A Tool for the Automatic Generation of MOISE Organisations From BPMN

Massimo Cossentino, Salvatore Lopes and Luca Sabatucci

ICAR-CNR, 153, Via La Malfa, 90146 Palermo, Italy

Abstract

Multi-agent systems proved successful in enacting business processes because of their inner properties (distribution of tasks, collaboration and coordination among agents). MAS adoption in enacting processes becomes even more interesting if they exhibit adaptation capabilities. The proposed approach consists in the automatic generation of a MOISE organisation from the BPMN specification of a business process. This organisation is conceived to support adaptation because of the possibility to adapt its configuration at runtime according to emerging needs. Here, we focus on the tool for processing BPMN specification and generating MOISE organization code.

Keywords

Multi-Agent Systems, Business Process, BPMN, Adaptation

1. Introduction

Traditionally, business processes are designed as static, rigid procedures, but in real production environment there are many events and/or exceptions that can not be foreseen at design time and can lead to the failure of the whole process. For instance, some web service could be not reachable, the network could be extremely busy with heavy delay in response etc. . . However, increasing the agility and the flexibility of business process is not trivial being in contrast to the current trend of over-specifying workflow details with the objective to detail every possible execution branches.

Multi-Agent Systems own many interesting features as autonomy, distribution of tasks, and collaboration/coordination, that proved successfully in a range of application fields. Historically, MASs have good records in implementing workflows because all those features perfectly match with enterprises' needs [1, 2, 3]. Moreover, the Belief Desire Intention (BDI) paradigm[4] allows developers to design applications with practical reasoning, facilitating the definition of business logic.

Most of the agent-based approaches to workflow design are based on services and semantic web description (i.e. DAML-S [5]) for defining the external behaviours of proactive agents, and social and communication abilities of agents to coordinate the flow of tasks. It remains open the issue of coordinating heterogeneous, autonomous agents, whose internal designs is not partially or full known a-priori.

WOA 2020: Workshop "From Objects to Agents", September 14–16, 2020, Bologna, Italy

massimo.cossentino@icar.cnr.it (M. Cossentino); salvatore.lopes@icar.cnr.it (S. Lopes); luca.sabatucci@icar.cnr.it (L. Sabatucci)

An important improvement in MAS design and implementation comes out with the JaCaMo platform [6]. It represents an interesting solution because it handle various aspects of agent programming. JaCaMo integrates in an unique platform three multi-agent programming dimensions (agent, environment, and organization levels) and provides Jason [7] for developing BDI agents [4], CArtAgO for defining artifacts [8], and MOISE [9] for specifying agent organizations.

In this paper we propose a tool able to automatically generate organizations of agents to implement a given workflow expressed in Business Process Modeling Notation (BPMN) [10]. A structured organization adds robustness to the MAS system without losing the nice properties of agents like reasoning, communication and coordination abilities. The aim is to produce a MAS implementation of the business process able to adapt itself to various internal/external unexpected conditions (unavailability of services, failure in reaching the expected end-condition, network problem and so on).

The BPMN is a high-level language that allows developer to model all the phases of a planned business process. It focuses on analysis activity and there is not any bounding between tasks and services at design time. A recent result has been that a business process, opportunely designed, may be automatically translated into goals [11]. This translation has the advantage that goals allow for breaking the rigid constraints of a BPMN: whereas sequence flows specify the precise order in which services are invoked, goals can relax the order of task execution, widening the space for adaptation [12]. This goes into the direction of defining several possibilities to attain the desired result.

Having a BPMN as a set of goals, it is possible to conceive a multi agent system able of addressing them [13]. The idea is that, given a set of goals, and a set of available services, it is possible to automatically generate one or more social organizations corresponding to the workflow enactment. Clearly, the number of organizations depends on the availability of services. A redundant service repository allows for different plans to pursue the same set of goals. However, when several organizations can be produced (each one targeted to the same desired result), it is possible to enable run-time adaptation strategies for continuing goal pursuit in case of failure.

The tool we propose uses a three steps procedure:

1. goal extraction from a BPMN definition [11],
2. exploiting a planner for composing available services (stored in a yellow pages service) and obtaining different wokflow solutions to the same set of goals, and
3. employment of each different solution for generating multi-faceted agent organizations voted to achieve the business goal.

