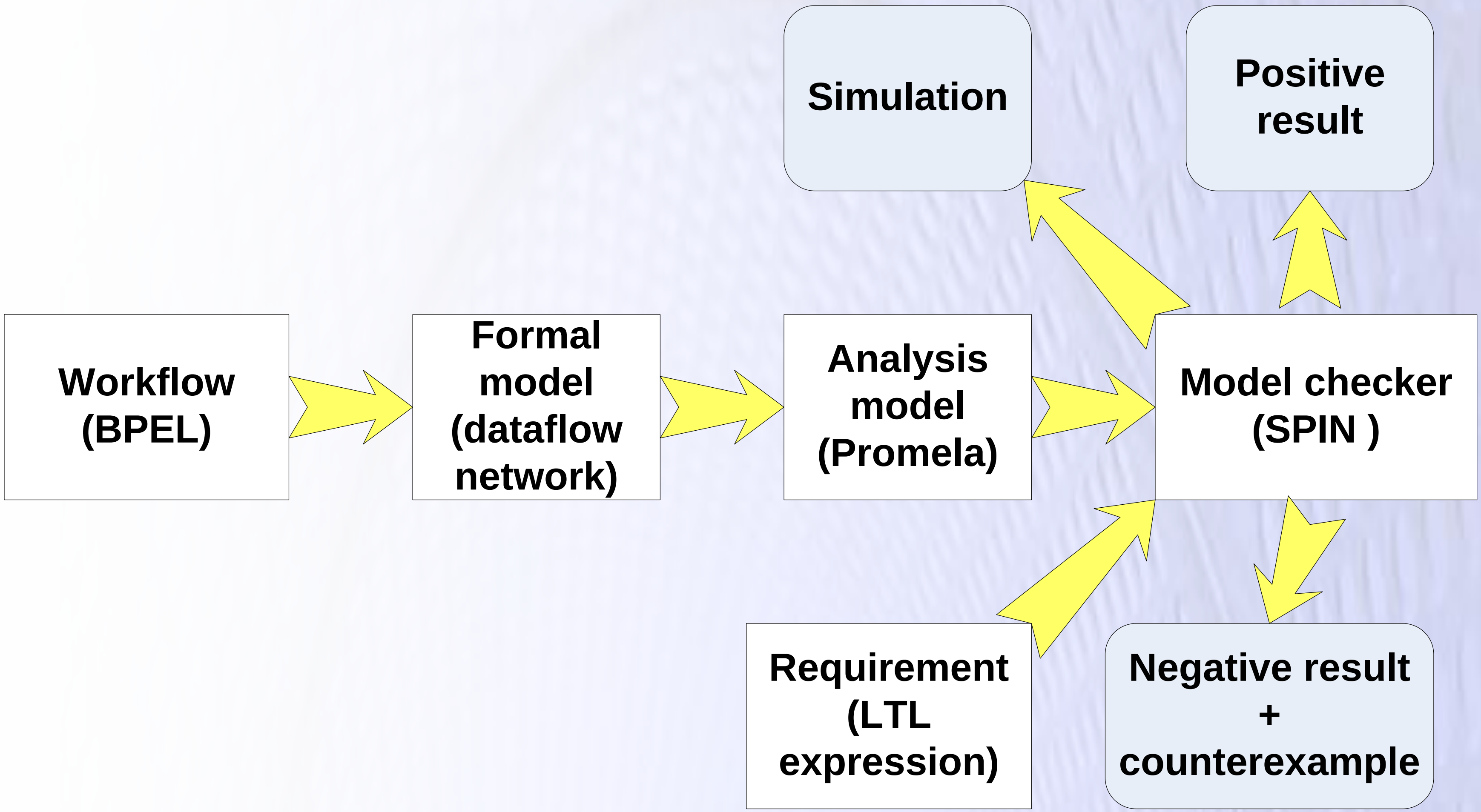
Objectives

- **Service composition (e.g. BPEL)**
  - **Widespread tool support**
  - **Design errors in choreography**
  - **Lack of formal verification**
- **Objectives:**
  - **Formal proof of compliance to the requirements on workflow**
  - **Derivation of mathematical analysis models by model transformations**
- **Formal analysis of workflows**
  - **Formal workflow semantics**
  - **Formal verification of properties**
    - **E.g. variable access**
  - **Fault simulation: assessment of error propagation**

A Workflow Example



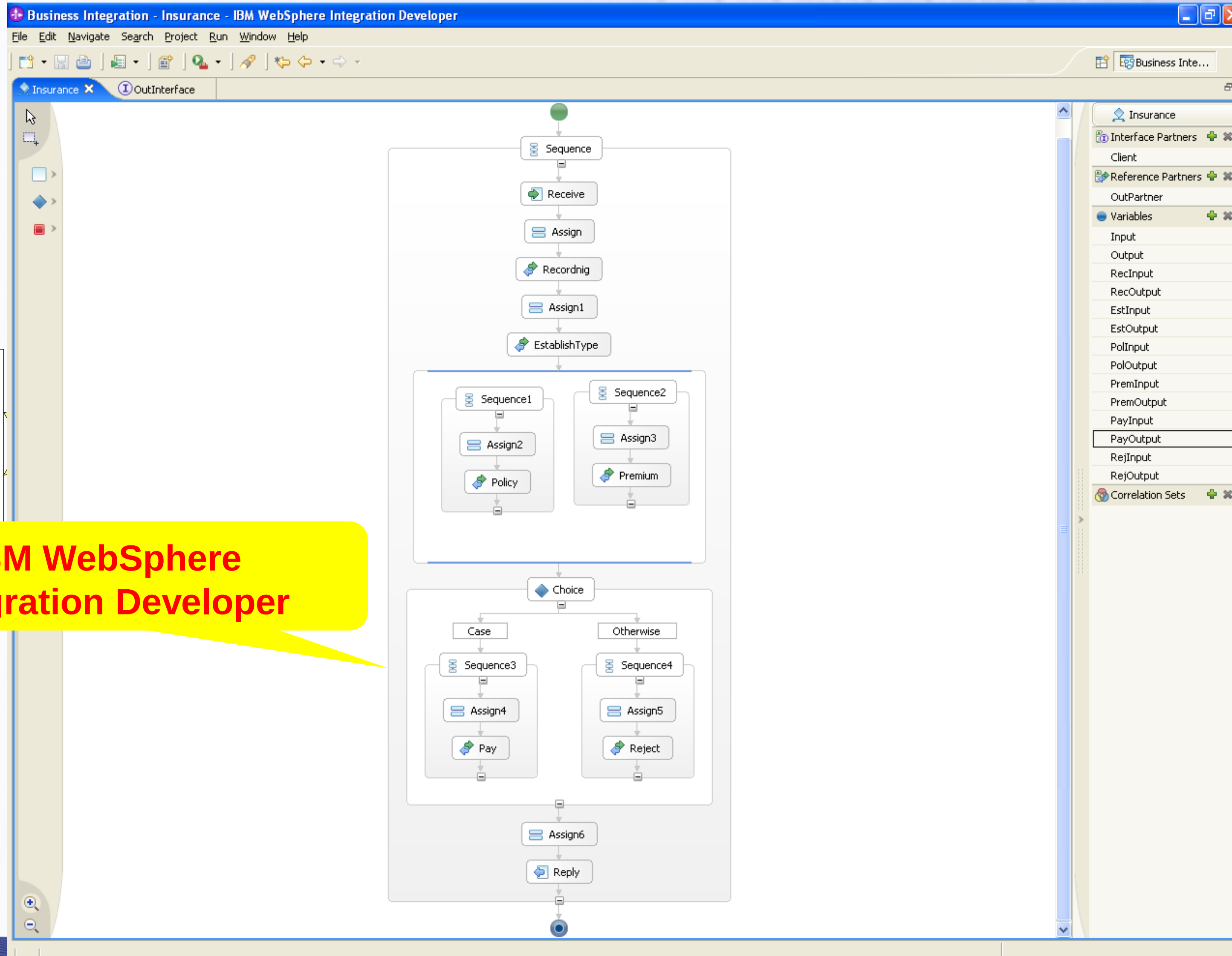
Verification of Workflows



Verification of Workflows

Workflow (BPEL)

IBM WebSphere Integration Developer

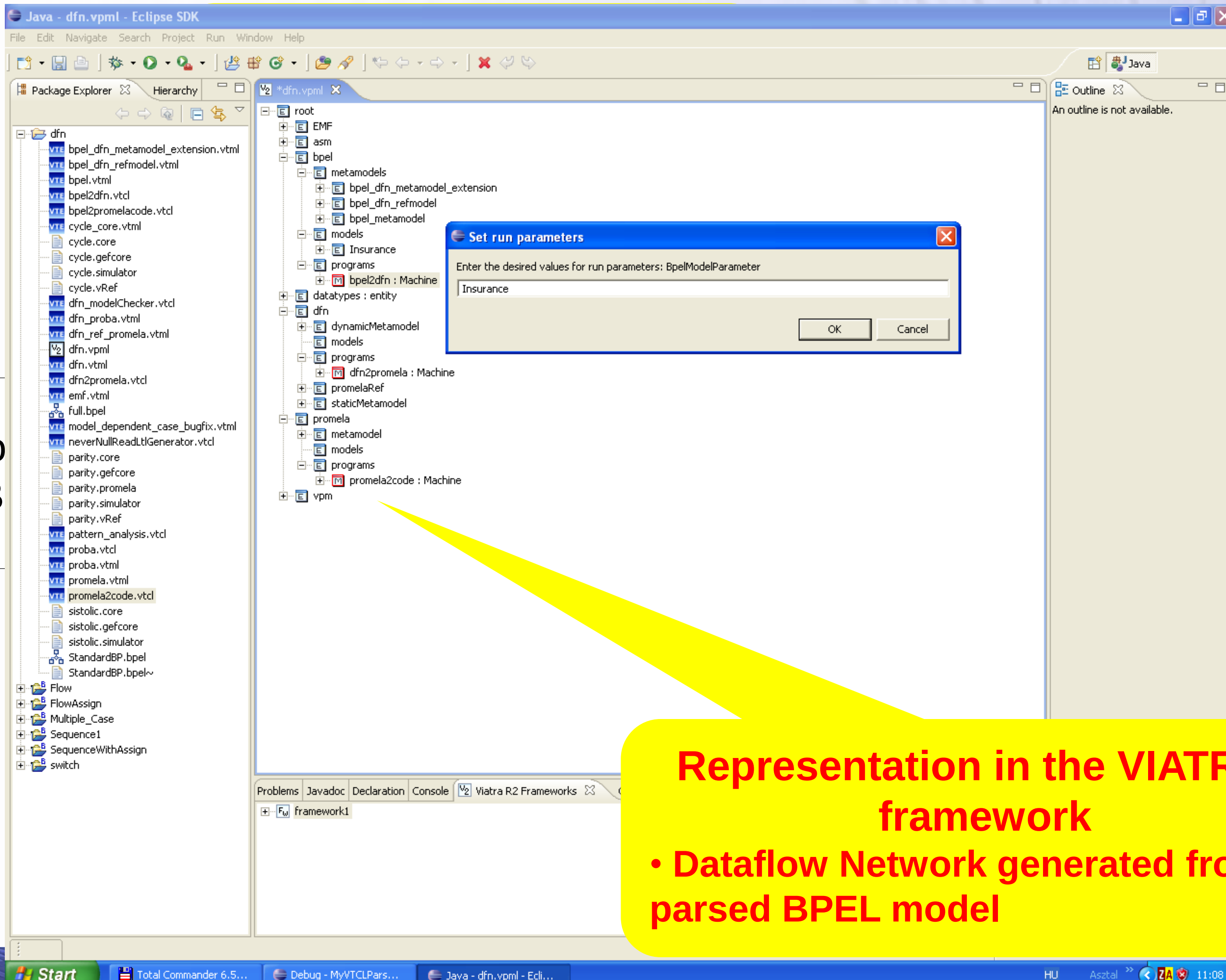


Verification of Workflows

Wo
(B

tive
ult

IN
hecker



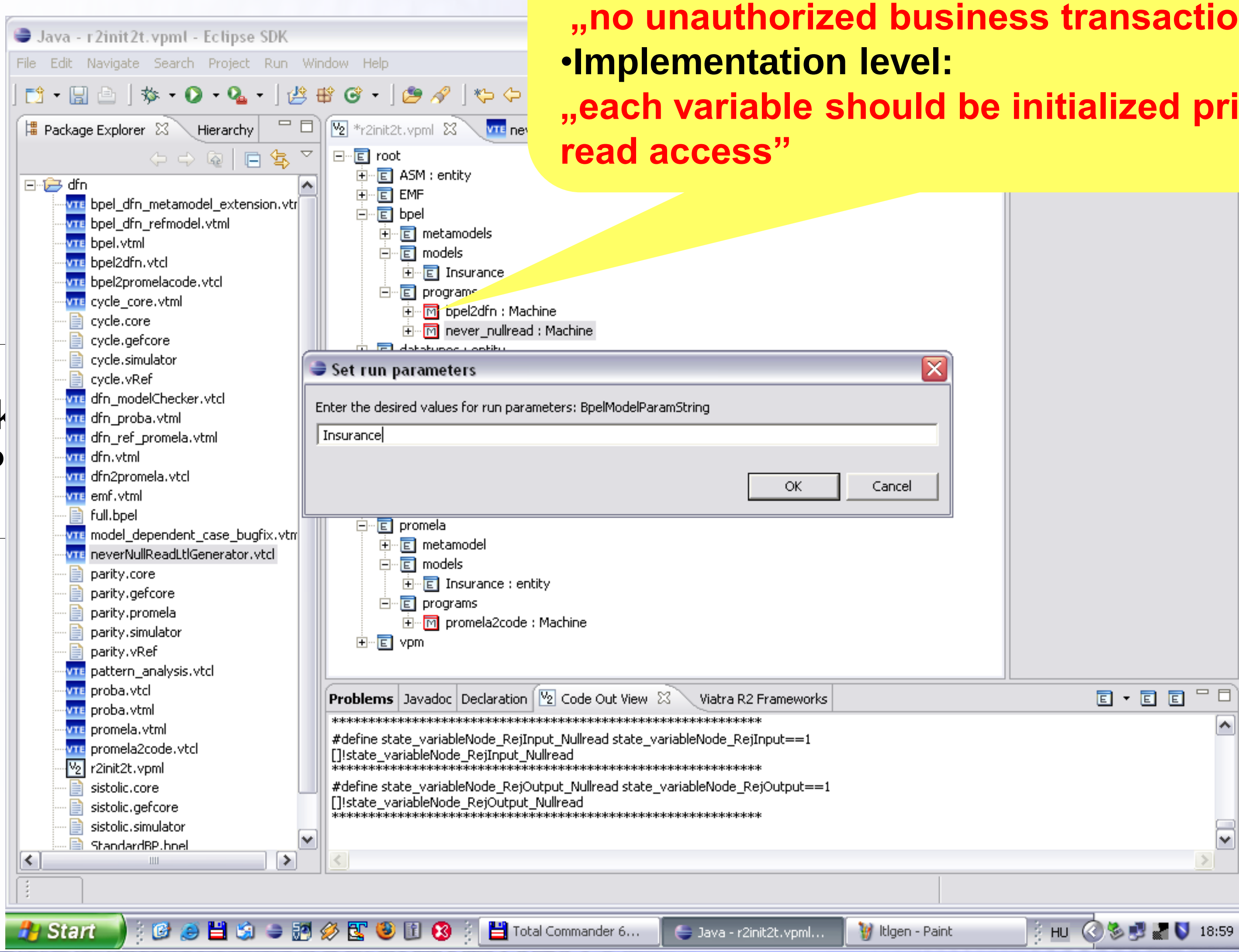
Representation in the VIATRA2 framework

- **Dataflow Network generated from parsed BPEL model**

Verification of Workflows

Target requirement

- **Business level:**
  „no unauthorized business transaction”
- **Implementation level:**
  „each variable should be initialized prior to a read access”



