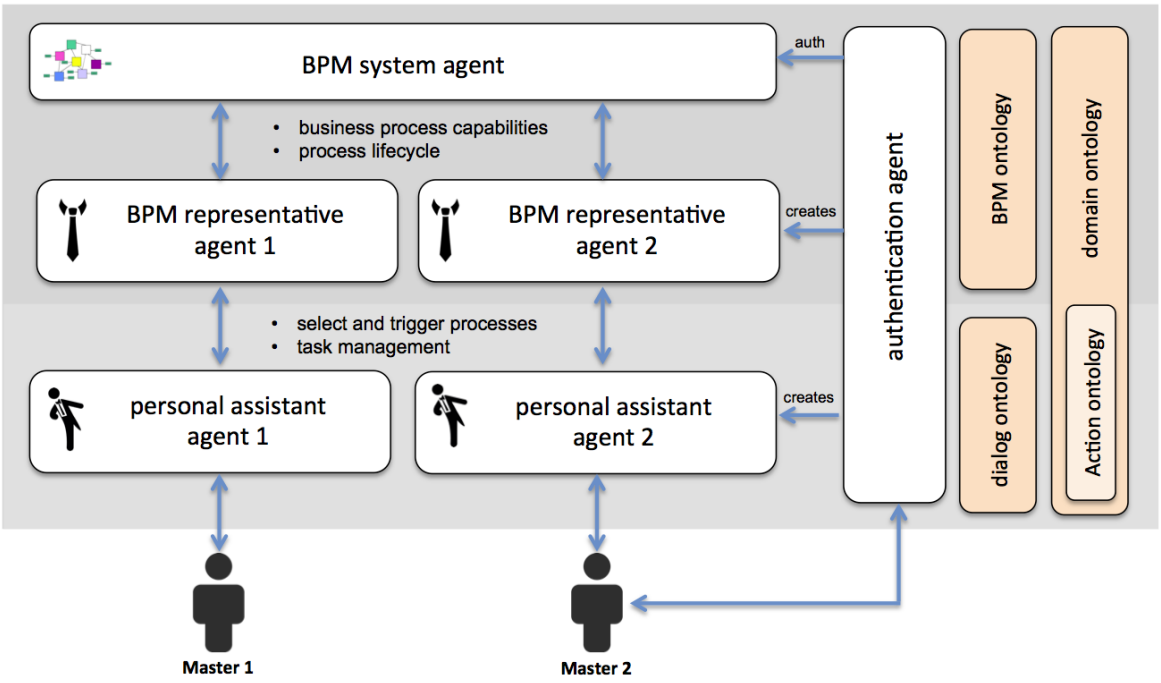
Par Marcio Fuckner

A Personal Assistant for the Enactment of Business Processes

Thèse présentée pour l'obtention du grade de Docteur de l'UTC



Soutenue le 22 avril 2016
Spécialité : Technologies de l'Information et des Systèmes

Soutenue publiquement le 22 avril 2016 devant le jury composé de :

M., BARTHES, Jean-Paul
Professeur, Université de Technologie de Compiègne
Directeur

Mme., DARENE, Nathalie
Professeur, Université de Technologie de Compiègne
Membre

M. MANDIAU, René
Professeur des Universités, Université de Valenciennes
Président et Rapporteur

M., PARAISO, Emerson Cabrera
Professeur, Université Pontificale Catholique du Parana
Membre

M., RAMOS, Milton Pires
Professeur, Institut de Technologie du Parana
Rapporteur

M., SCALABRIN, Edson Emilio
Professeur, Institut de Technologie du Parana
Directeur

A Personal Assistant for the Enactment of Business Processes

Márcio Fuckner

UMR CNRS 7253 Heudiasyc
Université de technologie de Compiègne

This dissertation is submitted for the degree of
Docteur de l'Université de technologie de Compiègne

September 2016

Acknowledgements

The present thesis is the product of more than three years of work. Many of the ideas behind this work were first sketched during my employment at the HSBC Bank as a software engineer and IT training leader, and then further developed upon starting a Ph.D. at the University of Technology of Compiègne. Needless to say, this project could not have been completed without the help and support of the many people that have accompanied me on this project.

I would like to gratefully acknowledge the efforts of my supervisors Prof. Jean-Paul Barthès and Edson Emilio Scalabrin. Their guidance and encouragements on my research work have been invaluable. They also contributed with advice on several revisions of this dissertation and played a key role in helping me improve the form and content of the final draft. I am particularly grateful to Marie-Hèléne Abel for welcoming me to the Information, Knowledge and Interaction group (ICI) at the Heudiasyc laboratory.

I naturally want to thank my colleagues for the great working environment and friendly atmosphere during all these years. I especially wish to express my gratitude to all my workmates for making me feel welcome and part of this fantastic group from day one. Likewise, I want to thank my previous colleagues of the HSBC Bank for many inspiring discussions on business processes and human interaction that contributed to shaping the direction of this thesis.

Last but certainly not least, I am without words to thank my wife and life partner, Ana Maria, for all her love and support through the years. Thank you for everything.

Abstract

Over the last few years, the advances in management science and information technology have transformed the business process management (BPM) discipline into an important topic for both industry and academy. BPM uses business processes as the means for improving the operational performance of organizations, and setting processes are at the heart of BPM allows linking together people, systems, and different organizations to deliver value to stakeholders. The target of our work is the family of BPM systems. A BPM system is a generic software system that is driven by explicit process designs to enact and manage operational business processes. Despite the wide range of topics addressed by the academy on business processes, there are still aspects not addressed by prior research. A particular problem in this regard is the mediation between BPM systems and humans. Human interaction in those systems follows a standard user interface based predominantly on work item lists and forms. Thus, there is little room for creativity for users. They have not only difficulties in enacting their processes but also for searching the most suitable one for their needs. It would be more efficient to let humans interact in natural language. However, process modeling languages are an insufficient means of capturing and representing the domain of discourse. The present thesis develops an original approach to agent dialog management for the problem of business process enactment. The overarching motivation for this work was to design a dialog model scalable to different domains. The model relies on domain and business process ontologies, and necessitates a minimum effort of adaptation on ontologies to improve the interaction. Results indicate the potential of our agent-based approach to generate natural language interfaces, without needing to rebuild the whole business process model.

