http://ansinet.com/itj

ITJ

ISSN 1812-5638

INFORMATION TECHNOLOGY JOURNAL

ANSInet

Asian Network for Scientific Information

308 Lasani Town, Sargodha Road, Faisalabad - Pakistan

Information Technology Journal 6 (8): 1181-1189, 2007
ISSN 1812-5638

Formal Spsecification and Proof of Multi-Agent Applications Using Event B

[1,2]Hong-Jiang Gao, [1,3]Zheng Qin, [4]Lei Lu, [1]Li-Ping Shao and [1]Xing-Chen Heng
[1]Department of Computer Science and Technology, Xi'an Jiaotong University,
Xi'an, Shaanxi, People's Republic of China
[2]School of Computer Science and Technology, Ludong University,
Yantai, Shandong, People's Republic of China
[3]School of Software, Tsinghua University, Beijing, People's Republic of China
[4]Library of Ludong University, Yantai, Shandong, People's Republic of China

Abstract: With increasingly complexity in Multi-Agent Systems (MAS), the problem of their verification and validation is acquiring increasing importance and rigorous design practices are needed in case of critical applications. Event B, which provides an accessible and rigorous development method, is ideal for the formal modelling of reactive systems. In this study, a practical approach for developing flexible and reliable formal specifications of MAS using Event B is described, exemplified on Contract Net Protocol (CNP) in the interaction of MAS and B models generated with evt2b supported by Atelier B are then proven in consistency and correctness. All the concepts of this approach are illustrated by a case study concerning the use of Event B for the modelling and verifying a multi-modal platform associating an intellectualized design system of shape and style in automobile. Moreover, the results of proof and evaluation of present method are presented.

Key words: Multi-agent systems, specification and proof, contract net protocol, roles-based, cases retrieval

INTRODUCTION

With increasingly complexity in MAS, rigorous design practices are needed in case of critical applications. The development process of these systems needs a sound methodology, which ensures quality, consistency and integrity. As a practically effective approach in recent years, formal methods provide systematic and quantifiable approaches to create coherent systems. Application of formal methods to modelling and verifying MAS is a practically effective approach in recent years.

A number of different formal approaches have been applied as candidates to model MAS. Most of them were using the Z method (d'Inverno *et al*., 1997) or Petri Nets (Moldt and Wienberg, 1997), but recently, the B method (Abrial, 1996) emerges as a promising choice (Mermet, 2002; Fadil and Koning, 2005) to model and reason about systems. Furthermore, Event B (Metayer *et al*., 2005), an adaptation of the B method, is more suitable for modelling reactive and distributed systems and verifying the correctness of functional distribution among its different components. Software development in Event B involves abstractly specifying the requirements of the system and then refining these requirements through several steps to create a concrete description of the system that can be translated to programming code. With the aid of B Tools (Atelier B, B-Toolkit, B4free and Click 'n' Prover) and associated UML tool (the U2B translator), Event B is ideal for the formal modelling of reactive systems, providing an accessible and rigorous development method that is suitable for reactive systems (Ball and Butler, 2006). The U2B tool, converting Rational Rose UML Class diagrams (including attached state charts) into the B notation, is a script file that runs within Rational Rose and converts the currently open model to B (Snook and Butler, 2000), which makes modelling with B more appealing to software engineers.

Ongoing research in present Laboratory, an Intellectualized Design System of Shape and Style in Automobile (IDSSSA), is aimed toward developing a distributed cooperative design platform to realize modernized design of complex manufacture. IDSSSA is a closed, dynamic, distributed agent-based system which contains autonomous agents, both stationary and mobile, that cooperate in order to provide an efficient and reliable intellectualized design in a web-based industrial design environment and that comprise design repositories

Corresponding Author: Hong-Jiang Gao, Institute of Software Theory and Technology, Department of Computer Science and Technology, Xi'an Jiaotong University, Xi'an, Shaanxi, 710049, People's Republic of China

(for example, cases repositories, knowledge bases and document repositories, etc.) agents, information retrieval agents, cases-reconstruction agents, supervisor (involved in authority, or document, etc.) agents, operator agents, consultant agents and so on. Present task is to achieve the both instant and reliable retrieval of principal characters of automobile parts, such as fashions (American styles, Japanese styles, etc.), models (Benz or Toyota), the sizes of parts (external length, width and height) and additionally, other customized automobile feature unlisted above-texture of parts, for example. Up to now, a prototype system has been developed and is under trial run. The prototype system can carry out concept-based parts retrieval, characteristic-based parts retrieval and content-based parts retrieval.

In this study, we will model retrieval system as multi-agent systems, using Event B and then verify correctness and consistency of the stepwise development.

OVERALL SPECIFICATION

After revising the E-CARGO model (Zhu *et al*., 2006; Liu and Zhu, 2006), we introduce a new roles-based collaboration model, which supports the agent evolutions, as the infrastructure of present further specification and verification.

Roles-based collaboration multi-agent model and its specification: Any multi-agent system encompasses the following elements (Ferber, 1999): the environment shared by all the system agents, a set of objects located within the environment, that can be created, deleted or modified by the agents, a set of agents that are the active entities of the system, a set of links that join the objects and the agents together and a set of operations that enable to create, manipulate and convert agents and objects. Such agents can play various roles during a communication process. In the meantime, they may belong to groups depending on their role.

