Flexible and Emergent Workflows using Adaptive Agents

Arcady Rantrua, Marie-Pierre Gleizes, Chihab Hanachi

Institut de Recherche en Informatique de Toulouse
University of Toulouse
Toulouse, France

{rantrua,gleizes,hanachi}@irit.fr

Abstract. Most of existing workflow systems are rigid since they require to completely specify processes before their enactment and they also lack flexibility during their execution. This work proposes to view a workflow as a set of cooperative and adaptive agents interleaving its design and its execution leading to an emergent workflow. We use the theory of Adaptive Multi-Agent Systems (AMAS) to provide agents with adaptive capabilities and the whole multi-agent system with emergent "feature". We provide a meta-model linking workflow and AMAS concepts, and the specification of agent behavior and the resulting collaborations. A simulator has been implemented with the Make Agent Yourself platform.

Keywords: workflow; multi-agent system; flexibility; adaptation; emergence

1 Introduction

In some dynamic and unpredictable process-oriented applications, processes can not be defined by explicitly specifying a priori all possible alternatives of execution. For example, in crisis management or innovative design, some guidelines should be followed and the goal reached, but the process definition and its execution with its possible alternatives should be subject to consultation and collective choices at runtime. In other words, the process emerges since its definition and execution interleave. A partial and high-level process is set up, detailed, maintained and adapted in a collective way according to the crisis evolution, appreciated through the feedback information from the field, the availability of resources and the tasks pre-conditions. Also, tasks pre-conditions could be relaxed to be more reactive and efficient, and resources may evolve in an unpredictable way (damages or reinforcements).

Although, these kinds of applications require adaptiveness and emergence, they also have to guarantee that we have followed a logical process (a set of coordinated tasks) since they could engage the responsibilities of actors in crisis situations and also because the process followed is a result to be reused, explained, improved (in both crisis and design situations).

Given this context, the problem addressed in this work is: *how to build a flexible workflow system supporting emergent processes i.e. that ensures the interleaving of definition, execution and adaptation of processes?*

Research in the area of workflow flexibility is becoming increasingly important as a result of organizational demands to develop applications that have to face dynamic

and unpredictable situations. A comprehensive survey of the area can be found in [2] and [3]. Four types of flexibility have been identified: flexibility by design, flexibility by deviation, flexibility by under-specification and flexibility by change. While most of them are still based on procedural languages that are rigid to represent domains with a lot of variability, we have to mention the case-handling approach, based on declarative rules, that supports the personalization and adaptation of pre-defined processes [4]. Also, tools (like WADE [5]) and applications [6] based on the agent paradigm have been built to support workflow flexibility, implementing them as co-operative agents. However, in the best of our knowledge, the idea to focus on emergent workflows and to support the interleaving of definition, execution and adaptation have only been investigated by a few studies (see [1]).

Our approach, described in this paper, is to agentify the components involved in a workflow according to the agent paradigm and have them cooperate and adapt to ensure the emergence of a relevant process. Our model is grounded on the AMAS theory (see Section 3).

The contribution of the paper is threefold:

- The definition of a meta-model that links the AMAS and the workflow concepts.
- A specification of an agent-based workflow system based on the AMAS theory.
- The WOrkflow Adaptive Simulator (WOAS) compliant with this specification and implemented with the MAY[1] Platform.

The paper is organized as follows. Section 2 provides a short state of the art on flexibility in workflows. We then introduce the AMAS theory and its adequacy to deal with adaptability and emergence. Section 4 presents the specification of our system: the meta-model, the agent structure and behaviour and agent collaboration protocol. Section 5 provides an overview of the WOAS simulator and experimentation. Finally, we discuss our approach and conclude the paper.

2 State of the Art

Workflows are designed through three different and interrelated perspectives giving rise to three conceptual models: process (behavior), organization and information. Also, design and execution are most of the time two separated phases.