Résumé

Ces dernières années, les progrès en sciences de la gestion et de l'information ont transformé la Gestion de Processus d'Affaires (*Business Process Management, BPM*) en un sujet important, tant du coté de l'industrie que de celui de la recherche. Le BPM utilise des processus métiers pour améliorer la performance opérationnelle des organisations. Les processus métiers établissent un lien entre les personnes, les systèmes, et les différentes organisations, dans le but de créer de la valeur pour les parties prenantes. La cible de notre travail est la famille des systèmes BPM. Un système BPM est un système logiciel générique guidé par des modèles explicites de processus métier avec pour objectif d'exécuter et de gérer des processus opérationnels. Malgré le vaste éventail de sujets traités par ce domaine de recherche, il reste encore quelques questions qui méritent une étude plus approfondie. Un problème particulier concerne la médiation entre les systèmes BPM et les humains. L'interaction homme-machine dans ces systèmes repose sur des interfaces standard basées sur des listes de tâches et des formulaires, ce qui est très contraignant pour les utilisateurs. Ceux-ci ont non seulement des difficultés à exécuter leurs processus métier, mais aussi à trouver le processus métier le mieux adapté à leurs besoins. Il serait beaucoup plus efficace d'utiliser des dialogues en langage naturel. Malheureusement les langages de modélisation de processus ne permettent pas de capturer ni de modéliser un domaine de discours. Le travail présent propose une approche originale de gestion du dialogue basée sur des systèmes multi-agents pour l'exécution des processus métier. La motivation globale pour ce travail fut de concevoir un modèle de dialogue extensible à différents domaines. Ce modèle s'appuie sur les ontologies de domaine, nécessitant un minimum d'effort d'adaptation pour améliorer l'interaction. Les résultats montrent tout le potentiel de notre approche multi-agent pour réaliser une médiation automatiquement, sans qu'il soit nécessaire de reconstruire les modèles de processus métier.

Table des matières

Table des figures

Liste des tableaux

Chapitre 1

Introduction

Over the last few years, the advances in management sciences and information technology have transformed the *business process management* (BPM) discipline into an important topic for both industry and academy. BPM uses *business processes* as the means for improving the operational performance of organizations, and processes are at the heart of BPM, linking together people, systems, and different organizations to deliver value to stakeholders.
BPM has become a mature discipline, with relevance acknowledged by practitioners (users, managers, analysts, consultants, and software developers) and academics (see Aalst in [103]). It could be confirmed by its constant presence among top-ranked information system conferences [86] [13] [76] [58]. Subjects range from organizational aspects (e.g. process model analysis, process flexibility and process reuse) to more specific issues (e.g. process modeling languages, process enactment infrastructures or process mining) [103].
We have a particular interest in a family of systems called *Process-Aware Information System* (PAIS). A PAIS is a software system that manages and executes operational processes involving people, applications, and information sources on the basis of process models (van der Aalst in [101]). In general, a PAIS follows a rigid approach, enforcing that the enactment of a process follow precisely what the process model specifies (e.g. traditional workflow management systems), without much room for flexibility. Needless to say, the process model is an essential artifact that determine the level of flexibility of information systems.

1.1 Problem Statement

Despite the wide range of topics addressed by the academy on business processes, there are still aspects not addressed by prior research. A particular problem in this regard is the

mediation between BPM systems and humans. The control structure of the process has a direct influence on the user experience, but the inverse is also true. According to Weske [110], human interaction in those systems follows a standard user interface based predominantly on work item lists and forms. Thus, there is little room for creativity for the user, since he or she must wait for the arrival of a work item to interact with the system. Weske also argues that this crude style of interface has not been considered appropriate for human workers.

Even with a high impact on the human resource perspective, influencing how people work, this topic does not draw much attention in research and industry (Pesic in [77], Russell and van der Aalst in [84] and [83]). Recent industrial standards such as BPEL4People [27] and *WS-Human Tasks* [26] are efforts aiming at enriching the resource perspective of workflow technology. However, they only take into account the allocation perspective, neglecting the assistance for humans during their activities. Also, the complexity of applications is still augmenting with the emergence of a new generation of pervasive and ubiquitous applications. This type of system makes use of different modalities of user interface (natural language with spoken and written interfaces, the use of gestures and virtual reality environments) that dramatically increase the combinations of actions during the usage of an information system. According to our vision, we believe that BPM requires research, leading to a new generation of tools for helping the enactment of processes. They should help users to reduce the ever-growing load of information, events and various commitments they need to handle. BPM enactment interfaces should hide the complexity of difficult tasks, by asking clarifying questions about the process, by performing tasks on behalf of the user, by training the user, by helping different users to collaborate and so on. These users have not only difficulties in enacting their processes but also for searching the most suitable one for their needs. Most of the reasons resides in the fact that process modeling languages like WS-BPEL [114] and BPMN [73] are an insufficient means of capturing and representing such a domain of discourse [44]. Several authors have worked on the problem of querying the process space (e.g. Leymann [60], Hepp et al. [45] [46], Kim and Suhh [52], and Ribeiro et al. [81]). Most of the approaches are dedicated to providing enough information to bridge the gap between domain experts and IT members. However, less attention has given on querying the process space based a story told by the user (e.g. based on assertions, questions, and directives). We argue that both the issue of poor interfaces and the difficulty in finding process are sufficient reasons for evaluating alternatives of interaction between BPM enactment systems and users.

Another aspect that is more a motivation than a problem, but could be an issue for organizations in the short term is the growing acceptance of natural language (written or

spoken) as a common interface for applications. We have conducted a survey, detailed in Chapter 3, applied to 50 participants of organizations. The results reinforce our perceptions : Only 5% of enterprise applications allow natural language written or spoken in their interfaces. But more than 80% believe that a natural language interface (written or spoken) as an input could improve the performance of daily activities and 90% have used personal assistants in their mobile phones (e.g. Siri from Apple, Google+ or Cortana from Microsoft). As can be seen, at the time of writing, this type of interface didn't reach organization at large scale (e.g. Workflow Systems, ERP's, CRMs and so on). However, companies will be naturally pushed to improve their systems, especially those intended for customer services (e.g. banking services, public services). One natural way to improve enterprise interfaces is to endow applications with a capacity to understand, even in a limited manner, the medium of information exchange that is most intuitive to human beings.