Therefore, we modify the model of Zhu *et al*. (2006) and model our multi-agent system using a roles-based collaboration system described as a 9-tuple. $\Sigma ::= \langle O, A, M, R, E, G, P, s0, H \rangle$, where

- O is a set of objects. An object $o :: = \langle n_o, s, F \rangle$, where n_o is the identification of the object, s is a data structure whose values are called states or attributes (or properties), F is a set of the function definitions or implementations;
- A is a set of agents. An agent $a :: = \langle n_a, I_r, I_g \rangle$, where n_a is the identification of the agent, I_r means a set of identifications of roles the agent is taking and I_g denotes a set of identifications of groups the agent belongs to;
- M is a set of messages. A message $m ::= \langle n_m, w, v \rangle$, where n_m is the identification of the message, w is the sender of the message expressed by an identification of a role, v is the receiver of the message;
- R is a set of roles. A role $r :: = \langle n_r, I_a \rangle$, where n_r is the identification of the role, I_a is a set of identifications of agents that are playing this role;
- E is a set of environments. An environment $e :: = \langle n_e, B \rangle$, where n_e is the identification of the environment and B is a set of tuples of role Nr and number range q, $B = \{\langle Nr, q \rangle\}$;
- G is a set of groups. A group $g :: = \langle n_g, e, J \rangle$, where n_g is the identification of the group, e is an environment for the group g to work and J is a set of relations of mapping an agent Na to a role Nr;
- P is a set of protocols;
- s0 is the initial state of a collaborative system, expressed by initial values of all the components O, A, M, R, E, G, P and H;
- H is a set of human users who are validly authorized.

In term of the collaboration model, we may model an agent as an abstract system in Event B, called S_Agents, containing a description of its state and a number of events.

```
SYSTEM S_ AGENTS
SETS
  AGENTS={agent_query,
    agent_base1, agent_base2};
  GROUPES={groupe_
        casequery};
  ROLES = {case_query,
    case_base};
  PROTOCOLS={cnp, other};
  HUMANS={designer, guest}
VARIABLES
  agents, roles, protocols,groupes,
  users, assigned_roles, current_
  roles, assigned_users, current_
  users, part_of, supported
INVARIANT
  agents⊆AGENTS∧
  roles⊆ROLES∧
  protocols⊆PROTOCOLS∧
  groupes⊆GROUPES∧
  users⊆HUMANS∧
  assigned_roles ∈ agents↔roles
  ∧current_roles∈agents⇸ roles
  ∧current_roles⊆ assigned_
  roles∧assigned_users∈roles
  ↔users∧current_users∈
  roles⇸ users∧current_users
  ⊆ assigned_users∧part_of∈
  agents↔ groupes∧
  supported∈agents⇸ protocols
INITIALISATION
  agents, roles, groupes, users,
  assigned_roles, current_roles,
  part_of, supported := ϕ,ϕ,ϕ,
  ϕ,ϕ,ϕ,ϕ,ϕ||
  protocols := {cnp}
```

We denote A, G, R, P and H of the model with five sets, named AGENTS, GROUPES, ROLES, PROTOCOLS and HUMANS, respectively. The INVARIANT clause identifies the variable types and the constraints they should satisfy, so it can express the static laws of the system, properties, requirements and relation between the variables. For example, we can deliver a message, in the

INVARIANT clause, that any agent belongs to one or several groups and a group has zero or more agents. The relationship between an agent and a group is denoted by part_of. Similarly, the current_role relationship indicates an agent's current role and supported denotes that the agents supporting one or several protocols. In this study, we only use the Contract Net Protocol.

The events of S_Agents, as the dynamic part, are defined in the EVENTS clause. Each atomic event is composed of a guard and a generalized substitution. The necessary events are the ones for creating and deleting an agent, adding a role to an agent, changing the role of an agent, create current role and so on.

```
EVENTS
  /* creating a new agent ag */
  create_agent(ag)
    when
      ag∈AGENTS∧
      ag ∉ agents
    then
      agents := agents∪{ag}
    end
  /* deleting an existing agent ag*/
  delete_agent(ag)
    when
      ag∈AGENTS∧ag∈agents
    then
      agents := agents-{ag}
    end
  /* creating a new role rl */
  create_roles(rl)
    when
      rl∈ROLES∧rl ∉ roles
    then
      roles := roles∪{rl}
    end
  /* adding a role rl to a agent ag*/
  add_roles(rl, ag)
    when
      rl∈roles∧ag∈agents∧
      {ag↦rl} ∉ assigned_roles
    then
      assigned_roles:=
      assigned_roles∪{ag↦rl}
    end
  /* deleting a role rl of agent ag*/
  delete_roles(rl, ag)
    when
      rl∈roles∧ag∈agents∧
      {ag↦rl}∈assigned_roles
      ∧{ag↦rl} ∉ current_roles
    then
      assigned_roles := assigned_
          roles-{ag ↦rl}
    end
  /* assigning a role rr as current
    role of an agent ag */
  creat_current(rr, ag)
    when
      rr∈roles∧ag∈agents∧
      {ag↦rr}∈assigned_roles
      ∧{ag ↦rr} ∉ current_roles
    then
      current_roles := {ag↦rr}
    end
  /* changing the current role of
     agent ag from r1 to r2 */
  change_roles(r1,r2,ag)
    when
      r1∈roles∧r2∈roles∧
      ag∈agents∧ (ag↦r1) ∈
      current_roles∧ (ag↦r2) ∉
      current_roles∧ (ag↦r2)∈
      assigned_roles
    then
      current_roles := {ag↦r2}
    end
  /* adding an interaction protocol pr
     for agent ag */
  add_protocol(pr,ag)
    when
      pr∈protocols∧ag∈agents
    then
      supporte := supporte∪
          {ag↦pr}
    end
```

FIPA contract net protocol and its specification: FIPA Contract Net Protocol (FIPA, 2002; Smith, 1980) is a minor modification of the original CNP proposed of the widely used CNP, originally developed by Smith and Davis, in that it adds rejection and confirmation communicative acts. The representation of FIPA CNP is given in Fig. 1, based on extensions to UML 1.x.