In this state of the art, we will compare existing works about workflow flexibility according to three criteria: the design approach followed (procedural, event driven, emergent), the flexibility techniques followed and their impact: life-cycle phase (design and/or execution), workflow model (process, organization, information).

Design approach. Traditional workflows are most of the time represented by procedural languages (Petri Net based formalism such YAWL, BPMN, BPEL, ...) relevant for routines, but too rigid to face dynamic and unpredictable applications. Some other systems, called declarative, are designed through an event-based approach that speci-

[1] Make Agents Yourself (http://irit.fr/MAY-en)

fies the workflow behavior through reactions to perform when some events occur. The Wide system of Casati [7] follows this approach by using the active rule formalism. The two previous approaches assume a centralized workflow engine controlling the execution of the workflow, and do not provide means to ease interactions between actors to negotiate and solve problems due to interdependant tasks. A third approach, called agent-based workflow, takes a more radical choice, fully rethinking the system in terms of cooperative and problem-solving agents and distributing the control over the agents. Even in this last approach, we can distinguish two classes:

1. *Design-execution separation:* the process models (control, information, organization) are pre-defined and distributed over different agents, each one being able to play a role, deciding by itself or after negotiation how to play it. The WADE system [5], one of the most well-known platforms, follows this approach.
2. *The emergent approach* involves agents having capabilities to design and execute the process. Even if some tasks, information and roles could pre-exist, agents have the capabilities to change or adapt all of them, and to elaborate the whole process in real time, interleaving design, adaptation and execution in an opportunistic way.

The ADEPT system [6], stands between these two cases. Indeed, in ADEPT, Agents' organization (abstracted as agencies), tasks, and the handled information by each task are predefined. On the contrary, the whole process is neither predefined nor designed, but the result of agents' interactions and negotiations. To the best of our knowledge, very few agent related works have fully investigated the emergent approach providing support for interleaving design and execution as required in dynamic and unpredictable situations. However, we can mention the Dynamic Engineering Design Processes (DEDP) approach used in the PSI project by [1]. Our work differs from [1] by its more autonomous and emergent approach made possible thanks to its theoretical background: the AMAS Theory in our case (see Section 3).

Flexibility Techniques. In addition to the previous approaches or paradigms, several works on workflow flexibility have been focused on adding flexibility to procedural languages. Four types of flexibility have been identified in [2]: flexibility by design, flexibility by deviation, flexibility by underspecification (late binding and late modeling) and flexibility by change.

The two most interesting techniques for our purpose are underspecification and flexibility by change, that are the most dynamic since they integrate the possibility to model or change part of a process during its execution. One of the most advanced concrete solutions, following the flexibility by change technique, is the "case handling" technique formalized by Van Der Aalst et al. [4] and implemented in the FLOWer system that allows the actors to change the process by focusing on what can be done instead of what should be done. This technique is dynamic but not emerging.

Although all these techniques improve workflow flexibility, they remain applied to the procedural approach that constraints their expressive power.

Flexibility impacts. Most research works discussed in this section has been focused on bringing flexibility to the workflow process model. The agent-based approach is probably the most dynamic and flexible in terms of control structure and organization. The Case Handling technique is one of the very few approaches that addresses flexibility regarding the three models. In fact, most of these works separate design and execution time, and whatever the approach followed (procedural, event based or agent), main decisions about the three models are decided at design time and the user has little influence on the three models during execution.

In conclusion, even if numerous works have addressed the workflow flexibility issue, none of them deals with emergent workflow which authorizes to interleave design and execution. This is regretful as this feature is known to be useful to face dynamic and unpredictable situations. However, the agent-based approach and some ideas from the case-handling technique are useful to support emergent workflows as we will show in our proposition.

3 Background on Adaptive Multi-Agent System

MAS are composed of interacting autonomous heterogeneous software entities or components called agents [9]. This approach makes the modeling, design and simulation of complex systems easier thanks to their capabilities: local representation of the knowledge of the agents (constraints and objectives); use of interaction / negotiation between agents; physical and processing distribution; decentralized decision making; autonomous behaviors, openness (new agents can appear and disappear at any time). All these capabilities contribute to accurately represent complex systems and simulate them. This is needed when the global behaviors (such as dynamic workflows) emerge from interdependent local interactions and reactions (such as resources breakdowns or unpredictable events).