The paper is structured as follows: Section 2 presents the theoretical background, providing a brief description of BPMN, MOISE and JaCaMo. Section 3 describes the automatic generation of MOISE organizations; in particular, Section 3.1 presents a running example; Section 3.2 briefly reports the approach by discussing the mapping between BPMN elements and MOISE metamodel elements; Section 3.3 enters into the details, by explaining the algorithm of conversion. Finally, some conclusion is drawn in Section 4.

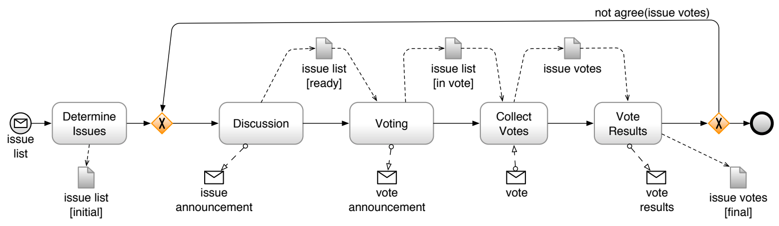

Figure 1: An example of worflow rielaborated from [14]

2. Theoretical Background

In this section we provide a brief description of the main background concepts used along the paper; namely the BPMN notation for the design of business processes and the MOISE framework for multi-agent systems organisations definition.

2.1. Process Modeling with BPMN

The *Business Process Model and Notation* (BPMN) [10, 14] is de-facto standard for business analysts to model a process. It contains a very articulated meta-model and an expressive notation for representing business processes of diverse nature. The graphical notation allows several modelling perspectives [10, 14]; this paper focuses on collaboration diagrams (similar to activity diagrams), in which a process is described as a collection of participants. Processes are composed of five categories of objects: activities, events, messages, data objects, and many kind of gateways. Every participant (each one in a different Swimlane) has her own flow of activities. Coordination occurs when processes exchange messages via message flows.

2.2. Organisations Definitions with MOISE

MOISE defines a collection of elements for modelling an organization. A MOISE organization is a specialized group that is devoted to pursue some goal. An organization is not only a collection of roles. Indeed, defining an organization implies the definition of structural, functional and normative perspectives [9].

The Structural Specification describes who (role) operates in the organization, the organization's hierarchy (groups and subgroups), which role belongs to each group together with its cardinality, and the kind of relationship between roles (authority, communication, acquaintance)

The Functional Specification specifies the scheme of activities associated to each group. A scheme represents the goal decomposition tree. Each scheme is characterized by a goal to attain, and one or more plan. A plan represents the modality (sequence, parallel, choice) to address the inner sub-goals. All the goals are contained into a mission.

The Normative Specification defines obligations and permissions that link roles to missions.

A brief description of the MOISE elements based on definitions proposed by Hubner in [9]. is reported here:

Structural Specification:

Role a role represents a placeholder for an agent that takes in charge to perform some activity. The number of agents that can play a role is constrained by a minimum and maximum number.

Group a group is composed by roles. A group is well formed when all its roles are played by agents.

Link A link specifies the type of relationship between agents playing two roles. It is possible to describe compatibilities between roles.

Functional Specification:

Goal semantic description of a result to obtain; MOISE goals may belong to two different types: achieve and maintain.

Scheme a Scheme is the decomposition of the organization's goals as an articulated gol-tree. It also includes the specification of Missions.

Plan is an operator, used within a Scheme, in order to specify the goal decomposition type (sequence, choice, parallel).

Mission A mission is a set of (sub-)goals.

Normative Specification:

Norm there are two types of norms: obligation and permission. They establish the link between the role and the mission. When an agent plays a role, it should or would commit to missions' goals.

The MOISE organisation (provided as a XML file) is not operational in itself. A set of CArtAgO Artifacts allows all necessary constructs and data structures to make a MAS operative, i.e. allowing agents to adopt ad play a role, to participate to a group, and to commit a mission [6]. For instance, the GroupBoard stores information about the MOISE elements the organization is made of. Once an agent adopt a role, it is subject to the constraints the organization imposes. Depending on the norm, an agent can commit or to be obliged to perform a mission.