Work (BP

N ecker

esult

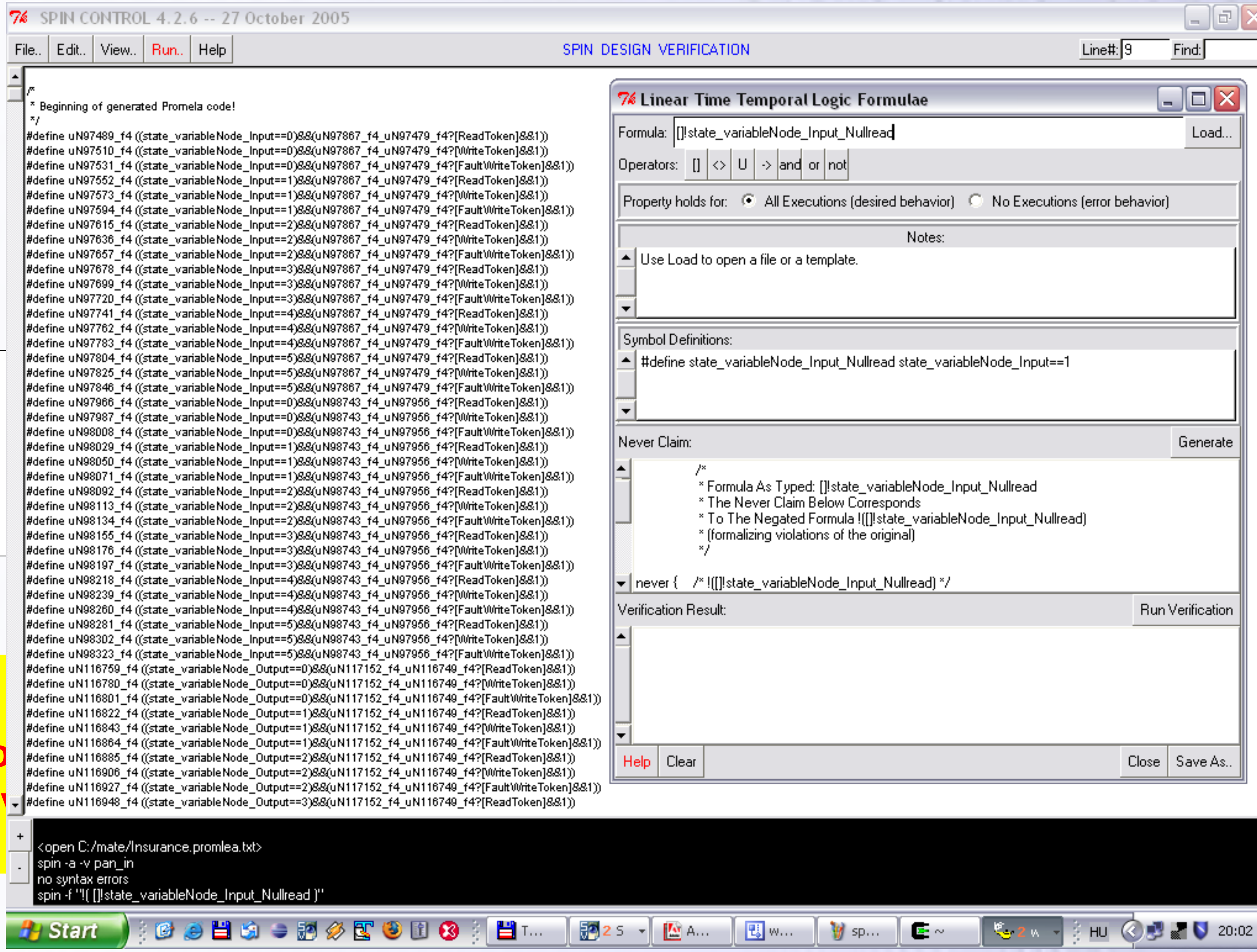
mple

Verification of Workflows

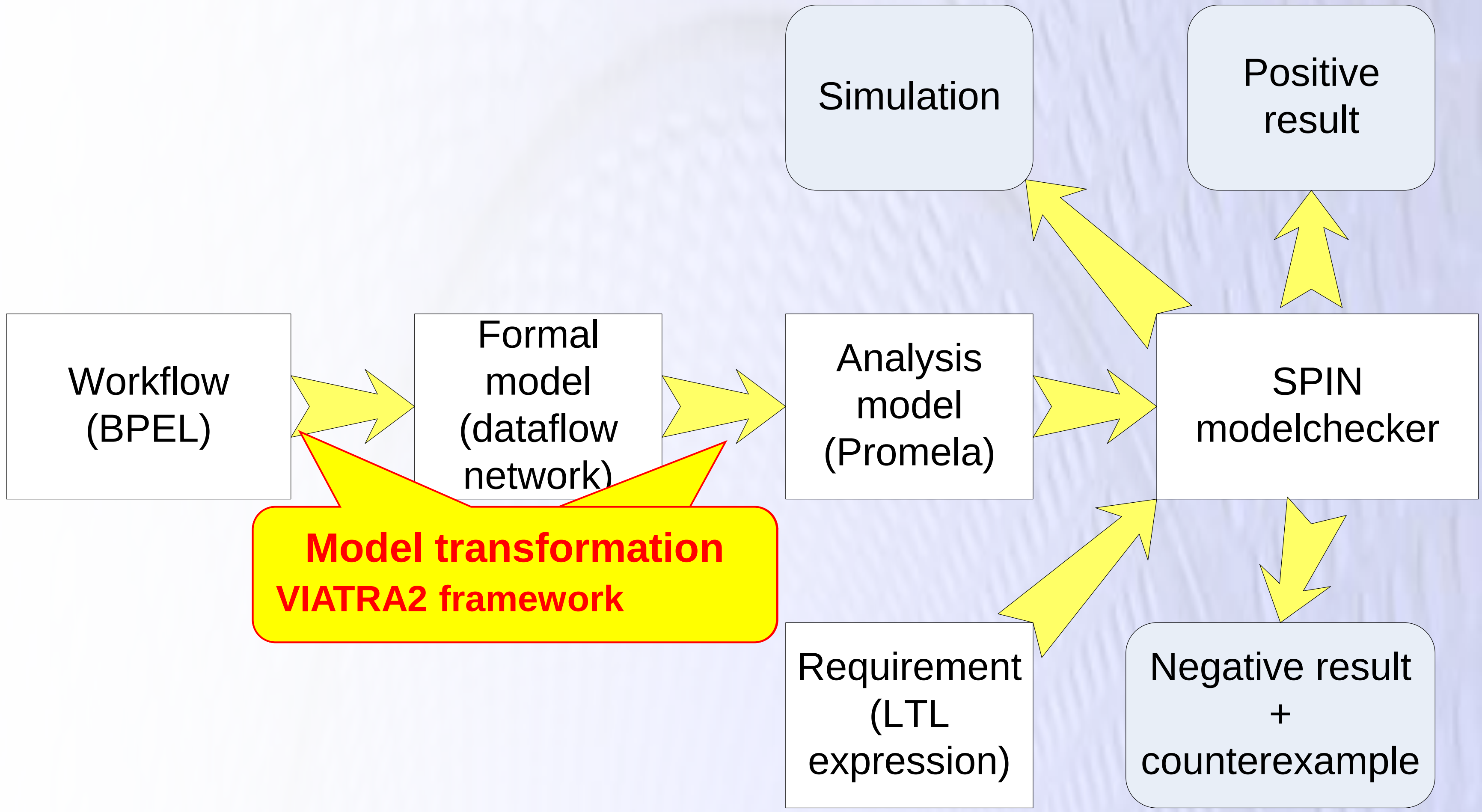
Workflow (BPEL)

er

- Evaluatio
- Exhaustiv



Verification of Workflows



Abstraction: qualitative modeling

- Formal methods have strict complexity limitations
  - Efficient, but still faithful abstractions are needed
- Qualitative abstraction:
  - A few of qualitative values out of an enumerated data type set
  - No detailed data representation
  - Drastic state space (analysis complexity) reduction
- Systematic methodology: predicate abstraction

Example

- **Full model:**
  IF credit_requested < 2.000.000 THEN approval(director) ELSE approval(board)

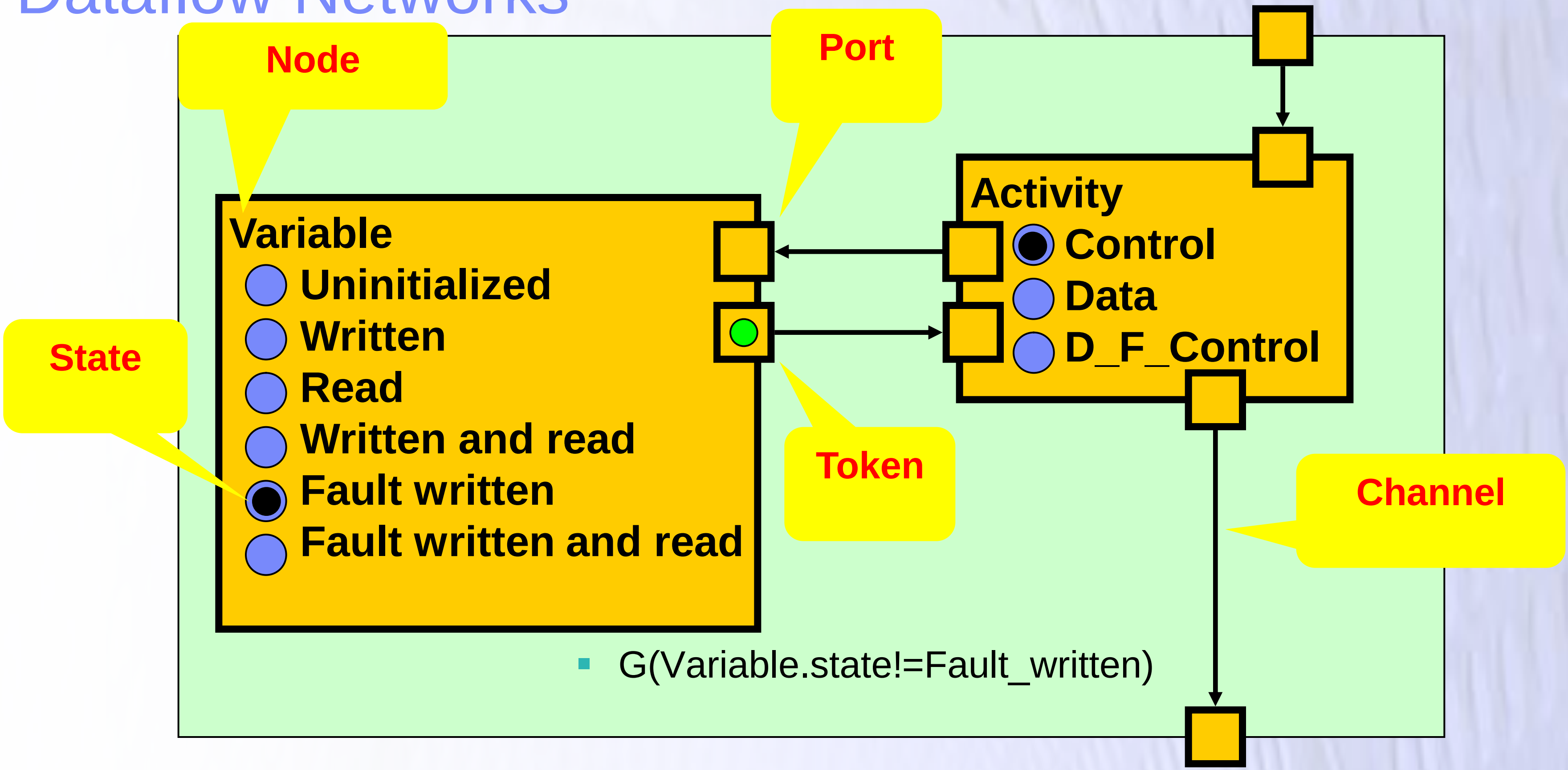
- **Deterministic abstraction:**
  IF minor_credit_requested THEN approval(director) ELSE approval(board)
  - **No representation is needed for value of credit_requested,**
  - **Only a single binary value (minor_credit_requested) representing the mode of operation**
  - **Invariant wrt. the limit of 2.000.000 changes**

- **Nondeterministic abstraction:**
  CHOOSE (approval(director), approval(board))
  - **No representation is needed for value of credit_requested,**
  - **No representation of the logic of selecting the mode of operation**
  - **Details -> random behavior**

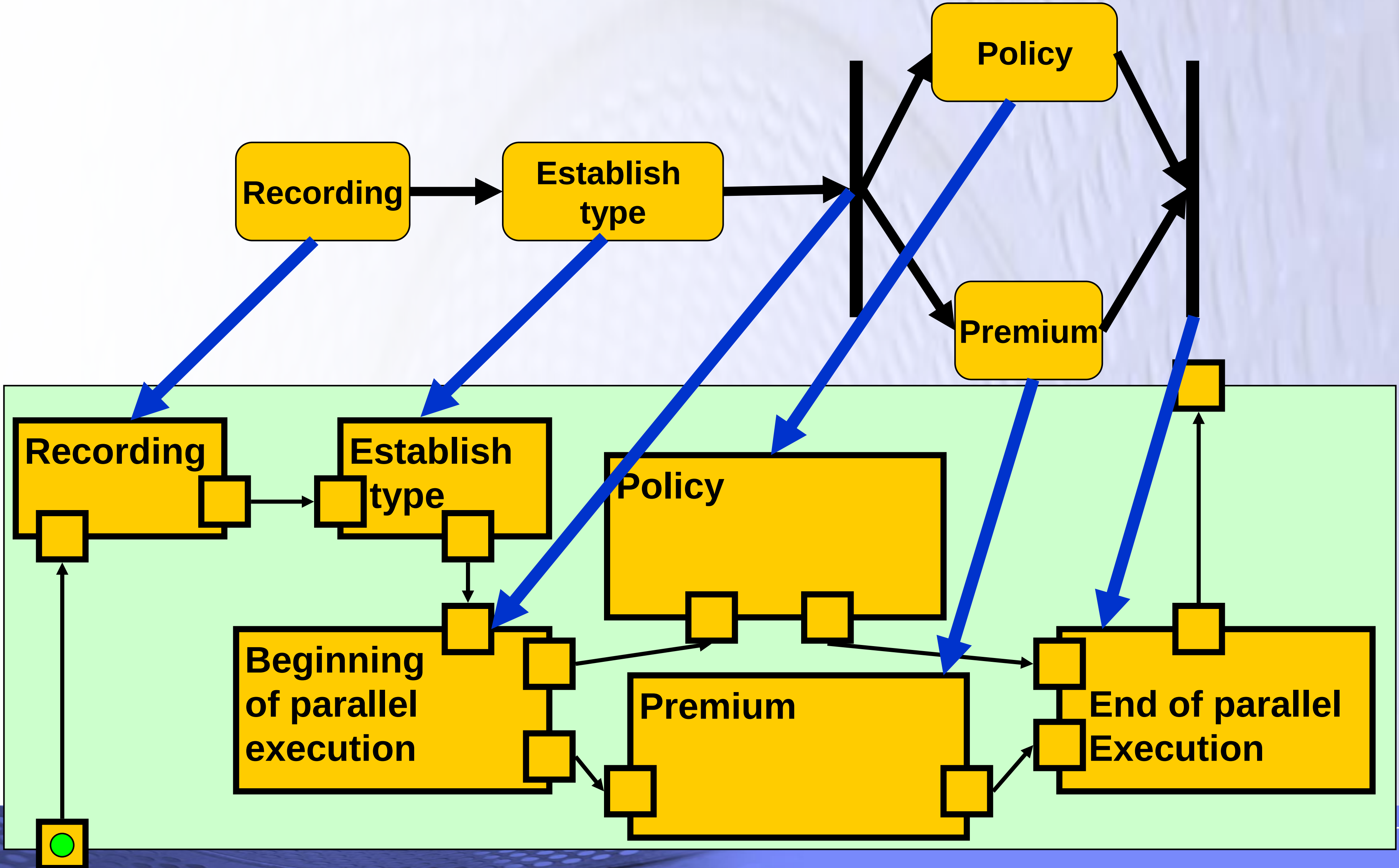
Estimation of the effects of a fault in a business workflow

- **A resource/operation is good / faulty / missing (FAULT)**
  → System behavior ?
- **Analysis principle:**
  - **Assign faults to resources / operations**
  - **Trace the flow of errors (ERROR)**
  - **Check: is a service to the user affected (FAILURE) ?**
- **Modeling and analysis:**
  - **Data items colored as good / faulty / suspicious**
  - **A component connected to another one in a potentially erroneous state is suspicious**
  - **Static worst case approximation:**
    Damage Confinement Region