The AMAS (Adaptive Multi-Agent System) theory is based on the general property asserting that the system realizes the required function when all its components (the agents) are in cooperative situation [10]. Consequently, a self-organized process locally guided by a cooperative attitude leads to a dynamic equilibrium such that each agent reaches (or tends to) its individual goals and the system reaches an adequate collective functionality. This assumption is the foundation of the AMAS approach that considers cooperation as the criterion for self-organization [11]. Basically, every agent pursues a local objective and interacts with others while respecting cooperative rules that guide its interactions in order to avoid and possibly remove situations that are judged as non cooperative from its local point of view. These situations are called Non-Cooperative Situations (NCS for short).

The benefits could be expressed in terms of efficiency, decentralization (minimizing shared knowledge, maximizing privacy), openness (adding/removing agents or knowledge) and robustness (minimizing problem-dependant parameters). This means that the system has the capability, through the agents composing it, to adapt to global or local changes at runtime, and to provide close solutions between changes, since changes are locally processed by agents; contrary to global centralized algorithms

requiring starting the resolution process again from scratch at each change. This cooperative approach has already been applied to several domains such as adaptive profiling, multi-criteria optimization, dynamic control of complex systems or manufacturing control.

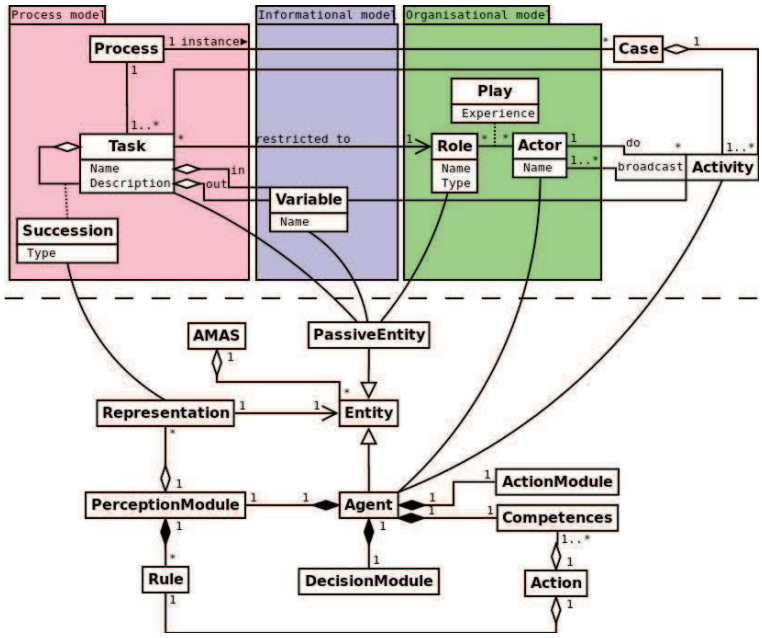

Fig. 1. Metamodels of AMAS and Workflows linked together

4 Modeling Emergent Workflow by Means of AMAS Theory

4.1 The Principles of our Approach

The required function of our system is to produce a workflow. We model this workflow as a set of entities where activities and actors are active entities (or agents) able to decide by themselves actions to perform and tasks, roles and variables are passive entities (or resources) handled by the agents.

More precisely, we consider two categories of agents:

- *Software agents* solve non-cooperative situations to try to stabilize their neighborhood following the AMAS theory. Their individual goal is to guarantee the local consistency of the workflow. They also automatically make emerge parts of the workflow that ease the agents' perception of the design evolution.
- *Human agents* are in charge of building the whole workflow. In our experimentation, they will be represented by artificial agents. Their goal is to reach a dynamic equilibrium corresponding to the situation where the designed workflow is consistent and accepted by every actor.