Here, we model FIPA Contract Net Protocol as an abstract system, called S_CNP, to manage the contract process between two agents. The abstract system S_CNP includes the S_Agent to enable the use of all the variables and operations as well as verify the invariants.

```
SYSTEM S_CNP
INCLUDES S_AGENTS
SETS
  MESSAGES={cfp,refuse,propose,
    reject-proposal, accept-
    proposal, inform-done:inform,
    inform-result:inform, failure};
  STATES={Participant,Initiator};
  DEADLINE={in_time,time_out}
VARIABLES
  agent_groupe, deadline, state,
  c_simi, messages_ exchanged
INVARIANT
  deadline ⊆DEADLINE∧
  agent_groupe⊆GROUPES
  ∧state ⊆ STATES∧state∈
  (agent_groupe↔agent_groupe)
  ∧messages_exchanged∈
  agent_groupe↔
  (MESSAGES ↔agent_groupe)
  ∧c_simi ∈agent_groupe→NAT
INITIALISATION
  agent_groupe, deadline,state :=
  ϕ,ϕ,ϕ ||
  c_simi^-1(agent_groupe):= 0
```

Some sets are defined for messages exchanges between the initiator agent and the other participating agents. The set MESSAGES contains all possible eight types of messages exchanged with CNP protocol. The set STATES indicates the possible agent states of the Initiator or the Participants. The set DEADLINE owns two opposite states indicating whether the deadline has expired or not, respectively.

Four necessary events of S_CNP are used to dispose of the exchanged messages according to the CNP protocol, i.e., starting_system, answer_cfg, response and inform_msg, described as follows.

```
EVENTS
  /* starting an interactive cycle
     with a cfp from the initiator
     ag_ini */
  starting_system(ag_ini)
    when
      ag_ini∈agent_groupe
    then
      deadline:= in_time||
      state:=((agent_groupe-
      {ag_ini})×{Participant})
      ∪{ag_ini↦ Initiator}||
      messages_exchanged:=
      messages_exchanged∪
      ({ag_ini}×{{cfp}×
      dom(state ▷ {Participant})
      })
    end
  /* answering the cfp from ag_ini,
     retrieving the required part,
     and then returning the
     message msg and a
     corresponding similarity sim*/
  answer_cfg(msg, ag_par,
             ag_ini,sim)
    when
      ag_par∈agent_groupe∧
      state(ag_par)=Participant
      ∧ag_ini∈agent_groupe∧
      state(ag_ini) = Initiator∧
      messages_exchanged^-1
      (ag_ini↦ ag_par) ≅ cfp ∧
      deadline=in_time∧
      msg∈MESSAGES∧
      msg∈{refuse, propose }
    then
      messages_exchanged :=
      messages_exchanged∪
      { ag_par↦(msg ↦ag_ini)}||
      c_simi^-1(ag_par):= sim
    end
  /* responding the propose from
     ag_par with the message msg */
  response(msg,ag_ini,ag_par)
    when
      ag_ini ∈agent_groupe∧par∈
      agent_groupe∧state(ag_ini)=
      Initiator∧state(ag_par)=
      Participant ∧messages_
      exchanged^-1 (ag_par↦ ag_ini)
      ≅ propose∧deadline= time_
      out∧msg∈{reject-proposal,
      accept-proposal}
    then
      messages_exchanged :=
      messages_exchanged∪
      {ag_ini↦{msg↦ag_par}
    end
  /* informing ag_ini about the
     interactive result with the
     message msg */
  inform_msg(msg, ag_par, ag_ini)
    when
      ag_par∈agent_groupe∧
      ag_ini∈agent_groupe∧
      state(ag_par) = Participant
      ∧ag_ini∈agent_groupe∧
      messages_exchanged^-1
      (ag_par↦ag_ini)≅ accept_
      proposal∧msg∈{inform-
      done:inform, inform-result:
      inform, failure}
    then
      messages_exchanged :=
      messages_exchanged∪
      {ag_par ↦(msg↦ag_ini)}
    end
```

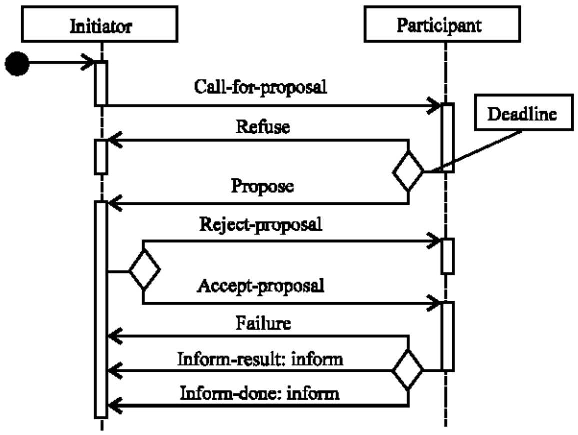

Fig. 1: FIPA contract net protocol

CASES RETRIEVAL SPECIFICATION OF INTELLECTUALIZED DESIGN SYSTEM OF SHAPE AND STYLE IN AUTOMOBILE

In this intellectualized design system of shape and style in automobile (IDSSSA), let us study the scenario of a characteristic-based cases retrieval, where the retrieval engine is an initiator agent (Agent_Query) that has to search automobile's parts with largest similarity in cases repositories according to the need of designer. It contacts cases repositories agents (Agent_Base1 and Agent_Base2), an operator agent (agent_user) and a cases-reconstruction agent (agent_redo). We only focus on the interaction about how to choose satisfactory parts by their similarities, as described in Fig. 2.

