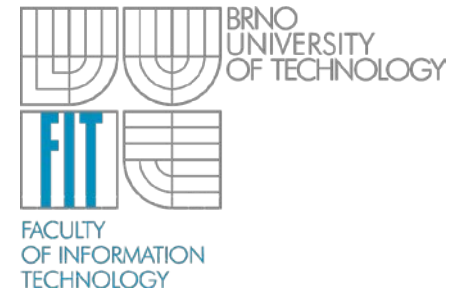


PNagent: a Framework for Modelling BDI Agents using Object Oriented Petri Nets

Z. Mazal, R. Kočí, V. Janoušek and F. Zbořil

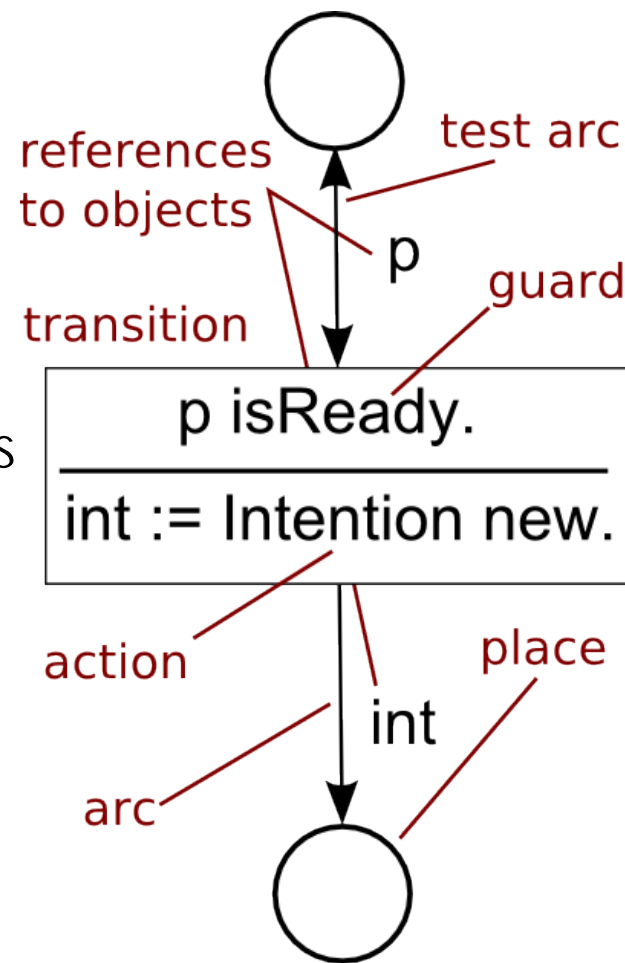
Brno University of Technology, Faculty of Information Technology
Božetěchova 2, 612 66 Brno
mazal@fit.vutbr.cz