1.2 Hypothesis

We hypothesize that the inclusion of a personal assistant agent bridging humans and traditional BPM systems during the enactment of processes could improve the user experience, resulting in more flexible and simple interfaces. More precisely, we argue that a conversational interface could be appropriate for business processes, since, although complexes, business processes are constrained. That is to say, most of the domain-related keywords are known, as well as concepts and their relations.
When using a personal assistant, users could request something to the assistant almost in the same manner that they do to a colleague, using natural language. In some situations, users have a clear goal in mind, and it is easy for them to formulate an appropriate request (e.g. "I want to make a deposit"). However, there are cases in which users do not have enough information to built a proper request. In these situations, a personal assistant could act as a colleague or a clerk that takes notes and finds a solution when the user goal gets clearer. So, users could feel more comfortable in describing their state ("I have some money to invest" or "I have lost my credit card"). The personal assistant might help users to find the most suitable process to execute, and ask clarifying questions when ambiguities are found. Personal assistants could also extract useful information during information filling. This set of facilities is virtually impossible using traditional BPM enactment system interfaces.
This natural interaction between users and BPM systems could allow a more self-service oriented and configurable user interface. Note that service orientation and

self-configuration is quite common in BPM systems, but mostly from the architectural and integration point of view. Several BPM systems have modern software architectures that use SOA principles and Cloud technologies to leverage the overall business process environment [110]. We believe that personal assistants could put user interface at the same level of flexibility that exists in the internal architecture of BPM systems. Moreover, personal assistants could improve the collaboration in knowledge-intensive business processes, leveraging the existing asset of organizations.

1.3 Motivation

The new generation of software architectures using SOA principles and the Cloud as a common infrastructure have opened the door to self-service and configurable applications. Most of our everyday activities are now relying on "smart" electronic devices of various kinds, from mobile phones to personal computers and navigation systems. Mobile services like Siri from Apple and Google Now are leveraging the natural language interface to enable a variety of functions to be accessed and controlled by end-users without worrying and memorizing sequences of commands to operate the software.
However, up to now these applications are mostly used for typical mobile applications like schedule management, Internet search, and entertainment. As these technologies gain in maturity, the design of appropriate user interfaces for enterprises becomes increasingly important. Human-computer interfaces of business applications should provide the user with rich and flexible communication channels while preserving the investment in existing software assets.

1.4 Contributions of our work

The present thesis develops an original approach to dialog management for the problem of business process enactment, called PA4Biz (*Personal Assistants for Business*). The overarching motivation for this work is to design a dialog model that is scalable to different domains. The model relies on domain and business process ontologies, yet necessitating a minimum effort of adaptation on ontologies to improve the interaction.
The contribution of our work is twofold, and could be broken into two categories. The first contribution is the characterization and description of business processes. The second contribution is the creation of a conversational interface for the selection and enactment of business processes.

A characterization of business processes

The contributions to the characterization of business processes are the following ones :

— An approach to describe and build a safe approximation of the capabilities of business processes. The novelty here resides in the way business processes are represented. They are defined in terms of preconditions and effects as inseparable parts. We allow the description of both deterministic and non-deterministic effects, the latter being quite useful for describing human tasks and services in non-deterministic scenarios. Our approach to semantic matching differs from other approaches since it defines business processes rather than web services.

— A method for querying the business process space. A flexible query mechanism based on preconditions and effects is a must for organizations with a growing business process space.

A conversational interface for business processes

The contributions to the conversational interface for business processes are the following ones :

— A dialog manager adapted to the business process selection problem. User sentences are interpreted and used as a source for selecting business processes. Users have different ways of expressing what they want. Sometimes they prefer using assertions to describe their current situation hoping that their interlocutor proposes a solution. Others are more direct and state their objectives using questions or directive acts. We propose an approach based on dialog acts and sentence analysis to select and discover processes.

— A dialog manager adapted to the business process enactment problem : Business processes have an independent lifecycle, a workflow engine performs its orchestration. That is to say, the dialog manager does not have control over events. Participants of the processes are allocated dynamically, following different strategies. Participants may be notified at any time, and the dialog manager can handle a series of asynchronous events.

1.5 Document Outline

The organization of the thesis is as follows :

— In Chapter 2, an overview of the discipline of Business Process Management is presented, which includes an overview of its lifecycle and the most known

techniques used to represent business process models. Moreover, we present an extensive analysis of the existing approaches towards flexibility in business process models.

— In Chapter 3, we provide an overview of concepts, tools, and techniques available in the literature to implement personal assistant agents. First, we discuss the concept of agents, multi-agent systems and their properties, with a particular focus on the business process aspect. We then present the evolution of a personal assistant platform developed in our laboratory, and its categorization using existing literature. We finish by showing the results of a survey, conducted to better understand users needs and their expectations about the capacities of a personal assistant in a corporate environment.

— In Chapter 4, we describe an approach for the characterization of business processes. We start by presenting a canonical business process model, describing the basic structure of business processes, the resource, and function perspective. A formalism to describe the capabilities of business processes is presented, allowing the definition of deterministic and non-deterministic effects. We finish by outlining our approach for querying the business process space.

— In Chapter 5, we explain in detail the conversational interface for the enactment of business processes. We first present our baseline for dialog management, as well as some examples of applications developed using this platform. We use an illustrative example to explain the dialog management mechanism and draw some conclusions and collect requirements to improve the existing baseline. Then, we present the overall architecture, our dialog manager, and the use of ontologies for the semantic annotation and information extraction. Finally, the mechanism for the selection, triggering and enactment of business processes using a conversational interface is described.

— In Chapter 6, we report the realization and evaluation of our functional prototype. First we present the technical architecture and its components. Next, we reviewed existing methods for evaluating dialog systems and built an equivalent counterpart application for comparison purposes regarding business process selection and enactment. We then describe the evaluation setup and the evaluation process. Results indicate the potential of our agent-based approach as an alternative interface, without needing to rebuild the whole business process model.

— Finally, we give our conclusion and discuss the future work in Chapter 7.

Chapitre 2

Business Process Management

The steadily growing interest in business process management from practice as well as from Academy clearly demonstrates the importance of this discipline in today's business environment [59]. According to van der Aalst [103], BPM can be seen as an extension of *Workflow Management* (WFM) that has a particular focus on the automation of business processes. BPM has a broader scope, providing tools and techniques to manage the whole lifecycle of business processes without neglecting the human aspect in the lifecycle of processes.
This chapter gives an overview of business process management and techniques to improve flexibility and usability. Section 2.1 provides a short historical outline and introduces the discipline of business process management by building on the business process lifecycle. Section 2.2 provides the basis for the characterization of process models. Section 2.3 classifies business processes according to the degree of human involvement and process structure. Section 2.4 gives an overview of the research towards flexibility in business processes. Section 2.5 explores in detail one approach towards process flexibility. Finally, Section 2.6 provides a brief summary of the chapter.

