16th Workshop on Software Engineering Education,
Jahorina, Bosnia, August 21-27, 2016

FROM BUSINESS PROCESS MODELS TO AGENT PROGRAMS

Amelia Bădică*, Costin Bădică*,
Florin Leon**, Ion Buligiu*

*University of Craiova, Romania
**Gheorghe Asachi Technical University, Iasi, Romania

TALK OUTLINE

- Introduction and Motivation
- Role Activity Diagrams - RAD
- AgentSpeak(L) and Jason
- Mapping RAD to Jason
- Conclusions

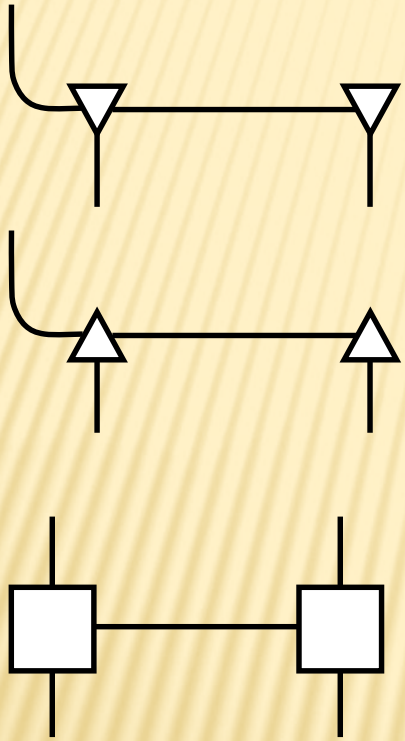
INTRODUCTION

- *Business organization* = society of cooperating agents that are collectively carrying out a set of business activities (business process) in order to meet business objectives

RESEARCH GOAL

- *Q: Can we apply state-of-the-art AOP languages for modeling and enactment of business processes ?*

