Approximation of (H)MSC semantics by Event Automata

Dr. Nikolai Mansurov

Dmitri Vasura

Department for CASE tools

Institute for System Programming

Moscow Russia

Outline

- Motivation
- Use Case Scenario Models
- Event Automata and requirement models
- What exactly they represent
- Use Case Studio toolkit
- Visualization of scenarios

Motivation

- Accelerated Development Methodology
  - improve adoption of formal modeling in industry
- Use (H)MSCs to formalize scenarios in use case methodology to improve requirements validation and transformation tools

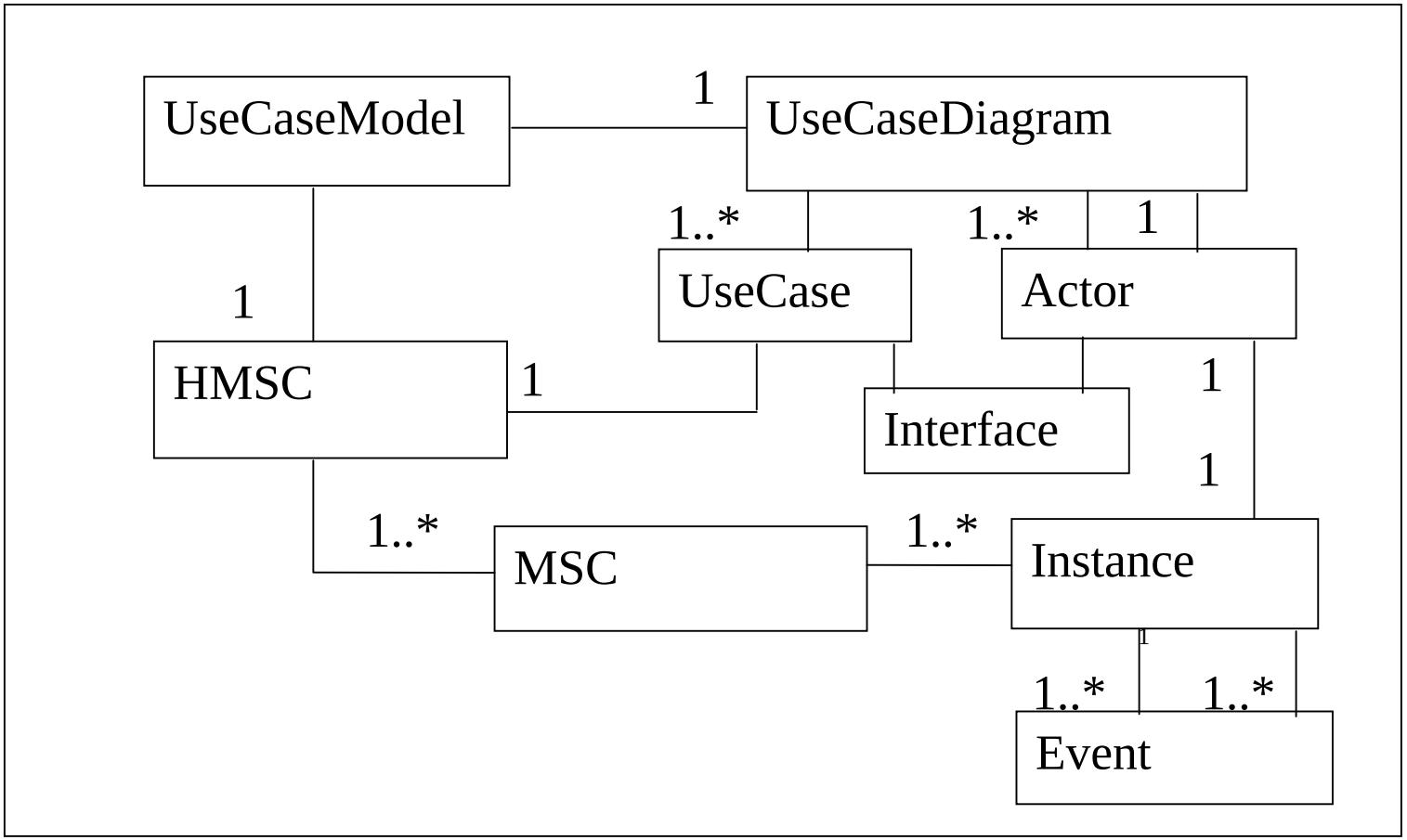
Summary

- MSCs are used to capture functional requirements
- “Instance-oriented” semantics of MSCs
  - we are interested in the behavior of the system actor with respect to all possible behaviors of all external actors
- Event automata are a simple representation of the “instance-oriented” semantics for a single instance
- Requirements model is a representation of the “collective” “instance-oriented” semantics (for multiple instances)
- “Instance-oriented” semantics provides an approximation of the standard (H)MSC semantics
- Gaps of the “instance-oriented” semantics are related to defects in requirements and can be detected by model checkers
- Tool support

MSCs are used to capture functional requirements

- Use Cases
  - black box specification of system
  - external actors and the system actor
  - exemplary behavior rather than complete
  - start-to-end behavior rather than individual operation (each use case should excite the customer)
- Use Case behavior is *non-symmetric*

Use Case Scenario Models



Semantics of use case scenario models

- Use case scenario models specify behavior of actors
- **Exemplary specified behavior** of an actor A in use case scenario model U is *a trace of events* S(A), corresponding to instance A in a ground MSC M(T), which corresponds to some traversal T(U)
- **Total specified behavior** of an actor A is a union of all exemplary specified behaviors S(A) in U
- **Semantics of a UCSM** U is a set of all total specified behaviors for all actors in the use case diagram, including the system actor

Outline

- Motivation
- Use Case Scenario Models
- **Event Automata and requirement models**
- What exactly they represent
- Use Case Studio toolkit
- Visualization of scenarios