2.1 Overview of Business Process Management

BPM has its roots in both management science and computer science with contribution from several disciplines. Figure 2.1 presents an overview of some disciplines that have contributed to its development. During the pre-industrial revolution age, Adam Smith (1723-1790) describes the production of a pin as a process in a chapter dedicated to labor division in his book called *The Wealth of Nations* [93, p. 11-14]. He analyzes how to increase the performance of a process, through the use of labor division, dividing a complex process into a simple set of activities performed by specialized workers,

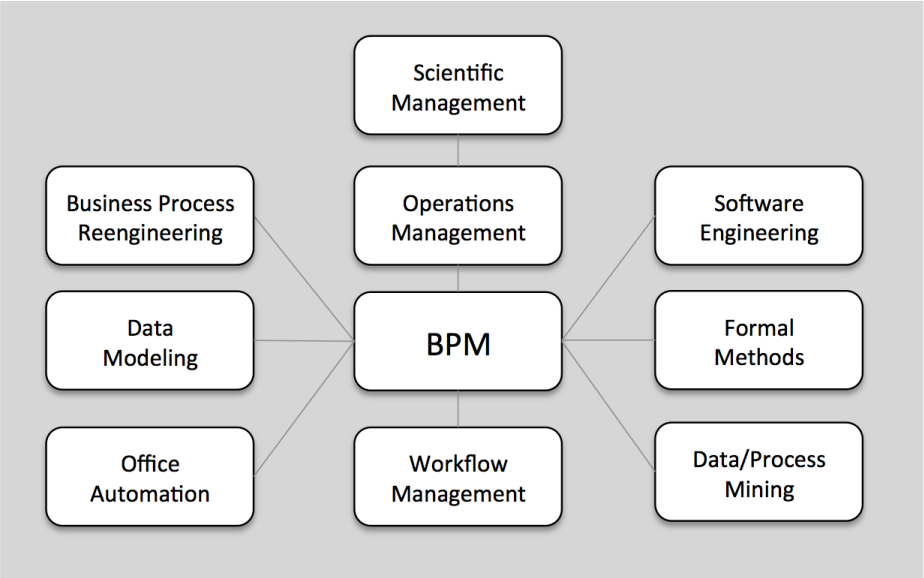


FIGURE 2.1 A list of some disciplines that contributed to the development of the BPM field [106]

registering the oldest known documented business process [47]. Almost a century later, Frederick Winslow Taylor (1856-1915) introduced the principles of **scientific management**, discussing the relevance of the division of work to the theory and practice around business processes. Henry Ford (1863-1947) had also introduced the production line for the mass production of cars. These events have given substantial contributions to BPM, especially the concept of *process-orientation* and *repetitiveness*.
From the fifties, the organization of work and business processes has been dramatically influenced by computers and digital communication infrastructures. Process modeling and **software engineering** techniques have become essential tools to manage the growing complexity of information systems. Several contributions have been proposed from the field of **formal methods**. Petri nets, introduced by Carl Adam Petri (1926-2010) in 1962, are one of the most prominent examples in the process-modeling domain. Nowadays, most of existing notations and process modeling standards use the same semantics adopted from his work such as the visual modeling of concurrency.
Data-driven approaches dominated the seventies and eighties. During this period, the focus of information technology systems was on data storage and retrieval [106]. Thus, **data modeling** was the starting point for modeling and building systems. In general, processes had to adapt to the existing information system and modeling was a neglected activity. Aalst et al. [106] argue that one of the reasons was the lack of consensus in the process modeling domain and the strong consensus on the modeling of data. Even today the

relational model by Codd and the Entity-Relationship Model by Chen [22] are academic and industrial references. Even with well-established formal methods such as Petri nets and process calculi, the industry has been pushing domain specific languages in their applications like BPMN (Business Process Modeling Notation) [73], BPEL (Business Process Execution Language) [114] and EPC (Event-Driven Process Chains).
In the nineties, Hammer and Champy introduced the discipline of BPR (Business Process Reengineering) [41]. They advocated the radical redesign of the business processes to improve customer service, cut operational costs and become world-class competitors. They claimed that information technology is the primary enabler of re-engineering programs of a company. Building on these management concepts, companies started to implement process orientation (see Weske [110, p 4]). Many WFM systems became available during this period. These systems focused on automating workflows with little support for process analysis, process flexibility, and process management. Years later, this technique has been criticized by several authors, claiming that BPR was a way to dehumanize the workplace, increase managerial control, and justify downsizing, i.e. major reductions of the workforce, as a rebirth of "Taylorism" under a different label (Vallabhaneni in [99]). Since then, some core concepts of the BPR have been used as a starting point for business analysis and redesign. However, it is adopted in a less radical way than originally proposed in the nineties. The concept of Business Process Management (BPM) has gained significant attention in the corporate world and can be considered as a successor to the BPR wave.
The core concept of BPM is the *business process*. We use the definition of *business processes* provided by Weske [110, p. 5] :

Definition 2.1.1. A *business process* consists of a set of activities, which are performed in an organizational and technical environment in a coordinated fashion. In this manner, they jointly achieve desired business goals. Although a single organization performs each business process, it may also interact with business processes from other organizations.

This definition is aligned with the value chain model proposed by Porter [78], where an organization produces value for its stakeholders by executing a chain of business processes. BPM helps companies to manage their processes and can be defined as a methodology to manage their lifecycle. In our work, we have adopted the BPM definition of van der Aalst [106], which comprehensively summarizes the scope of BPM :

Definition 2.1.2. *Business process management* supports business processes using methods, techniques, and software to design, enact, control and analyze operational processes involving humans, organizations, applications, documents and other sources of information.

Note that not all business processes are appropriate to be controlled by the BPM. A sufficient level of detail related to activities, rules and people involved is necessary. This definition restricts the universe of BPM to operational processes. More specifically, processes at the strategic level or processes that cannot be made explicit are not considered.
BPM activities are typically organized in the context of a lifecycle (see van der Aalst in [106]). Since this work is mainly concerned with the human aspect in the BPM context, we follow the lifecycle proposed by Dumas et al. [33]. Figure 2.2 shows the lifecycle as a continuous process consisting of four phases that are related to each other.