3. A Tool for the Automatic Generation of MOISE Organisations from BPMN

This section discusses the tool for the automatic generation of agent organization, by using a running example to present details of the approach.

3.1. Running Example: The Simplified Email Voting Process

In the remaining sections of this paper, we refer to the example workflow reported in Fig. 1. That is a simplification of a well known BPMN example [14]. In this process, members of a

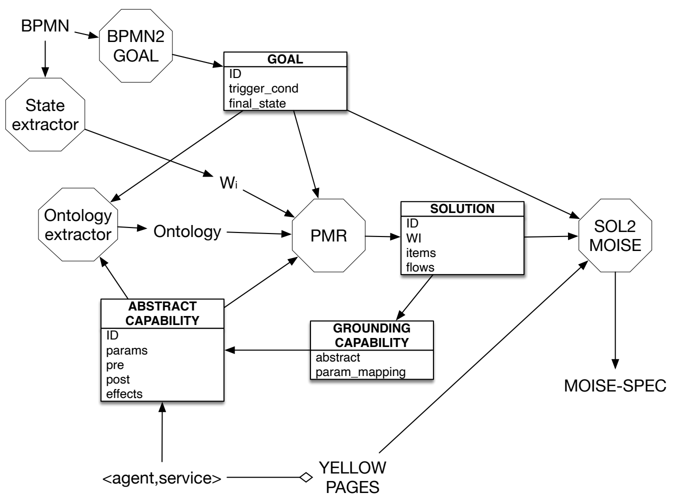

Figure 2: An overview of the main elements involved in the process to generate MOISE organizations from BPMN

committee discuss an issue list, and express a vote: if the majority is not reached, a new iteration of discussion/voting is performed.

The process description reports some events (emails) connecting the reported activities with the external voting members, represented as a swimlane shading the behaviour of the voting members that is of low interest for the current example. Data flow in the process is represented by Data Objects (like the Issue_list) that can evolve their refinement during the process, and that is represented by a state (initial, in_vote,. . .).

3.2. The Proposed Approach

This section describes the proposed approach for generating MOISE organizations starting from the BPMN description of a workflow.

The main tools and artifacts of the architecture are depicted in Fig. 2. They contribute to the generation of the organization according to the following sub-steps:

Goal Extraction The first step consists in the extraction of goals from the BPMN process that is depicted using some BPMN compliant tool (and exported using the XMI format). This becomes the input of the *BPMN2GOAL* module[1]. The tool generates a list of goals by inspecting BPMN elements and their relationships. Indeed, single elements (activities, gateways,. . .) contribute to the advancement in the world state in a way that is dependent on both endogenous factors (the result provided by the work done inside the element) and exogenous ones (the influence that other process elements have on it by creating its input, and constraining its output for compatibility with the remaining part of the process). In other words, a kind of balance of forces is to be solved in order to represent the mutual influence of each BPMN element on the other,

[1]Available online at: http://aose.pa.icar.cnr.it:8080/BPMN2Goal/

GOAL: <*name of the goal (from BPMN activity*>
WHEN: <*conditions for goal triggering, from preconditions of the activity and post-conditions of predecessor BPMN elements*>
THEN: <*intended state of the world, from activity postcondition and preconditions of successor BPMN elements*>

Figure 3: The general structure of a goal extracted from the BPMN process

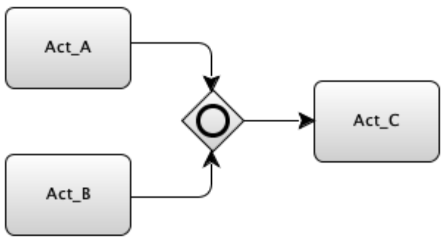

Figure 4: A fragment of BPMN process

and the single element contribution to what becomes the collective outcome of the process execution.