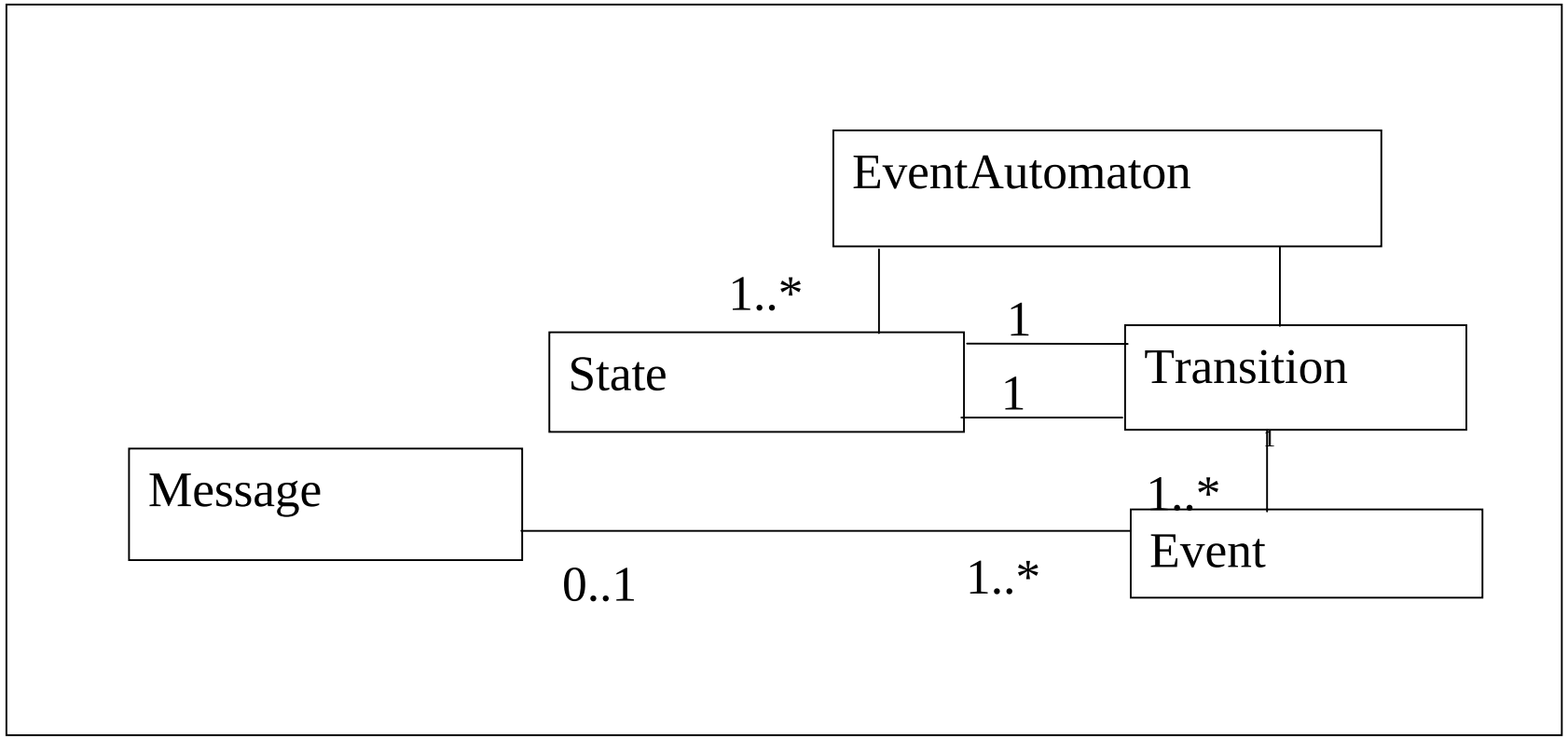
Event Automata

- Event Automata represent the total specified behavior of actors
- Event Automata are related to the theory of formal languages :
  - total specified behavior is a *language* of event sequences
  - sets of sequences of events are represented by finite state *recognizer automata*

Event Automata

- Event Automaton, corresponding to an actor A in a UCSM U is a finite state automaton, such that its alphabet of input symbols is equivalent to the set of events for A, and its input language is equivalent to the total specified behavior of A