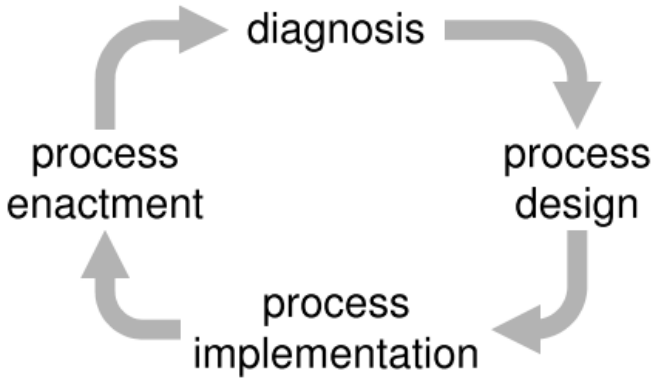


FIGURE 2.2 Business process management lifecycle [33]

This lifecycle starts with *process design*. During this phase, business processes and their inter-relations are identified, reviewed, validated and represented by *process models* (see Weske in [110]). Those models are expressed in a graphical notation to facilitate the communication between different stakeholders, so they can communicate efficiently and also refine their processes. Although several processes models are encoded using a traditional top-down approach, (i.e., providing a prescriptive model, that contains what "should" be done), they could be also discovered. The field of process discovery is a very active theme of research that aims at encoding processes based on recorded events. These events are used as input for specialized process mining algorithms. The paper of van der Aalst [102] presents a comprehensive survey of process discovery techniques. Note that the traditional modeling and process discovery are not mutually exclusive. For example, Maruster and Jorna in [67] proposes an approach to business process modeling design called SCT (Sensory, Coded Knowledge, and Theoretical knowledge). This approach puts process data as a first-class citizens and use it as a source for the encoding and the generation of theoretical knowledge. The new theoretical knowledge can be used for analyzing, diagnosing and reorganizing the business process. As can be seen, the concept

of *Process Model* is fundamental to this work because it can be used not only to analyze, design and improve the process, but also to *implement* information systems.
The process model is used as a source to configure information systems during the *implementation* phase. If the organization uses a conventional software system, it must develop or adjust its software according to the process model. If the organization already has a BPM system, it could benefit from the existing model, and configure its system with few adjustments.
Once the system is adapted, processes are executed in the organization environment (process *enactment* phase). During the execution of processes, event data are collected. These data are useful to analyze running processes and detect anomalies such as bottlenecks and exceptions.
The process *diagnosis* phase uses information about the actual enactment of processes to evaluate them. The results from the diagnosis phase are used to continuously improve business processes.
As mentioned in the introduction, we have a particular interest in *Process-Aware Information Systems* (PAIS). These systems use the business process model as a source, following exactly what the process model specifies.

2.2 Business process model

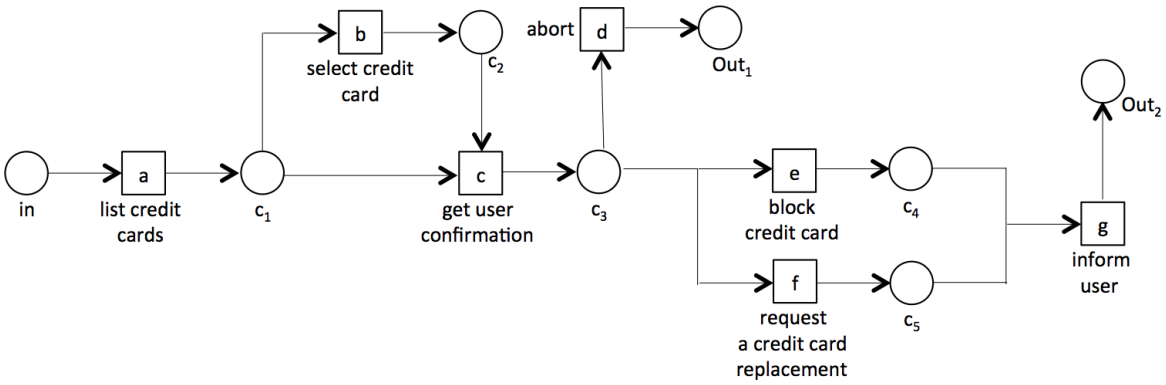


FIGURE 2.3 A process modeled in terms of a Petri net

The business process model is a core artifact in the BPM discipline. The solution adopted to improve flexibility depends on how the process model is structured. More precisely, one could model a process using an *imperative approach*, focusing on defining how tasks will be executed, linking them by connectors specifying causal or temporal relations.

Conversely, one could model a process using a declarative approach, defining constraints at the task level.
Figure 2.3 presents an example of a process model used to *report a stolen or lost credit card* expressed in terms of a Petri net. The model allows different scenarios. For instance, in the scenario *[a,b,c,e,f,g]*, the system retrieves the list of *credit cards* that belong to the customer (activity *a*). Next, the user selects one *credit card* from a list of available ones (activity *b*), the user confirms that she wants to *report the stolen or lost card* (activity *c*). Activities *e* and *f* are executed in parallel, *blocking the selected credit card* and *requesting a new one*. Finally, the user is notified when task *g* is finished with success. Note that, this model focuses more on the *control-flow*, rather than the data and resource perspective. The *control-flow* perspective is often the backbone of a process model (see van der Aalst in [103]). Industry standards such as BPMN [1], WS-BPEL [2], and EPC [3] follow the same approach and can be viewed as extensions of Petri Nets with some conceptual differences. These differences determine the degree of expressiveness and suitability because each one adds different semantics to model other aspects of a process model, such as activities, events, and roles.
Another technique for structuring a process is by using a *declarative approach*. This type of approach focuses more on specifying what is allowed during the enactment of the process than defining how it works by linking activities. The "what" part is determined using constraints that link tasks. As more constraints are defined for a process, fewer execution paths are possible (Schonenberg et al. in [90]). One could improve flexibility by establishing optional constraints that can be violated if needed. Figure 2.4 provides an example of modeling tasks using an *imperative* and a *declarative* approach. The trace of the imperative approach (a) seems to be more rigid, allowing only the execution of task A followed by B. On the other hand, the trace of the declarative approach (b) is apparently more flexible because of its implicitness, allowing different combinations of executions. Both approaches allow a degree of flexibility by using different techniques. To be flexible, an imperative approach must provide at design time a set of explicit paths. In the case of declarative approaches, flexibility means reducing the number of constraints.
Despite conceptual differences, both paradigms can be used for implementing processes that involve humans and applications and have the potential for improvement. Before discussing the existing approaches for modeling processes, we first present some classification of processes with particular focus on human interaction, to delimit the scope of our research.

1. Business Process Model and Notation
2. Web Services Business Process Execution Language
3. Event-driven Process Chain

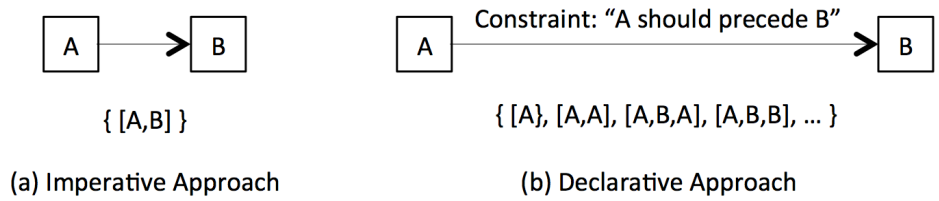


FIGURE 2.4 Examples of (a) imperative and (b) declarative approaches [90]

2.3 Classification of business processes

Business processes are at the heart of organizations and are used for a variety of situations. Not all processes are the same, and the degree of involvement of humans determines how processes are structured. Bring flexibility, by finding an optimal tradeoff between the inherent complexity of humans and the need for control of organizations is a challenging task recognized by several researchers (e.g. Schonenberg et al. [89], Heinl et al. [43], and Soffer [94]). Thus, we first classify business processes according to the level of human involvement and the resulting level of structure of business processes.