Details about the goals extraction procedure may be found on [11]. Briefly, the structure of a goal is reported in Fig. 3. Usually the goal name is taken from the activity it refers to (GOAL line), while the WHEN clause defines the trigger condition of the goal; this depends on the elements before the activity under analysis. Let us consider the process fragment reported in Fig. 4, an OR gateway merges the control flows of activities Act_A, Act_B that are placed before Act_C: the input of Act_C may be provided by only one of the other two activities feeding the OR gateway or even by both of them. This has to be reported in the WHEN condition by connecting the two output conditions of Act_A, and Act_B with an OR logical operator. We may also suppose Act_C requires some other condition to hold in order to be executed and that is to be expressed in the WHEN specification as well. This may happen when the condition in the original process was generated by other activities/tasks placed before Act_C. Finally, the THEN clause defines the postcondition after the execution of Act_C. This could also depend on the expected input of the following activities as well, in order to ensure the correct execution of the workflow.

State of the World Extraction The current state of the world is of paramount importance for the execution of a MAS solution. This is generated by the *State Extractor* module by processing the input workflow. The method considers the event that is usually sent to the workflow in order to trigger its execution together with any Data Input (an input Data Object for the whole process) as specified in the XMI file.

Ontology Extraction Goal specifications naturally define a vocabulary of terms that may be usefully employed to specify a part of the world where the agents live (predicates in the

WHEN/THEN conditions of goals), and the actions that can be done in it (BPMN activities). This generates a primordial ontology that is generated by the *Ontology Extractor* module. A more accurate processing of some optional details of the BPMN notation greatly contributes to this issue. As an example we could consider Data Objects (that could represent ontology concepts), their states (that could generate predicates), Data type (items) that could contribute to the generation of a IS-A tree. Despite the interesting implications, these features are not relevant to the current work.

Solutions Calculations The possibility to execute the workflow with the available agents in the MAS depends on the capabilities such agents register in the system's Yellow Pages. We suppose each agent, entering the system, registers its own capabilities in such register by using a tuple <*agent, service*>. For each service, an *Abstract Capability* is automatically generated by considering the service interface specifications like pre and post conditions. Of course, services with the same specifications are considered as belonging to the same abstract capability thus creating a redundancy in the capability instantiation that can profitably support some adaptation degree (see [13]). Indeed, the instantiation of the capability also depends on some parameters that are part of the solution and they contribute to the definition of the *Grounding Capability*. A Grounding Capability is the concretization of an Abstract one, and many Grounding capabilities may correspond to one Abstract Capability, according to the available agent-service tuples in the Yellow Pages.

Solutions are computed by the Proactive Means-end Reasoning (PMR) algorithm [15]: each solution is a workflow including capabilities, decision nodes, and so on.

The PMR algorithm, in each solution, defines a world transition system (WTS), like the one shown in Fig. **??** from the input workflow of Fig. 1. The initial state corresponds to the initial event of the workflow (the reception of the issue list in this example). The next state corresponds to the availability of an issue list arranged for the discussion. The PMR identifies the capability *prepare* as the one that could transform the state of the issue list from *[received]* to *[initial]*. This capability is the abstraction of a service registered in the Yellow Pages by one or more agents. Actually this likely means the *prepare* capability proposes the received issue list to the committee chair for editing. Once the chair completes her editing, the list moves to the *[initial]* state and it is ready for discussion in the committee. Again the PMR algorithm identifies one capability (*discussion*) for managing the debate and producing the expected output (the issue list in the *[ready]* refinement state).

Now, if the voting achieves a majority consensus, the final state is reached, otherwise another branch is activated with the possibility to iterate modifications and votes.

This WTS ma be used to deduce the workflow underpinned by the solution. That step is easily done by looking at the capabilities listed in the WTS and reporting them as activities just like it is shown in Fig. 6. Control flows are similarly found in order to complete the design.

It is worth to note that this workflow is equivalent to the input BPMN one in terms of the results it produces but the type and number of services may not (and usually do not) exactly map one-to-one to the initial process activities. This solution is specifically conceived to be executed by the agents in the MAS and it has been designed to only use services that are possessed by those agents.