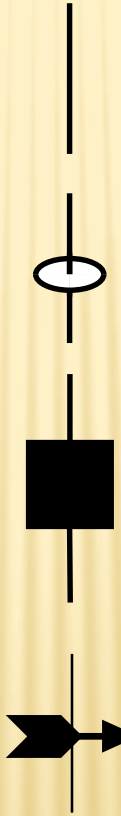
RAD NOTATION



Case refinement

Part refinement

Interaction



State

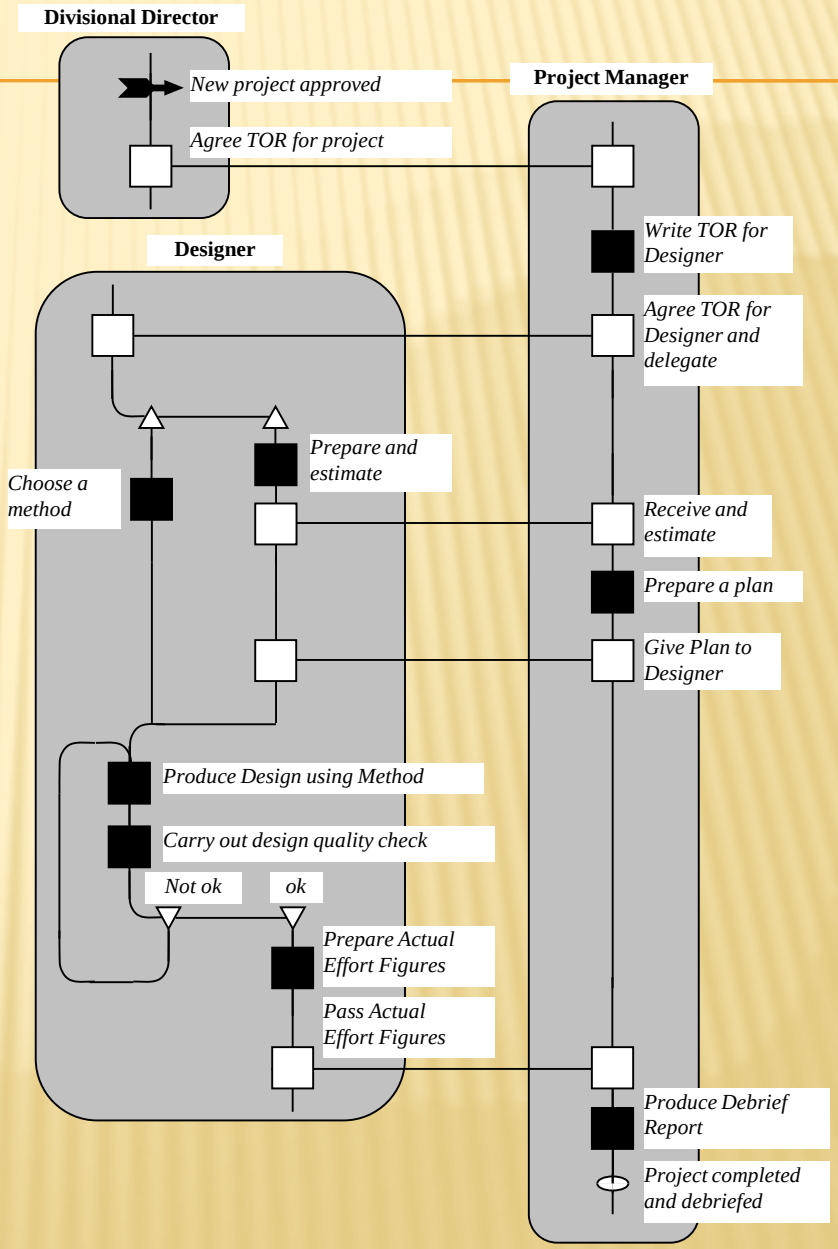
State description

Activity

External event

EXAMPLE RAD

- Three roles:
  - *Divisional Director*
  - *Project Manager*
  - *Designer*



AGENT ORIENTED PROGRAMMING

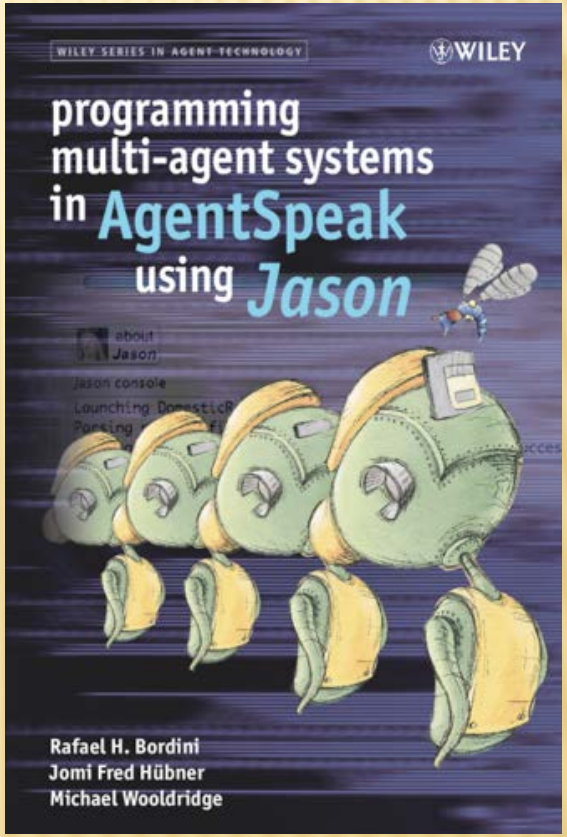
- Historically, AOP was firstly proposed more than 20 years ago (Shoham, 1990) as:

  A new programming paradigm, one based on cognitive and societal view of computation

- Many models and implementations of AOP.

AGENTSPEAK(L) AND JASON

- AgentSpeak(L) is an abstract AOP language, introduced by Rao in 1996.
- Jason is an implementation, as well as an extension of AgentSpeak(L), based on Java.



AGENT BELIEF BASE

- The agent *belief base* is composed of Prolog-like facts and rules and it represents the "agent memory". The belief base is continuously updated during the agent reasoning cycle.