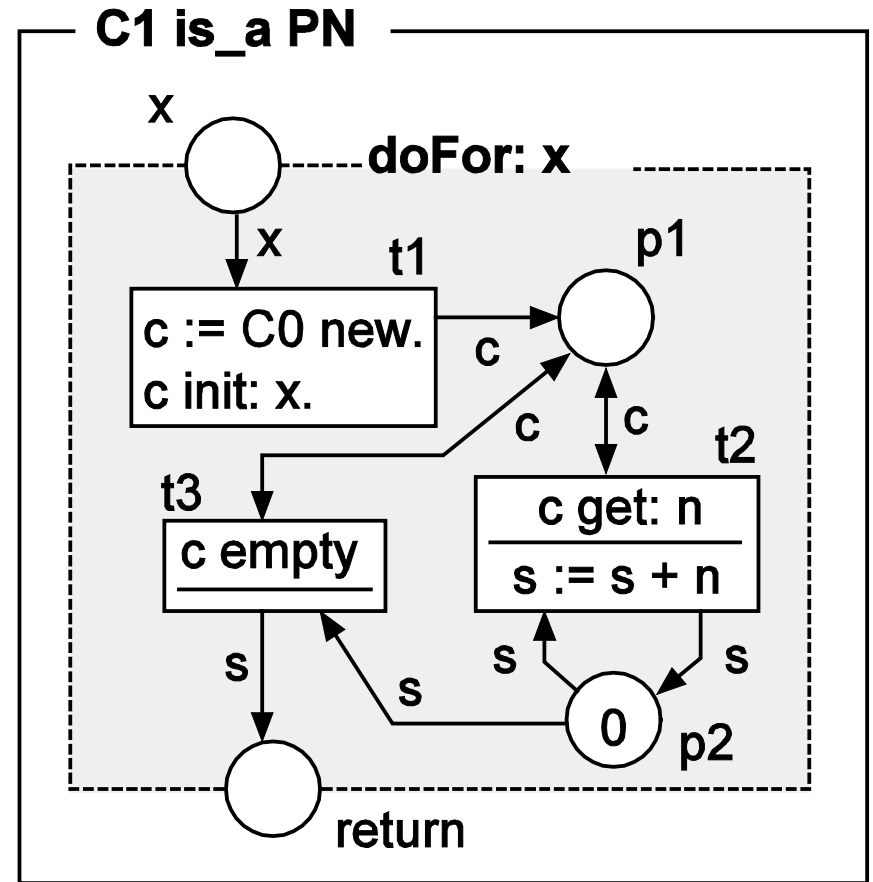
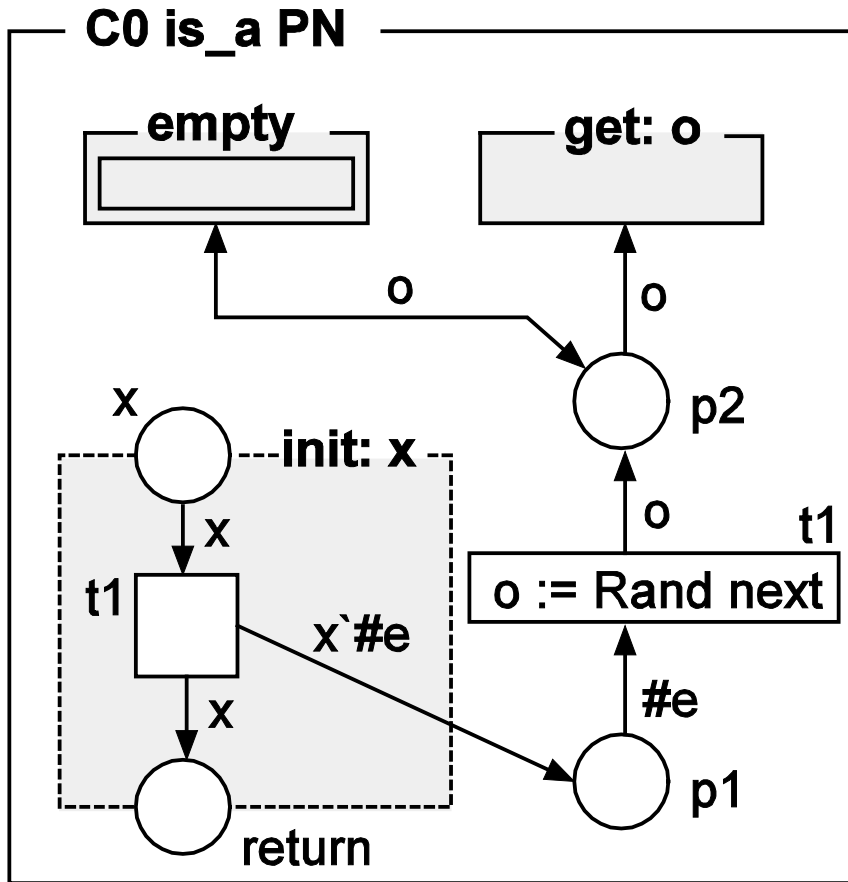