Dataflow Networks



Mapping a Workflow to Dataflow Networks

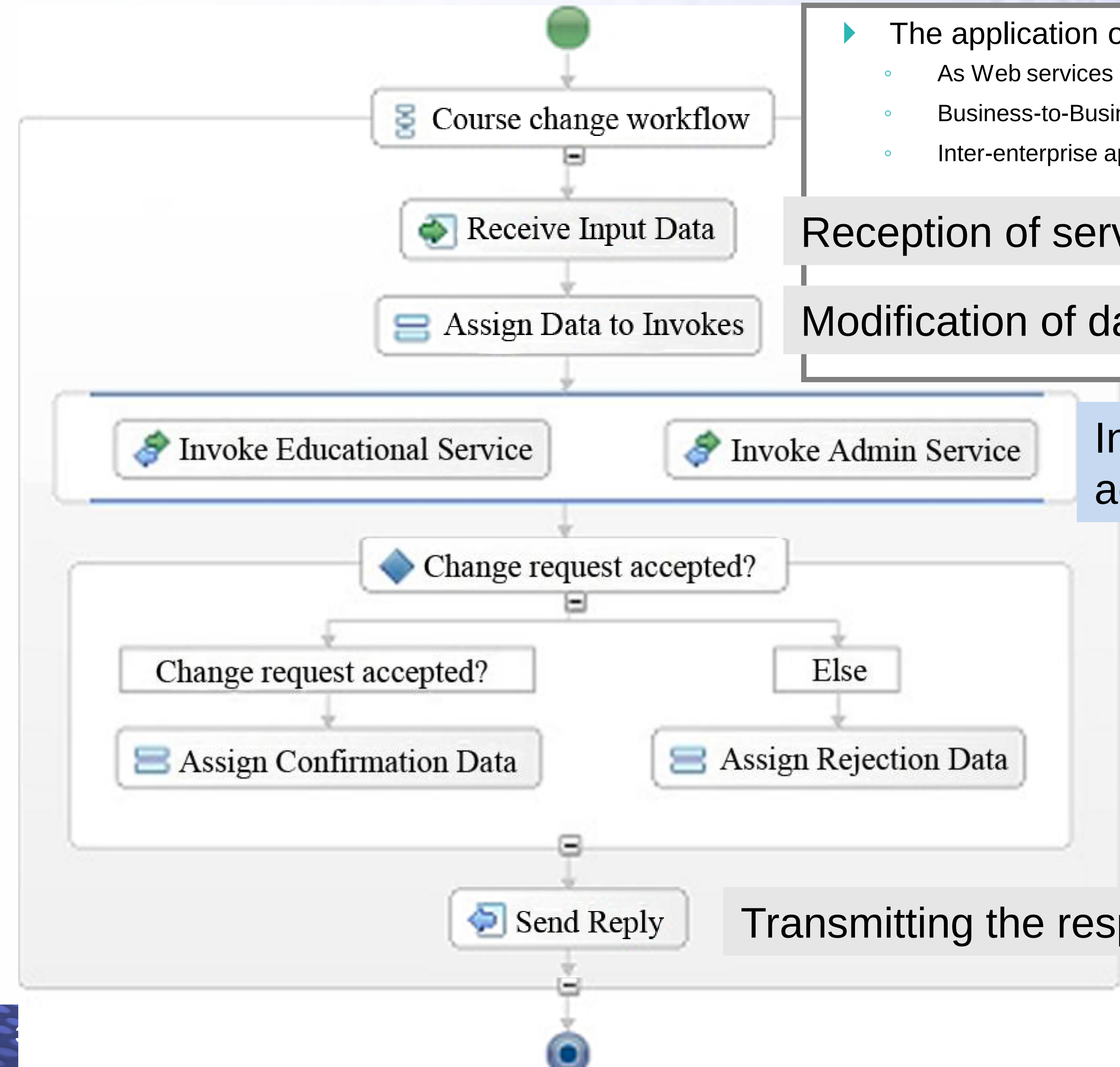


Formal Verification of Cooperating BPEL 2.0 Processes

Tibor Bende, Ábel Hegedüs, *Máté Kovács*

SENSORIA Workshop Munich, February 9-11

BPEL Processes



- The application of workflows
  - As Web services
  - Business-to-Business
  - Inter-enterprise application integration (EAI)

Reception of service request

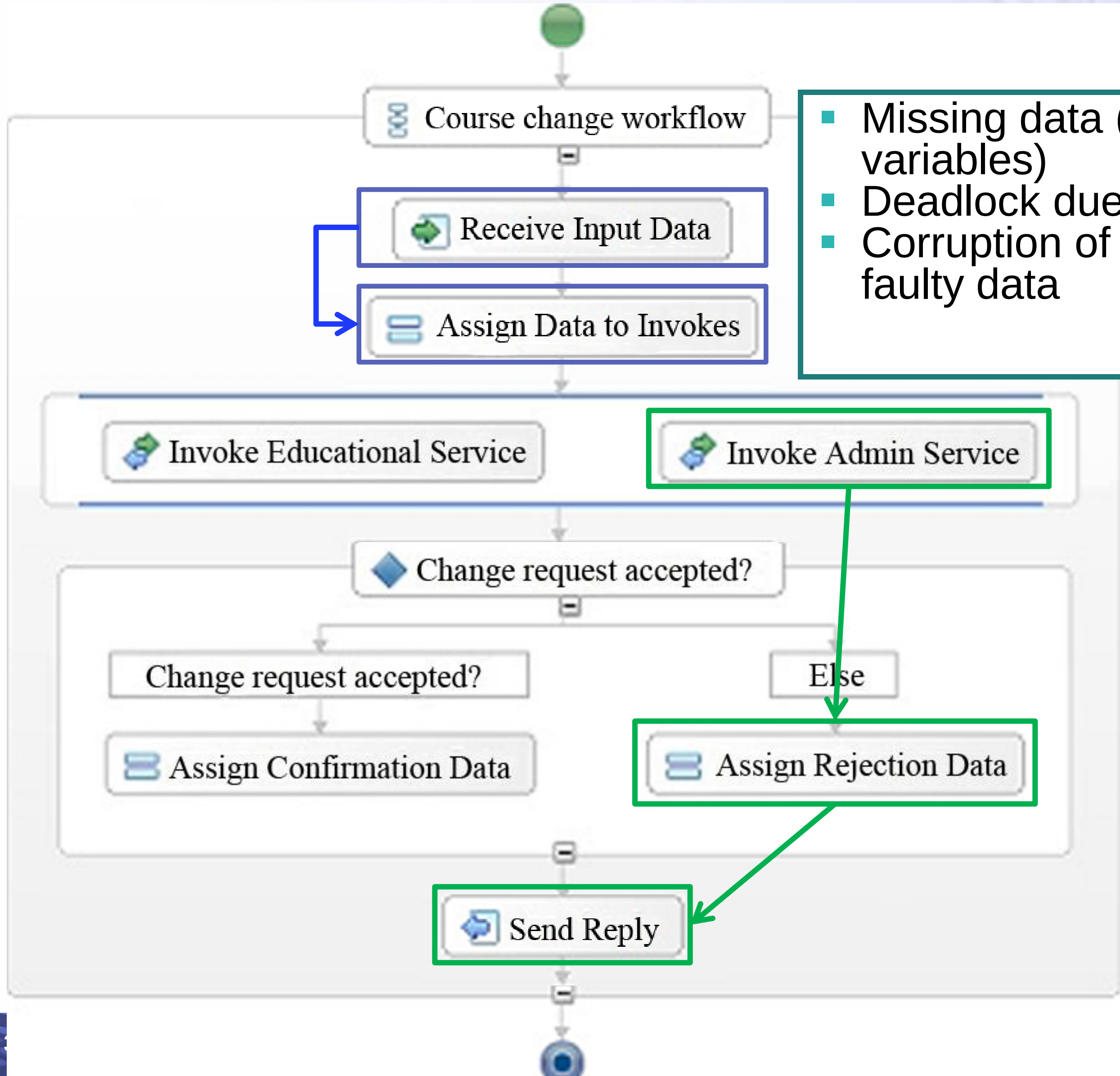
Modification of data

Invocation of educational and administrative processes

Is the request accepted?

Transmitting the response

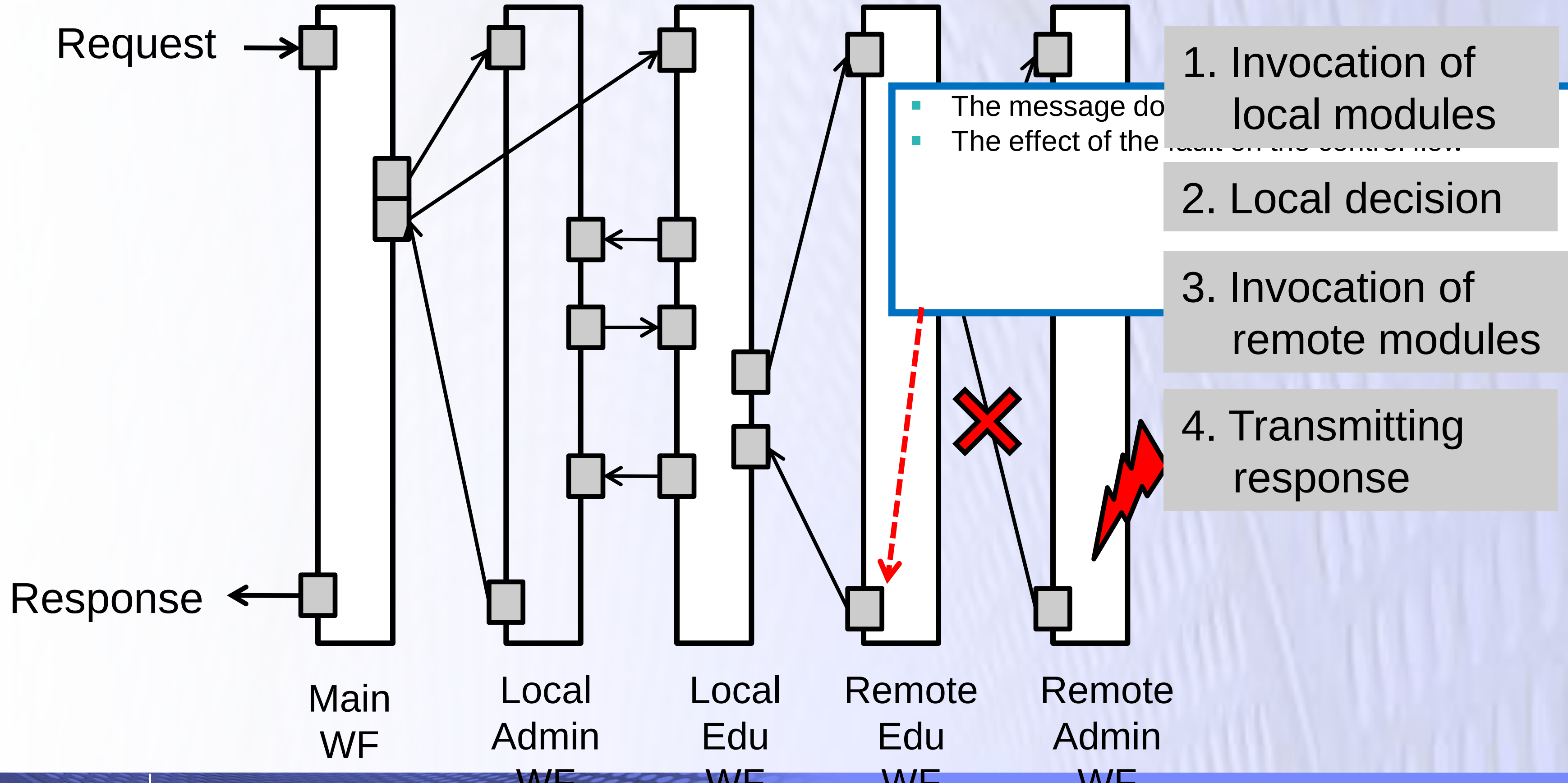
Potential Faults in BPEL Workflows



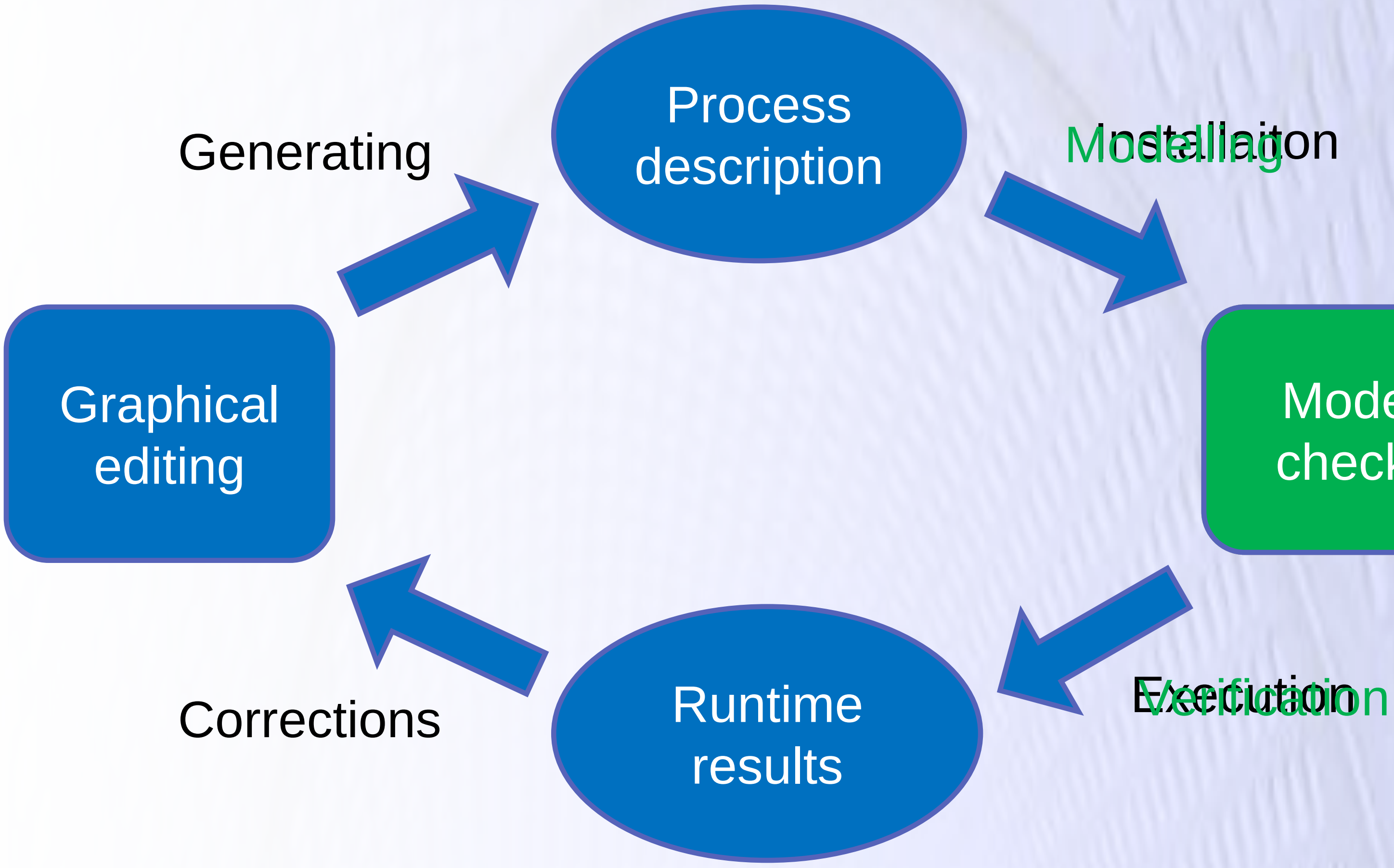
- Missing data (Reading uninited variables)
- Deadlock due to loss of data
- Corruption of the control flow due to faulty data

Cooperating Business Processes

Processing an application to participate in a course on a different university



Analysis of Workflows



Summary of the Contribution

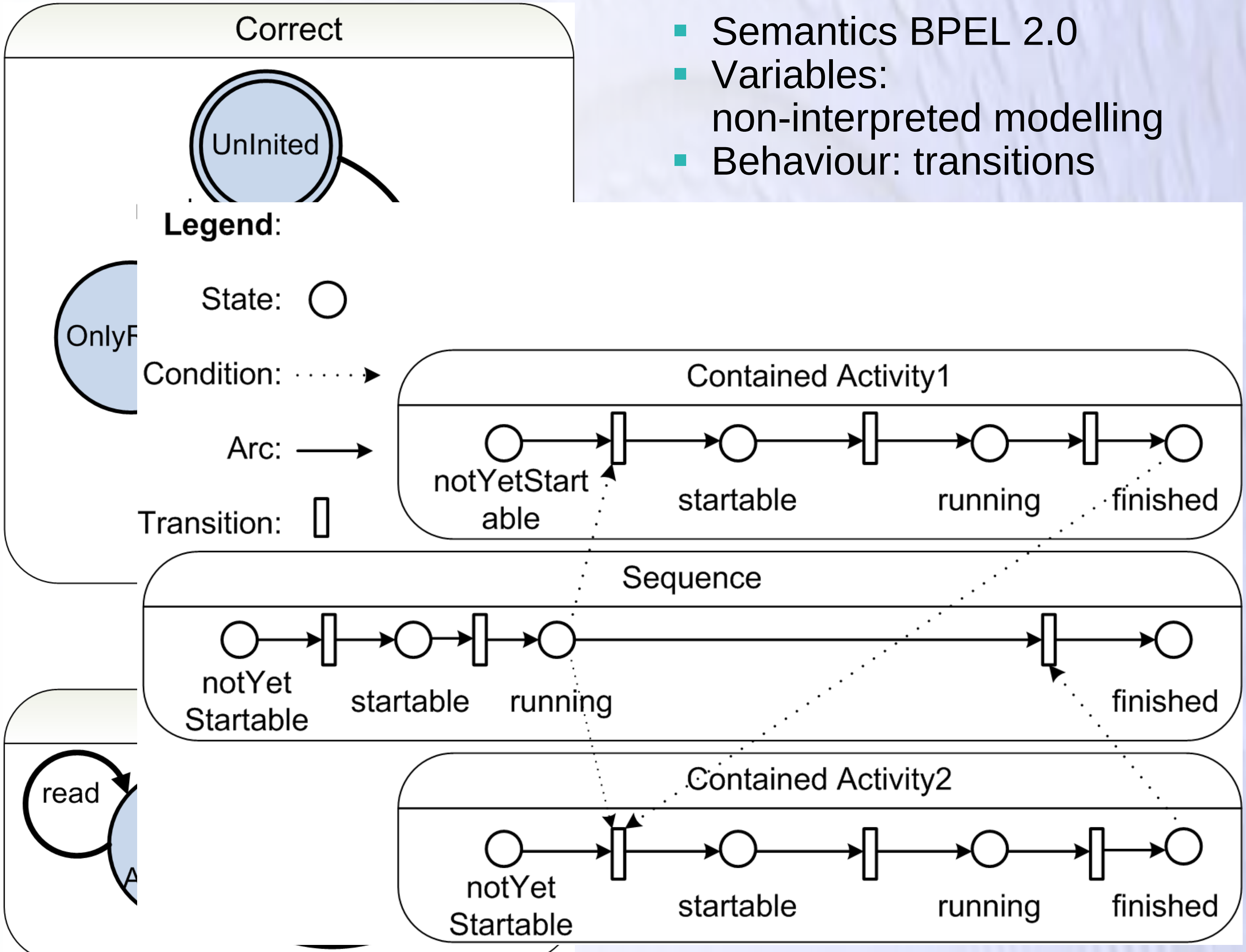
- The application of design time verification
  - There are various approaches
- Formal Verification of BPEL 1.1 processes