We will model each agent and each role as an Event B system and eventually construct a whole system using the INCLUDES clause. The overall specifications are incrementally constructed using composition, as shown in Fig. 3.

The Agent_Query agent plays the role of Case_Query and Agent_Base1 or Agent_Base2 takes the role of Case_Base. The following specifications only involve the scenario relating with the CNP. For space reasons, we discuss only necessary events, with detailed specifications omitted.

Specification of the Agent_Query agent: When invoked, Agent_Query taking the Case_Query role starts the communication by sending a cfp message to all other agents in the same work group, i.e., Agent_Base1 and Agent_Base2, which play the Case_Base role.

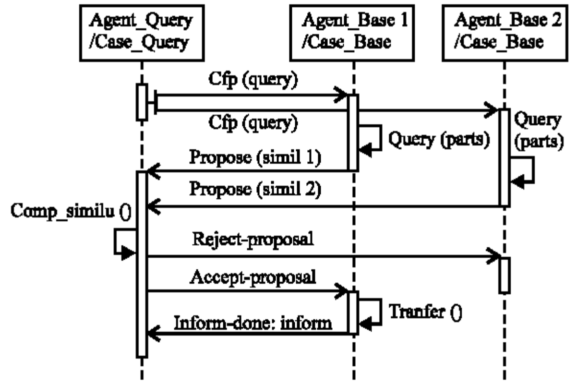

Fig. 2: Scenario of selecting an alternative similarity

```
SYSTEM  Agent_Query
INCLUDES Case_Query
VARIABLES
   task, ag_role, ag_name, users,
   choice_propose
INVARIANT
   task∈TASKS∧ag_role∈
   ROLES∧ag_name∈AGENTS∧
   ag_name∈agent_groupe∧
   state(ag_name)=Initiator∧
   choice_propose∈agent_groupe
   →BOOL∧users∈HUMANS
INITIALISATION
   task, ag_role, ag_name :=
   search_parts, case_query,
   agent_query || choice_propose
   := φ || users := designer
EVENTS
   /* starting the communication by
      emitting cfp to all the
      participating agents */
   cfp_agent_query(sty,par,mod,
        leng, wid,heig, fea,users)
      when
         task= search_parts∧
         ag_role= case_query∧
         sty∈STYLES∧par∈PARTS∧
         mod∈MODELS∧leng∈NAT
         ∧wid∈NAT∧heig∈NAT∧
         fea∈FEATURE∧
         users=designer
      then
         cfp_query(ag_name, sty, par,
         mod, leng, wid, heig, fea,users)
      end
/* responding propose from the
   participating agent */
resp_propose
      any ag_par where
         deadline=time_out∧
          ag_par∈agent_groupe∧
          state(ag_par)=Participant
         ∧ag_name ∈agent_groupe
         ∧messages_exchanged⁻¹
          (ag_par ↦ ag_name) ≅
          propose∧choice_
          propose(ag_par)=TRUE
      then
        send_response( ag_name,
             ag_par)
      end
```

We use only two necessary events in the system Agent_Query. Event cfp_agent_query is used to emit cfp to invoke the cfp_query event, with the parameters of which leng, wid, heig are the dimension of the automotive parts and sty, par, mod, fea are related pieces of information such as style, name of parts, model and feature description. In turn, the event cfp_query in the Case_Query role module calls the starting_system event of the S_CNP system.

Event resp_propose responds the propose message from the participating agent with send_response from the Case_Query module. After all participating agents have responded the cfp message with a propose, the initiator agent selects the appropriate participating agent which has retrieved a required part with the largest similarity.

Later, the accept-proposal message is sent to the selected agent, whereas, the reject-proposal message is sent to the unselected agent.