- Object Oriented Petri Nets
 - Basics
 - Example
 - PNtalk
- Deliberative Agents
 - BDI Architecture
- PNagent
 - Vision
 - Main Concepts
 - Belief Representation
 - Plans, Intentions
 - Plan Example
 - Events, Goals
 - Interpret
 - Infrastructure
 - Applications
- Conclusions

- Basic PT Petri nets (C. A. Petri ~1960)
 - Graphical modelling tool, mathematical foundations
 - Problems with description of large models
 - No means for hierarchical modelling
 - Several extensions proposed – High level Petri nets
 - Coloured Petri nets, Hierarchical coloured Petri nets
- Object oriented Petri nets (V. Janoušek ~1993)
 - PN: formal and intuitive description of the model
 - OO: model structure and dynamic net invocation
 - Smalltalk-like style – class based, single inheritance, message passing, variables contain references
 - Concurrency – active objects, offer services

- Places – hold tokens
- Transitions = guard and action
- Nets = places and transitions connected by arcs
 - Tokens are references to objects
 - Arcs inscribed by multiset expressions
 - Variable bindings
- Objects = object net, method nets, synchronous ports and negative predicates
 - Method nets can access object net places





- Framework for simulation based development
 - Based on OOPN formalism – simulation, verification
 - Model as a part of target application
- Implemented in Squeak Smalltalk
 - Interoperability: access to Smalltalk objects from OOPN objects and vice versa
- Meta-object protocol
 - Access to each PNtalk object programmatically
 - Modification of classes and instances during runtime
- Wrapped into DEVS formalism
 - As an atomic component





- Artificial intelligent agents
 - Autonomous, reactive, proactive, social
- Classification by inner architecture:
 - Reactive (Subsumption architecture, etc.)
 - Percepts mapped directly to actions
 - No explicit internal representation of the world
 - **Deliberative** (PRS, dMars, AOP, etc.)
 - Classical AI approach
 - Contain symbolic representation of environment
 - Planning, reasoning, etc.
 - Hybrid (InteRRaP, etc.)
 - Combination of both approaches

- Based on the work of philosopher M. Bratman
 - Models of behaviour based on utility maximization
 - B = Beliefs, D = Desires (Goals), I = Intentions + Plans (procedural knowledge)
- Formalisation
 - Rao and Georgeff – multimodal logics BDI CTL
 - Reiter, Levesque – situation calculus
- Implementations
 - Vague connection with theory
 - Use modalities as data structures
 - Programming language, world representation – FOL
 - IRMA, PRS, dMARS, JACK, JAM, JADEX,...

