Behavioral Modeling Languages

Ákos Horváth and Dániel Varró With Contributions from István Majzik, Gergely Pintér, András Vörös, Gábor Bergmann, Ábel Hegedüs

Model Driven Software Development





Budapest University of Technology and Economics Department of Measurement and Information Systems

An Overview of Behavioral Modeling Languages





Dynamic Languages: An Overiew

System

- State-based reactive
- Dataflow-based
- Event & Rule-based
- Agent-based
- Block diagrams
- Other

Property

- Requirements
- Scenarios

Analysis techniques:

- Simulation, Static analysis, Model checking,
- Symbolic computation, ODE (Diff. Eq)

- Engineering languages:
 - Statecharts, Statemate, Business Process Models, Simulink Block Diagram, Message Sequence Charts, KAOS, Drools, CQL, Esterel, AnyLogic, Modelica, Ptolemy-II, ...

Formalisms:

 Petri nets, Finite automata, Timed automata, Cellular autom. Bond graph, Process algebra, Queuing network, Kahn process network



Characteristics of Dynamic Languages

Specification

- Consistency
- Completeness
- Unambiguity

Time

- o Untimed
- o Discrete
- Continuous

Communication

- Synchronous
- Asynchronous

- Determinism
 - Stochastic
 - o Deterministic
- Causality
 - Causal
 - Non-causal
- Analysis
 - Exact vs. Approximative
 - Complete vs. Incomplete
- Other concepts
 - Conflict, priority
 - Dependency,



Property Specification Languages





- Requirements
 - Human readable
 - Structured text (DOORS, SysML)
 - Requirements modeling notations (i*, KAOS)

Scenarios

- Specify permitted / forbidden execution paths
- LTL, Temporal OCL
- UML Sequence Diagrams
- Message sequence charts



State-based languages



- Main concepts:
 - State , Transition
 - Event, Action
 - State hierarchy, history
- Examples:
 - Finite automata
 - Timed automata
 - Cellular automaton
 - Statemate (Harel)
 - UML Statecharts
 - Matlab SimulinkStateflow

Dataflow-based languages



- Main concepts:
 - Process, activity channel, queue, token/message
 - Examples:
 - Activity Diagrams
 - Business Process
 Models
 (also event-based)
 - o Petri nets
 - Queuing networks
 - Kahn process networks
 - o Esterel



Event-based Rule languages

File	Edit Source Status: [D	raft]
Attrib	Ites Edit	Mair 🗧
Load Template Data		
WHEN	+	
	There is an Applicant with:	
	age less than\$max_age bar	
1.	age greater than or equal to 🔽 \$min_age 📴	
	creditRating equal to \$cr ₽ ₂ ■	Т.
2.	There is a LoanApplication [\$a]	
THEN	+	
1.	Modify value of LoanApplication [\$a] approved false 🗾 🗖 🆆 🤑	🛯 🔰 Ο Κι
(show options	·)	
		🚢 🗉 Fxan
(nroc	2000TX	
<		
<	rules>	OBL
	<query id="helloworldRule"></query>	
<	! [CDATA [o Gr
S	elect T.StockName as shortName, T.LastPrice as price	
I	rom stockinputchannel	0.50
1.	PARTITION BY shortName	C -
	MEASURES A.shortName as StockName, A.price as LastPr	
	PATTERN (ABBBA)	
	DEFINE Definition CQL-	REGISTER STREAM str
	A as A.price > prev(A.price),	STREAM (
	B as B.price < prev(B.price) Data manipulation CQL-	SELECT id, SUM(val
)	as T	GROUP BY id)
		Deletine energian
		Relation operation

- Main concepts:
 - Events (atomic, complex)
 - Event queue/stream
 - Timestamp, Time window
 - Rule(Precondition, Action)
- Examples:
 - Business rules (Drools)
 - Graph transformation
 - Stream processing (CQL)
 - Complex event processing

Stream operation

Windowoperation



(id INT, val INT)

AS s

Agent-based languages



- Concepts
 - Agents + Connections
 - Behavior

(create, destruct)

- Space, Mobility,
- Environment
- Characteristics
 - Decentralized
 - Individual-centric
 - Examples:

- o AnyLogic
- Social simulators



Continuous-time Languages

Block diagrams (causal) (Simulink, Ptolemy)



 Multi-Physics (non-causal) (Modelica, Bond Graphs)



Other Dynamic Languages

- Population dynamics
 - N(t+1) = N(t) + B D + I + E
 (birth, death, immigrants, emmigrants)
 - Calculation of rates



- Forrester System Dynamics
 - Stocks, Flows
 - Feedback, Time delays





Dynamic Metamodeling in DSLs





Dynamic Metamodeling in DSLs



Complement Static Metamodel with

 Dynamic metamodel: currentState, configuration, etc.
 Execution trace metamodel: previous state, replay



Example 1: Business Processes



MÚEGYETEM 1782

Example 2: Petri Nets



MÚEGYETEM 1782



Metamodeling of Execution Traces



Representation for

Hierarchy of steps (simple, compound)

- Old value → New value
- Aim: Replayable



Statecharts for Modeling Reactive Behavior

Statecharts





State-based behaviour modeling

State partition (AKA state space)

- A set of distinguished system states
- Examples
 - {Mon, Tue, Wed, Thu, Fri, Sat, Sun}
 - States of microwave oven: {full power, defrost, off}
- DEF: A state partition is a set, <u>exactly one</u> element of which characterizes the system at any time.

Current state

• E.g. today is Wed, the microwave is on defrost, etc.

• **DEF**: At any given moment, the current state is the element of the partition which is currently valid.