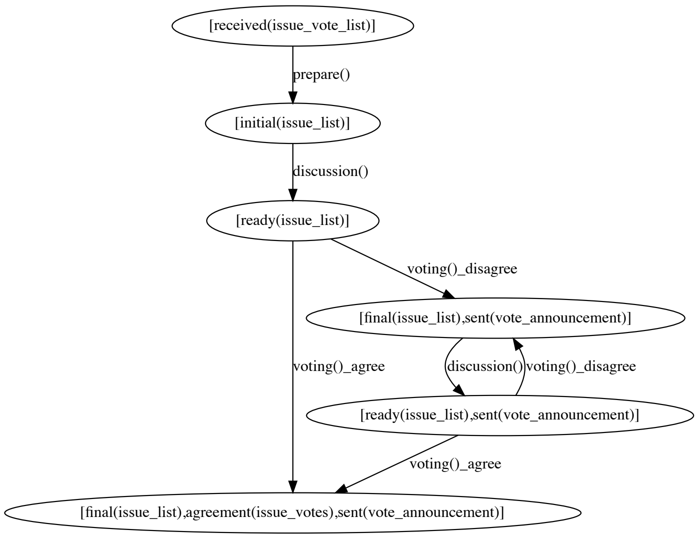
[received(issue_vote_list)]
prepare()
[initial(issue_list)]
discussion()
[ready(issue_list)]
voting()_disagree
[final(issue_list),sent(vote_announcement)]
voting()_agree
discussion()
voting()_disagree
[ready(issue_list),sent(vote_announcement)]
voting()_agree
[final(issue_list),agreement(issue_votes),sent(vote_announcement)]

Figure 5: The world transition statechart extracted by the PMR algorithm from the process reported in Fig. 1

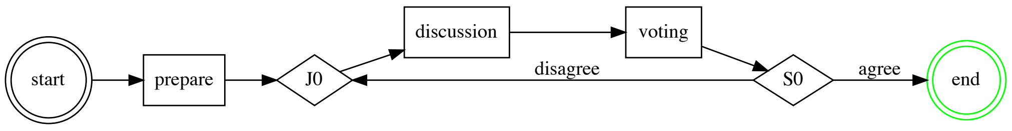

Figure 6: The workflow defined by the PMR algorithm to obtain the prescribed results using the services in the Yellow Pages

Of course the proposed example is very simple and solutions could be easily found even by hand but the approach works for large and complex input BPMN processes that would be very hard to solve using a long list of not exactly one-to-one matching services.

Moreover, the PMR algorithm, if the Yellow Pages repository is huge enough, may produce several different solutions by differently composing all the existing capabilities. This generates an even larger number of concrete solutions (the realization of a solution in terms of concrete capabilities).

MOISE Organization Definition The last step in the adopted process consists in the actual definition of the MOISE organization. The main input for that is: 1) the set of goals, 2) the set of solutions generated by the PMR algorithm and 3) the content of the yellow pages. These are processed according to the algorithm detailed in the next subsection.

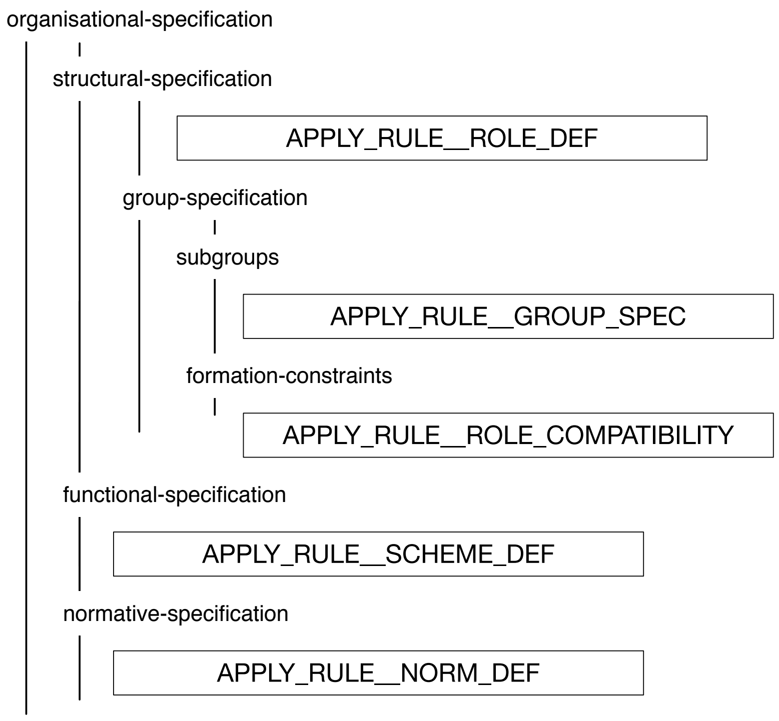

Figure 7: Abstract Representation of the XML output generated by the tool by applying specific rules.