In the Case_Query role module, we aid the agent Agent_Query with three events to communicate with the participating agent. The last event comp_simil is used to select the largest value of similarity in the retrieved parts obtained from the two participating agent. Only all the propose messages has been sent to Agent_Query and the largest similarity is obtained, can the send_response event send out accept-proposal or reject-proposa message by invoking event response from the S_CNP system.

```
SYSTEM   Case_Query
INCLUDES S_CNP
SETS
   TASKS={search_parts,
      case_redo};
   STYLES={American, European,
      Japanese}; FEATURE;
   PARTS={engine, assembly, tyre,
      underpan, dynamo, radiator,
      seat…};
   MODELS={Benz, BMW, Audi,
      Ford, Porsche, Cadillac,
      Toyota, Honda…}
VARIABLES
   style, parts, model, length, width,
   height, features, simi1, simi2,
   c_similu, proposed, user
INVARIANT
   style∈STYLES∧parts∈PARTS
   ∧model∈MODELS∧length∈
   NAT∧width∈NAT∧height∈
   NAT∧features∈FEATURE∧
   ∧similu∈NAT∧simi1∈NAT
   BOOL∧user∈HUMANS
INITIALISATION
   long,wide,high,simi1,simi2,
   c_similu:= 0,0,0,0,0,0 ||
   style, parts, model, user, features
   :=ϕ,ϕ, ϕ,ϕ,ϕ||
   proposed := FALSE
EVENTS
   /* performing the startup of
      the communication */
   cfp_query (ag_initi, sty, par,
      mod,leng, wid, heig, fea,users)
      when
        ag_initi∈agent_groupe∧
         sty∈STYLES∧par∈PARTS
         ∧mod∈MODELS∧leng∈
         NAT∧wid∈NAT∧heig∈
        NAT∧fea∈FEATURE∧
        users=designer
      then
        starting_system(ag_initi)||
          style:= sty|| parts:= par ||
          model:=mod||length:=leng||
          width:= wid|| height:=
          heig|| features:= fea||
          user:= users
      end

/* computing the maximum
   similarity c_similu of retrieved
   parts from all the participating
   agents */
comp_simil
   any ag_par where
     ag_par∈agent_groupe∧
     state(ag_par)= Participant∧
     ag_name∈agent_groupe∧
     c_simi^-1(ag_par)>0∧
     messages_exchanged^-1
     (ag_par↦ ag_name) ≅
     propose∧choice_
     propose(ag_par)=TRUE∧
     simi1∈NAT∧simi2∈NAT
     ∧proposed ∈BOOL
   then
     if simi1=0
        then simi1:=
              c_simi^-1(ag_par)
      else
        simi2:= c_simi^-1(ag_par)||
        proposed:=TRUE ||
        c_similu :=
              max({simi1,simi2})
      end
   end
/* responsing the propose from the
   participating agents */
send_response(agi, agp)
   when
      deadline=time_out∧agi∈
      agent_groupe∧agp∈
      agent_groupe∧state(agp)=
      Participant∧state(agi)=
      Initiator∧messages_
      exchanged^-1(agp↦agi)≅
      propose∧proposed=TRUE
   then
    if  c_simi^-1(agp)= c_similu
      then
         response(accept-proposal,
              agi, agp)
      else
         response(reject-proposal,
              agi, agp)
      end
   end
```

Specification of the Agent_Base agent: Because of symmetry, we will discuss only the specification of Agent_Base1, which plays the role of Case_Base.

Three events are used in Agent_Base1 to respond cfp and accept_ proposal message from Agent_Query. The event verif_avail_agent and propose_simil are used to answer the cfp request jointly. In the event verif_avail called by event verif_avail_agent, if a matched part is found according to this part's external size (length, width and height) and other compositive information (style, parts name, model and feature description), the event propose_simil finishes the answer for the cfp via propose_si event of Case_Base module and in turn, answer_cfp event of S_CNP module.

The third event is accept_propo. When receiving an accept_proposal message, the agent Agent_Base1 begins to transfer the associated picture related with the satisfying part to the initiator agent, Agent_Query, by triggering firstly the event accept_propo of Agent_Base1 system and then the event send_inform of Case_Base system. Finally, an inform-done: inform message is sent to the initiator agent after transferring the picture successfully.

```
SYSTEM  Agent_Base1
INCLUDES Case_Base
VARIABLES
   ag_role, ag_name, available
INVARIANT
   ag_role∈ROLES∧ag_name∈
   AGENTS∧ag_name∈agent_
   groupe∧state(ag_name)=
   Participant∧available∈BOOL
INITIALISATION
   ag_role, ag_name := case_base,
   agent_base1 ||
   available := FALSE
EVENTS
   /*searching a required part
      after receiving cfp */
   verif_avail_agent(sty,par,mod,
         leng,wid,heig, fea)
      when
         ag_name∈agent_groupe
         ∧state(ag_name)=
         Participant∧ag_role=
         case_base∧sty∈STYLES
         ∧par∈PARTS∧mode∈
         MODELS∧leng∈NAT∧
         wid∈NAT∧heig∈NAT
         ∧fea∈FEATURE∧sty ↔
         par ↔ mod ↔ fea∈query
         ∧messages_exchanged^-1
         ({state^-1(Initiator)↦
         ag_name}) ≅ cfp
      then
         verif_avail(sty, par, mod,
            leng, wid, heig, fea) ||
         available:= avail
      end
/* evaluating the similarity of the
   retrieved part from the available
   participant agent after receiving
   cfp and finding a required part */
propose_simil(sty, par, mod,
        leng, wid, heig, fea)
   when
      deadline=in_time∧
      ag_name∈agent_groupe∧
      state(agent_name) =
      Participant∧sty∈STYLES∧
      par∈PARTS∧mode∈
      MODELS∧leng∈NAT∧
      wid∈NAT∧heig∈NAT∧
      fea∈FEATURE∧
      messages_exchanged^-1
      ({state^-1(Initiator) ↦
      ag_name}) ≅ cfp∧
      available=TRUE
   then
      propose_si(ag_name,state^-1
      (Initiator), similu((sty ↔ par
      ↔ mod ↔ fea)))
    end
/* accepting the proposal of the
   initiator agent after receiving
   accept_ proposal */
accept_propo(sty, par, mod,
        leng, wid, heig, fea)
    when
      deadline=time_out∧
      ag_name∈agent_groupe
      ∧state(agent_name) =
      Participant∧ag_role=
      case_base∧
      messages_exchanged^-1
      ({state^-1(Initiator) ↦
      ag_name}) ≅ accept_
      proposal
    then
      send_inform(ag_name,
         state^-1(Initiator), heig)
     end
```