The components of the workflow cooperate to decide the workflow structure. The workflow itself may be viewed as a multi-agent system producing its own design through self-organization.

4.2 The Metamodel for Linking Adaptive Multi-Agent Systems and Workflow Concepts

Figure 1 links AMAS concept (top part) and workflow concepts (bottom part) with cross-domain association. Actors are active entity, or agent (entity that can autonomously make decisions). Actors have been agentified because they have knowledge about what can and should be done. Activities are also agentified because they actively work to solve the conflicts that can occur between the agents. The other entities, namely variables, tasks and roles, are passive. They are part of a library of resources which is used by the agents to represents the environment (the workflow).

Fig. 2. Use Case Diagram of the System

4.3 Agents' Actions and Behavior

Agents' actions. As shown in figure 2 we have three types of agents that can perform several actions impacting the three perspectives through the manipulation of data (inputs, outputs, profile), groups, activities and coordination pattern.

The "simulate" actions implements the interleaving of runtime and design time

concepts by allowing a software agent to simulate the execution of an activity and see the consequences. That is how our system can check the validity of the design and try to fix it before the real execution.

Activity Structure. An activity has a name (usually a verb representing the action that it does, like "firefighting") and a set of input/output data named profile. Those activities can be executed by an actor if he/she has the corresponding capacity. In order to represent the different ways for executing an activity our model accepts the coexistence of multiple capacities with the same name but with different profiles. In case of a conflict, actors trigger a negotiation protocol to agree on a profile.

An activity also features two coordination patterns, linking it to its previous and following activities. It may be: a sequence, an alternative (if), a parallel control structure or a synchronization. An activity is associated to a group that contains actors with common interests (it usually means that each group member can execute the activity).

4.4 Agent Collaboration and Conflict Resolution

Cooperation is triggered by NCS. The Petri net of figure 3 describes the protocol that handles an NCS from its detection to its repair. When an agent detects a NCS, it triggers a repair action to reach a cooperative situation.

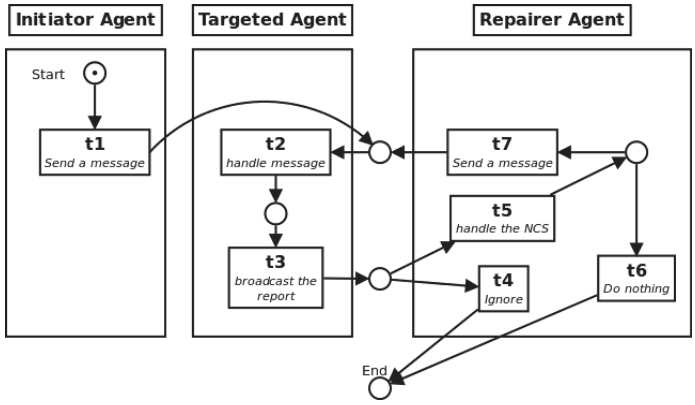

Fig. 3. Petri net of NCS handling

At the beginning an initiator agent (IA) sends a message to a targeted agent (TA) asking it for a modification (e.g. if the targeted agent is an activity it could be asked to adapt its profile). TA handles the requests (t2) and then broadcasts the report (t3) to all the agents interested by the change (e.g. any member of the activity's group). One agent among them will play the role of repairer. The repairer agent (RA) can find the change cooperative and ignore it (t4). Or, it could discover an NCS and handle it (t5). It can choose between: a) doing nothing (t6), sometimes there is no other way and one of the agents must accept a sub-optimal situation for itself, and b) asking TA to modify itself again by sending it a message (t7). This protocol could loop until one of the

following situations is met:

- All the potential repairer agents agree with the change: there are no more conflicts (they all perform t4).
- Some repairer agents don't agree but decide to do nothing (they perform t6) to help the convergence of the system.