Event Automata



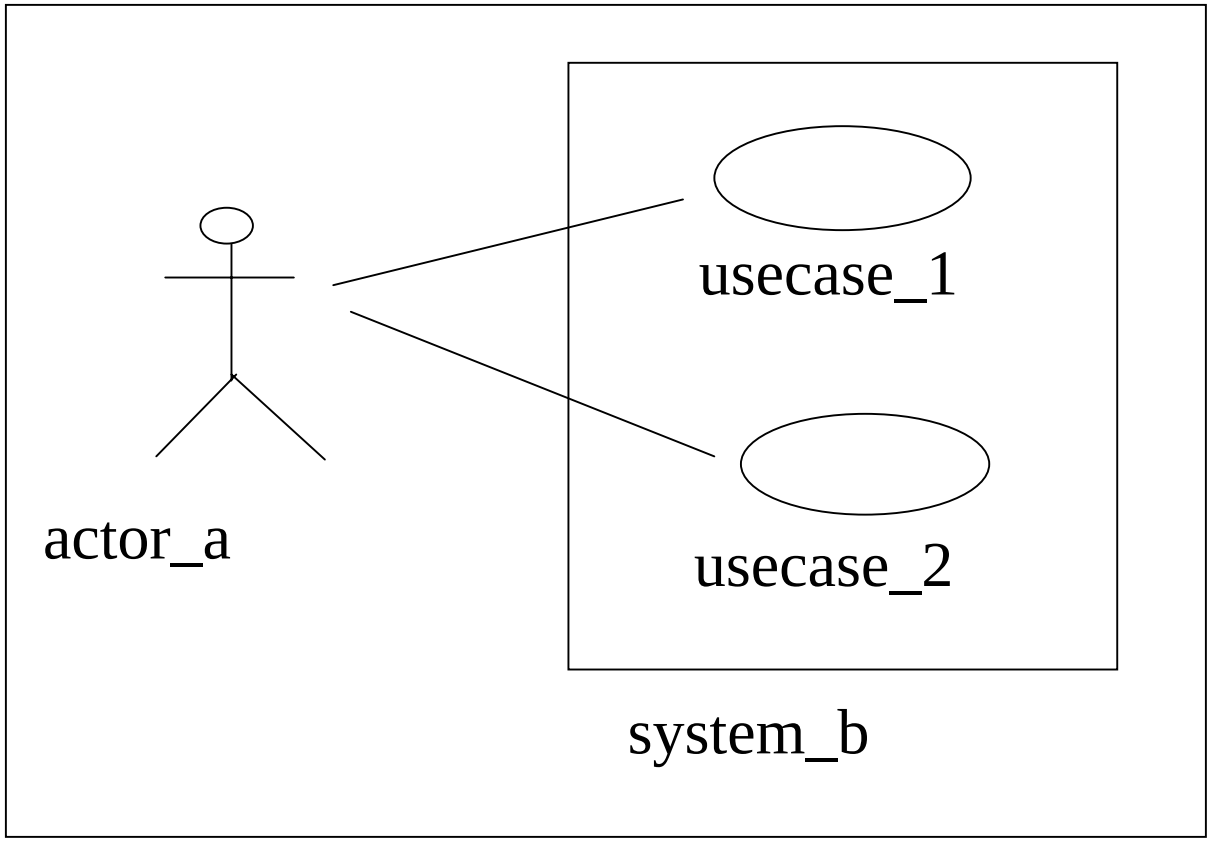
Semantics of Event Automata

- **Exemplary accepted behavior** of an event automaton E is a sequence of events M, corresponding to a sequence of transitions through E from start state to end state
- Exemplary accepted behavior is a “sentence” in the language, accepted by E
- **Total accepted behavior** of E(A) is equivalent to total specified behavior of A

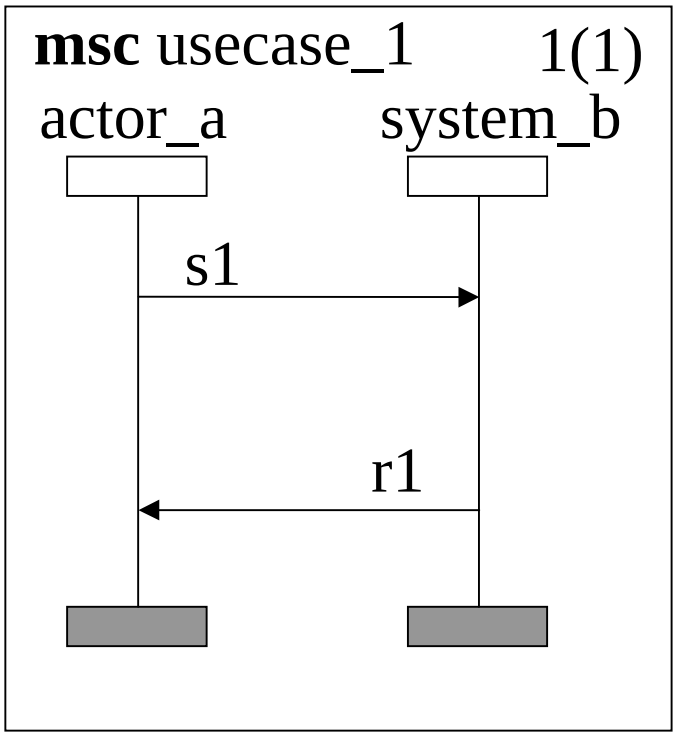
Algorithm for construction of an Event Automaton

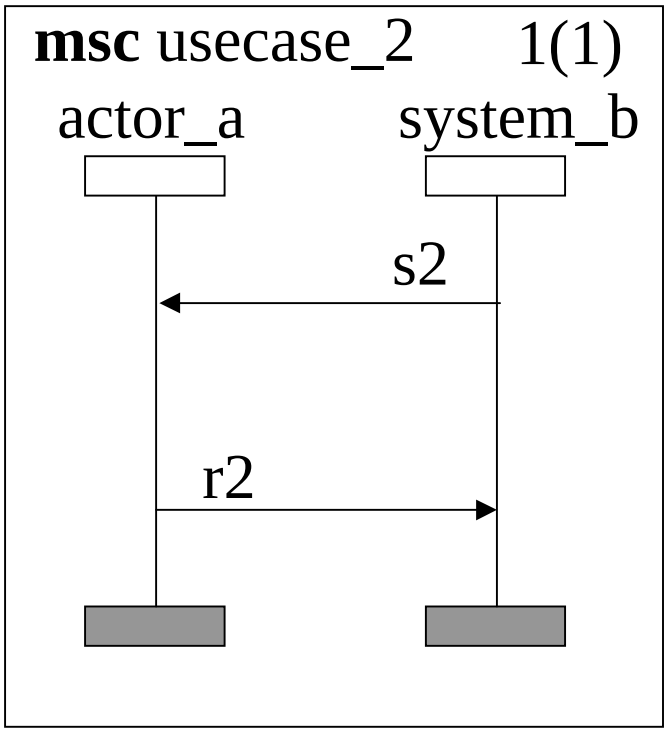
- Construct initial states and transitions of the event automaton, equivalent to the nodes and flow of the HMSC. Transitions are empty
- For each referenced basic MSC, create an event automaton, corresponding to the sequence of events involving the given instance
- Replace each state, corresponding to an MSC reference by the event automaton for the referenced MSC
- Minimize the resulting event automaton by eliminating empty transitions