3.3. An Algorithm for MOISE Organisation Definition

The automatic definition of the MOISE organization relies upon the XML template represented in Figure 7, in which XML elements and rules interleave. The algorithm has been written in Scala, a language derived by Java but closely integrated with XML. This way a XML rule is defined as a function that receives some parameter and returns the XML element to be used to complete the schema.

For instance, the "role definition rule" is coded as the following Scala function:

```
def apply_rule__role_def(yp:List[ServiceDescr]): Elem = {
<role-definitions>
  {yp.map(service =>
    <role id={service.id + "_role"}>
            <extends role="worker"/>
    </role>
    )}
</role-definitions>
}
```

The function receives a list of Capabilities (extracted from the system Yellow Pages) and returns a 'role-definitions' tag where children are generated from the list of services (yp parameter). Each service in yp generates (*map* method) a new 'role' xml element extending the 'worker' parent role.

The following function contains the rule for generating the group specification corresponding to a Solution as generated by the PMR algorithm.

```
def apply_rule__group_specification(sol: Solution): Elem = {
  <group-specification id={get_group_id} min="0">
  {apply_rule__group_roles(sol) ++ apply_rule__group_links(sol)}
  </group-specification>
}
def apply_rule__group_roles(s: Solution): Elem = {
  <roles>
    <role id="manager" min="1" max="1"/>
    {solution.wftasks.map(capability =>
      <role id={capability.id + "_role"} min="1" max="1"/>
        )}
  </roles>
}
def apply_rule__group_links(s: Solution): Elem = {
  <links>
    <link from="manager" to="worker" type="authority"
    extends-subgroups="false" bi-dir="false"
    scope="intra-group"/>
    <link from="worker" to="manager" type="acquaintance"
    extends-subgroups="false" bi-dir="false"
    scope="intra-group"/>
  </links>
}
```

Briefly, the rule produces a 'group-specification' by applying two sub-rules: the first one is for generating roles of the group; it is done by looking at the specific tasks of the input solution. The PMR solution is an assembly of Capabilities which execution ensures full goal satisfaction. Each of the involved Capabilities produces a new role into the group. Clearly, more agents of the society could own the same Capability, being able of play that role. The second sub-rule is to define structural links among these roles.

An interesting rule is that for generating the scheme, in the functional specification, corresponding to a Solution.

```
def apply_rule__scheme(s: Solution): Elem = {
  val sol_tree = new SolutionPattern(s)
  val opt_tree: Option[WorkflowPattern] = sol_tree.get_tree

  if (opt_tree.isDefined) {
    val tree : WorkflowPattern = opt_tree.get
    val capabilities = get_solution_capabilities(s)
    val scheme_id = get_scheme_id

    <scheme id={scheme_id}>
    {
    apply_rule__plan(tree)++
    apply_rule__management_mission(scheme_id)++
    capabilities.map(c=>apply_rule__mission(c))
    }
    </scheme>
  } else {
    <scheme id={get_scheme_id}>
    </scheme>
  }
}
```

The function uses the SolutionPattern class, responsible of identifying basic workflow patterns[16] in the Solution. Figure 8 shows an example of translation applied to Figure 6. The translation

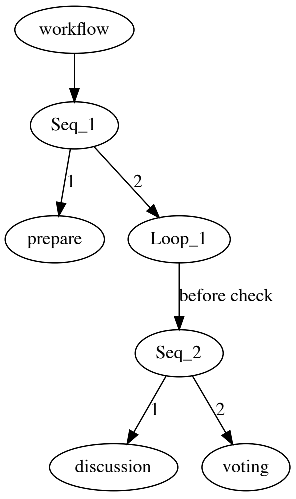

Figure 8: Result of Solution Pattern transformation. This algorithm identifies some of the Workflow Patterns [16] into a Solution, and it renders the workflow as a structured tree.

algorithm exploits the control-flow perspective of the solution in order to identify some basic constructs: Sequence patterns, Choice patterns, Loop patterns.