- Formal Verification of BPEL 2.0 processes
  - New control flow structures
  - Event and fault handling
  - Algorithm for dead path elimination

The analysis of the cooperation of such processes

Modelling a Stand Alone Workflow

- Semantics BPEL 2.0
- Variables: non-interpreted modelling
- Behaviour: transitions

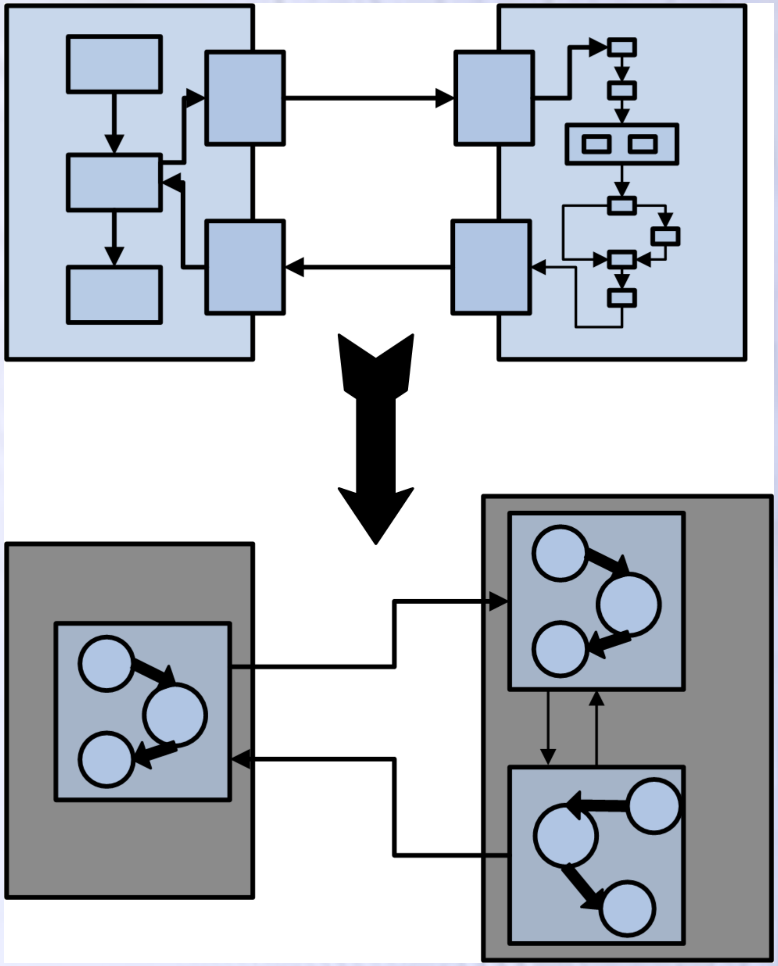


Based on the Presentation of András Pataricza @ University Relations -- Academic Days

Modelling the Cooperation of Workflows

Abstraction

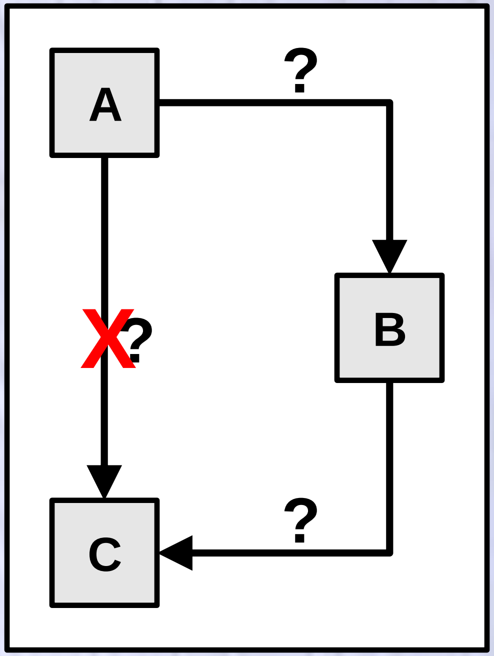
- \+ Based on communicating activities
- \+ Smaller statespace
- − Overestimation of the behaviour

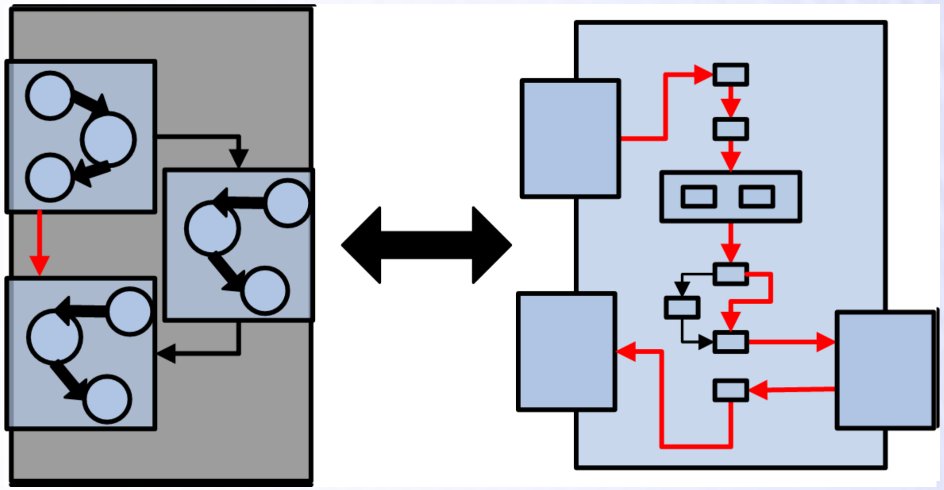


 Based on the Presentation of András Pataricza @ University Relations -- Academic Days

Steps of Model Refinement

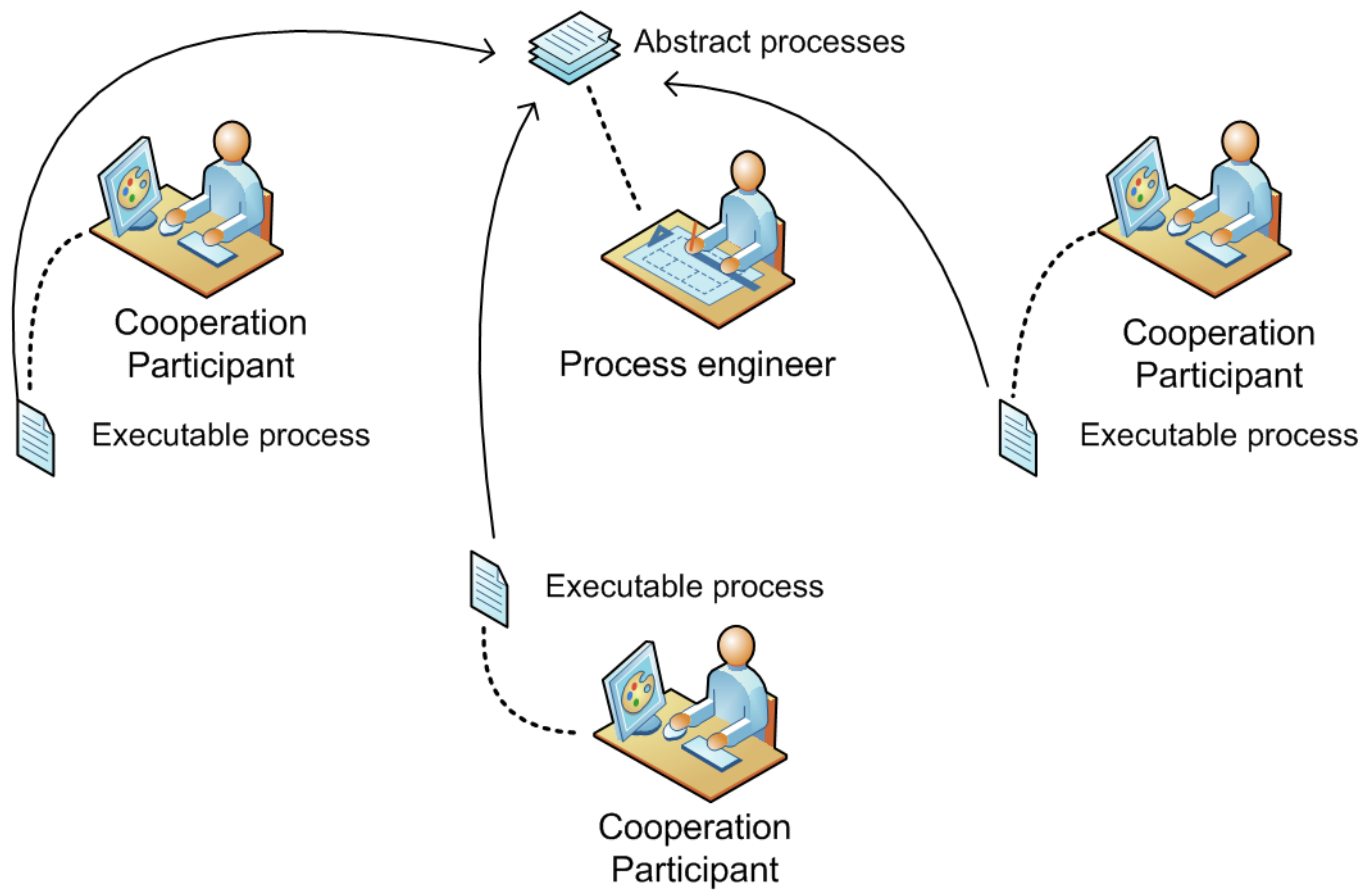
1. Abstract model
2. Verification of the requirement
3. Refinement
4. Refined model





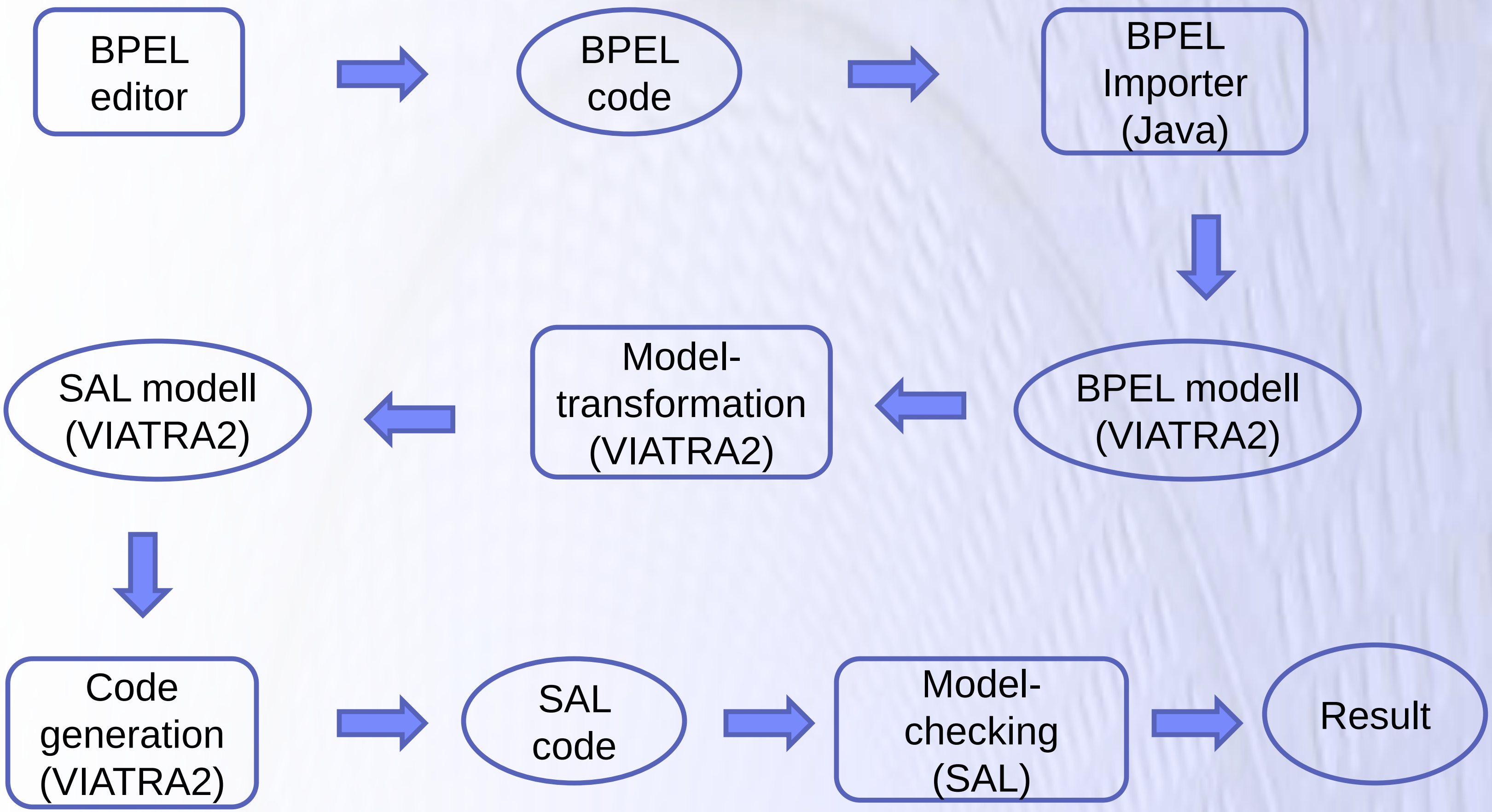
 Based on the Presentation of András Pataricza @ University Relations -- Academic Days

Roles

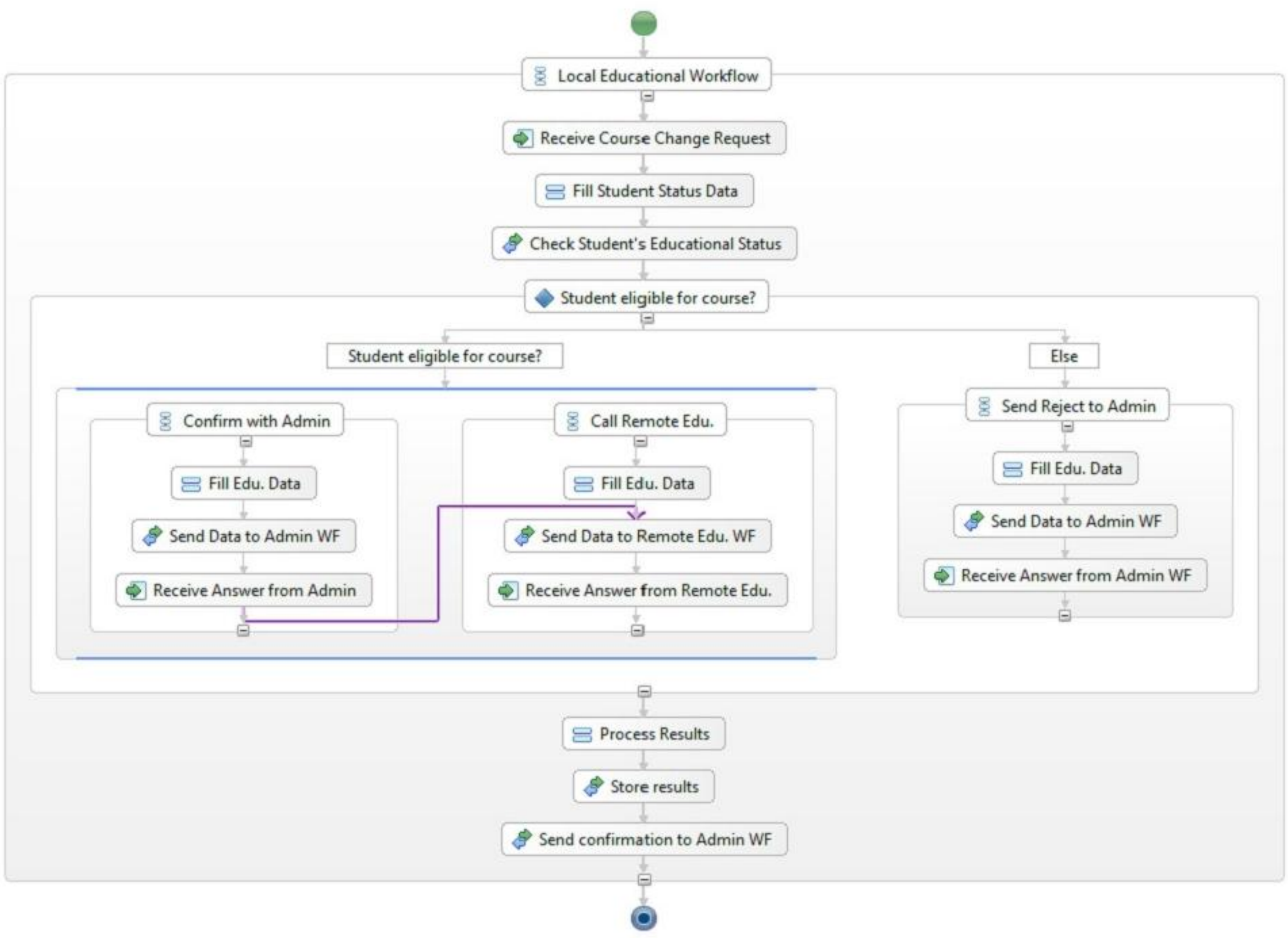


Abstract processes
Cooperation Participant
Process engineer
Cooperation Participant
Executable process
Executable process
Executable process
Cooperation Participant