2.3.1 Degree of human involvement

Business processes can be classified according to the level of involvement between humans and systems (human-centric or system-centric) (Georgakopoulos et al. [40]). More precisely, they can be classified into Person-to-Person (P2P), Person-to-Application (P2A) and Application-to-Application (A2A) processes (Dumas et al. [33]).

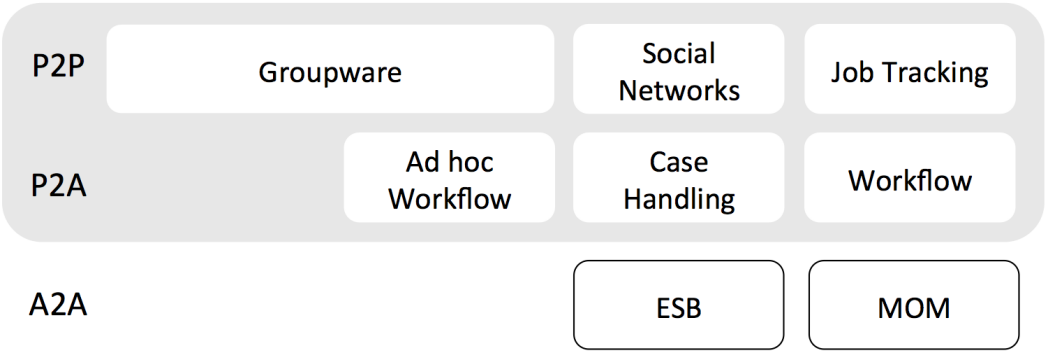


FIGURE 2.5 Business processes according to the level of human involvement and examples of application

P2P processes are those in which participants are primarily humans. More specifically, this type of process requires that humans collaborate to reach a business goal. For this kind of process, software tools are used towards supporting computer-mediated interactions.

Indeed, processes managed by these applications usually do not involve entirely automated tasks (Dumas et al. [33]). Examples are groupware tools, content management systems, and social networks. For instance, a virtual team could make a peer-to-peer evaluation of requirements of global software, using a video conference for meetings. To interact with other members, they could use a groupware system for the exchange of ideas, diagrams and documents. Note that they have some limited benefit to tacit knowledge transfer. Some groupware tools like MEMORAe [7] makes use of semantic modeling and ontologies to improve the support for the codification of knowledge, which could be useful to discover and redesign existing processes.

A2A processes, on the opposite side, are those involving only software systems, without needing human intervention during their execution. This type of business process is very common in financial, insurance and supply chain systems. Some examples : a robot purchase an action at the market system based on its user preference ; two financial systems exchange messages during an electronic funds transfer. ESB (*Enterprise Service Bus*) platforms and MOM (*Message-Oriented Middleware*) applications are examples of technologies that support A2A processes.

The majority of information systems fall in the *P2A category*. This type of process has a more complex nature and involves both human interactions seen in P2P processes and also interactions involving applications that work without human intervention (A2A). In fact, the P2A category represents the real objective of information systems, making people and applications work in an integrated manner (Dumas et al. [33]). Examples of process-aware information systems are traditional workflow management systems, case handling systems, and *ad hoc* workflow systems.

The focus of our research is to develop an approach to the mediation between humans and process-aware information systems. Thus, the A2A category is out-of-scope of this work. This type of application solves a different kind of problem, much more related to the non-functional aspect of a system, like performance, portability, scalability and so on. Figure 2.5 shows the nature of process participants (P2P, P2A, and A2A) and a non-exhaustive list of examples. The region in gray indicates the interest of our research.

2.3.2 Degree of structure

The degree of structure is often used as a dimension for classifying process-aware information systems (Georgakopoulos et al. [40]). Due to its sequential nature, a well-structured process is easier to implement if compared to an unstructured process. We have used a classification of Dumas et al. [33] that proposes four levels of structure of processes namely *unframed*, *ad hoc framed*, *loosely framed*, and *tightly framed*.

An *unframed* process does not have an explicit process model associated with it. Typically, the type of system where these processes are managed does not even allow the specification of processes. It is the case of groupware systems, where users are free to select and trigger activities as well as control their ordering on demand.
A process is considered *ad hoc framed* if a process model is built or changed at runtime to attend a particular business need. In general, an *ad doc framed* process is executed only once for a small number of times. Project management systems have some examples of *ad hoc framed* processes. Each project is different, and depending on its nature, budget and objectives, activities should be conducted in a different fashion. Thus, the project management team could create a distinct process to attend this project lifecycle (e.g. change management, communication management, and escalation procedures).
A *loosely framed* process has a predefined process model and a set of constraints. The predefined process model describes what is done in typical situations but does not enforce a specific set of possible paths to follow. The set of constraints gives a degree of freedom to the enactment of such processes, allowing deviation from the common path within certain limits. Examples of this type of processes can be found in health-care and emergency management, where workers should have some autonomy during the enactment of processes.
Finally, a *tightly framed* process is one that strictly follows what is defined in the process model. Examples can be found in production, banking, insurance and administrative processes. Most of the traditional workflow management systems fall into this category.
The degree of *structure* has a positive correlation with the degree of *human involvement* in several situations. The more humans are involved, the more unpredictable and unstructured are processes. For example, a robot buying some stock options (A2A) has a high degree of predictability because of his nature. The financial company must comply with several regulations and policies. As a consequence, the resulting business processes have a very well defined set of rules and limited variations during the execution flow (*tightly framed* process). A surgery process (P2A or P2P) should be flexible enough to allow working in emergency situations. Workers can deliberate in particular cases and follow a different (unusual but allowed) flow. For example, the doctor could decide to skip the exam activity to proceed with the surgery, saving time and augmenting the life expectation of the patient (*ad hoc framed* or *loosely framed* process). It is easy to infer that the degree of structure is strongly linked to the degree of predictability (Dumas et al. [33]).
Figure 2.6 presents the correlation between the degree of *human involvement* and the *degree of structure of processes*. Highly predictable processes tend to be *tightly framed* and automated (van der Aalst et al. [103]). For this type of process, the complexity to