- Beliefs are similar to logic programming rules:

  h :- b_1 & ... & b_k $\quad\quad$ $k \geq 0$

AGENT PLANS

- *Plans* define the *agent know-how*. A plan is:

  e : *c* <- *b*

- A plan is composed of event, context and body:
  - The *event* triggers the plan if *context* is matching beliefs.
  - The plan *body* is a sequence of actions: *internal actions*, *external actions*, and *goals*.

AGENT GOALS

- The agent is working towards the reaching of achievement goals !G.
- Test goals ?G are used to retrieve information from the belief base using Prolog-like reasoning and unification.

EVENTS

- Events trigger plan execution.

  +belief

  -belief

  +!goal

  -!goal

  +?goal

  -?goal

AGENT REASONING ENGINE

- Gets *perception* and *communication*
- Updates beliefs
- Selects an event
- Selects an applicable plan (*option*) and adds it to the agenda
- Selects an *intention* (a stack of partially instantiated plans) for execution from agenda
- Executes the *next step* of the top of the currently selected intention.

RAD SYNTAX

- *Role model* = bipartite directed graph $\langle A \cup S, E \rangle$ s.t.:
  - A = finite set of action nodes
  - S = finite set of state nodes
  - E = finite set of arcs, $E \subseteq (A \times S) \cup (S \times A)$
- *Action nodes*:
  - *Activities* (including *interactions*)
  - *External events*
  - Conditions of *case refinements*
  - *Part refinements:* originating points (forks), joining points (joins)
- *State nodes* => process states represented by *state lines*

RAD SEMANTICS

- *Current state* = "tokens" assigned to state lines.
- *State transitions* = tokens flowing from a state line to a successor state line
- *Action* "consumes" and "produces" tokens.
- This behavior closely resembles Petri nets.

MAPPING OUTLINE

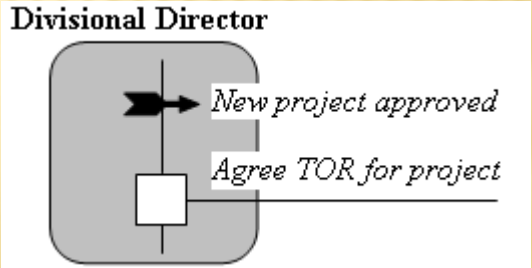
- *RAD role* => Jason agent. Example: *d*, *dd* and *pm* agents.
- *State* => agent belief base. Example: *dd0*, *dd1*, *dd2*, *d0*, *d1*, …
- *Action* => agent plan:

```
+!advance : current state <-
    state update
    do activity
    !!advance.
```

- *RAD process* => multi-agent program

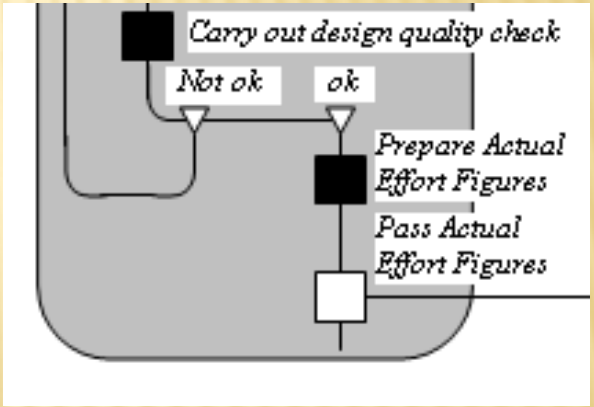
MAPPING STATES AND ACTIVITIES

```
+!advance : dd0 <-
   -dd0;
   ?task("New project approved");
   +dd1;
   !!advance.
+!advance : start <-
   -start;
   ?task("Starting ...");
   +dd0;
   !!advance.
+!advance : dd2 <-
   -dd2;
   ?task("Stop").
```