Example: Use Case Diagram

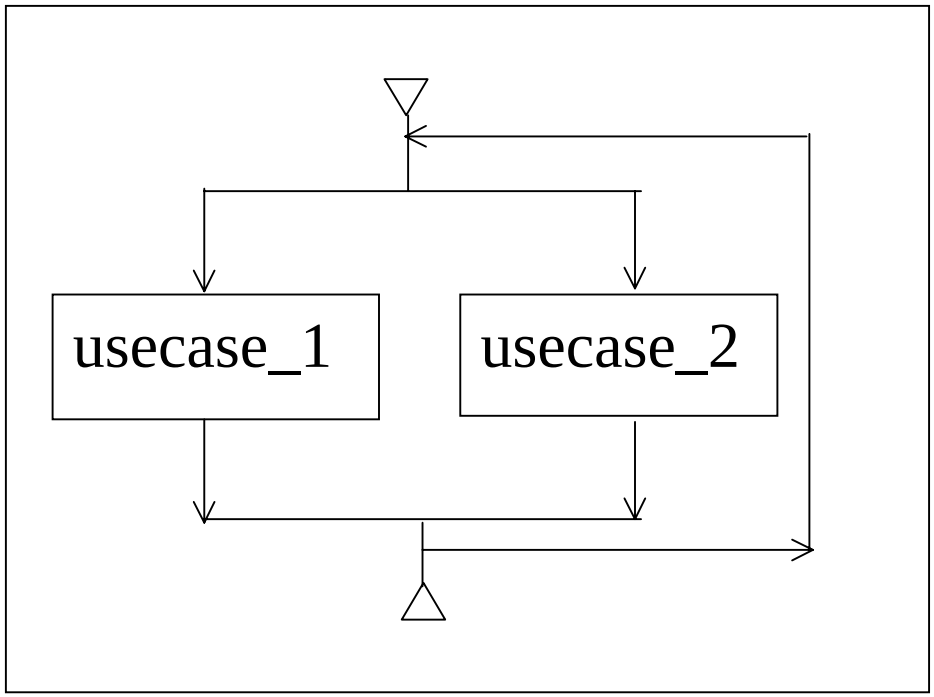


Example: MSCs

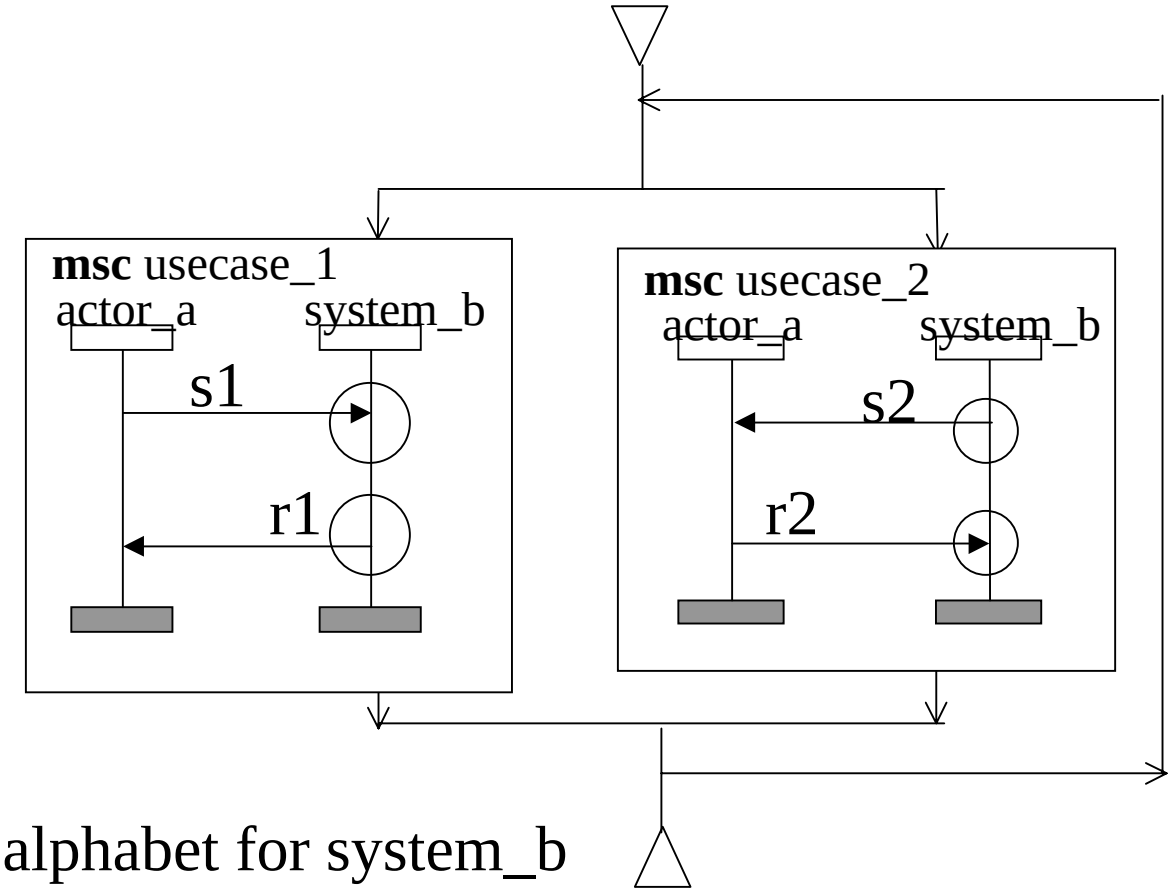




Example: HMSC

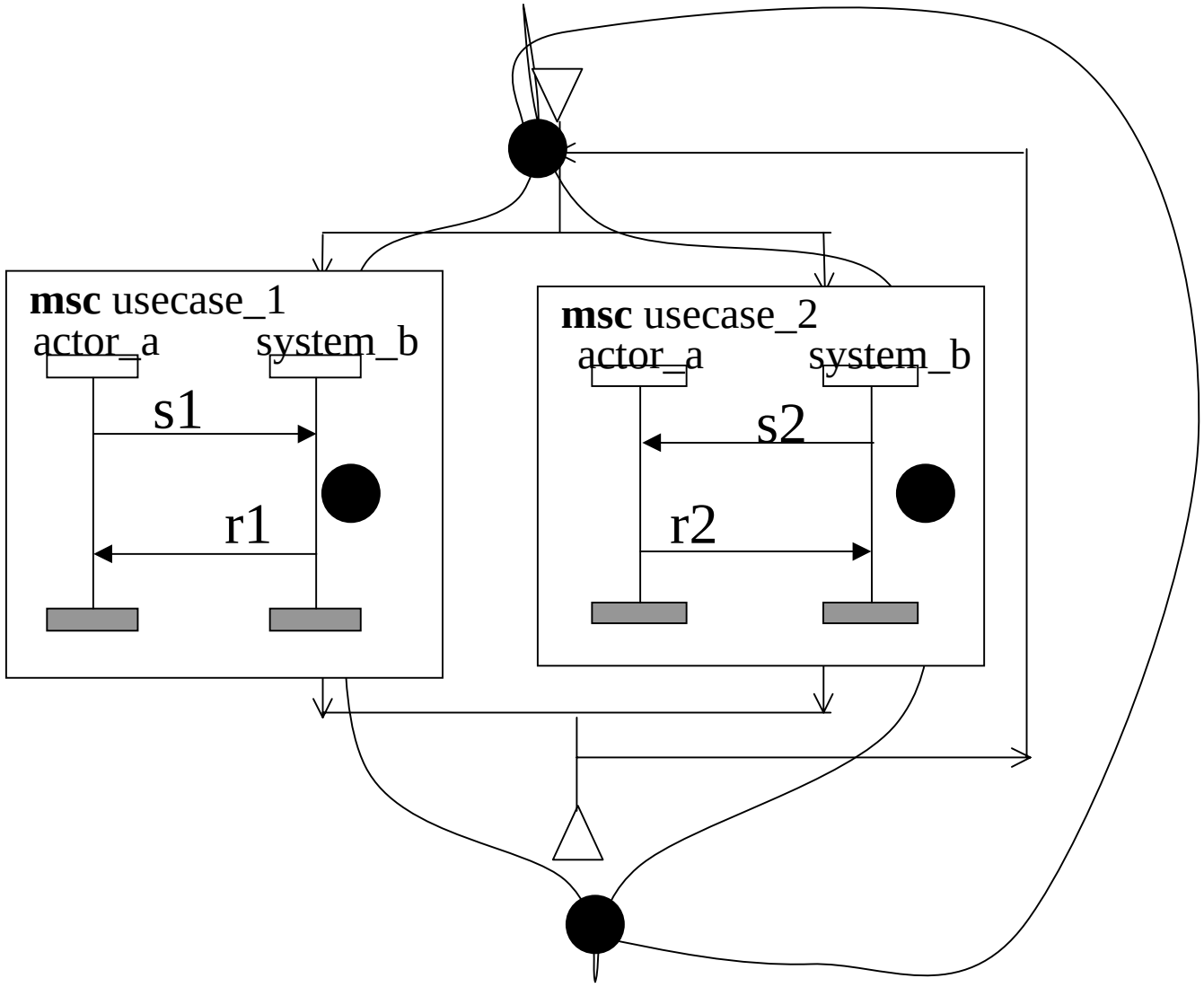


Example

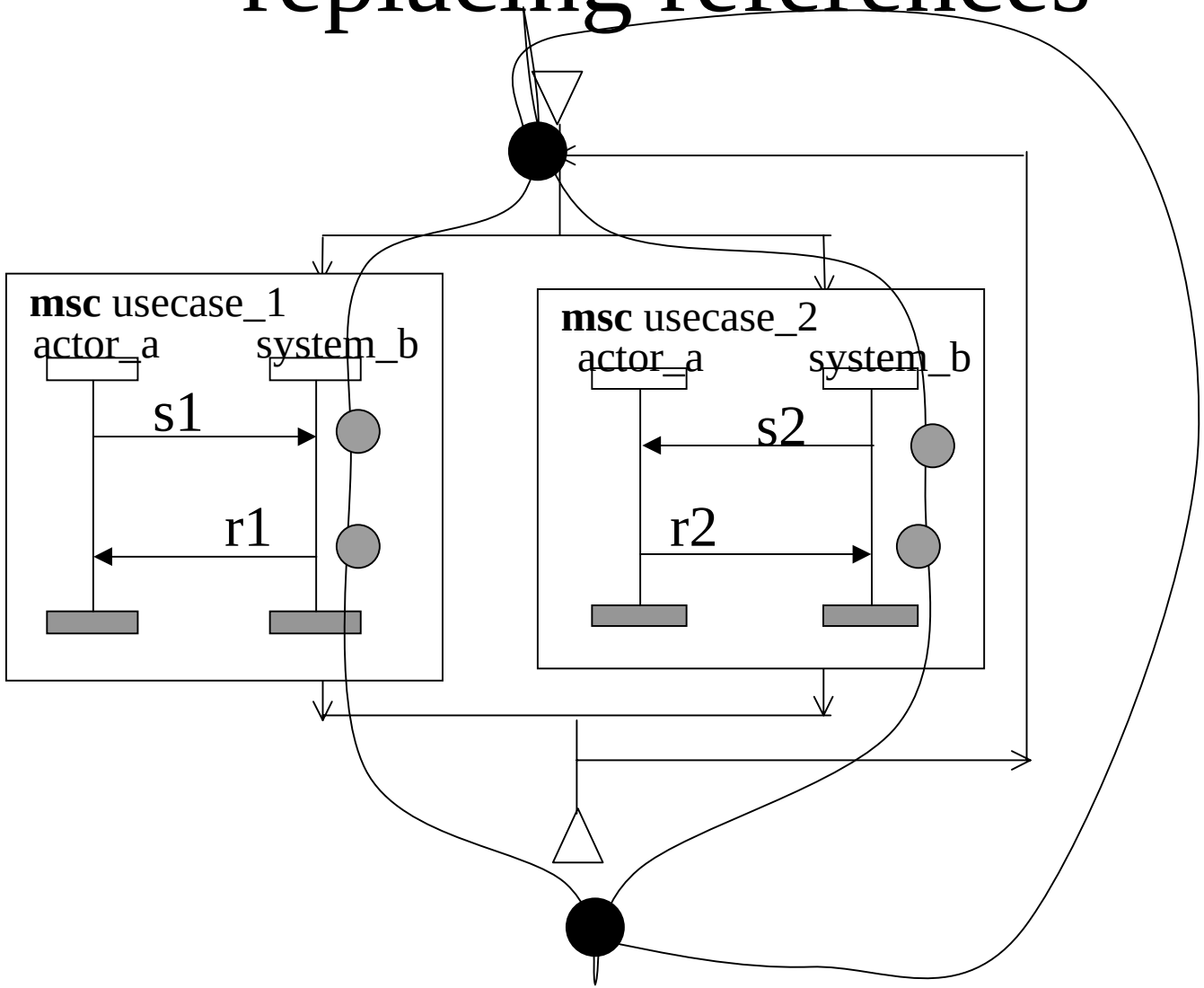


Event alphabet for system_b

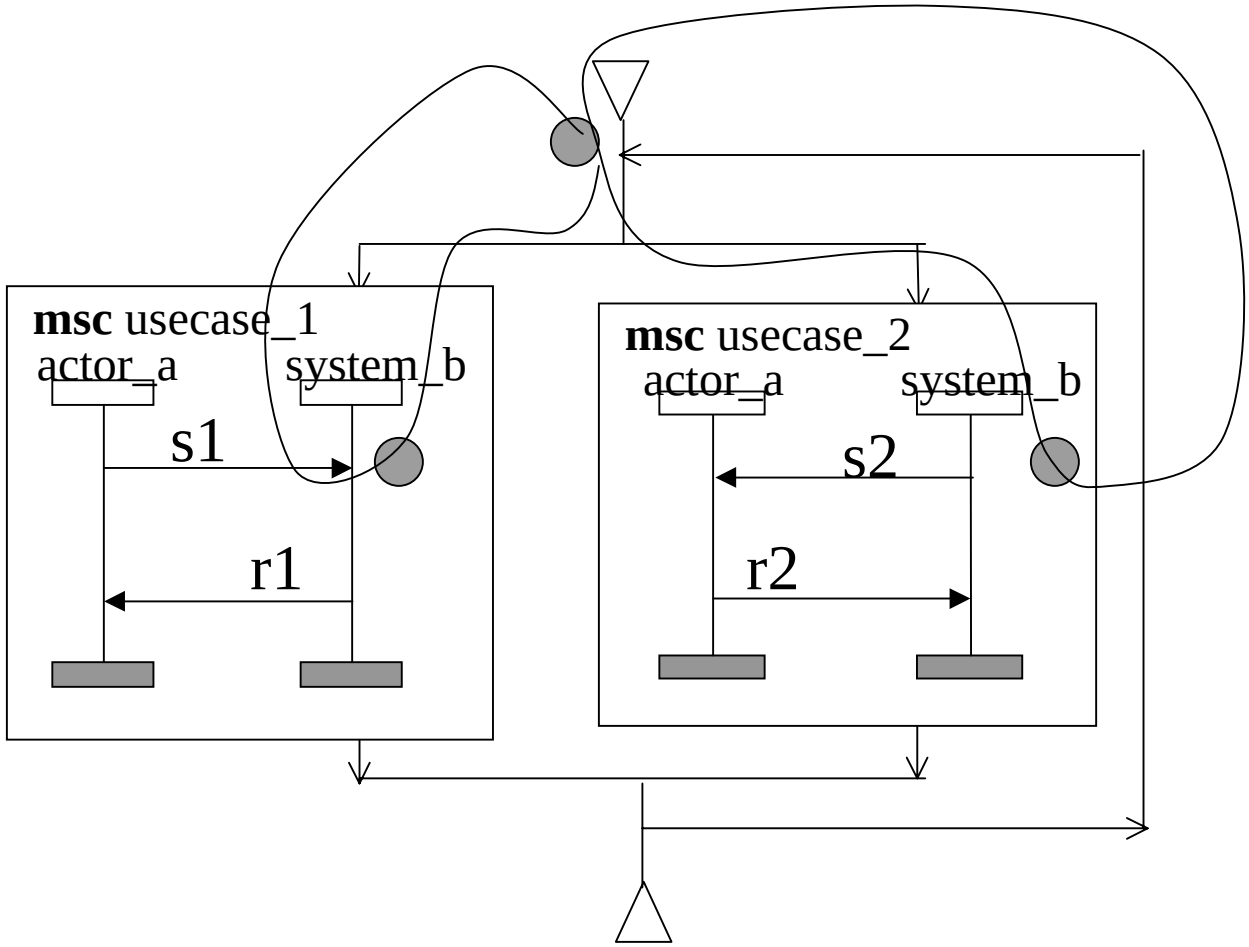
Example: initial automaton



Example: automaton after replacing references

