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

TUCS

Åbo Akademi

Outline

- 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

TUCS

Åbo Akademi

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
    $\text{Event}_1 \ \hat{=}$
      when condition **then** ... **end**
    ...
end

Specification and Refinement Patterns

- An Event-B model is a *specification pattern* if it is 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, ..., P_n such that

$$P_1 \textbf{ is refined by } P_2 \textbf{ is refined by } \ldots \textbf{ is refined by } P_n$$

- Generic model development, with all generic data structures of P_1, ..., P_n as its parameters

- P_2, ..., P_n can be also considered as *refinement patterns* since they define generic model transformations

- **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:

$$Goal \neq \varnothing \quad \text{and} \quad Goal \subset GSTATE$$

Goal Modelling Pattern

- 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 $\widehat{=}$
status *anticipated*
when
$gstate \in GSTATE \setminus Goal$
then
$gstate :\in GSTATE$
end

gstate $\in$ *GSTATE* – the current state of the system goal

gstate $:\in$ *GSTATE* non-deterministic assignment from *GSTATE*

Goal Decomposition Pattern

- **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

TUCS

Åbo Akademi

Goal Decomposition Pattern (cont.)

- 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**
- $State\_map \in SG\_STATE1 \times SG\_STATE2 \times SG\_STATE3 \rightarrowtail\!\!\!\!\twoheadrightarrow GSTATE$
- $\forall sg1, sg2, sg3 \cdot \; sg1 \in Subgoal1 \wedge sg2 \in Subgoal2 \wedge sg3 \in Subgoal3$
$$\Leftrightarrow State\_map(sg1 \mapsto sg2 \mapsto sg3) \in Goal$$

  (the **main goal** is reached $\Leftrightarrow$ **all three subgoals** are reached)

Goal Decomposition Pattern (cont.)

- The abstract variable **gstate** is refined by the new variables:

$$gstate_i \in SG\_STATE_i, \;\; i \in 1..3$$

– model the state of the subgoals

- **gluing invariant:**

$$gstate = State\_map(gstate1 \mapsto gstate2 \mapsto gstate3)$$

Machine M_GD
Reaching_SubGoal$_i$ $\widehat{=}$ **refines** Reaching_Goal
status *anticipated*
when
$gstate_i \in SG\_STATE_i \setminus Subgoal_i$
then
$gstate_i :\in SG\_STATE_i$
end
...

TUCS

Åbo Akademi

Goal Decomposition Pattern (cont.)

- 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:

$$Stable(P) \;\Leftrightarrow\; \text{“}\textit{once } P \textit{ becomes true, it remains true}\text{”}$$

- Abstract Goal Modelling Pattern and Goal Decomposition Pattern $\rightsquigarrow$ allow to specify **goals/subgoals**
- In MAS, particular (sub)goals are usually achieved by system **agents**
- New Agent Modelling Pattern $\rightsquigarrow$
  - introduces a representation of **agents**
  - defines **eligible agents** – the agents that are capable of achieving a certain subgoal

TUCS

Åbo Akademi

Agent Modelling Pattern

- **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
- $\mathbf{elig}_i \subseteq EL\_AG_i, \;\; i \in 1..3$ – the dynamic sets of eligible agents
- System invariant: $\mathbf{elig}_i \neq \varnothing, \;\; i \in 1..3$

Agent Modelling Pattern (cont.)

$\mathsf{Success\_in\_Reaching\_SubGoal_i} \;\widehat{=}$
refines $\mathsf{Reaching\_SubGoal_i}$
status *convergent*
any ag
when
$gstate_i \in SG\_STATE_i \setminus Subgoal_i$
$ag \in elig_i$
then
$gstate_i :\in Subgoal_i$
end

$\mathsf{Fail\_in\_Reaching\_SubGoal_i} \;\widehat{=}$
refines $\mathsf{Reaching\_SubGoal_i}$
status *convergent*
any ag
when
$gstate_i \in SG\_STATE_i \setminus Subgoal_i$
$ag \in elig_i \wedge card(elig_i) > 1$
then
$gstate_i :\in SG\_STATE_i \setminus Subgoal_i$
$elig_i := elig_i \setminus \{ag\}$
end