There are four necessary events used in the Case_Base module to cooperate with the Agent_Base1 and the S_CNP. Obviously, the event verif_avail is used to explore a matched part and the similu could compute and return the similarity value of a particular part according to a certain retrieval condition. The event propose_si is used to calculate the similarity of retrieved part and answer the initiator agent with answer_cfp event. The last one, send_inform, is used to transfer the data of retrieved automobile part and when finished the transfer, finish this communication by sending message inform-done: inform to the initiator agent, Agent_Query.

```
SYSTEM  Case_Base
INCLUDES S_CNP
SETS
  STYLES={American,European,
     Japanese}; FEATURE;
  PARTS={engine,assembly,tyre,
    underpan, dynamo,radiator,
    seat…};
  MODELS={Benz, BMW, Audi,
    Ford, Porsche, Cadillac,
    Toyota, Honda…}
VARIABLES
  avail, query, style, part,model,
  long, wide, high, features, si,
  c_similu, part_id
INVARIANT
  style∈STYLES∧part∈PARTS
  ∧model∈MODELS∧long∈
  NAT∧wide∈NAT∧high∈
  NAT∧features∈FEATURE
  ∧si∈NAT∧avail∈BOOL
  ∧query∈STYLES ↔ PARTS
   ↔MODELS↔ FEATURE∧
  c_similu∈query⇸NAT∧
  part_leng∈query⇸NAT∧
  part_wid∈query⇸NAT∧
  part_heig∈query⇸NAT∧
  dom(part_leng)=dom(part_
  wid) ∧dom(part_leng)=
  dom(part_heig) ∧
  part_id∈NAT
INITIALISATION
  avail:=FALSE|| query, style,
  part, model,long, wide, high,
  features, c_similu:= ϕ, ϕ,ϕ,
  ϕ,ϕ,ϕ,ϕ,ϕ,ϕ||
  si, part_id:=0,0
EVENTS
  /* whether a matched part is
     found */
  verif_avail ( sty, par, mod,
       leng, wid, heig, fea)
     when
       sty∈STYLES∧par∈
       PARTS∧mod∈MODELS
       ∧leng∈NAT∧wid∈
       NAT∧heig∈NAT∧fea
       ∈FEATURE∧sty ↔ par
        ↔ mod ↔  fea∈query
     then
       avail:=BOOL(part_leng⁻¹
       (leng)=(sty ↔ par ↔ mod
        ↔ fea) ∧part_wid⁻¹(wid)=
       (sty ↔ par ↔ mod ↔ fea)
       ∧part_heig⁻¹(heig) =
       (sty ↔ par ↔ mod ↔ fea)
       ∧part_leng⁻¹(leng)=part_
       wid⁻¹(wid) ∧part_leng⁻¹
       (leng)=part_heig⁻¹ (heig) );
       part_id := part_heig⁻¹(heig)
     end
/* calculating the similarity si of
   the retrieved part */
si β similu(sty,par,mod, leng,
      wid,heig,fea)
   when
     sty∈STYLES∧par∈PARTS
     ∧mod∈MODELS∧leng∈
     NAT∧wid∈NAT∧heig∈
     NAT∧fea∈FEATURE∧
     sty ↔ par ↔ mod ↔ fea∈
     query
   then
     si:= c_similu((sty ↔ par
        ↔ mod ↔ fea))
   end
/* computing the similarity of the
   retrieved part and answering the
   initiator agent with proposal  */
propose_si(agp, agi, si)
   when
     deadline=in_time∧si∈NAT
     ∧agp∈agent_groupe∧
     agi∈agent_ groupe∧
     state(agp)= Participant∧
     state(agi)= Initiator∧
     messages_ exchanged⁻¹(agi
     ↦ agp) ≅ cfp
   then
     answer_cfp(proposal, agp,
                agi, si)
   end
/* after transferring the data of
   retrieved part, ending this
   contact cycle */
send_inform(agp,agi,heigth)
   when
     deadline=time_out∧heig∈
     NAT∧agp∈agent_groupe
     ∧agi∈agent_ groupe∧
     state(agp)= Participant∧
     state(agi)= Initiator∧
     messages_ exchanged⁻¹(agi
     ↦ agp) ≅ accept_proposal∧
     part_id=part_heig⁻¹ (heigth)
   then
     transfer(agp, agi, part_id);
     inform_msg( inform-done:
       inform, agp, agi)
   end
```

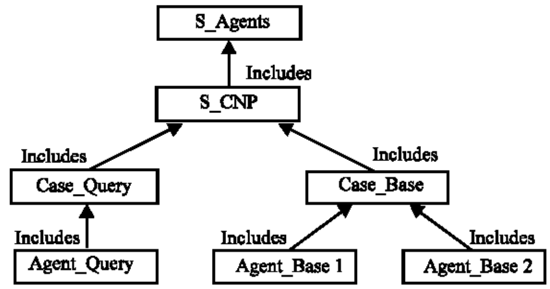

Fig. 3: The includes relationship among subsystems

MECHANICAL PROOF ON THE CASES RETRIEVAL OF IDSSSA

After the abstract systems in Event B have been translated into classical abstract machines using evt2b (MATISSE, 2001), we may use Atelier B to check system consistency and correctness during the initialization and refinement. With evt2b tool we can generate also the additional proof obligations that are needed to prove correct refinement of MAS.