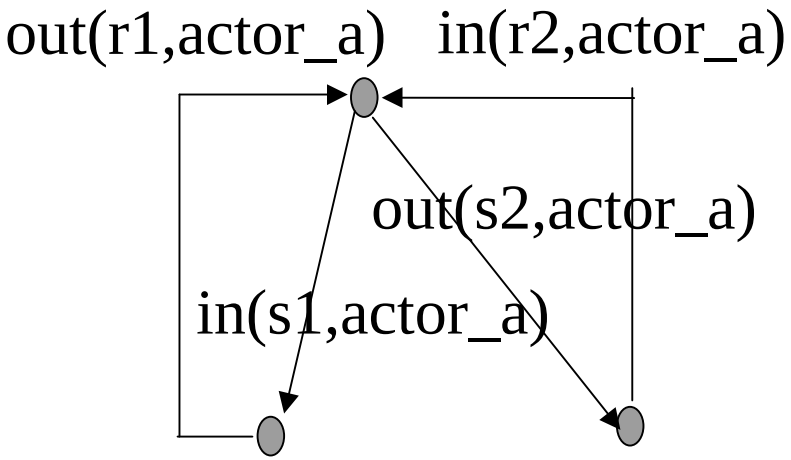


Example: minimized automaton



Example: resulting event automaton

EA system_b



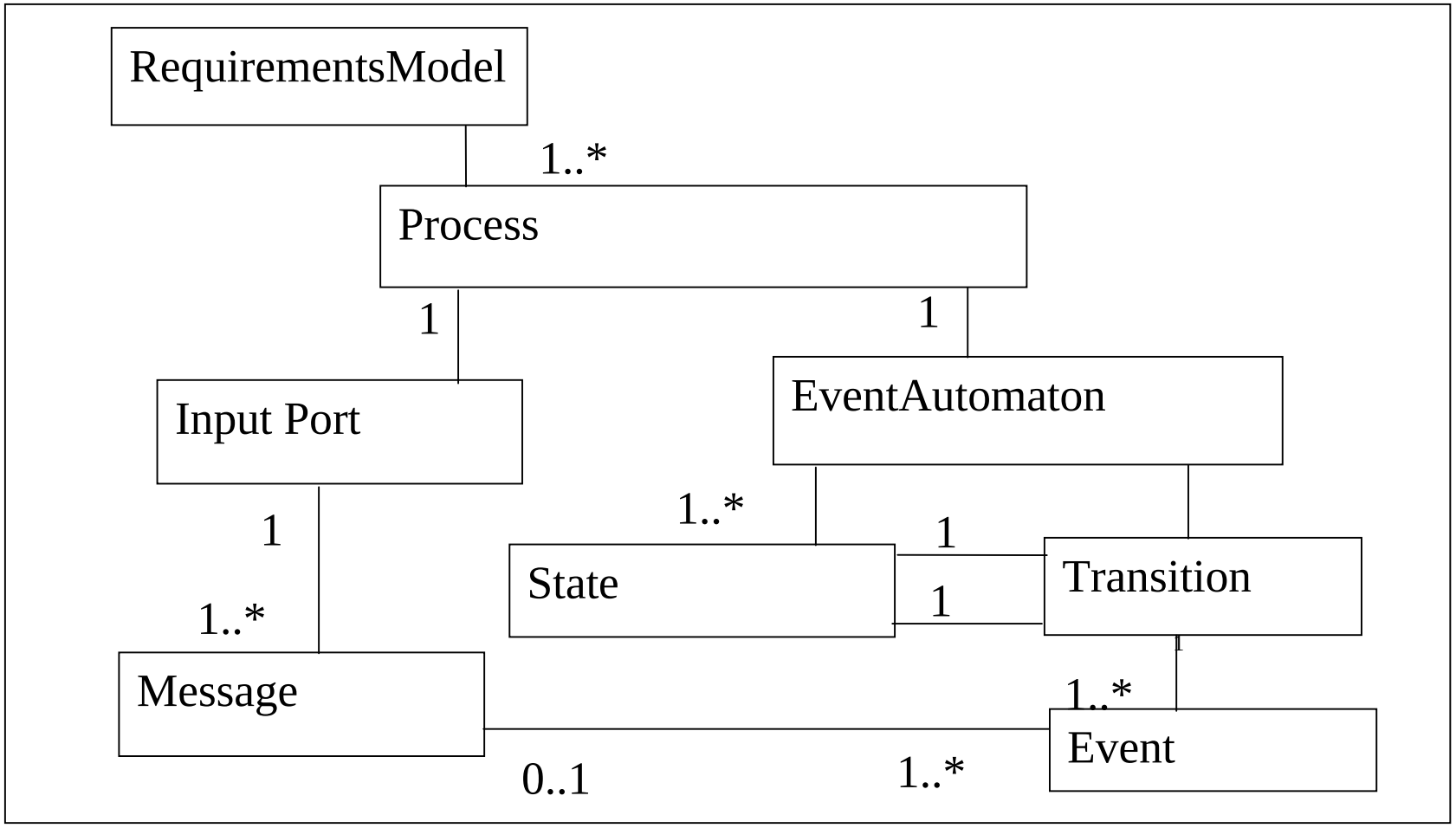
Outline

- Motivation
- Use Case Scenario Models
- **Event Automata and requirement models**
- What exactly they represent
- Use Case Studio toolkit
- Visualization of scenarios

Requirements model

- Requirements model represents “collective” behavior of several event automata
- Concurrent processes, communicating by asynchronous messages; each process has a single input port
- Event Automata are generators of the corresponding behaviors
- Dependencies on input port for send and receive events are considered

Requirements Model



Semantics of requirements model

- **Exemplary implemented behavior** of an actor A in requirements model R is a sequence of events S(A), performed by the process P(A), corresponding to some sequence of transitions through E(A) starting from the start state
- In contrast to accepted behavior, receive event can be part of some implemented behavior if there was a corresponding send event
- exemplary implemented behavior is an equivalence class of paths through the state space of the requirements model