Steps of the Verification Process



Case Study



Local Educational Workflow
Receive Course Change Request
Fill Student Status Data
Check Student's Educational Status
Student eligible for course?
Student eligible for course?
Else
Confirm with Admin
Fill Edu. Data
Send Data to Admin WF
Receive Answer from Admin
Call Remote Edu.
Fill Edu. Data
Send Data to Remote Edu. WF
Receive Answer from Remote Edu.
Send Reject to Admin
Fill Edu. Data
Send Data to Admin WF
Receive Answer from Admin WF
Process Results
Store results
Send confirmation to Admin WF

Analysis of the Case Study

Requirement	# of states	Verification time [s]	Execution time [s]	Result
Not reading uninitialized variable?	532336	0,5	4,6	True

Requirement	# of states	Verification time [s]	Execution time [s]	Result
Not reading uninitialized variable?	532336	0,5	4,6	True
A variable is not needed?	98784	0,3	5,8	False

Conclusion

- Verification of business processes and their cooperation in design time
- Advantages:
  - Finding usual practical mistakes and unhandled exceptions
  - Compatible with SOA
  - Further research directions
    - Graphical requirement specification
    - Domain specific fault model
    - Automatic model refinement

Early dependability assessment

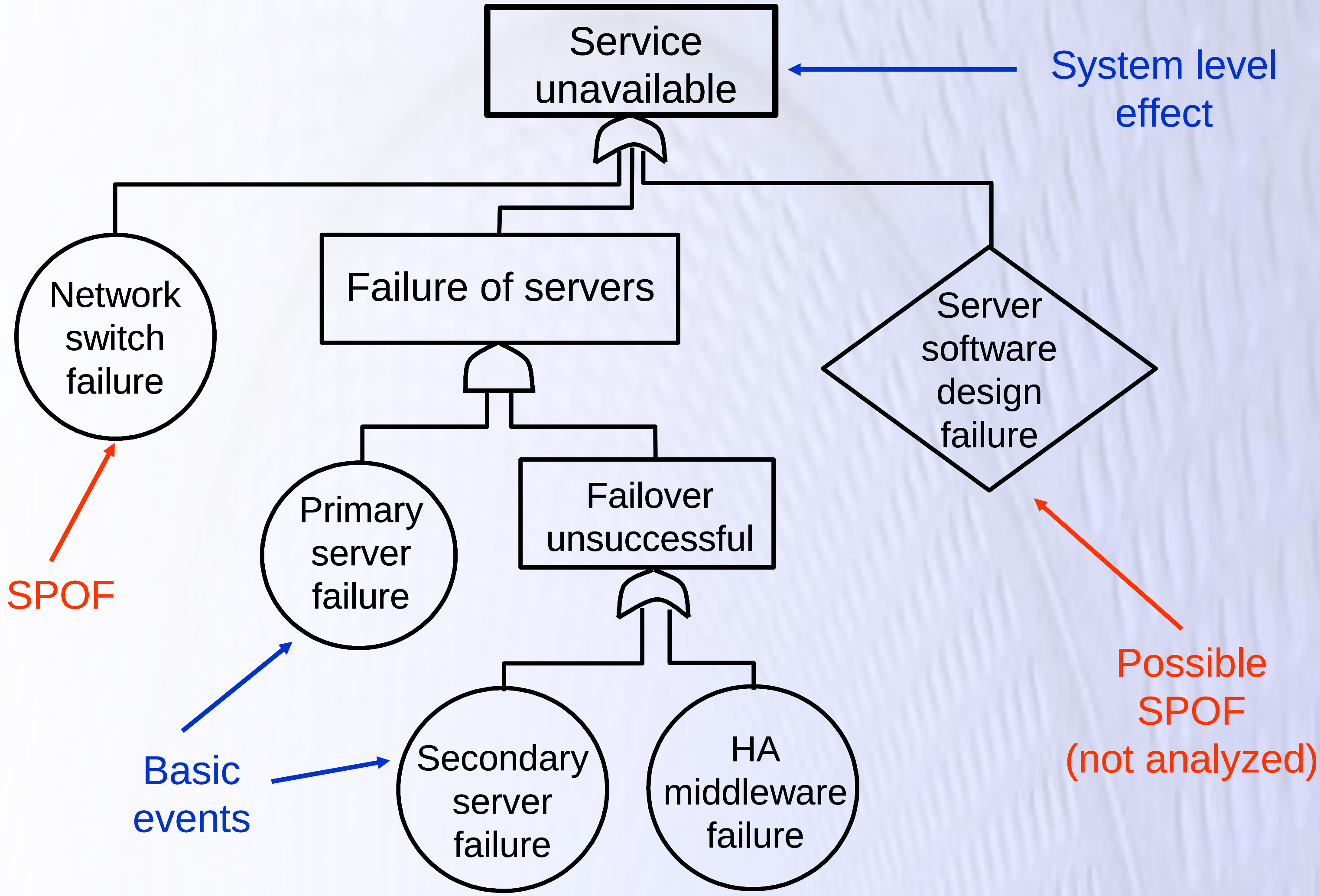
Objective: identification of the critical processes and their dependence on services and resources
Reinforcement of the workflow (ABFT)

Systematic analysis techniques (overview)

1. **Fault tree analysis**
2. Event tree analysis
3. Cause-consequence analysis
4. **Failure mode and effects analysis (FMEA)**

→ **Risk matrix with acceptable hazards**

Fault tree analysis



Quantitative analysis

- Probabilities assigned to basic events
  - Component data book, estimation, measurements
- Probability of system level failure is computed
  - AND gate: product of probabilities
  - OR gate: sum of probabilities

  Independent basic events are assumed here.
- Problems:
  - Correlated basic events
  - Handling the scenario of basic events
    (fault tree is a static snapshot)

Failure mode and effects analysis

- Failures and system level effects are listed
- Advantages:
  - Systematic analysis of component failures
  - Efficiency of redundancy can be estimated

Component	Failure mode	Probability	Effect
Network switch	- garbage out - broken link	35% 65%	- access denied - service timeout
...	...	...	...

Based on the Presentation of András Pataricza @ University Relations -- Academic Days

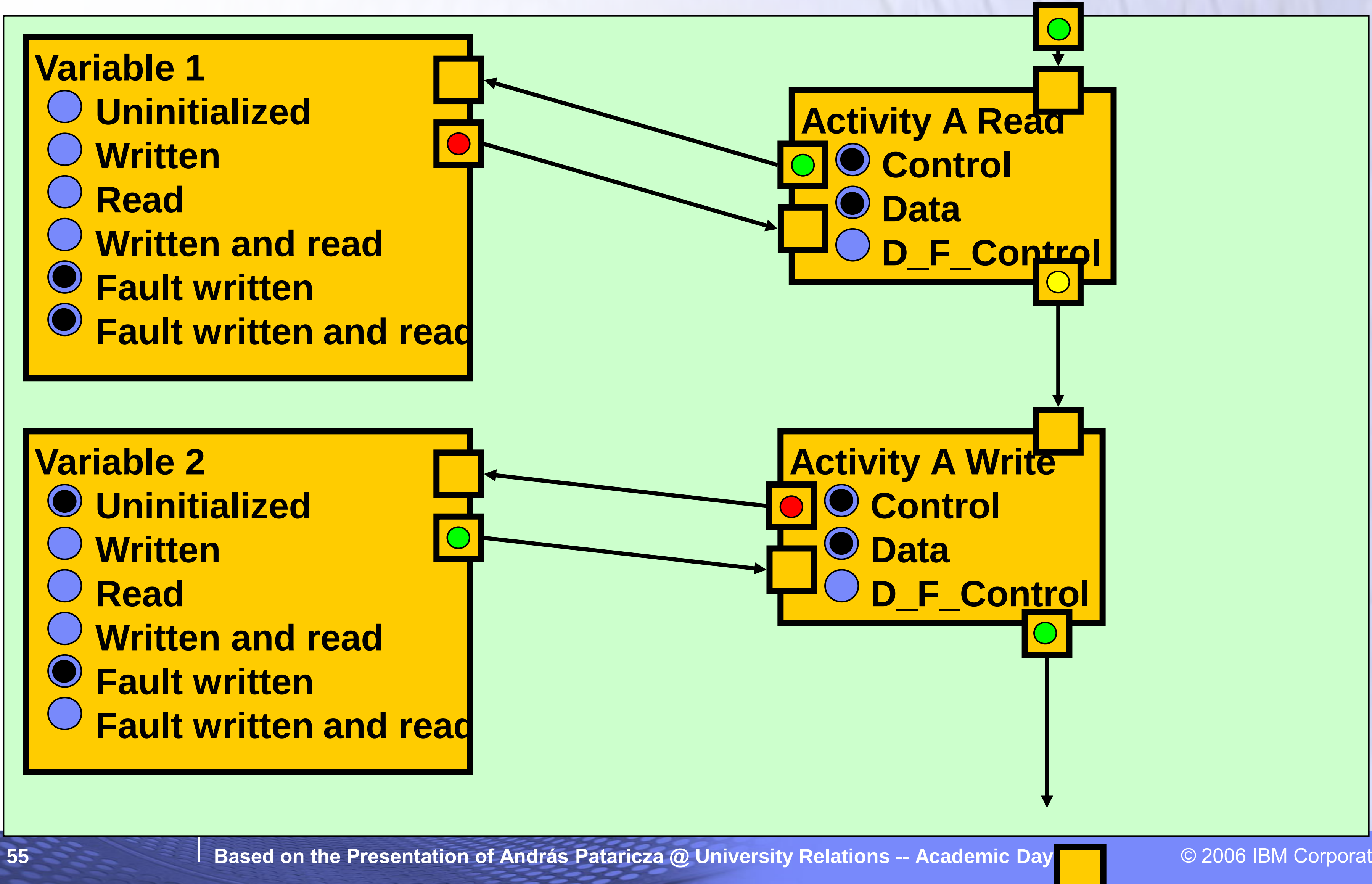
Risk matrix and acceptable hazards

Hazard effect	Catastrophic	Critical	Marginal	Negligible
Frequent				
Probable	Network switch failure		Primary server failure	
Occasional		Server sw. failure		Secondary server failure
Remote	HA middle-ware failure			
Improbable				
Impossible				

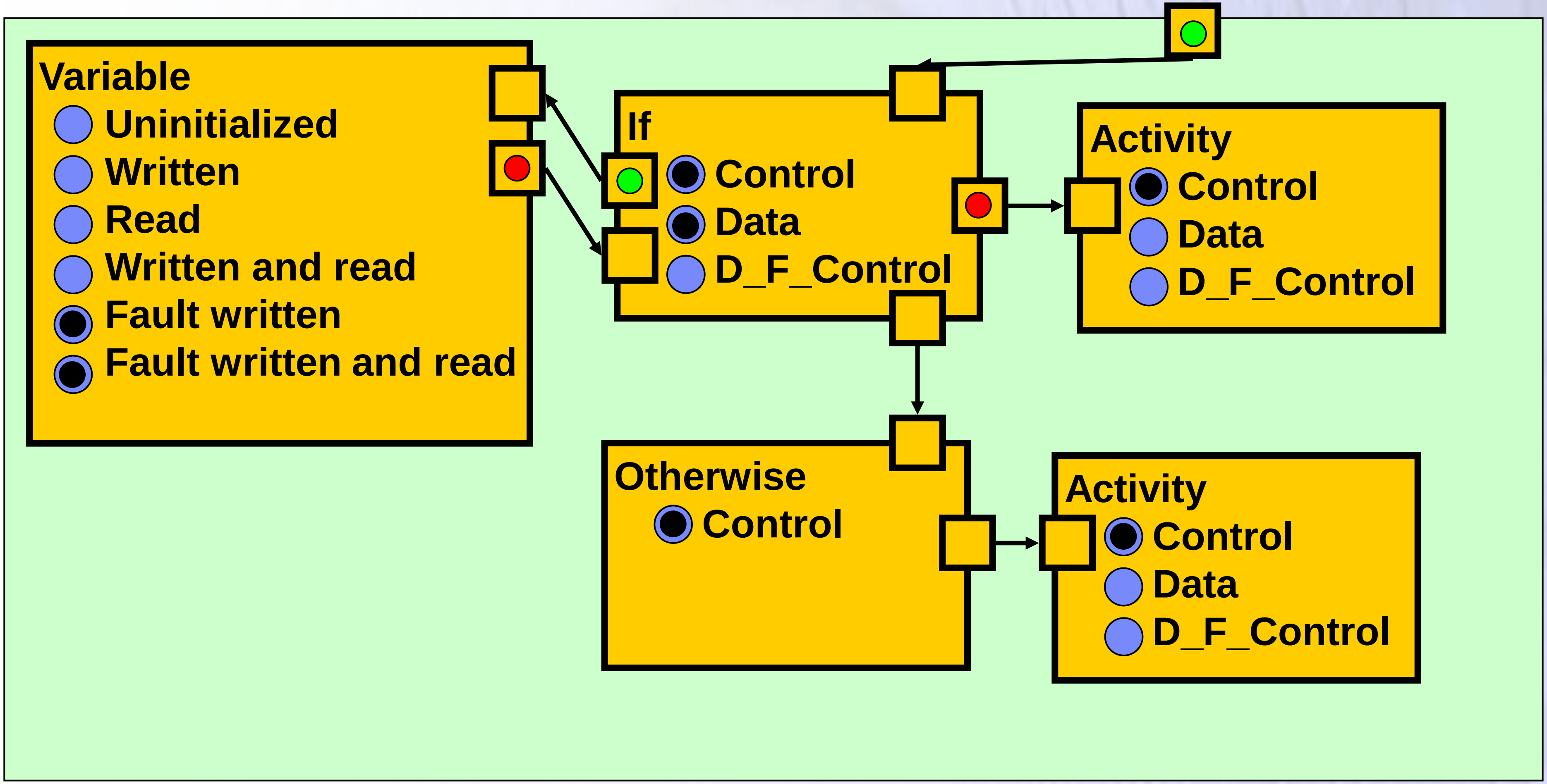
Protection level

Based on the Presentation of András Pataricza @ University Relations -- Academic Days

Dynamics: Qualitative Data Fault Simulation / Model Checking

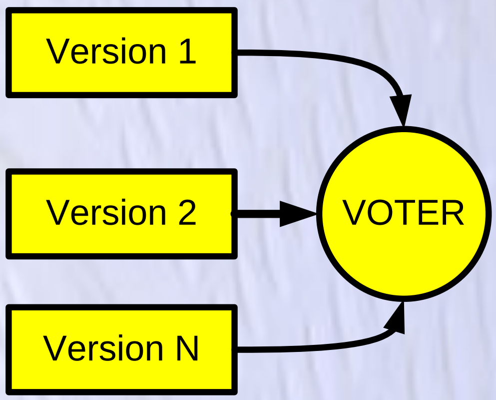


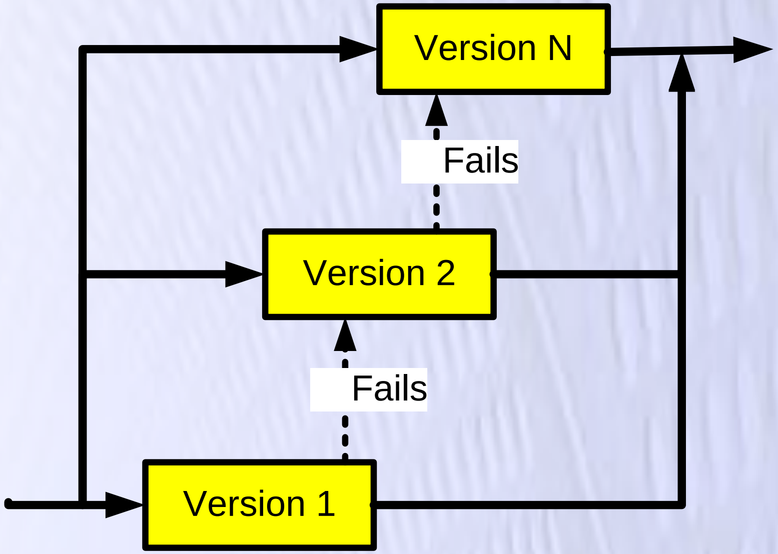
Simulation of the Error Propagation Dynamics