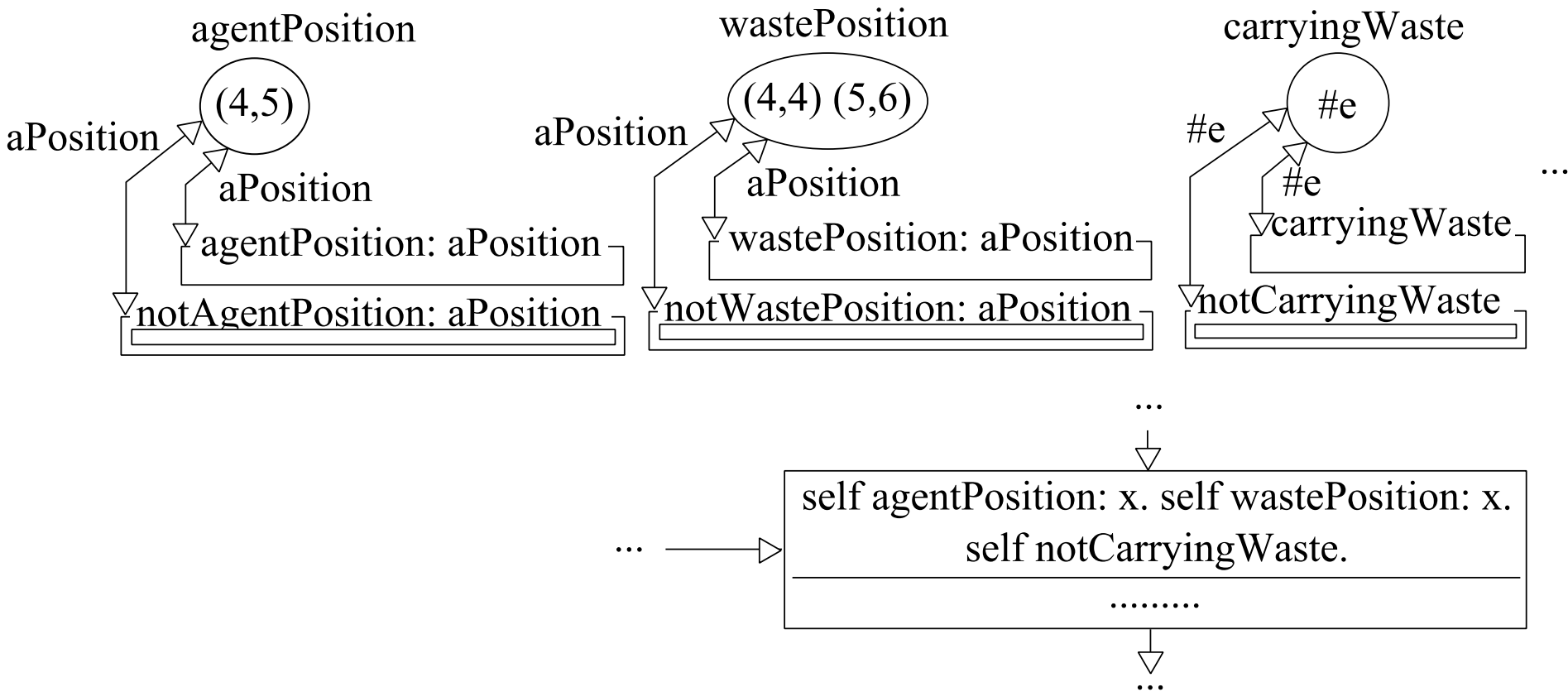
- Most current implementations – loose connection with theory
 - OOPNs provide both and in a graphical intuitive way
- One means for description of the whole system
 - Representation of agents' beliefs as PN instead of FOL
- Agent systems are complex and strongly parallel
 - Parallelism inside the agent
 - Verification – communication protocols, etc.
- Experiments with both architecture and applications
 - Model based design
 - Open system, embedded in DEVS formalism

- Provides base classes for BDI agents development
 - Interpreter (agent class)
 - Abstract plans
 - Internal events, goals, intentions
 - Platform (environment)
 - Communication (based on FIPA ACL language)
- Application programmer adds:
 - Representation of the world – agent's beliefs
 - Agent capabilities
 - Concrete plans
 - External events, goals
 - Environment physics (or connection to real world)

- Example: cleaner world
 - Classical test application
 - Robot cleans 2D grid
 - Fields contain waste, bin or agent
 - Waste appears randomly
- Representation in FOL
 - $wastePosition(5,6)$.
 - $wastePosition(4,4)$.
 - $binPosition(2,3)$.
 - $agentPosition(4,5)$.
 - $carryingWaste$.
- Querying beliefs:
 - $canDropWaste :- binPosition(X,Y), agentPosition(X, Y), carryingWaste$.