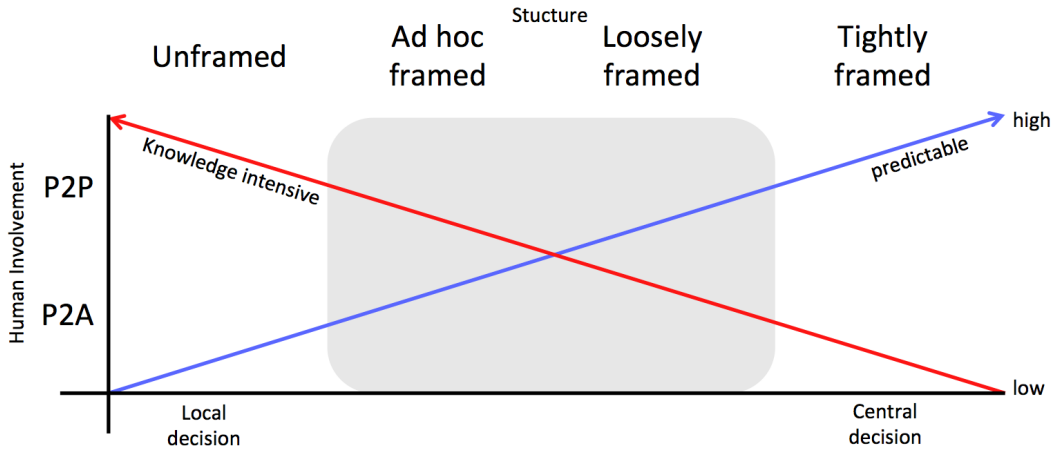


FIGURE 2.6 The correlation between the degree of human involvement and the degree of structure of processes. (in gray, the interest of our research)

mediate a conversation between humans and systems (if necessary) is rather low and does not demand a special approach. At the other end, knowledge-intensive processes tend to be less framed and more people-centric.
The gray area of Figure 2.6 indicates the scope of our work : *ad-hoc framed* and *loosely-framed* processes brings an inherent complexity, requiring mediation and assistance during their enactment. *Loosely-framed* processes can benefit from a mediation approach that guides the conversation between humans and applications, respecting user preference and context. *Ad-hoc framed* processes can benefit from the planning capability of the mediation approach, for example, learning from past experiences or build an ad hoc process based on user expectation. Because of the lack of process structure, *Unframed processes* are out-of-scope. Because of its simplicity and predictability, *tightly framed* processes are also out-of-scope.

2.3.3 Tradeoff between support and flexibility

According to Pesic [77], the flexibility that users have and the support that users get while working with BPM systems have a significant influence on both satisfaction and productivity. In this context of business processes, *support* refers to the degree to which a system makes decisions during the execution of processes. Traditional workflow management systems have a high degree of support. In this situation, decisions are predominantly made by the system and the level of influence of users is quite low. At the opposite side, *flexibility* refers to the degree to which users can make local decisions about how to execute business processes. Groupware systems, exemplified by products developed on top of Lotus Notes are systems with a high degree of flexibility and low

degree of support. Users have a great level of influence and decision in this type of system. They offer excellent possibilities for handling documentation, but is difficult to extend for supporting dynamic features, like live interaction, the use of legacy software, and pro-activity (Barthès et al. [12]).

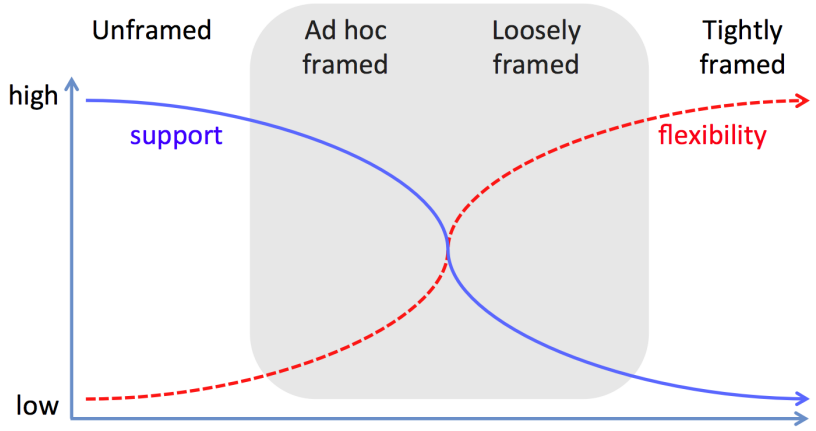


FIGURE 2.7 The tradeoff between support and flexibility

As depicted in Figure 2.7 a centralized decision-making process has a high degree of support and a low degree of flexibility, and a local decision-making process has a low degree of support and a high degree of flexibility. Organizations must find an optimal balance between both styles of decision. As stated in Section 2.3.2, knowledge-intensive processes tend to be more unpredictable, so localized decision-making is necessary.
As can be seen, flexibility has a great impact on our work. Many researchers have acknowledged the importance of this topic, and we address it in the next section.

2.4 Process flexibility

As stated in previous sections, the dynamic nature of business processes requires the development of flexible approaches to deal with occurring variations. Effective business processes must be able to accommodate changes in the environment in which they operate, for example, new laws, changes in business strategy, or emerging technologies (van der Aalst et al. [103]). Modern PAIS must balance a certain degree of flexibility and support at the same time during the enactment of business processes. Flexibility in this context can be seen as the ability to deal with both foreseen and unforeseen changes, by varying or adapting those parts of the business process that are affected by them, while retaining the core format of parts that are not impacted by the variations (Schonenberg et al. [89]).

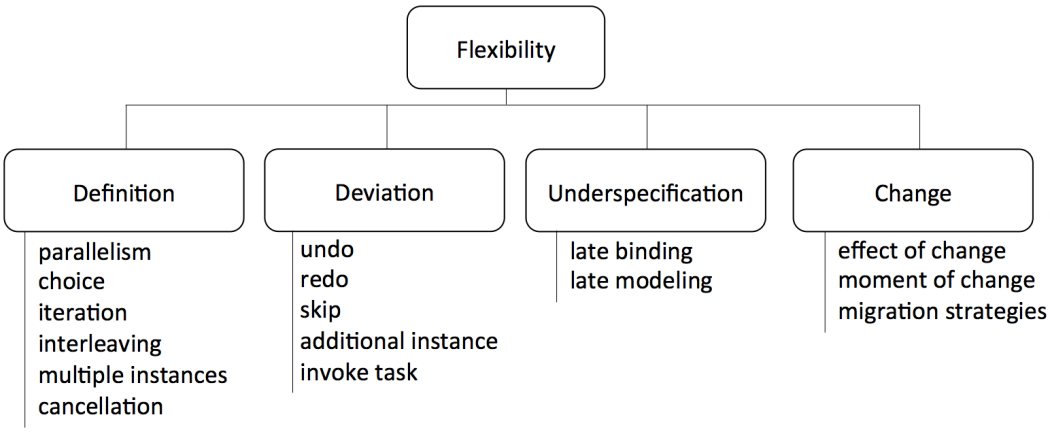


FIGURE 2.8 Taxonomy of process flexibility proposed by Schonenberg et al. [89], identifying four main flexibility types : flexibility by definition, flexibility by deviation, flexibility by underspecification, and flexibility by change

We have adopted the taxonomy provided by Schonenberg et al. [89]. It is an extension of the taxonomy proposed by Heinl et al. [43] and was applied in different state-of-the-art implementations. They suggest four types of flexibility : Flexibility by definition, flexibility by deviation, flexibility by underspecification, and flexibility by change. Figure 2.8 presents the taxonomy and the most important properties investigated for each theme.