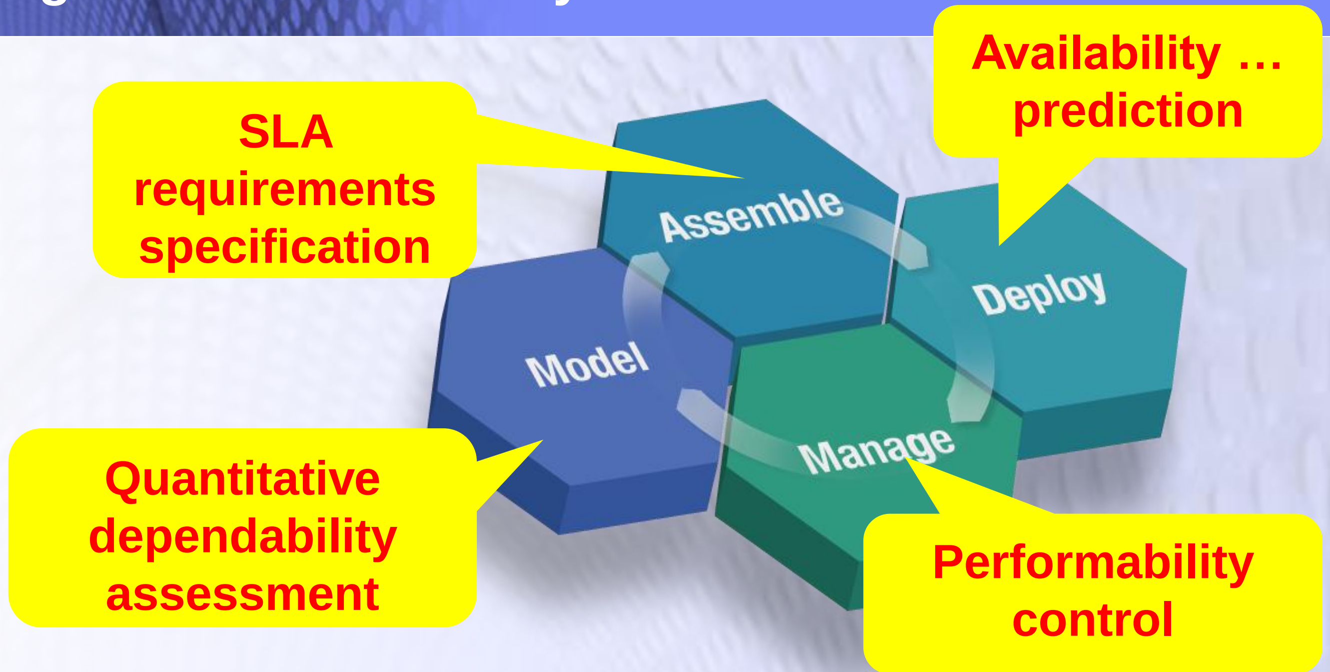
Dependable design patterns

- Critical elements of the model can be replaced
- N-Version programming
  - Here: N-Version Invocation
  - Simultaneous invocation of multiple service implementations (*variants*)
- Recovery Block
  - Here: Sequential invocation of variants
    - Until the result is acceptable
  - Adjudicator?





Quantitative Dependability Analysis



Prediction of quantitative service/business level characteristics

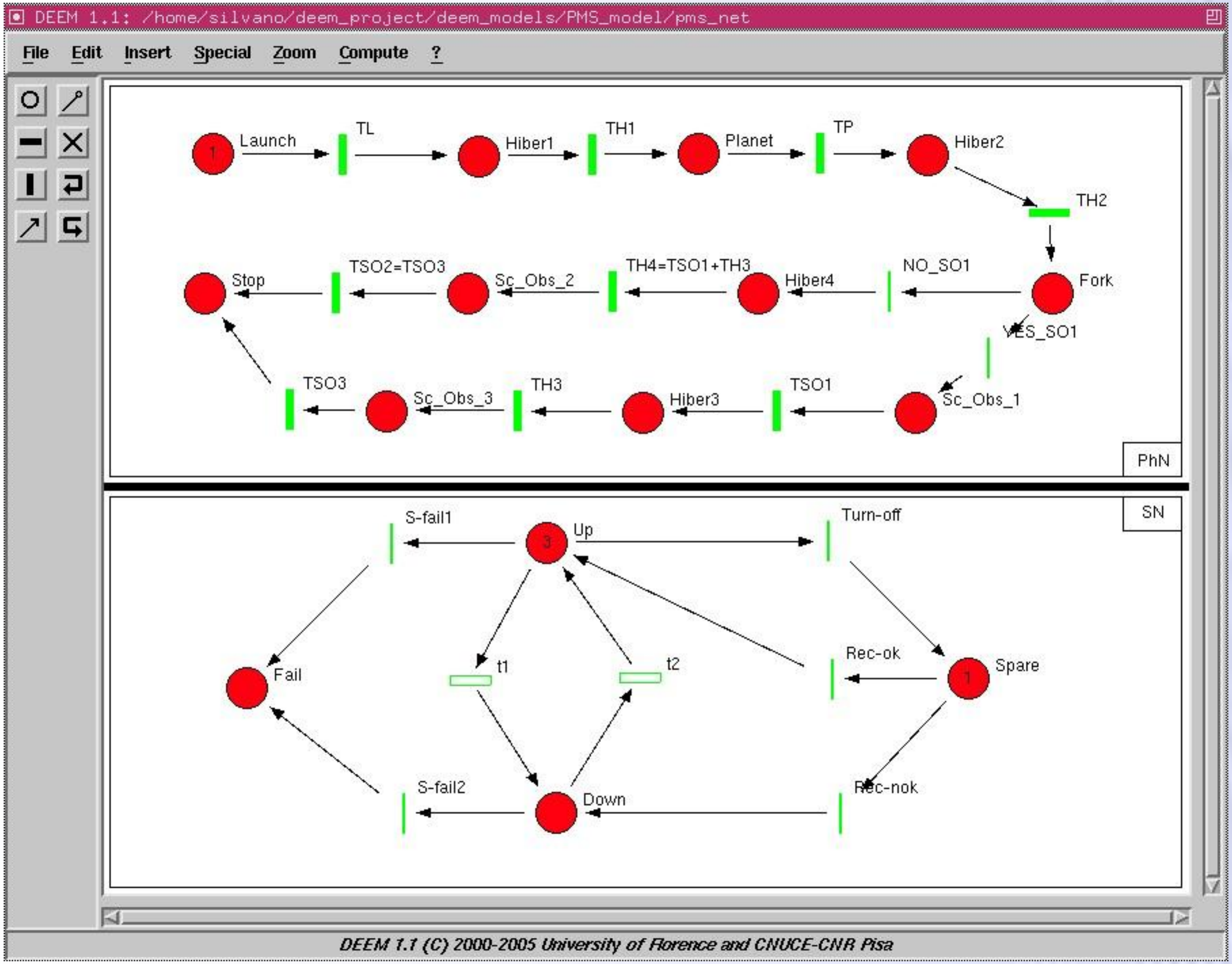
Objectives

Composite services
- Composed of basic service components
- Only partial control over the different services

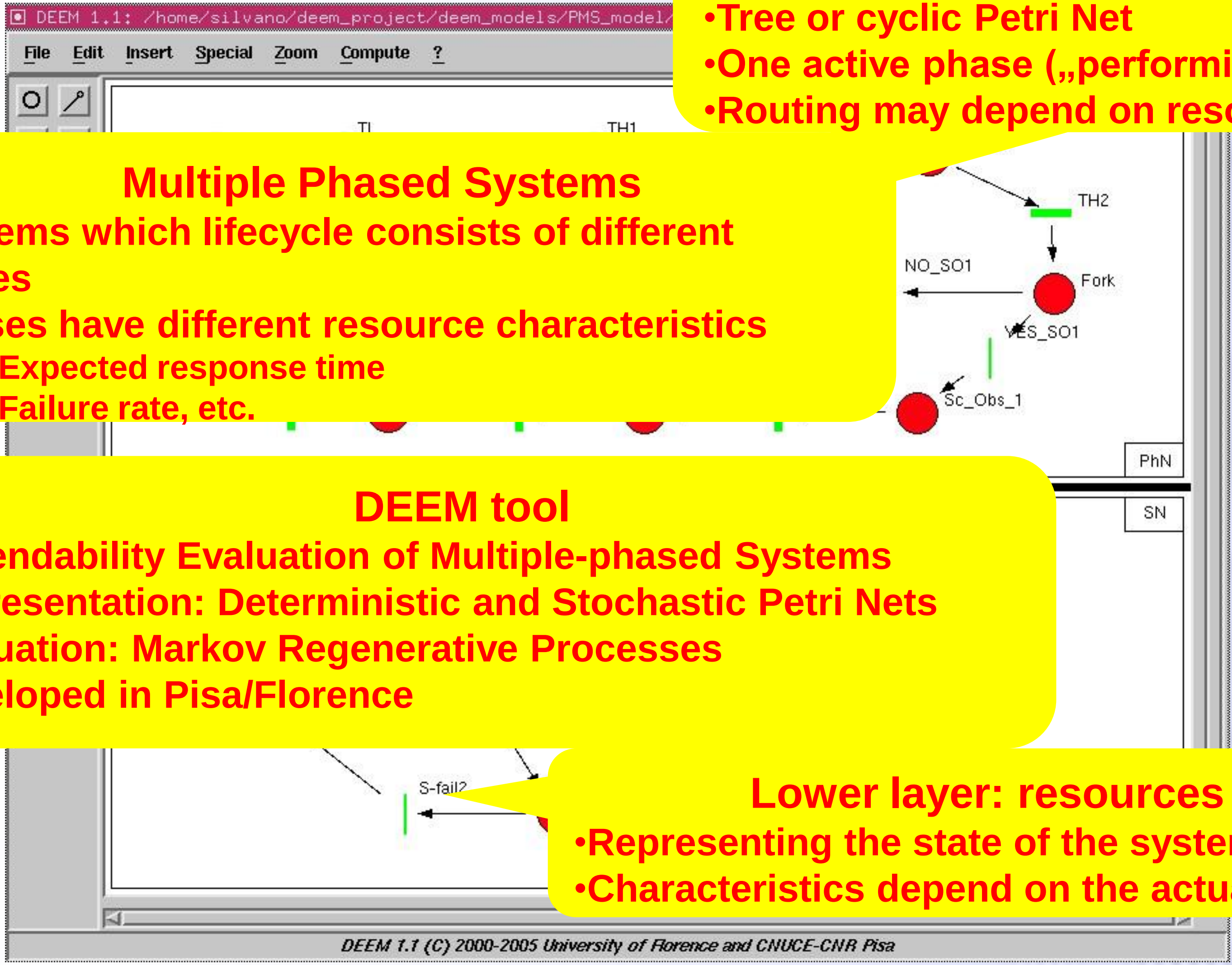
Analysis of composite services
- According to SLA parameters of services
  (e.g. throughput, reliability, availability)
- User perceived service:
  potentially different service levels for different users
- Required parameters for the invoked services
- Guaranteed parameters for the main service

Non-functional analysis
- PREDICTION of
  - Dependability metrics for the services
  - Business impacts
- WHAT-IF analysis

Phased Mission Systems



Phased Mission Systems



Upper layer: phases

- Operational life
- Tree or cyclic Petri Net
- One active phase („performing action X")
- Routing may depend on resource states

Multiple Phased Systems

- Systems which lifecycle consists of different phases
- Phases have different resource characteristics
  - Expected response time
  - Failure rate, etc.

DEEM tool

- Dependability Evaluation of Multiple-phased Systems
- Representation: Deterministic and Stochastic Petri Nets
- Evaluation: Markov Regenerative Processes
- Developed in Pisa/Florence

Lower layer: resources

- Representing the state of the system
- Characteristics depend on the actual phase

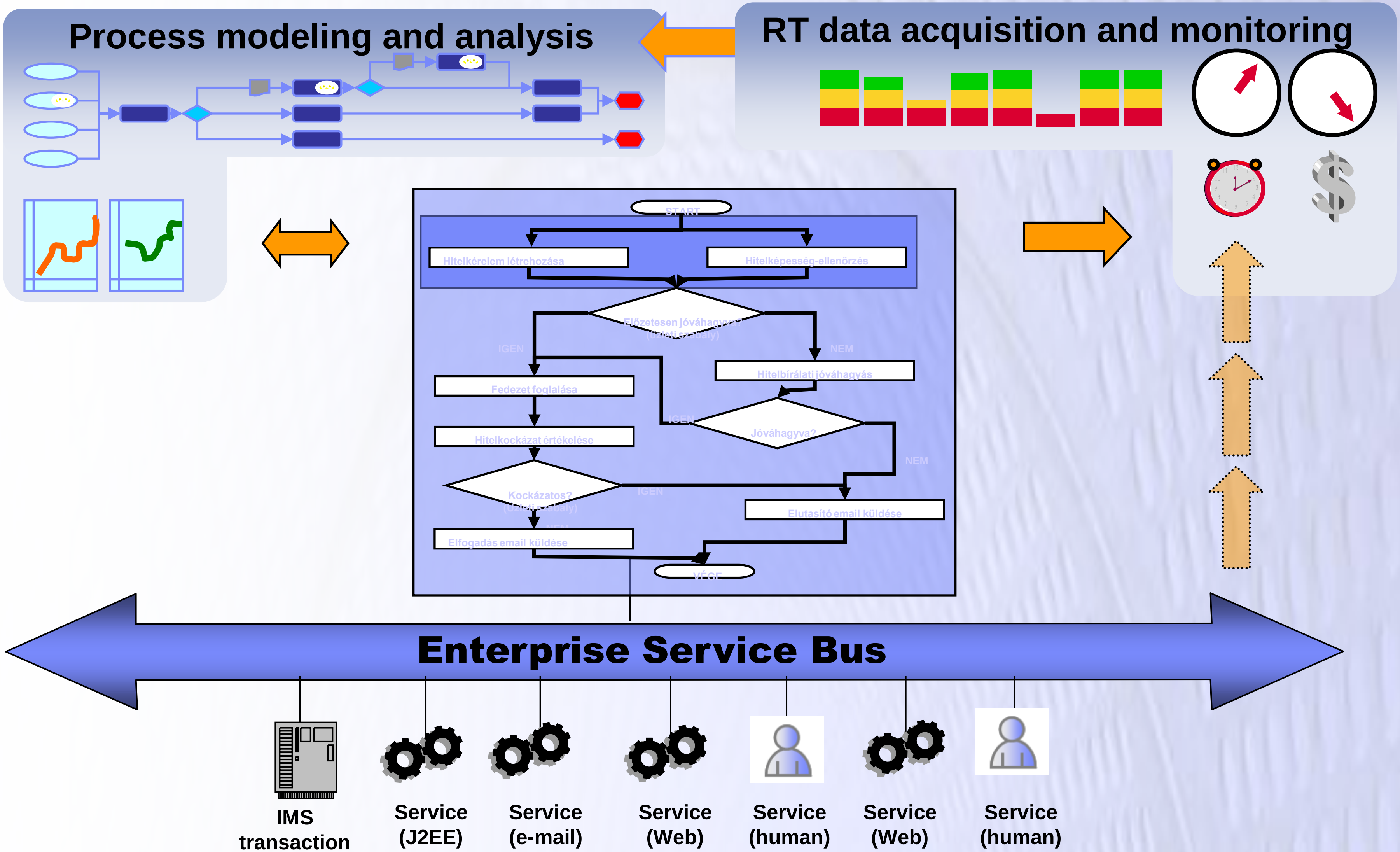
Example: Phased Mission Systems

- **Stochastic modeling**
- **Phased operational life**
- **System changes during the phases**
  - **E.g. resource states**
- **System characteristics depend on the actual phase**
  - **E.g. phase-dependent failure rates**
- **Mission goal depends on system state**
  - **Degradation**
- **Dependability modeling and analysis**
  - **Described as GSPN**
  - **Originally for mission-critical systems**