Repairer agents detect when they are in an NCS and trigger the corresponding rule. We distinguish two categories for these rules:

- Those that keep the workflow valid. For example, finding a compatible profile when actors' capacities are in conflict. Currently the (very trivial) repair action is to have the profile of any agent be a subset of the activity's profile.
- And those that allow the workflow to be built faster. For example, if an actor A participates in an activity T and realizes that the outputs of T are compliant with the inputs of one of its capacities. Then A will instantiate this capacity as a new activity T' and connect it to T.

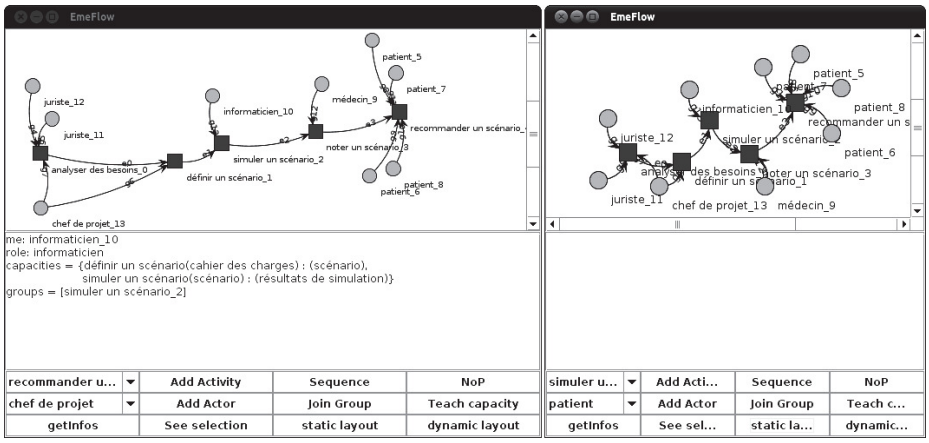

Fig. 4. Two WOAS' windows representing the same workflow (left: 4a and right: 4b)

5 Validation: The WOAS Simulator

To validate our approach we have built the WOAS simulator and tested it on a case study. This case study is about the co-design of an healthcare product (improving homecare). It involves actors from several domains: medical, law, computer scientists, sociologists and patients.

During our tests some of those roles were played by human actors. Each of them could access a shared board (see the interface in figure 4a) in which he/she could co-design a workflow.

Figure 5 shows part of a design session as a sequence diagram with real human actors. The computer scientist creates a group (1) and adds himself to this group (2). He then allocates this group to the activity "user requirement analysis" (3). As a default reaction to any allocation, the activity, which is a software agent decides to respond

(4) with its information (profile, allocated actors, …). The computer scientist (a human agent) decides to add the project manager to his/her group (6) so they can work together on the "user requirement analysis". Again, the activity responds with its information (7). By accessing to the profile of the activity, the project manager checks if he/she accepts the profile of the activity (8). This is not the case and he/she decides to modify it (9, 10).

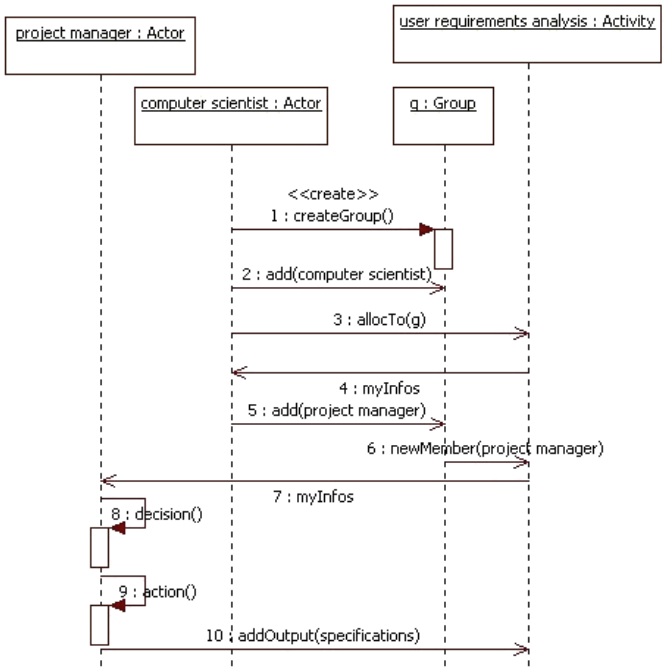

Fig. 5. Sequence diagram of a co-design session

Our system provides the following services:

- It allows each agent to add an activity in a shared space, connect it to other activities with a coordination pattern (sequence, parallelism, synchronization, conditional structure), allocate it to a group of actors, modify the profile (input data, output data) of an activity.
- It allows the user to add actors to the workflow and inside a group.
- It allows actors to negotiate the activities' profile and the workflow structure.
- It allows visualizing the workflow evolving in real-time.

Our system is implemented using SpeADL/MAY [8], a tool allowing to build architectures that are adequate for the development and the execution of multi-agent systems. It also comes with a set of reusable components providing features like communications or scheduling leading to a more lightweight development.

6 Discussion and Conclusion

This paper defines and specifies a new approach for supporting flexible and emergent workflow systems. This is made possible with the interleaving of design, adaptation and execution phases. Actors can decide and specify in a cooperative way, using a negotiation process, the tasks profiles and the group allocated to a task. This work is grounded on the AMAS theory that supports adaptation and conflict resolution. AMAS and workflow concepts have been linked through a unifying meta-model. A simulator called WOAS has been implemented and it demonstrates the feasibility of our solution in a distributed mode. Nevertheless several issues remain open:

- WOAS remains a simulator. It requires, for real world applications, the integration of an execution module and to take into account real world feedbacks that could probably lead to improvements of the tool.
- In our model and simulator some negotiations protocols are explicit and simple, others are complex and emergent. It would be interesting to discover and represent the emergent ones to better understand the emergence phenomenon and to make them available as resources for agents.

7 References

1. Vadim Ermolayev, Eyck Jentzsch, Oleg Karsaev, Natalya Keberle, Wolf-Ekkehard Matzke, Vladimir Samoilov: Modeling Dynamic Engineering Design Processes in PSI, Seventh International Bi-conference Workshop on AOIS, 119-130 (2005)
2. H. Schoneneberg, R. Mans, N. Russell, N. Mulyar and W. van der Aalst: Process flexibility: asurvey of contemporary approaches, International Workshop on CIAO/EOMAS, International Conference on Advanced Information Systems, 16-30 (2008)
3. S. Nurcan: A survey on the flexibility requirements related to business process and modeling artifacts, Hawaii International Conference on System Sciences, 378 (2008)
4. W. van der Aalst, M. Weske and D. Grünbauer: Case handling: a new paradigm for business process support, Data Knowl. Eng. 53(2), ACM, 129-162 (2005)
5. F. Bergenti, G. Caire and D. Gotta: Interactive workflows with WADE, WETICE (2012)
6. N. R. Jennings, T. J. Norman, P. Faratin, P. O'Brien and B. Odgers: Autonomous agents for business process management, Applied Artificial Intelligence, 14(2), 145-189 (2000)
7. F. Casati, L. Baresi, S. Castano, M.G. Fugini, I. Mirbel and B. Pernici: WIDE Workflow Development Methodology, 19-28 (1999)
8. V. Noël: Component-based software architectures and multi-agent systems: mutual and complementary contributions for supporting software development, Ph.D. Thesis (2012)
9. G. Weiss: Multiagent systems. A modern approach to distributed artificial intelligence, The MIT Press (1999)
10. V. Camps, M.-P. Gleizes and P. Glize: A theory of emergent computation based on cooperative self-organization for adaptive artificial systems, Fourth European Congress of Systems Science (1999)
11. P. Glize and G. Picard: Self-organisation in constraint problem solving, Self-organising Software From Natural to Artificial Adaptation, Giovanna Di Marzo Serugendo, Marie-Pierre Gleizes, Anthony Karageorgos (Eds.), Springer-Verlag, 347-377 (2011)