Composite state modeling

- Modeling complex systems
 - Asynchronous components
 - Composite state space as product of state spaces
- Challenge: scalability
 - Exponential explosion of state space
 - 10 components of 6 local states each \rightarrow 6¹⁰ states!
 - More concise notation required
- Solution: statechart languages
 - Hierarchical refinement with history
 - Concurrent regions



Statecharts = States + Transitions

- Describes the states and state transitions of the system, of a subsystem, or of one specific object.
 - hierarchical and concurrent systems
- States
 - Concrete state:
 - Combination of possible values of attributes
 - Can have an infinite state space
 - Abstract states: (like in Statecharts)
 - Predicates over concrete states

 - Hierarchical states:
 - Frequent in embedded apps (e.g. control of car brake)
- Transitions
 - Triggering Event
 - Guard
 - Action





Statechart - introduction

- For defining reactive behavior of objects
 - Responds to events: state transitions and actions
 - Traditional approach: state machine
- Statechart: extension to state machine
 - <u>State hierarchy</u>: refinement of states
 - <u>Concurrent behavior</u>: parallel threads
 - <u>Memory</u>: last active state configuration





States I.

Attributes:

- \circ entry action
- o exit action
- static reaction
- State refinement
 - Simple state
 - OR refinement: auxillary state machine, only one active state
 - AND refinement: concurrent regions (state machines), all regions are active in parallel







Example for state refinement: TV







State II.

History state

- Stores the last active state configuration
- Incoming transition: it sets the object to the saved state configuration
- Outgoing transition: defines the default state, if there were no active state since
- Initial state: becomes active when entered to the region
 - One in each OR refinement
 - One in each AND region
- Final state: state machine terminates





Statechart elements

- State
- (Transition)
- History state
- Initial State
- Final State







Transition I.

- Defining state changes
- Syntax:

trigger [guard] / action

<u>trigger</u>: event, triggered operation or time-out

- o guard: transition condition
 - Logic formula over the attributes of the objects and events
 - referring to a state: IS_IN(state) macro
 - Without trigger: if becomes true the transition is active
- \circ <u>action</u>: operations \Rightarrow action semantics





Transition II.

Time-out trigger:

 becomes active if the object stays in the source state for the predefined interval

e.g., tm(50), based on system time

Complex transitions







Transitions between different hierarchy levels





Transition example







Complex Example

- Traffic light for an intersection with a prioritized road
 - Off: (blinking yellow)
 - On: green for the priority road
 - Green, yellow, red etc. Different timerange (timer)
 - 3 waiting vehicle on priority road: green light despite the timer's ticks
 - Automatically take photos of vehicles crossing the piority road on red light. Manual on/off for this feature.





1. Basic state machines







2. Hierarchy







3. Concurrent states







4. History States







Complete System







Example Concrete State



ΜÚΕ

Semantics: How does it work?

- Basics:
 - Hierarchical state machine (state chart)
 - Event queue + scheduler
- Semantics defines:
 Behavior in case an event occurs
 → one step of the state chart
 - o (concurrent) transitions fire
 - State configuration changes in all region in the active state and also one substate in the OR refinement (recursively)





Semantics of State Transitions

Separately processed events:

 Scheduler only triggers the next event if the previous one is completely processed stable configuration: there is no state change without an event

Complete processing of events:

- The largest set of possible fireable transitions (all enabled transitions fire, if they are not in conflict)
- How does it work?:
 - →Steps of the event processing





Steps of event processing I.

Scheduler triggers an event for the statechart in a stable state configuration

Enabled transitions:

- Source state is active
- The event is their trigger
- Guards are evaluated to true

Based on the number of fireable transitions

- Only one: fire!
- None: do nothing
- More than one: select transitions to fire?





Steps of event processing II.

- Selection of fireable transitions:
 - Fireable = Enabled + Max priority
 - Conflict: Has the same source state
 - Formally: the intersection of their left (exit) states is not empty
 - \rightarrow Conflict resolution \rightarrow <u>priority</u>:
 - Defined between two transitions (t₁ and t₂)
 - t₁ > t₂, if and only if the source state of t₁ is a substate within the state hierarchy of t₂ ("lower level")
 - \rightarrow Priority insufficient to resolve conflict if
 - Same source state (or parallel subregions)





Steps of event processing III.

- Selection of transitions to fire:
 - Parallel execution of concurrent transitions
 - Maximal set of fireable transitions (= cannot be extended any further)
 - There is no conflict between any two transitions
 - Selection of this set:
 - <u>Nondeterministic!</u>





Steps of event processing IV.

- Selected transitions fire: in nondeterministic order
- Firing one transition:
 - Leaving the source states from the bottom to top and execute all their exit operations
 - Execute the action of the transition
 - \odot Entering the target states from top to bottom and execute the entry actions \rightarrow new state configuration





Steps of event processing V.

- Entering a new state configuration:
 - Simple target state: part of the state configuration
 - Non-concurrent superstate: direct target of one of its substate or its initial state
 - Concurrent target state: all of its regions have to have an active state either as direct target state (maybe via fork) or as initial state
 - History state : the last active state configuration if there is none: the target state of the history state





















































Yakindu Statechart Tools

Example tool support: Yakindu Hierarchical state chart language







Yakindu Statechart Tools

Java/C++ code generation from statechart • *Magnetron* switches to state *On* (simplified) /* The reactions of state On. */ private void reactMagnetron On() { if (sCITimer.timeout) { sCIBeeper.operationCallback.beep(); stateVector[0] = State.magnetron Off; } else { if (sCIUser.open) { stateVector[0] = State.magnetron Off;





Summary

- Effective technique to model certain dynamic systems
- Hierarchic refinement allows iterative development
- Already used in many application domains
 Avionics, automotive, ...