SOA service flows as PMS

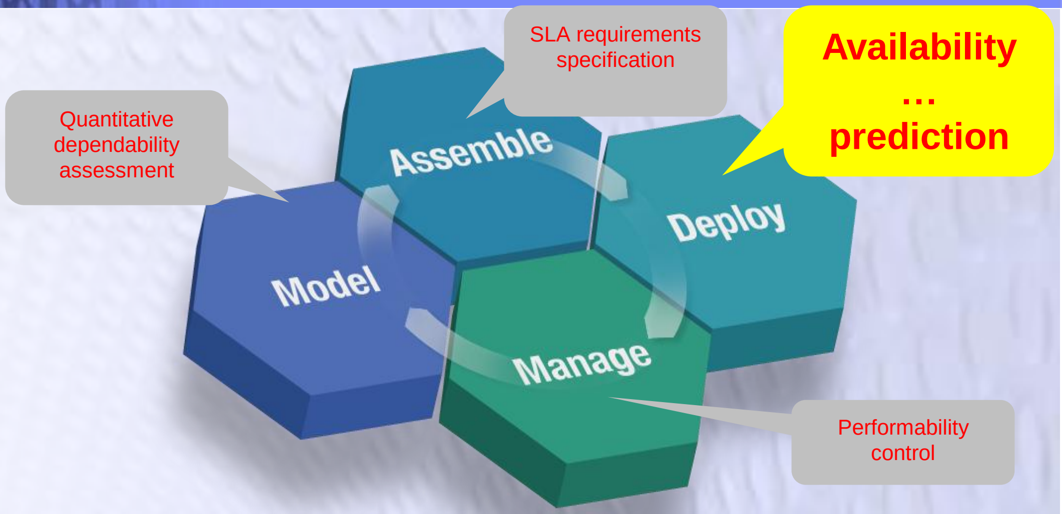
- **SOA service flow as PMS**
- **The operational life is built of distinct steps**
  - **Web service invocations are the phases**
  - **The dependability requirements of the phases are different**
    - **Based on Service Level Agreements**
  - **The execution of the phases depends on the result of previous steps**
    - **Restricted operation if a service invocations fails**
- **Dependability of the main service**
- **Bottleneck analysis**
- **Sensitivity analysis**
  - **Component's failure rate → System dependability**

Modeling vs monitoring based SOA lifecycle



IBM

Model-based optimization of Service Deployment

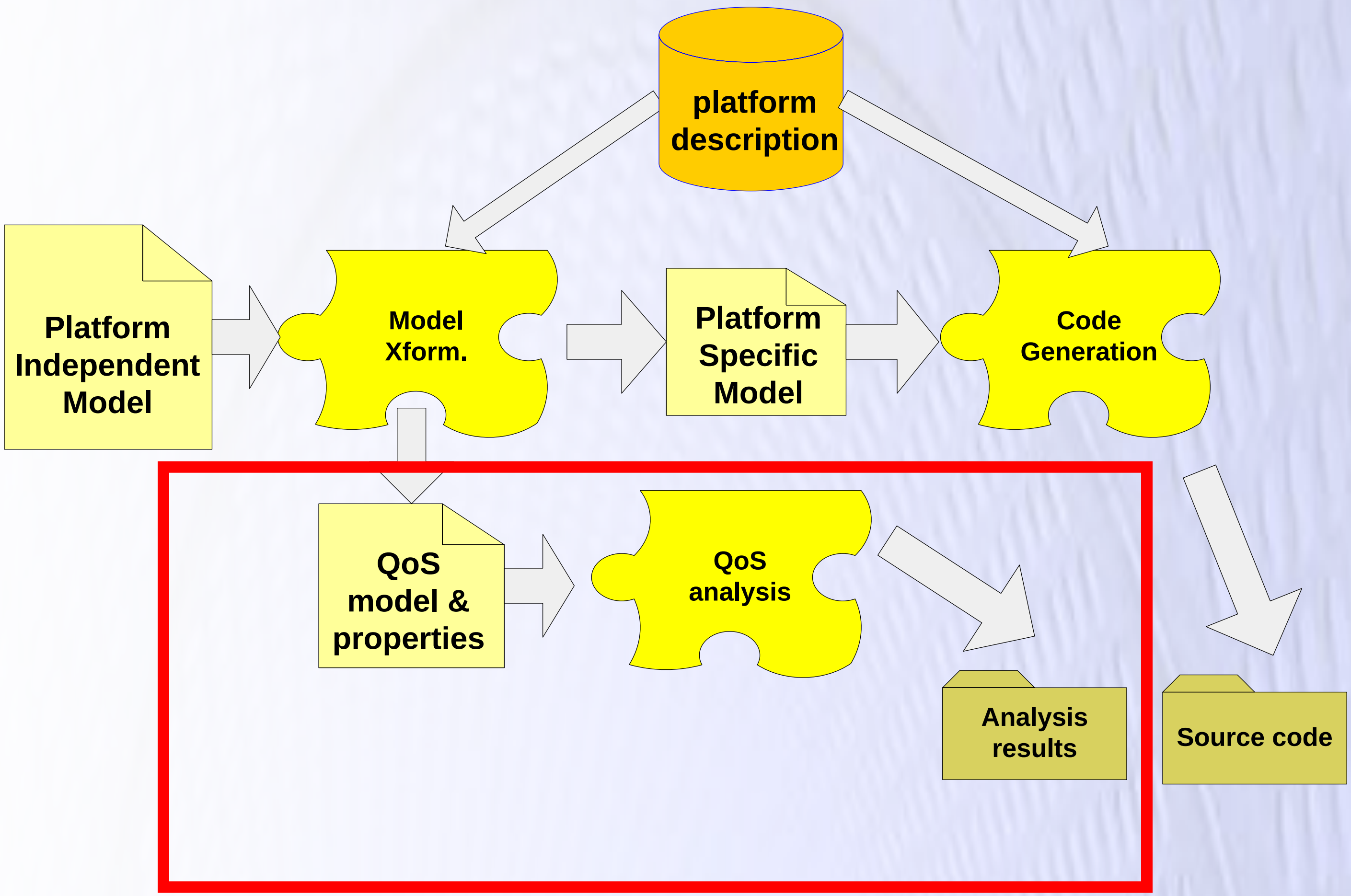


Availability aware deployment of services: integrated deployment design and optimization under cost and performance constraints

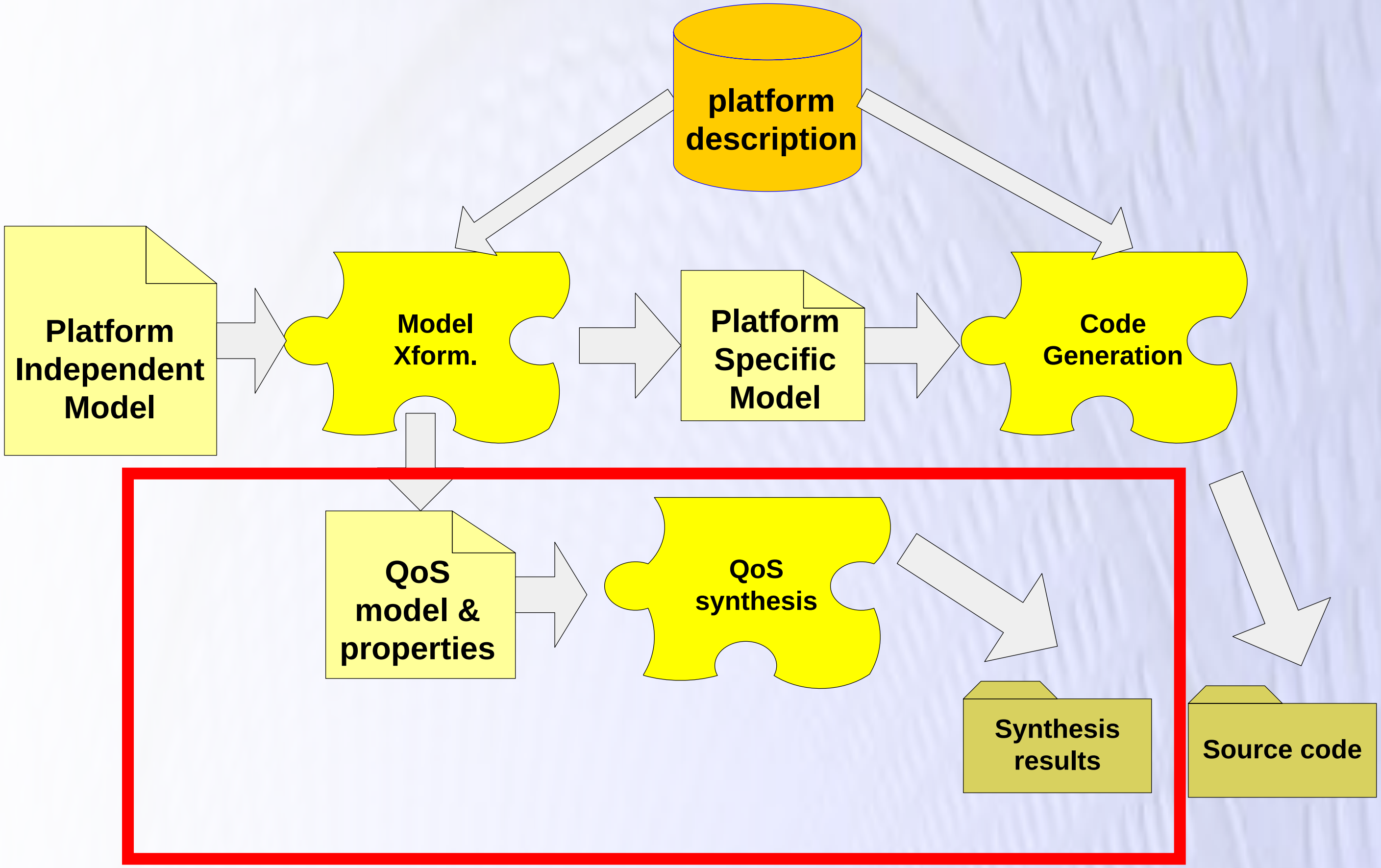
Objective

- Enterprise Information Systems
  - Towards Service Oriented Architecture
- System development
  - Model-Driven Architecture
- Quality aspects of services
  - Growing importance
- Simultaneous assurance of
  - Required availability level
  - Performance
  - Cost minimization

Model-Driven Architecture + QoS analysis

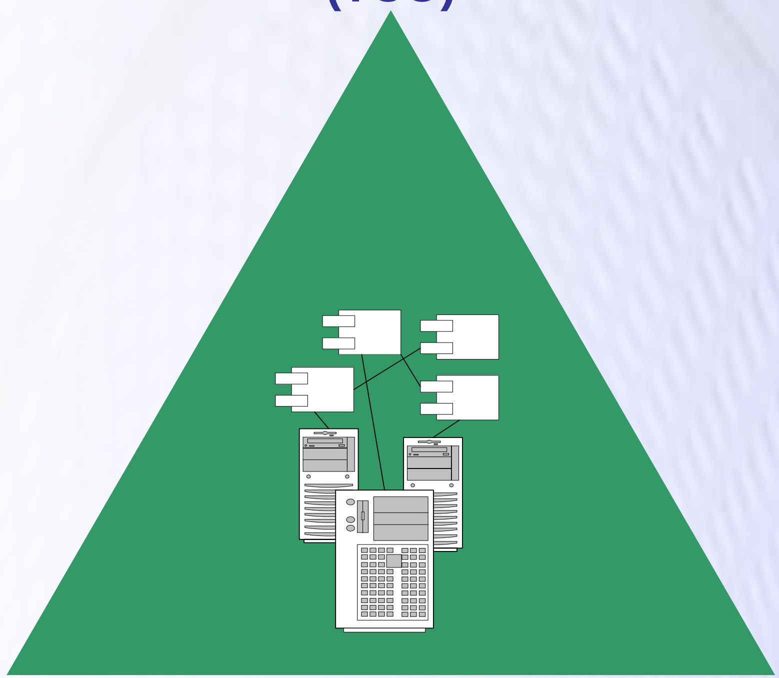


Model-Driven Architecture + QoS-based synthesis



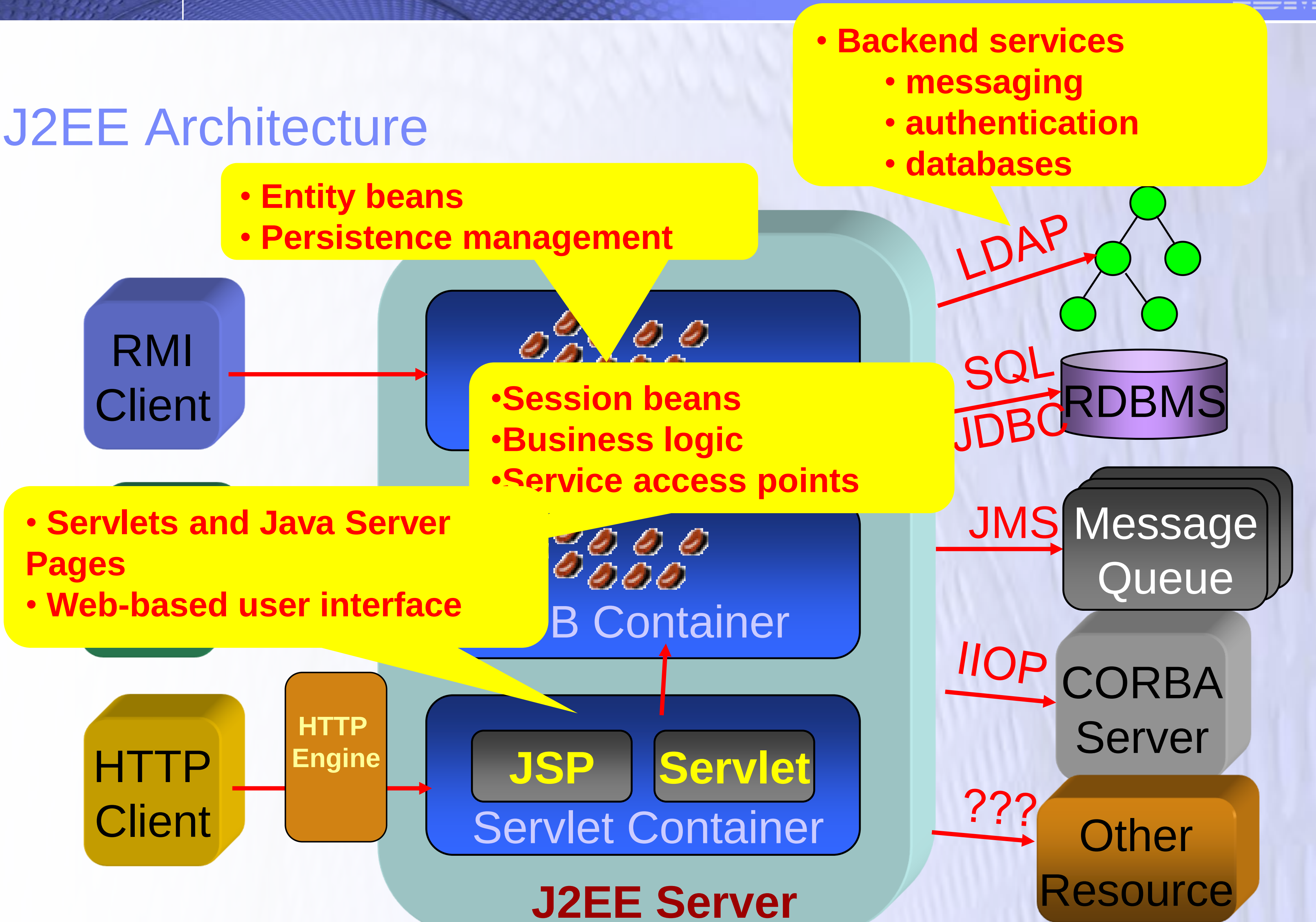
Architecture Synthesis – Design space

Total Cost of Ownership (TCO)