Sequence Pattern. Description: a ‘sequence’ models consecutive steps in a workflow. Activities A,B,C represent a sequence if the execution of B is enabled after the completion of A, and, the execution of C is enabled after the completion of B.
Identification. The translation algorithm supposes the sequence is the default approach for implementing a workflow.

Exclusive Choice Pattern. Description: a point in the workflow where the control may pass to one among several branches, depending on the evaluation of a condition.
Identification. The translation algorithm supposes a split exclusive gateway starts an Exclusive Choice pattern. Branches are identified as possible sequences that terminates either with the same join exclusive gateway (Simple Merge pattern) or an end event.

Structured Cycle Pattern. Description: a point in the workflow where one or more activities can be executed repeatedly.
Identification. The translation algorithm supposes a join exclusive gateway possibly begins a loop, whereas it contains a ’before check’ sequence of activities until a split exclusive gateway that provides at least one exit condition, and a loop condition that begins one or more ’after check’ sequences that rolling back to the first split gateway.

Once the SolutionTree has been built (as in Figure 8), the apply_rule_scheme function easily converts the tree structure into a MOISE ‘scheme’ via goals and plans. This requires the function below:

```
def apply_rule__plan(pattern: WorkflowPattern) : Elem = {
  pattern match {
    case SequencePattern(children) =>
```

```
      <goal id={get_goal_id}>
        <plan operator="sequence">
          {children.map(c => apply_rule__plan(c))}
        </plan>
      </goal>

    case ChoicePattern(children) =>
      <goal id={get_goal_id}>
        <plan operator="choice">
          {children.map(c => apply_rule__plan(c))}
        </plan>
      </goal>

...
   case ActivityPattern(task) =>
     val id = task.grounding.capability.id
       <goal id={id}></goal>

  }
}
```

It is a recursive rule, that browse the tree structure and incrementally builds the ‘goal/plan’ decomposition. A SequencePattern is translated into a sequence ‘plan’, a ChoicePattern is translated into a choice ‘plan’, whereas the Structured Cycle Pattern is a bit more articulated composition of sequences and choices. The exit condition for the recursion is the Activity (i.e. the leaves of the tree): each activity is translated into a simple ‘goal’ element.

Just to provide and example, the output of the previous rule application is reported below:

```
<functional-specification>
  <scheme id="scheme1">
    <goal id="root_goal">
      <plan operator="sequence">
        <goal id="prepare"></goal>
        <goal id="goal_1">
          <plan operator="sequence" type="maintain">
            <goal id="goal_2">
              <plan operator="sequence">
                <goal id="discussion"></goal>
                <goal id="voting"></goal>
              </plan>
            </goal>
            <goal id="check_exit_goal_3"></goal>
          </plan>
        </goal>
      </plan>
    </goal>
    <mission id="management_scheme1" min="1" max="1">
      <goal id="root_goal"></goal>
    </mission>
    [...]
  </scheme>
</functional-specification>
```

4. Conclusions and Future Works

The proposed approach consists in the automatic generation of a MOISE organisation from the BPMN specification of a business process with the support of a specific tool. The organisation is conceived to provide system adaptation because it includes several alternative plans (schemes) for pursuing goals. The approach is complemented by a tool for processing BPMN (XMI) code and generating the MOISE organization specification code. The approach is based on a few fundamental steps: 1) the automatic extraction of goals by processing the BPMN activities and their dependencies, 2) the identification of the initial state for agents execution (and solutions computation), 3) the extraction of the ontological vocabulary from goal definitions, 4) the calculation of one or more solutions for the achievement of process objectives by using agents' capabilities, 5) the generation of a MOISE organization composed of several alternative schemes, one for each computed solution. Alternative schemes in the agent organization can be selected according to performance criteria or to overcome a failure thus achieving some system adaptation. The generation of the organization is supported by a tool that uses well known workflow patterns to process the input BPMN XMI code and to generate the MOISE specification.

