Formal Goal-Oriented Development of Resilient Multi-Agent Systems in Event-B

Inna Pereverzeva**, Elena Troubitsyna*, Linas Laibinis*

*Åbo Akademi University, Turku, Finland

**Turku Centre for Computer Science, Turku, Finland

Ada Europe, Stokholm, June 14th, 2012
Motivation and Our Approach

Our Formal Framework – Event B

Formalising Goal-Oriented Development in Event-B

Case Study

Conclusions
Goal-Oriented Development

- **Goal-Oriented Requirements Engineering** – a useful framework that facilitates structuring complex requirements

- Goals – objectives that a system should achieve

- Changes in the system environment or system faults might hinder achieving the desired goals

- Need for techniques to ensure system resilience
Our approach

- We aim:
  - to formalise the notion of goal and goal achievement
  - to provide a set of patterns to facilitate formal goal-oriented development (of multi-agent systems)

- Application of patterns is illustrated by a case study – an autonomous multi-robotic system
Our Formal Framework: Event-B

- Even-B – a state based formal approach based on
  - correct-by-construction development, and
  - formal verification by theorem proving

- Designed to model and reason about distributed and reactive systems

- Gradual elaboration on the system data and dynamics by correctness-preserving steps – refinements

- Tool support: the Rodin platform
Event-B Model

- Usually consists of two components: context (static) and machine (dynamic)
- Context: description of types and constants
- Machine: variables, events and preserved properties (invariants)

**Context C**
- Sets
  - ...
- Constants
  - ...
- Axioms
  - ...
end

**Machine M**
- Sees C
- Variables
  - ...
- Invariants
  - ...
- Events
  - Event\(_1\) =
    - when condition then ... end
  - ...
end
An Event-B model is a specification pattern if it contains generic (abstract, underspecified) types, constants and variables, which can be instantiated by concrete counterparts during development.

The context defines abstract constraints that should be satisfied during instantiation.

The invariant properties, once proven for a pattern, are true for any valid instantiation.
Let us have a collection of specification patterns $P_1, \ldots, P_n$ such that

$P_1$ is refined by $P_2$ is refined by $\ldots$ is refined by $P_n$

Generic model development, with all generic data structures of $P_1, \ldots, P_n$ as its parameters

$P_2, \ldots, P_n$ can be also considered as refinement patterns since they define generic model transformations
Goal Modelling Pattern

- **Purpose**: to abstractly define system goals and its achieving
- 2 generic parameters: **GSTATE** and **Goal**
- **GSTATE** – the system state space
- **Goal** – the given system goals
- 2 constraints in the context:

\[ \text{Goal} \neq \emptyset \text{ and } \text{Goal} \subset \text{GSTATE} \]
The whole execution is modelled as an iterative event
Reaching\_Goal

The event status is *anticipated*: goal reachability is postulated rather than proved. Obligation to prove event convergence

**Machine** M\_AGM

**Variables** gstate

... 

Reaching\_Goal ≜

**status** anticipated

when

\[ gstate \in GSTATE \setminus \text{Goal} \]

then

\[ gstate \in GSTATE \]

end

**gstate** ∈ GSTATE – the current state of the system goal

**gstate** :∈ GSTATE

non-deterministic assignment from GSTATE
**Purpose:** to decompose the high-level system goals into a set of subgoals

**Subgoals** define intermediate stages of achieving the main goal

**We have to ensure that high-level goals remain achievable:**
- the relation between goals and subgoals
- goals reachability is achieved via reachability of subgoals
• System goal **Goal** is achieved by reaching three subgoals
  **SubGoal1**, **SubGoal2**, **SubGoal3**

• The state space **GSTATE** is partitioned into three
  **SG_STATE1**, **SG_STATE2**, **SG_STATE3**

• \( \text{State\_map} \in \text{SG\_STATE1} \times \text{SG\_STATE2} \times \text{SG\_STATE3} \mapsto \text{GSTATE} \)