- **Note:** the event status changed to convergent

TUCS

Åbo Akademi

Proving Goal Reachability

- Restriction: at least one agent associated with the subgoal remains operational: $card(elig_i) > 1, \ i \in 1..3$
- 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:

$$card(elig_1) \ + \ card(elig_2) \ + \ card(elig_3)$$

- The constraint to have at least one operational agent associated with our model can be dropped by probabilistic modelling of goal reachability (future work)

Agent Refinement Pattern

- **Purpose**: to define agent types and agent statuses

$$\underset{\{OK,KO\}}{AG\_STATUS} \longleftarrow \textbf{Agent} \longrightarrow \underset{\{Type1,\ Type2,\ Type3\}}{AG\_TYPES}$$

- $\forall ag \cdot ag \in EL\_AG_i \Leftrightarrow atype(ag) = TYPE_i,\ \ i \in 1..3$

- Dynamic agent status: **astatus** $\in AGENTS \rightarrow AG\_STATUS$

- Refine the abstract variables $elig_i, i \in 1..3$:

$$elig_i \ = \ \{a | a \in AGENTS \wedge atype(a) = TYPE_i \wedge astatus(a) = OK\}$$

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

Case Study: a Multi-Robotic System

- 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

TUCS

Åbo Akademi

Multi-Robotic System: Abstract Model

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

CleaningTerritory $\widehat{=}$
  status *anticipated*
  when
    $completed = FALSE$
  then
    $completed :\in BOOL$
  end

completed $\in BOOL$ – the current state of the system goal

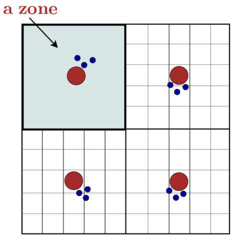
Multi-Robotic System: Representation of Subgoals

- The territory is divided into n zones
- The current status for every zone:

  $\textbf{zone\_completed} \in 1..n \rightarrow BOOL$

- Gluing invariant:

$$completed = TRUE \Leftrightarrow$$
$$zone\_completed[1..n] = \{TRUE\}$$



CleaningZone $\widehat{=}$ **refines** CleaningTerritory
status *anticipated*
any *zone*, *zone_result*
when
$zone \in 1..n$
$zone\_completed(zone) = FALSE$
$zone\_result \in BOOL$
then
$zone\_completed(zone) := zone\_result$
end

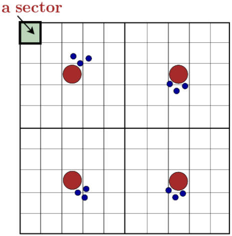
TUCS

Åbo Akademi

Multi-Robotic System: Representation of Subsubgoals

- Every zone is divided into k sectors
- The current status for every sectors:

  sector_completed $\in 1..n \rightarrow (1..k \rightarrow BOOL)$



- Gluing invariant:

$$\forall\ zone \cdot zone \in 1 \mathinner{..} n \Rightarrow (zone\_completed(zone) = TRUE \Leftrightarrow sector\_completed(zone)[1 \mathinner{..} k] = \{TRUE\})$$

Multi-Robotic System: Agent Modelling

- Introduce the abstract set **AGENTS**
- **ELIG** $\subseteq$ **AGENTS** – the eligible agents for executing the tasks
- **elig** $\neq \varnothing$ – the dynamic set of eligible agents
- The abstract event **CleaningSector** is refined by two events **SuccessCleaningSector** and **FailCleaningSector**

TUCS

Åbo Akademi

- Two types of agents – robots and base stations

$$\mathbf{AGENTS} = \mathbf{RB} \cup \mathbf{BS}$$

- Set of eligible agents is represented by robots: $ELIG = RB$
- Dynamic robot status:

$$\mathbf{astatus} \in RB \rightarrow \{active, failed\}$$

- Dynamic set of eligible agents:

$$\mathbf{elig} = \{a \mid atype(a) = RB \wedge astatus(a) = active\}$$

TUCS

Åbo Akademi

Future work

- Pattern list is far from complete $\Rightarrow$ 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)

TUCS

Åbo Akademi

Thank You!

Questions?