Consistency check: The consistency of the abstract system is established by proof obligations. The semantics of an abstract system rise from its invariant and is guaranteed by proof obligations. The consistency of the system is established by the invariant of the system holding in every state reached by each event. More precisely, it should be proved that the initialization establishes the invariant and new invariant is preserved when each event modifies state variables, provided that invariant holds with older variable values and event guard holds. For example, to show that the initialization of S_AGENTS is correct, we would have the following proof obligation:

$$\phi \subseteq AGENTS \wedge \phi \subseteq ROLES \wedge \phi \subseteq GROUPES \wedge \phi \subseteq HUMANS \wedge \{cnp\} \subseteq PROTOCOLS \wedge \phi \subseteq \phi \wedge \phi \in \phi \leftrightarrow \phi \wedge \phi \in \phi \nrightarrow \phi \wedge \phi \in \phi \nrightarrow \{cnp\}$$

and the proof obligation for the first invoking of event create_agent(ag) is as follows:

$$\phi \subseteq AGENTS \wedge \phi \subseteq ROLES \wedge \phi \subseteq GROUPES \wedge \phi \subseteq HUMANS \wedge \{cnp\} \subseteq PROTOCOLS \wedge \phi \subseteq \phi \wedge \phi \in \phi \leftrightarrow \phi \wedge \phi \in \phi \nrightarrow \phi \wedge \phi \in \phi \nrightarrow \{cnp\} \wedge ag \in AGENTS \wedge ag \notin \phi$$
$$\Rightarrow$$
$$\phi \subseteq AGENTS \wedge \phi \subseteq ROLES \wedge \phi \subseteq GROUPES \wedge \phi \subseteq HUMANS \wedge \{cnp\} \subseteq PROTOCOLS \wedge \phi \subseteq \phi \wedge \phi \in \phi \leftrightarrow \phi \wedge \phi \in \phi \nrightarrow \phi \wedge \phi \in \phi \nrightarrow \{cnp\} \wedge agents := \phi \cup \{ag\}$$

Refinement check: Dynamic constraints are very difficult to express and verify in B. A solution for checking such constraints is to use the refinement mechanism. The B refinement process of an abstract system may introduce new variables and new events in the resulting abstract system. The abstract system and its refinement are related by a gluing invariant. During the refinement, the non-determinism will be reduced. The preconditions have to be relaxed in order to take into account all the possible cases. After the 11th refinement, the formal specification together with the refinement process give an executable C code that is correct with respect to the specification. The demonstration of the proof obligations in each refinement provides evidence that related events fulfils the invariant predicates and correctness properties are formally proved. For example, a desired property for the subsystem Agent_Query is responsibility: at least one participant will respond to the call-for-proposal before a certain deadline:

$$\mathrm{cfp(query)} \Rightarrow \neg((\mathrm{simi1} = 0) \vee (\mathrm{simi2} = 0)) \wedge \\ \mathrm{c\_simi}^{-1}(\mathrm{ag\_par}) > 0 \wedge \mathrm{deadline} \subseteq \mathrm{DEADLINE}$$

Another approach of checking dynamic constraints, (Darlot, 2002), combines the Event B method with the temporal logic PLTL (Pnueli, 1981), whereas the temporal properties are verified by model checking, namely the SPIN tool.

EVALUATION OF THE FORMAL DEVELOPMENT

Here, we provide metrics about the formal specification and proof of the cases retrieval of IDSSSA. During this subsystem is entirely proved, a number of proof obligations are generated automatically, from the gluing invariant and the definitions of the abstract and concrete operations. This validation process we performed includes the verification of the abstract B system, interfaces between B models and non-B, the added proof rules and the translation of concrete B0 models into C source code. Table 1 synthesises a part of metrics related to the development.

There are 439 Proof Obligations (POs) in to prove to establish the consistency, 35 have to be proven with the Atelier B interactive prover. This means a 92% of automatic proof, which is a high rate and a good result. Over the remaining 35 POs, some of them, that the case for the operation machine and its refinements, are similar, which ease the manual proof process. After gradually introducing new variables and new events, more POs are produced in the specifications, as shown in Table 1, including the mathematics rules added by the programmer during the interactive proof.

Table 1: A part of metrics on the formal specification and proof

Metrics	Spec.	Ref. Step 1	Ref. Step 2	Ref. Step 3	…
Lines of B	520	1650	1960	2090	…
Basic modules	7	11	16	18	…
Generated POs	439	2883	315	1537	…
POs automatic proofs (%)	92	96	91	52	…
Lines of C code	112	380	407	430	…
Workload (Men days)	5	10	13	14	…

Table 2: Comparing between formal development and conventional development

Elements	Formal development	Conventional development
Errors discovered by reading	9	20
Errors discovered by proof	26	Not applicable
Errors discovered by testing	11	61
Main development (days)	30	30
Proof activity (days)	18	Not applicable
Testing (days)	3	12
Integration (days)	3	6

One goal of the IDSSSA project is to compare formal and traditional development in order to show the benefits and drawbacks of using formal techniques in industry. For this reason, the cases retrieval subsystem was completed by two different teams with the same starting point: one used formal techniques and the other used traditional techniques. A test phase was provided to each development to check the correspondence between the informal requirements and the produced C code. Table 2 describe the elements of the comparison.

From the Table 2 it concluded that the formal development produces fewer errors than the traditional development, which indicates a better quality in terms of compliance with the original requirements. This is because the modelling activity requires a good understanding of the informal specification and a lot of work required by the refinement method and thus, reduces the risk of introducing errors.

Another significant different is testing time, proof activity time and the errors number found by testing between the two development methods. This means that although the formal development can save us testing time, the proof is very long and costly. However, this can be decreased if Atelier B is improved and particular rules as well as proof tactics are developed. If we manage to capitalize on experience obtained from previous developments, or to adopt a methodology dedicated to MAS applications, we should decrease the time consumed in the proof, or the development time required to build models. Moreover, even if the time and cost needed for the formal development are greater than those for a conventional development, it is worthwhile to be adopted in the development of safety-critical or life-critical systems, to guarantee the expected safety.