• \( \forall \text{sg}1, \text{sg}2, \text{sg}3 \cdot \text{sg}1 \in \text{Subgoal1} \land \text{sg}2 \in \text{Subgoal2} \land \text{sg}3 \in \text{Subgoal3} \)
  \( \Leftrightarrow \text{State\_map}(\text{sg}1 \mapsto \text{sg}2 \mapsto \text{sg}3) \in \text{Goal} \)

  (the **main goal** is reached \( \Leftrightarrow \) **all three subgoals** are reached)
The abstract variable \textit{gstate} is refined by the new variables:

\[ gstate_i \in SG\_STATE_i, \ i \in 1..3 \]

- model the state of the subgoals

\textbf{gluing invariant:}

\[ \text{gstate} = \text{State\_map}(gstate_1 \mapsto gstate_2 \mapsto gstate_3) \]

\begin{verbatim}
Machine M_GD
Reaching_SubGoal_i ≜ refines Reaching_Goal
  status anticipated
  when
    gstate_i \in SG\_STATE_i \setminus \text{Subgoal}_i
  then
    gstate_i :\in SG\_STATE_i
  end
...
\end{verbatim}
Refinement pattern, i.e., refines the $M_{AGM}$ pattern

The pattern can be repeatedly applied to build the goal hierarchy

We assume that subgoals are independent of each other

Moreover, while a certain subgoal is reached, it remains reached, i.e., the system always progresses towards achieving its goals. This can be expressed as a stability property:

\[
Stable(P) \Leftrightarrow \text{“once } P \text{ becomes true, it remains true”}
\]
Further Development

- **Abstract Goal Modelling Pattern and Goal Decomposition Pattern**  \(\Rightarrow\) allow to specify **goals/subgoals**

- In MAS, particular (sub)goals are usually achieved by system agents

- **New Agent Modelling Pattern**  \(\Rightarrow\)
  - introduces a representation of **agents**
  - defines **eligible agents** – the agents that are capable of achieving a certain subgoal
Purpose: to model agents and abstractly define agent eligibility

AGENTS – the set of the system agents

EL_AG1, EL_AG2, and EL_AG3 – the eligible agents for each subgoal

 elig_i ⊆ EL_AG_i, i ∈ 1..3 – the dynamic sets of eligible agents

System invariant: elig_i ≠ ∅, i ∈ 1..3
Success_in_Reaching_SubGoal_i \triangleq
\text{refines Reaching}_{\text{SubGoal}_i}
\text{status } \textit{convergent}
\text{any } ag
\text{when}
\quad gstate_i \in SG_{STATE_i} \setminus \text{Subgoal}_i
\quad ag \in \text{elig}_i
\text{then}
\quad gstate_i :\in \text{Subgoal}_i
\text{end}

\text{Fail_in_Reaching}_{\text{SubGoal}_i} \triangleq
\text{refines Reaching}_{\text{SubGoal}_i}
\text{status } \textit{convergent}
\text{any } ag
\text{when}
\quad gstate_i \in SG_{STATE_i} \setminus \text{Subgoal}_i
\quad ag \in \text{elig}_i \land \text{card(} \text{elig}_i \text{)} > 1
\text{then}
\quad gstate_i :\in SG_{STATE_i} \setminus \text{Subgoal}_i
\quad \text{elig}_i := \text{elig}_i \setminus \{ag\}
\text{end}

\textbf{Note:} the event status changed to \textit{convergent}
Restriction: at least one agent associated with the subgoal remains operational: \(\text{card}(elig_i) > 1, \ i \in 1..3\)

This assumption allows us to change the event status from anticipated to convergent. I.e., for each subgoal, the process of reaching it eventually terminates.