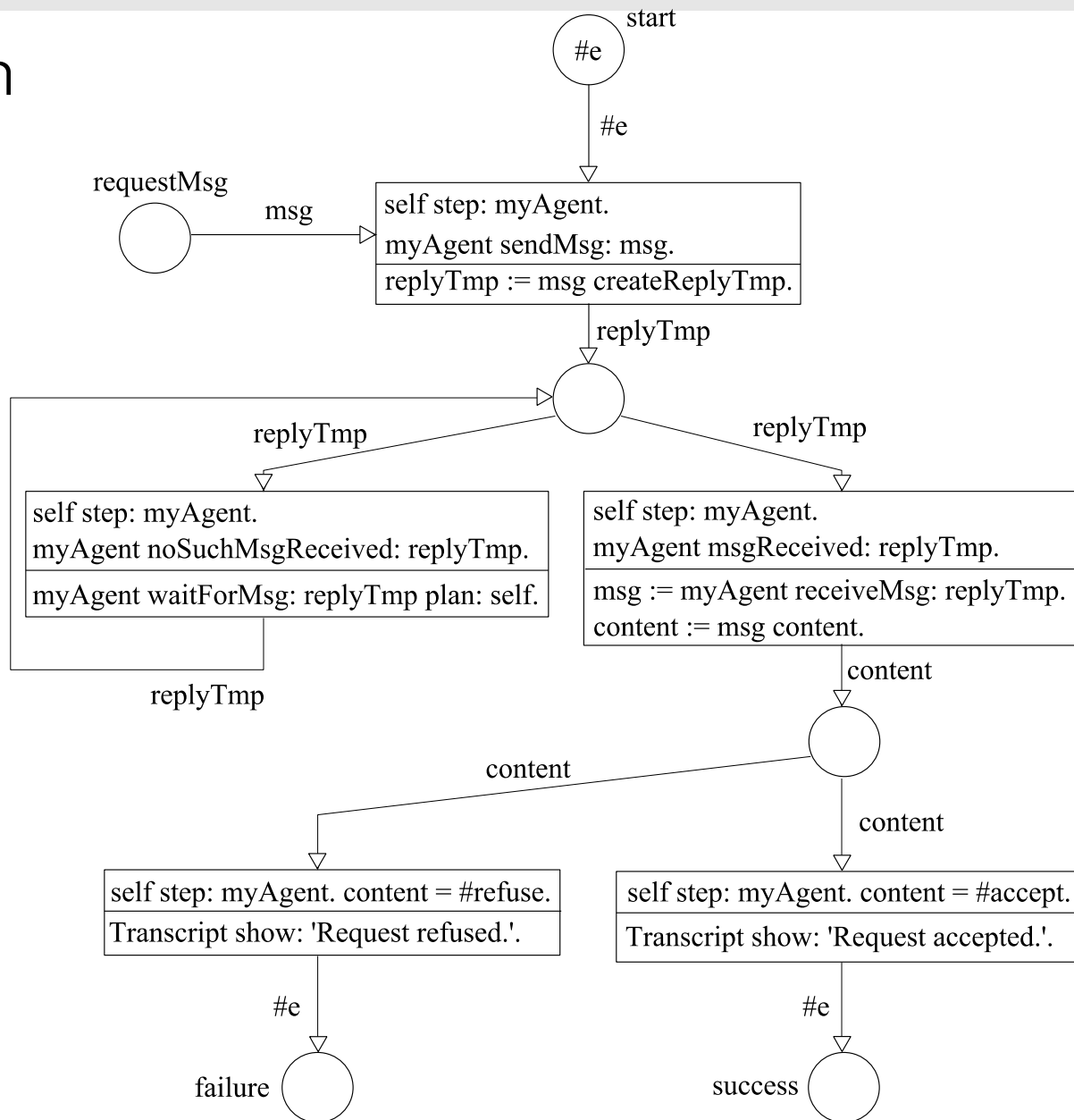
	1	2	3	4	5	6
1						
2						
3						
4						
5						
6						

- PNagent
 - Places = predicate types
 - Tokens = individual predicates
 - Synchronous ports / negative predicates = queries