The availability of different organization's schemes allows to select the scheme (goal decomposition tree and set of missions) that provide the best performance, according to the quality attributes registered in the yellow pages. The approach also allows to replace the scheme that is in execution with another one, in case of agent/service failures thus obtaining a runtime adaptation feature.

So far, the tool supports the whole BPMN 2.0 specification for defining the process, but the Sub-Process element, that is going to be used to create hierarchical groups. The language for specifying Capabilities is a proprietary format. Very soon, we will move towards open-access action specification languages (PPDL for instance). Currently, the tool has some limitations: 1) parallel gateways are understood by the BPMN2Goal parser but not supported in phase of Goal2MOISE generation; 2) timer/clock events are partially supported because they are translated into first-logic predicates that require the agent society uses a specific internal synchronization mechanism.

References

[1] P. A. Buhler, J. M. Vidal, Towards adaptive workflow enactment using multiagent systems, Information technology and management 6 (2005) 61–87.

[2] M. P. Singh, M. N. Huhns, Multiagent systems for workflow, Intelligent Systems in Accounting, Finance & Management 8 (1999) 105–117.

[3] S. Ceri, P. Grefen, G. Sanchez, Wide-a distributed architecture for workflow management, in: Research Issues in Data Engineering, 1997. Proceedings. Seventh International Workshop on, IEEE, 1997, pp. 76–79.

[4] A. S. Rao, M. P. Georgeff, et al., Bdi agents: from theory to practice., in: ICMAS, volume 95, 1995, pp. 312–319.

[5] D.-S. Coalition, A. Ankolekar, M. Burstein, J. R. Hobbs, O. Lassila, D. Martin, D. McDermott, S. A. McIlraith, S. Narayanan, M. Paolucci, et al., Daml-s: Web service description for the semantic web, in: The Semantic Web-ISWC, Springer, 2002, pp. 348–363.

[6] O. Boissier, R. H. Bordini, J. F. Hübner, A. Ricci, A. Santi, Multi-agent oriented programming with jacamo, Science of Computer Programming 78 (2013) 747–761.

[7] R. H. Bordini, J. F. Hübner, M. Wooldridge, Programming multi-agent systems in AgentSpeak using Jason, volume 8, John Wiley & Sons, 2007.

[8] A. Ricci, M. Viroli, A. Omicini, Programming mas with artifacts, in: International Workshop on Programming Multi-Agent Systems, Springer, 2005, pp. 206–221.

[9] M. Hannoun, O. Boissier, J. S. Sichman, C. Sayettat, Moise: An organizational model for multi-agent systems, in: Advances in Artificial Intelligence, Springer, 2000, pp. 156–165.

[10] M. Chinosi, A. Trombetta, Bpmn: An introduction to the standard, Computer Standards & Interfaces 34 (2012) 124–134.

[11] L. Sabatucci, M. Cossentino, Supporting dynamic workflows with automatic extraction of goals from bpmn, ACM Transactions on Autonomous and Adaptive Systems (TAAS) 14 (2019) 1–38.

[12] L. Sabatucci, M. Cossentino, Self-adaptive smart spaces by proactive means–end reasoning, Journal of Reliable Intelligent Environments 3 (2017) 159–175.

[13] M. Cossentino, S. Lopes, L. Sabatucci, Goal-driven adaptation of moise organizations for workflow enactment, in: Proc. of the 8th International Workshop on Engineering Multi-Agent Systems (EMAS 2020), National Research Council of Italy, 2020.

[14] Object Management Group (OMG), Business Process Model and Notation (BPMN 2.0) by Example, Available online at https://www.omg.org/cgi-bin/doc?dtc/10-06-02.pdf, 2010.

[15] L. Sabatucci, M. Cossentino, From means-end analysis to proactive means-end reasoning, in: Proceedings of the 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS), IEEE Press, 2015, pp. 2–12.

[16] W. M. van Der Aalst, A. H. Ter Hofstede, B. Kiepuszewski, A. P. Barros, Workflow patterns, Distributed and parallel databases 14 (2003) 5–51.