MAPPING CASE REFINEMENTS

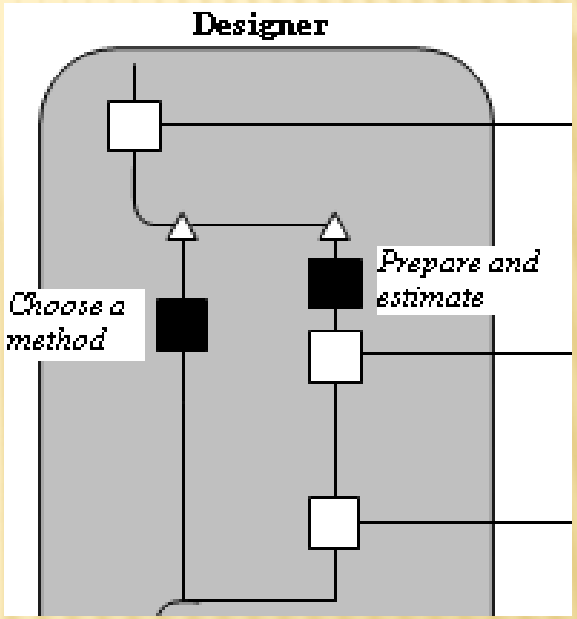
```
+!advance : d9 <-
   -d9;
   ?task("Carry out design quality check");
   rad.choice([ok,nok],Result)
   +d10(Result);
   !!advance.
+!advance : d10(nok) <-
   -d10(nok);
   ?task("Design quality not ok");
   +d8;
   !!advance.
```



MAPPING PART REFINEMENTS

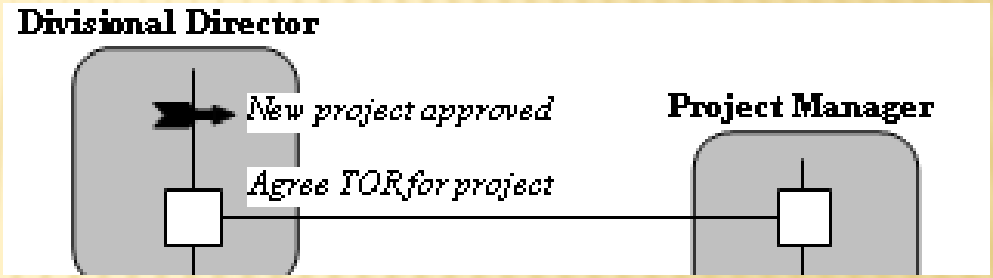
```
+!advance : d1 <-
    -d1;
    ?task("Fork parallel threads");
    +d2;
    +d3;
    !!advance.

+!advance : d6 & d7 <-
    -d6;
    -d7;
    ?task("Join parallel threads");
    +d8;
    !!advance.
```



MAPPING INTERACTIONS

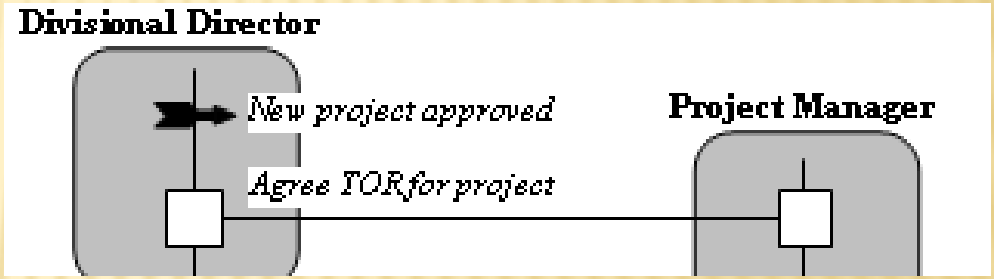
```
+!advance : start <-
   -start;
   ?task("Starting ...");
   +pm0;
   .send(dd,tell,pm0);
   !!advance.
+!advance : dd1 & pm0 <-
   -dd1;
   -pm0[source(pm)];
   ?task("Agree TOR for project");
   +dd2;
   !!advance.
```



CONTINGENCY PLAN

```
+!advance : dd1 & pm0 <-
   -dd1;
   -pm0[source(pm)];
   ?task("Agree TOR for project");
   +dd2;
   !!advance.


-!advance : true.
+pm0 : true <- !!advance.
```



MAPPING SUMMARY

- *One proactive plan* for agent starting and *one proactive plan* for agent stopping.
- *A proactive plan* for each action node.
- *One contingency plan* to deal with shared beliefs that have not yet arrived from peer agents.
- *A reactive plan* for handling the arrival of each shared belief from peer agents.