- Plans = agents' procedural knowledge
 - No first order planning, prepared by application programmer in design time, partially specified (unbound variables)
- Framework provides abstract plan classes
 - PlanTemplate – means for creating plan instances
 - AP adds: triggering event and goal templates
 - PlanInstance – lifecycle, event handling, identification, ...
 - Provides places start, success, failure
 - Plan concretization
 - By inheritance, AP adds places and transitions for a concrete application
 - Each transition = atomic interpretation step
 - CW: Collect waste plan, Move to waste plan
- Intentions = stacks of plan instances (subplans)

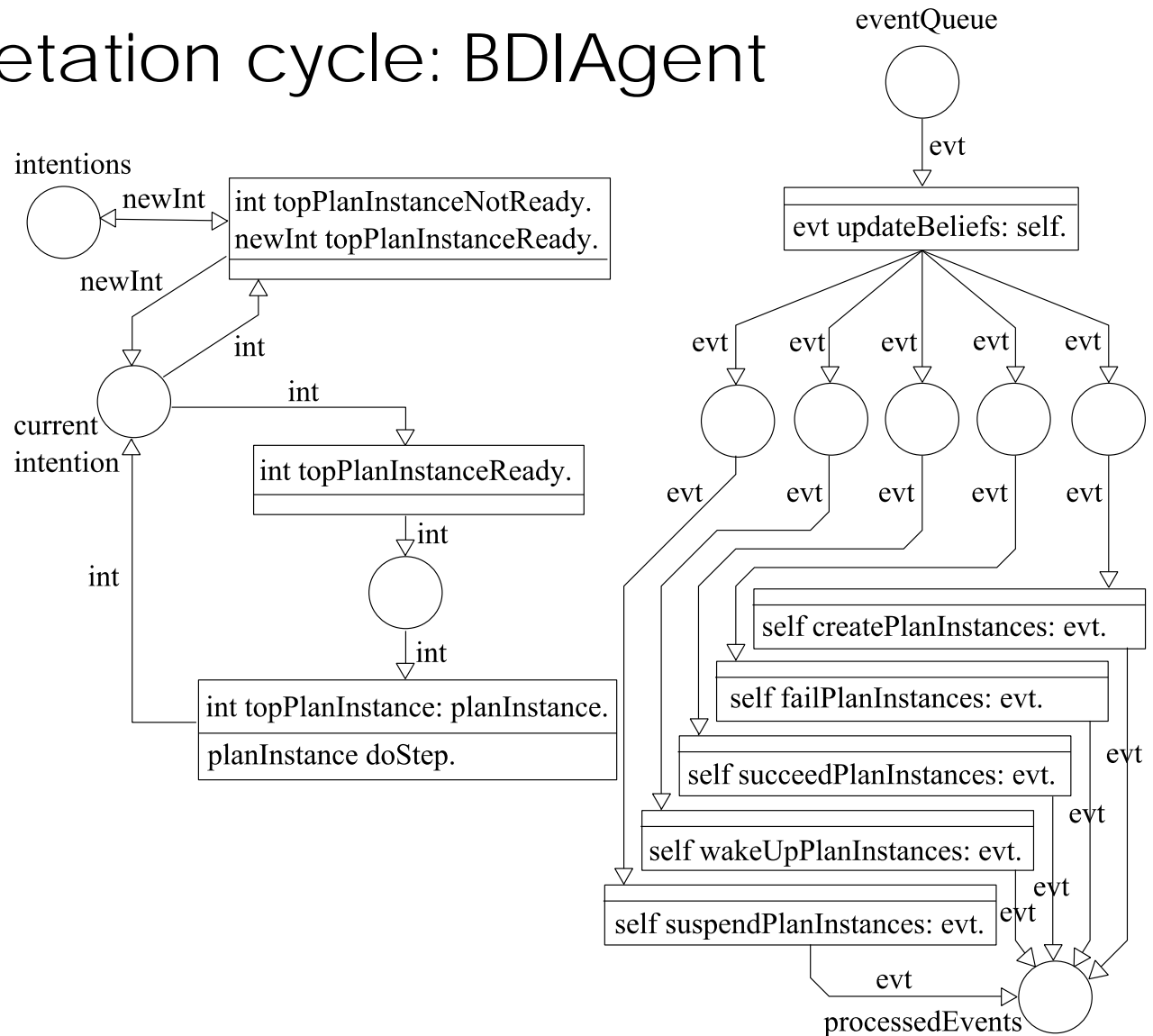
- Only application specific part
- Simple message sending and waiting for answer
- Plan lifecycle – changes triggered by events