Accordingly, we have also shown that the formal methods are much more competitive with a strong involvement in tool improvement and in methodologies. Experience gained in previous developments should improve our skills and speed up future developments too. And then, using formal methods could be a real advantage as it is no more costly than conventional development while providing high-quality code.

Finally, it needs to be emphasized that B method is more suitable for the development of our applications. Compared with other formal methods, an advantage of the B method, as well as Event B, is the iterative refinements, so specifications are developed in a top-down fashion. Another advantage is the component-based approach to developing the specifications, which maps well to component-based architectures and development methodologies. An additional strength of the B method is its tool support. From a B specification, we can generate code, analyze the correctness and perform an animator as well as proof of correctness. The ability to easily reuse specifications has also been cited as a plus to the B method and tools (Rouff *et al.*, 2006).

CONCLUSIONS AND FUTURE WORK

In this study, we present a complete study on formal specification of MAS applications using Event B, with CNP as interactive protocol, based on a roles-based collaboration model. Then we prove the correctness of the specification and its refinement using Atelier B. In the end, the results of proof are analysed and evaluation of our method is presented.

Current works focus on the combination of model checkers with Event B to verify key properties of the cases retrieval of IDSSSA.

One of this future tasks is to import the reasoning mechanism such as the BDI model into our model. We also envisage the embedding of the Petri Nets or CSP into Event B to verify the liveness and safety in the communicating process. Another point, which would be interesting to study, is other techniques to simplify the increasing complexity of the modelling MAS and the method for the practical formal development of MAS.

ACKNOWLEDGMENTS

The present research is funded by the Major State Basic Research Development Program of China (973 Program) under Grant No. 2004CB719401 and also supported by the National Natural Science Foundation of China under Grant No. 60542004. The authors would like to acknowledge gratefully the valuable suggestions and constructive comments provided by the anonymous referees.

REFERENCES

Abrial, J.R., 1996. The B-Book: Assigning Programs to Meanings. Cambridge University Press, Cambridge.

Ball, E. and M. Butler, 2006. Using decomposition to model multi-agent interaction protocols in Event-B. In: Proceedings of Fm'06 Doctoral Symposium, Hamilton Ontario, Canada.

Darlot, C., 2002. Reformulation et vérification de propriétés temporelles dans le cadre du raffinement de systémes d'événements (Reformulation and verification of temporal properties during refining system events). Thèse de doctorat, soutenue à l'université de Franche Comté, France, pp: 96-100.

d'Inverno, M., D. Kinny, M. Luck and M. Wooldridge, 1997. A formal specification of dMARS. Intelligent Agents IV: Proceeding of 4th International Workshop on Agent Theories, Architectures and Languages (ATAL'97), Providence, Rhode Island, USA., pp: 155-176.

Fadil, H. and J.L. Koning, 2005. A formal approach to model multiagent interaction using the B formal method. Advanced Distributed Systems: 5th International School and Symposium (ISSADS 2005), Guadalajara, Mexico, pp: 516-528.

Ferber, J., 1999. Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence. 1st Edn. Addison-Wesley Longman Inc., New York.

FIPA, 2002. Fipa contract net interaction protocol specification. Technical Report, FIPA, http://www.fipa.org/specs/fipa00029/sc00029h.pdf.

Liu, L. and H. Zhu, 2006. Implementing agent evolution with roles in collaborative systems. Proceedings of the 2006 IEEE International Conference on Networking, Sensing and Control, Ft. Lauderdale, Florida, USA., pp: 819-824.

MATISSE, 2001. Event B to B translator user manual. Matisse Project (IST-1999-11435), http://www.atelierb.societe.com/ressources/evt2b/evt2b_user_manual.pdf.

Mermet, B., 2002. Formal model of a multiagent system. In: 3rd International Symposium From Agent Theory to Agent Implementation (AT2AI-3). 16th European Meeting on Cybernetics and Systems Research, Vienne, Autriche, pp: 653-658.

Metayer, C., J.R. Abrial and L. Voisin, 2005. Rodin deliverable 3.2, Event-B language. Technical report, University of Newcastle upon Tyne, UK, http://rodin.cs.ncl.ac.uk/deliverables/D7.pdf, May 2005.

Moldt, D. and F. Wienberg, 1997. Multi-agent systems based on coloured petri nets. In: Proceedings of 18th International Conference on Application and Theory of Petri Nets (ICATPN'97), Toulouse, France, pp: 82-101.

Pnueli, A., 1981. The temporal semantics of concurrent programs. Theor. Comput. Sci., 13: 45-60.

Rouff, C.A., M. Hinchey, J. Rash, W. Truszkowski and D. Gordon-Spears, 2006. Agent Technology from a Formal Perspective (NASA Monographs in Systems and Software Engineering). Springer-Verlag, New York Inc., USA.

Smith, R.G., 1980. The contract net protocol: High-level communication and control in a distributed problem solver. IEEE Trans. Comput., C-29: 1104-1113.

Snook, C. and M. Butler, 2000. Verifying dynamic properties of UML models by translation to the B method and toolkit. Technical Report DSSE-TR-2000-12, Declarative Systems and Software Engineering Group, University of Southampton, September 2000. 38.http://www.dsse.ecs.soton.ac.uk/techreports/95-03/2000-12.html.

Zhu, H., M. Zhou and P. Seguin, 2006. Supporting software development with roles. IEEE Trans. Systems, Man and Cybernetics. Part A, 36: 1110-1123.