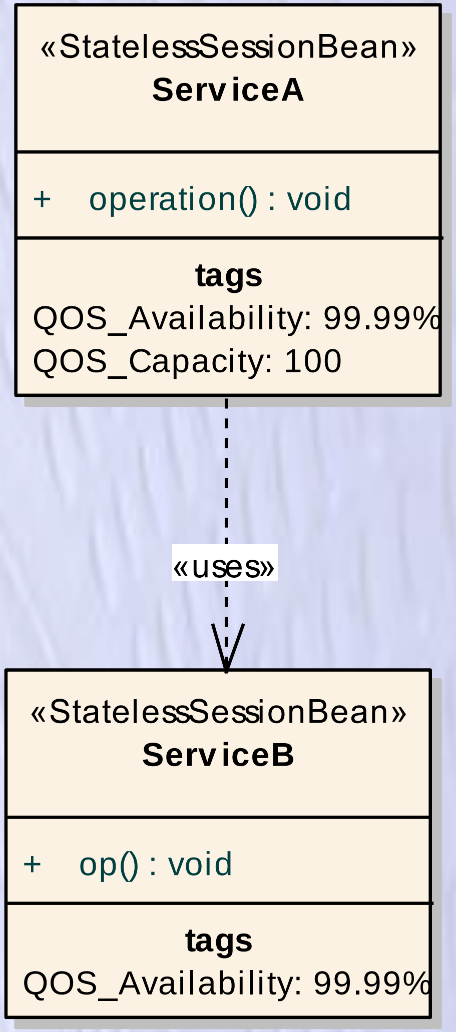
Availability **Capacity**

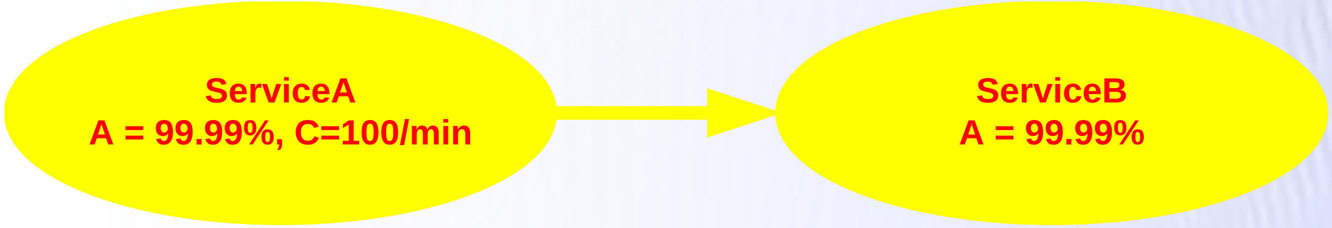
J2EE Architecture



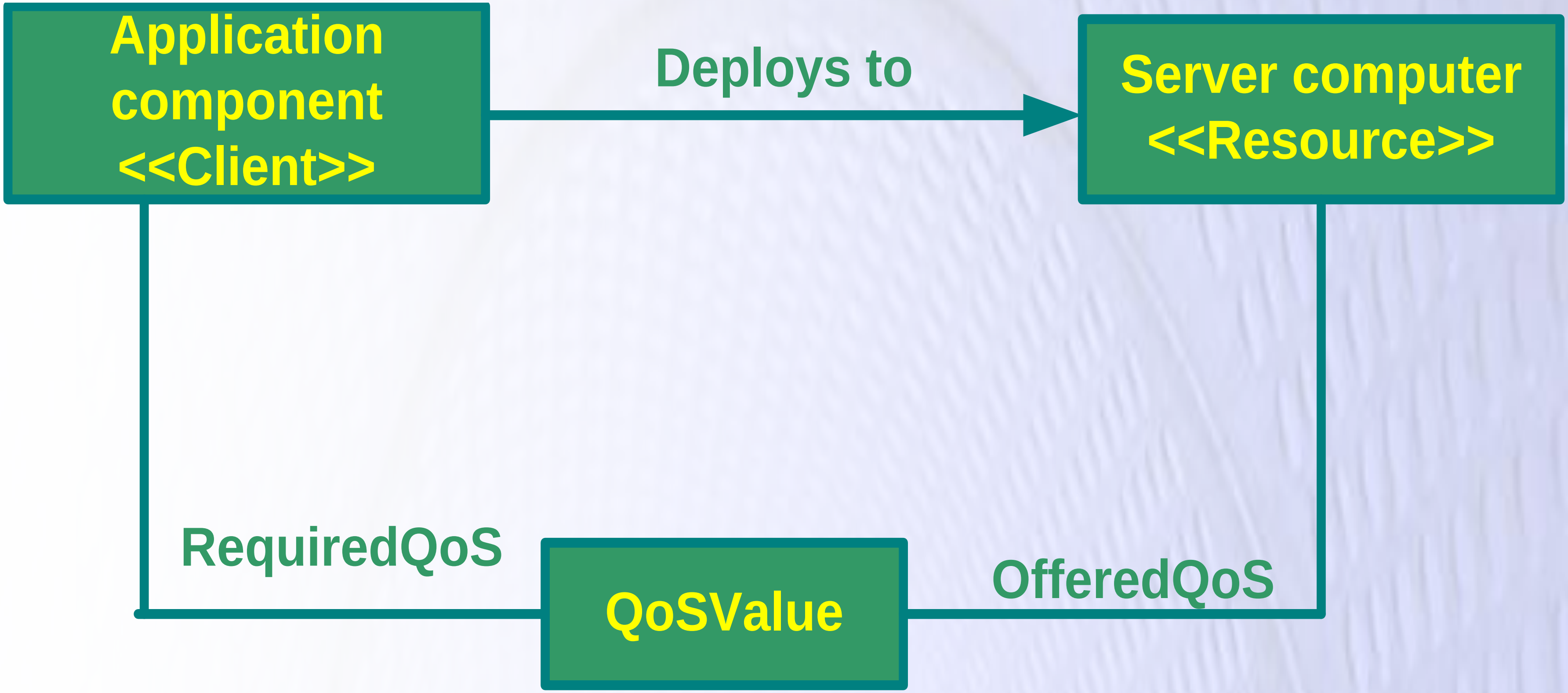
QoS model for services

- Stand-alone components
  - QoS attributes
    - Capacity requirement (throughput)
    - Availability requirement
- Links
  - Represents single usage relationship
  - Directed
  - QoS attributes are propagated





General Resource Model



Mapping components to the QoS Model

- Fault model
  - Hardware components
    - Independent faults
  - Operating system
  - Application server software
  - The application components are treated fault-free
    - Majority of the code
      - automatically generated
      - Formally verified

- QoS Service Component
  - EJB Module
    - atomic deployment unit
- Component availability
  - Max(required availability for the services) <
    minimal availability of the runtime platform running the beans in the module
- Capacity requirements
  - Sum of capacity reqs. of the contained beans

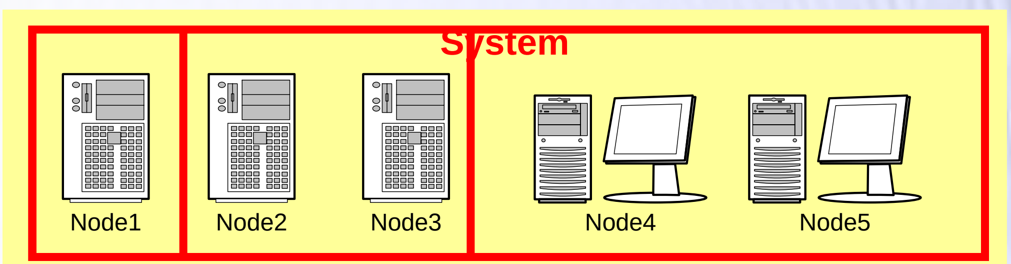
Architecture Optimization – Objective Function

$$TCO_{System} = \sum_{m \in HW} TCO(m) * number\_used(m)$$

Total Cost of Ownershi
the system

Total Cost of Ownership o
the specific node type

Actual number of nodes
from the specified type



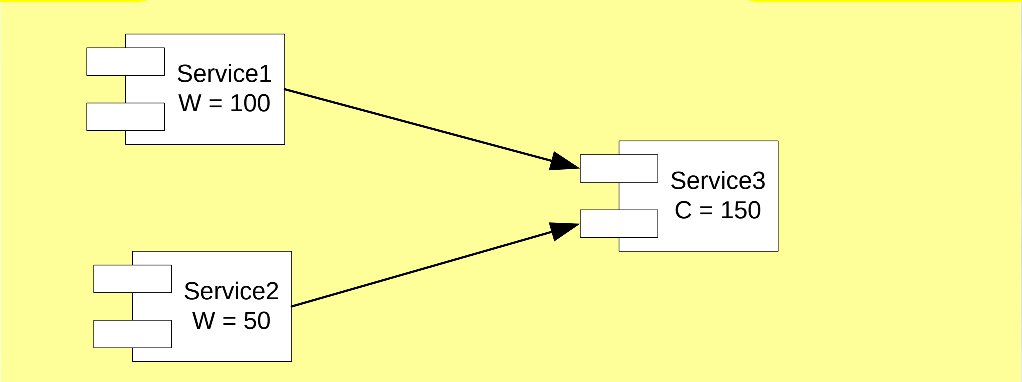
Architecture Optimization – Component Workload Balance

$$Workload(i) = Capacity\_need(i) + \sum_{j \in depends(i)} Workload(j)$$

Actual workload of the specified component

Defined direct load of the component

Indirect workload from depending services



Architecture Optimization – Component Throughput Limits

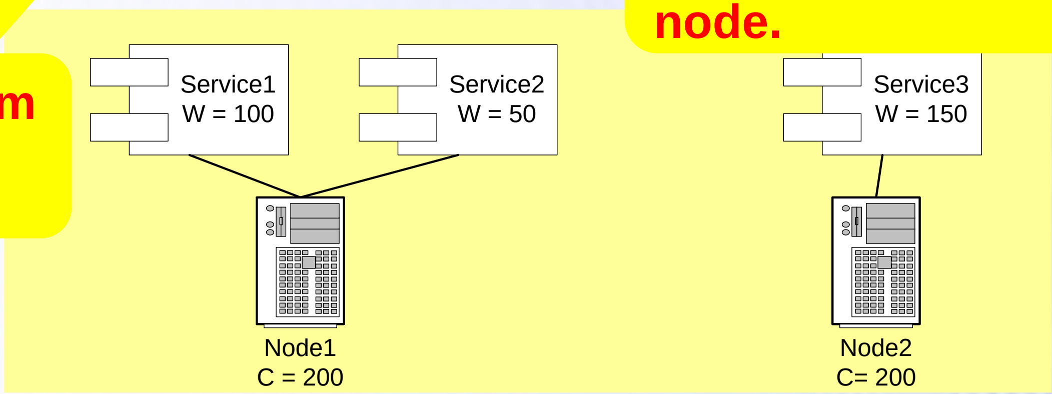
A specific hardware node of the system

Saturation factor
The maximum load (0..1) on the machine

$$\forall m \in HW : Capacity(m) * SF \geq \sum_{s \in deployed(m)} Workload(s)$$

Aggregate workload of all components running on the node.

Capacity of node m (from benchmark)



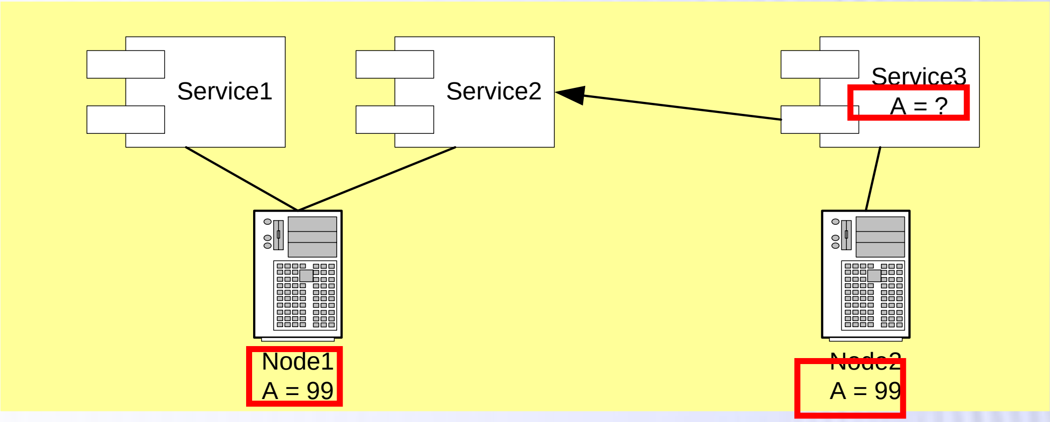
Architecture Optimization – Availability Effect of Interactions

$$A_{act}(i) = P(HW available \wedge All\ needed\ services\ available) =$$

$$= P(HW\ available) * \prod_{\substack{all\ HW \\ running \\ req.srvc}} P(HW\ avail) = A_{HW(i)} * \prod_{\substack{\forall j, HW(j)\ running \\ needed service}} A_{HW(j)}$$

Availability of the deployment targ

Availability of the deployment target HW of invoked services



Architecture Optimization – Availability Requirements

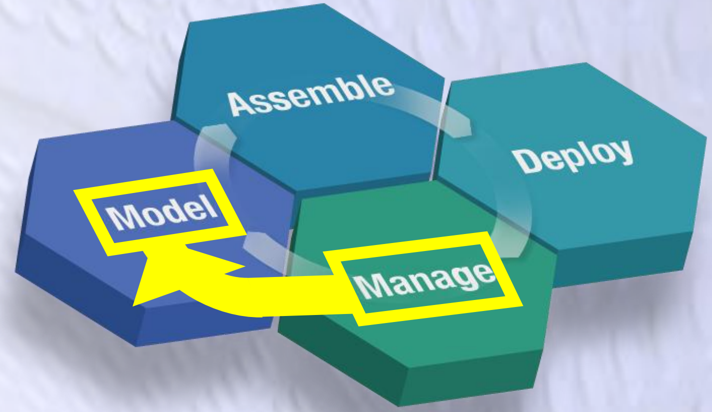
- Constraints cont'd

$$\forall i \in services: A_{act}(i) \geq A_{required}(i)$$

Actual availability of the component

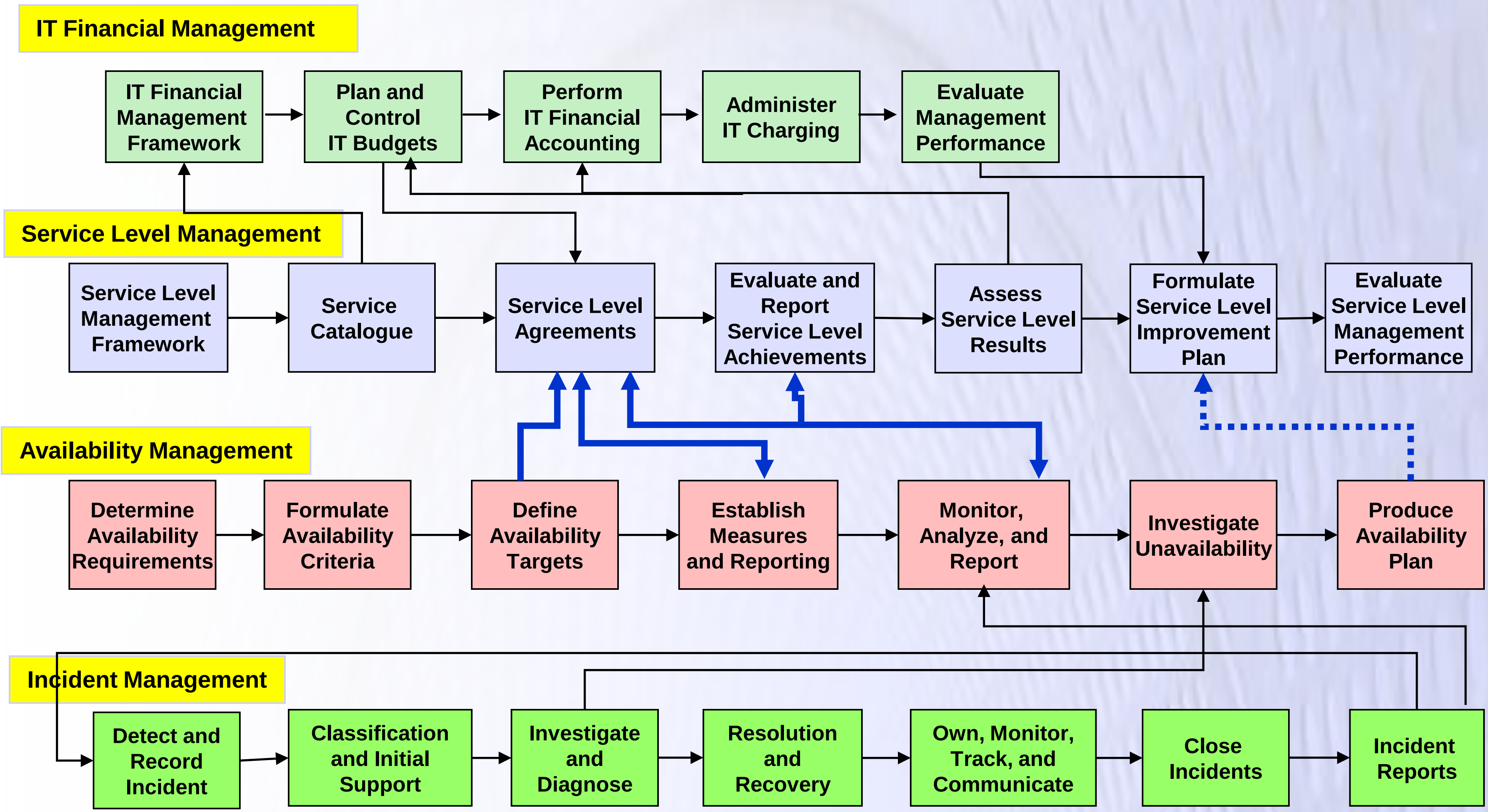
Required availability of the component

Dependability consolidation



Numerous applications implemented with no dependability considerations, but delivering business critical services ?

IT Service Management Processes are *Interdependent*



 Based on the Presentation of András Pataricza @ University Relations -- Academic Days