- All changes in the system relevant for the agent
 - Internal, external
 - Triggering, suspending, waking up plans, etc.
 - Updating agents' mental states
- Framework provides internal events
 - Goal achieved, plan instance failed, new sub-goal posted..
- AP creates application specific events
 - CW: new waste appeared, bin moved...
- Abstract class Event
 - Unifications (analogous to beliefs representation approach).
- Goals = persistent events

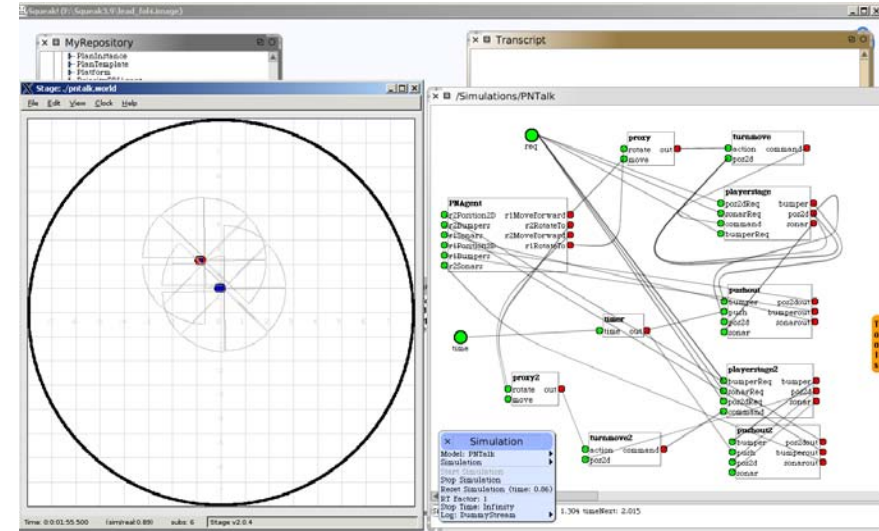
- Main interpretation cycle: BDI Agent

- Dispatch events
- Interpret intention structure



- Platform class – represents agents' environment
 - Generates events
 - Implements environment physics / connects the model to real environment
 - Provides services (as defined by FIPA)
 - Naming, white pages, message transport..
- Communication
 - Implemented as a layer above BDI core
 - Based on message passing
 - Subset of FIPA ACL language currently supported

- Common test „Toy apps“
 - Cleaner world
 - Blocks world
 - Contract Net Protocol
- Mobile robot control
 - Leader-follower formation establishing
 - Multi paradigmatic modelling
 - PNagent for „brain“
 - DEVS for communication
 - Player/Stage for robot behaviour simulation
 - Extensive use of plans & communication
 - Problems with model efficiency
 - PNTalk optimizations needed



- PNagent
 - A framework for modelling BDI agents by OOPNs
 - Visual modelling
 - High level formal description
 - Provides base classes
 - AP inherits classes, adds application specific part
 - Possibilities of verification, connections to other paradigms
 - Experiments with various approaches to BDI
 - Model based design
 - FIPA-compatible applications
 - Test use in mobile robots control

This work was supported by:

- Czech Grant Agency, under the contracts:
 - GA102/05/H050
 - GP102/07/P306
 - GP102/07/P431
 - GA102/07/0322
- Ministry of Education, Youth and Sports under the contract:
 - MSM 0021630528

Thank you for your attention.