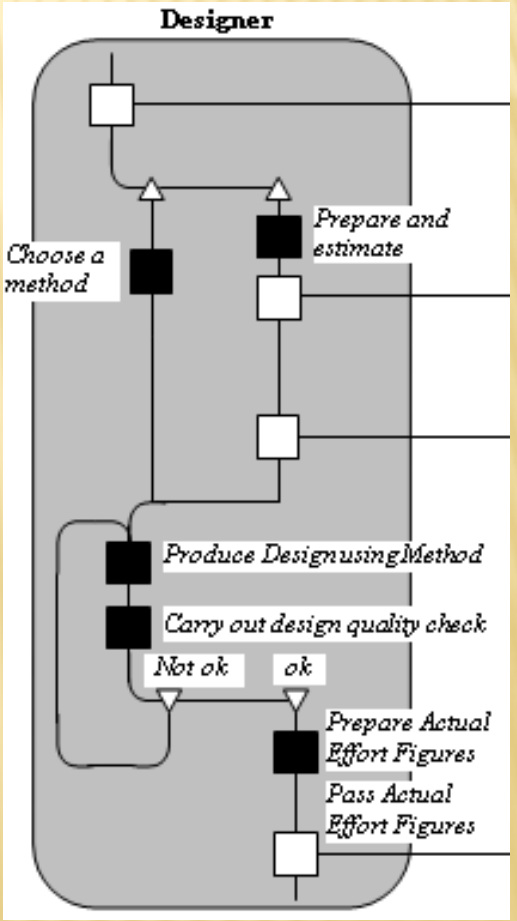
KNOWLEDGE-BASED BUSINESS AGENTS

- Generic *knowledge-based business agent architecture* – $\mathcal{KB}^2\mathcal{A}^2$.
  - A *knowledge base* that captures the operational knowledge of the agent according to a given RAD business process.
  - A *set of template plans* that capture the generic behavioral patterns of business agents.

KNOWLEDGE BASE

rule(*Action*, *StateIn*, *StateOut*).

```
rule(task("Fork parallel threads"), [d1],
  [d2,d3]).
rule(task("Prepare and estimate"), [d3],
  [d4,s(pm,d4)]).
rule(task("Receive and estimate"),
  [d4,r(pm,pm3)],[d5,s(pm,d5)]).
...
rule(choice("Carry out design quality check",
  [ok,nok]), [d9], [[ok,d10(ok)],
  [nok, d10(nok)]]).
```



PROACTIVE TEMPLATE PLANS

- Proactive template plans for handling agent actions that do (not) represent case refinements:

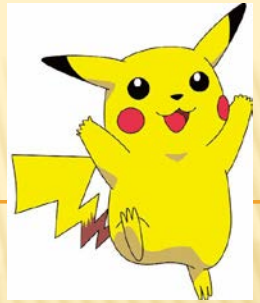
```
@plan_task[atomic,all_unifs] +!advance :
   rule(task(Name),In,Out) & match(In) <-
   !remove(In);
   ?task(Name);
   !append(Out);
   !continue(Out).
```

REACTIVE TEMPLATE PLAN

- Reactive template plan for handling shared belief assertions from peer agents.

```
@plan_wake +X :
   rule(_,In,_) & member(r(_,X),In) <-
   !!advance.
```

- $\mathcal{KB}^2\mathcal{A}^2$ includes also the contingency plan introduced in the previous section.



CONCLUSIONS

- We *introduced a mapping* of RAD business processes expressed => Jason AOP language.
- We proposed a new architectural model *knowledge-based business agents* entitled $\mathcal{KB}^2\mathcal{A}^2$.
- We provided an *intuitive presentation* of $\mathcal{KB}^2\mathcal{A}^2$ by example.
- Future works:
  - *to provide formal support* to mathematically assess the mapping correctness
  - *to develop a software tool* to assist knowledge engineers in creating and configuring $\mathcal{KB}^2\mathcal{A}^2$ agents.


THANK YOU
THANK YOU
memegenerator.net