To prove the convergence, we define the following variant expression:

\[\text{card}(elig_1) + \text{card}(elig_2) + \text{card}(elig_3)\]

The constraint to have at least one operational agent associated with our model can be dropped by probabilistic modelling of goal reachability (future work)
Purpose: to define agent types and agent statuses

\[ \forall ag \cdot ag \in EL_AG \iff atype(ag) = TYPE_i, \ i \in 1..3 \]

Dynamic agent status: \( \text{astatus} \in AGENTS \rightarrow AG\_STATUS \)

Refine the abstract variables \( elig_i, i \in 1..3 \):

\[ elig_i = \{ a | a \in AGENTS \land atype(a) = TYPE_i \land astatus(a) = OK \} \]
Outline

- **Goal Modelling Pattern:**
  abstractly define system goals and its achieving

- **Goal Decomposition Pattern:**
  decompose the high-level system goals into a set of subgoals

- **Agent Modelling Pattern:**
  introduce system agents and agent eligibility

- **Agent Refinement Pattern:**
  introduce agent attributes: statuses, types
The overall goal: to coordinate a number of mobile robots to clean a certain area.

The area is divided in zones, which further divided in sectors.

Each zone has a base station – a static computing and communicating device – that coordinates the cleaning of a zone by giving assignments to robots.

Robots may fail; Another operative robot is then given the failed task.
Multi-Robotic System: Abstract Model

- The state space $GSTATE \equiv BOOL$
- The system goal $Goal \equiv \{TRUE\} –$ the whole territory is cleaned
- The process of cleaning the territory:

  $CleaningTerritory \equiv$
  
  $\text{status \ anticipated}$
  
  $\text{when}$
  
  $\text{completed} = FALSE$
  
  $\text{then}$
  
  $\text{completed} \in BOOL$
  
  $\text{end}$

$completed \in BOOL –$ the current state of the system goal
The territory is divided into n zones.

The current status for every zone:

\[ \text{zone\_completed} \in 1..n \rightarrow BOOL \]

Gluing invariant:

\[ \text{completed} = TRUE \iff \text{zone\_completed}[1..n] = \{ TRUE \} \]
CleaningZone \equiv \text{refines} \text{CleaningTerritory}

\begin{align*}
\text{status} & \textit{anticipated} \\
\text{any} & \text{zone, zone\_result} \\
\text{when} & \\
\text{zone} & \in 1..n \\
zone\_completed(\text{zone}) & = \text{FALSE} \\
zone\_result & \in \text{BOOL} \\
\text{then} & \\
zone\_completed(\text{zone}) & := \text{zone\_result} \\
\text{end}
\end{align*}
Every zone is divided into k sectors

The current status for every sectors:

\[ \text{sector}\_\text{completed} \in 1..n \rightarrow (1..k \rightarrow BOOL) \]

Gluing invariant:

\[ \forall \ zone \cdot \text{zone} \in 1..n \Rightarrow (\text{zone}\_\text{completed}(\text{zone}) = \text{TRUE} \iff \text{sector}\_\text{completed}(\text{zone})[1..k] = \{\text{TRUE}\}) \]
Introduce the abstract set \( \text{AGENTS} \)

\( \text{ELIG} \subseteq \text{AGENTS} \) – the eligible agents for executing the tasks

\( \text{elig} \neq \emptyset \) – the dynamic set of eligible agents

The abstract event \( \text{CleaningSector} \) is refined by two events \( \text{SuccessCleaningSector} \) and \( \text{FailCleaningSector} \)
Two types of agents – robots and base stations

\[ \text{AGENT} = \text{RB} \cup \text{BS} \]

Set of eligible agents is represented by robots: \( \text{ELIG} = \text{RB} \)

Dynamic robot status:

\[ \text{astatus} \in \text{RB} \rightarrow \{\text{active}, \text{failed}\} \]

Dynamic set of eligible agents:

\[ \text{elig} = \{a \mid \text{atype}(a) = \text{RB} \land \text{astatus}(a) = \text{active}\} \]
Future work

- Pattern list is far from complete ⇒ to create extended library of patterns
- Integrate stochastic reasoning in our formal development
- Experiment with different schemes for goal decomposition and dynamic goal reallocation (dynamic system reconfiguration)
Thank You!

Questions?