Outline

- Motivation
- Use Case Scenario Models
- Event Automata and requirement models
- **What exactly they represent**
- Use Case Studio toolkit
- Visualization of scenarios

Failed behavior

- **Exemplary failed behavior** of an actor A in requirements model R is a sequence of events S(A), performed by the process P(A), corresponding to an unfinished sequence of transitions through E(A), and P is not able to perform any further transition
  - all other processes have reached their end states without sending the message M, responsible for the progress of P
  - the set of processes, capable of sending the message M, exercise some failed behavior themselves (deadlock)

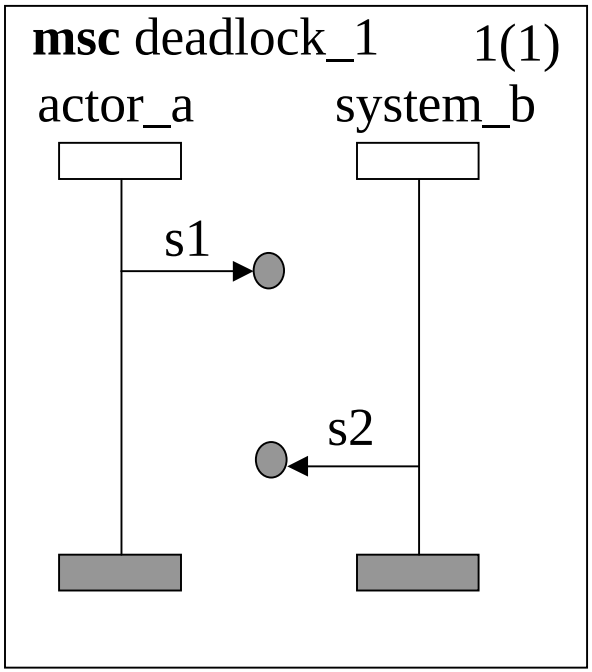
Approximation of the standard semantics

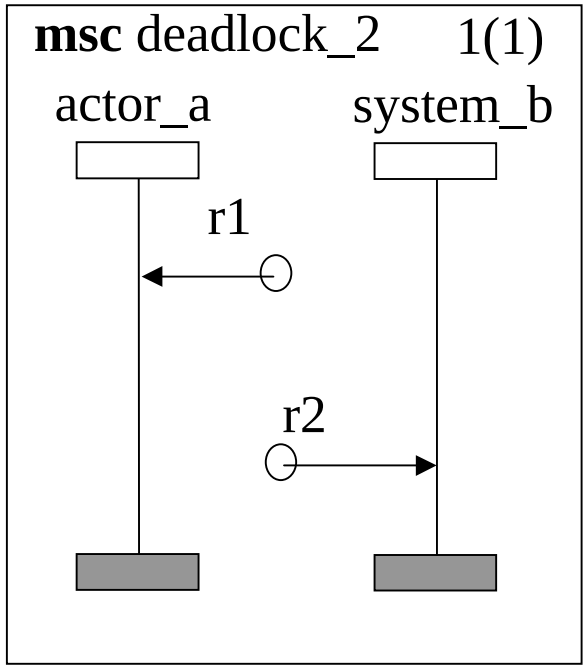
- Semantics of Use Case Models is already an approximation of the standard (H)MSC semantics
- Total implemented behavior of an actor A is equivalent to total accepted behavior of E(A) and total failed behavior of E(A)
  - total accepted behavior is included into the total implemented behavior
  - there is no other behavior than accepted of failed
  - failed behavior exists

Approximation of the standard semantics

- complete implemented behavior of all instances in requirements model (“symmetric” traces involving events from all instances) is more complex:
  - there exists specified behavior, which is not implemented (e.g. message overtaking)
  - there exist implemented behavior, which is not specified (interleaving of events in referenced MSCs)

Example: Failed behavior





Implications for requirements validation

- Approximation of scenarios by event automata adds failed behavior
- This failed behavior can be related to poor understanding of requirements (e.g. distributed choice implies either a missing actor or missing synchronization requirement)
- Failed behavior can be discovered by model checkers

Implications for requirements validation

- Use case scenario models allows designers to concentrate on typical primary scenarios (success stories)
- Subsequent approximation of scenarios by event automata automatically discovers gaps in requirements and generates failed scenarios
- Simulation of synthesized event automata models can discover unwanted scenarios
- Corrected scenarios are added to the initial set of scenarios, and the requirements model is synthesized

Requirements engineering process

- capture initial system requirements in tabular form, capture the set of external actors and use cases
- specify partial functional requirements by providing primary scenarios for each use case
- provide secondary scenarios where necessary
- identify individual transactions or operations
- specify complete behavior of the system by arranging transactions into an behavior graph
- validate requirements

Outline

- Motivation
- Use Case Scenario Models
- Event Automata and requirement models
- **What exactly they represent**
- **Use Case Studio toolkit**
- Visualization of scenarios

Use Case Studio

- The methodology is implemented in Use Case Studio toolkit
  - Validation and Generation Kernel
  - Visualization tools

Validation and Generation Kernel

- UCD, MSC, HMSC analyzer
- Synthesizer of Event Automata
- Code Generation platform (API to Event Automata)
- Simulator of Event Automata
- Generator to SDL
- Generator to TTCN

Visualization tools

- UML Use Case Diagram Editor
- MSC/ UML Sequence Diagram Editor
- HMSC/ UML Activity Diagram Editor
- Interface Editor
- Scenario Recorder
- HMSC “Episode Simulator”
- Model Navigator

Conclusions

- We presented a framework for requirements engineering based on formal use case scenario models
- Event Automata representation of instances is non-symmetric and thus provides only an approximation of the standard (H)MSC semantics
- However it leads to intuitive structure of the synthesized model, allows to use model checkers for requirements validation and allows to build transformation tools, e.g. to generate TTCN test from requirements

Visualization: Interface Editor

Visualization: Scenario Recorder