2.4.1 Flexibility by definition

Flexibility by definition refers to the ability to specify at design time different alternatives of execution in the process model. As a consequence, users can choose the most suitable alternative during the enactment of a process. Schonenberg [89] enumerates the most common options for realizing this type of flexibility :

— *Parallelism :* the ability to execute a list of tasks in parallel.
— *Choice :* the ability to choose one or more tasks from a list of tasks.
— *Iteration :* the ability to contiguously perform a task several times
— *Interleaving :* the ability to perform a list of tasks in any order.
— *Multiple instances :* the ability to perform multiple concurrent instances of a task.
— *Cancellation :* the ability to cancel a task at any point of the execution.

Flexibility by deviation is the most fundamental type of flexibility, and all BPM systems support this kind of flexibility. However, according to van der Aalst [103], declarative approaches make it easier to defer choices at runtime instead of at design time.

2.4.2 Flexibility by deviation

Flexibility by deviation refers to the ability to deviate at runtime from the execution alternatives specified in the process model, without changing the process model. Users might ignore the execution of some parts of the prescribed process. For example, it could be useful to invert the ordering of the *send proposal to customer* and *register customer* when the customer wants a quick response. In general, models are more descriptive than prescriptive, stating what is the common or normal flow of execution, allowing a certain degree of variation.

- *Undo task* : Shift the control to the moment before the execution of the specified task.
- *Redo task* : Repeat a preceding task.
- *Skip task* : Skip the current task, passing the control to a subsequent task.
- *Create an additional instance of task* : Allow the creation of a new instance of a task that will run in parallel.
- *Invoke task* : Allow the execution of a task that is not currently enabled.

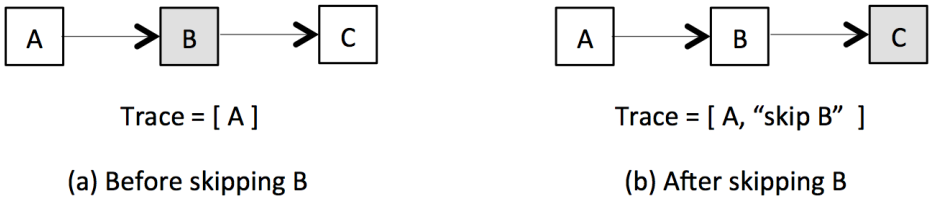


FIGURE 2.9 Flexibility by deviation : the user has the control over the flow of execution (Pesic [77])

Figure 2.9 presents an example of flexibility by deviation by using a skip operation. In (a) task B is enabled. After performing the skip B operation as shown in (b), it is possible to execute the successor of the task B (task C in this case). Case handling [107] is an imperative approach that allows such type of flexibility.

2.4.3 Flexibility by underspecification

The flexibility by underspecification allows modeling some parts of a process model as “black boxes” or *placeholders*. They could be added later during the enactment of a process.
Figure 2.10(a) shows a model specifying that activity A must be executed, followed by B, and finished by an unspecified activity or subprocess. Figure 2.10(b) shows a possible scenario where A and B are performed followed by the placeholder D, selected at runtime.

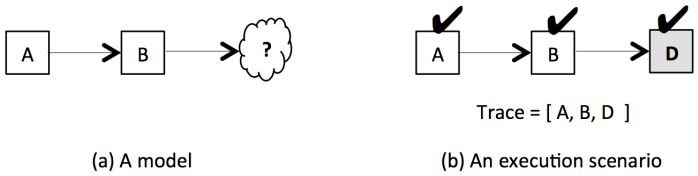


FIGURE 2.10 Flexibility by underspecification : design and runtime perspectives (Pesic [77])

Note that, each time the process executes, the user can select a different activity or even an entire sub-process. This example puts in evidence only the control-flow perspective, but the approach is valid also for the resource or data perspective. For example, one should declare an activity without assigning a resource to perform the action. Only at runtime, the resource will be identified. The same occurs for data, omitting one output generated by some activity during design-time. The reference to the data element can be specified each time the process is executed.
Placeholders can be invoked during runtime using two distinct methods :

- *Late binding :* where the realization of a placeholder is selected from a set of existing process fragments.
- *Late modeling :* more sophisticated form of realization, allowing choose not only a fragment from a set of existing processes but also model of a placeholder at runtime.

An example of a system that allows this functionality is the YAWL language [104]. Using the *Worklet* extension, some activities can be considered as unspecified parts. During the enactment, the *Worklet Service* allows users to choose the exact specification that will be executed.

2.4.4 Flexibility by change

Flexibility by change is the ability to change a process definition at run-time. Systems that supports this type of flexibility allow the migration of all processes that have pending instances to the new definition. The following variation points are observed in this kind of flexibility :

- *Effect of change :* Could be a temporary change, affecting only a set of instances, or an evolutionary change, migrating all active instances and the original model.
- *Moment of change :* Could be at entry time, meaning that only new instances of the process will conform to the new version, and current ones will not migrate. They

could occur *on-the-fly*, meaning that existing instances of the process need to migrate to the new model.

— *Migration strategies :* Define what to do with running process instances that are impacted by an evolutionary change. The most commonly implemented techniques are : (i) Forward recovery, aborting existing instances. (ii) Backward recovery aborting, compensating if necessary, and restarting current instances. (iii) Proceed with change, ignoring existing process instances. (iv) Transfer, migrating the state of existing process instances to a compatible one of the new version.

Flexibility by change is a very challenging theme and has been investigated by many researchers. Depending on the scope of changes, several anomalies may be inserted into the environment like missing data, conversion problems, and deadlocks (van der Aalst [103]).

2.4.5 Choosing flexibility requirements to improve mediation

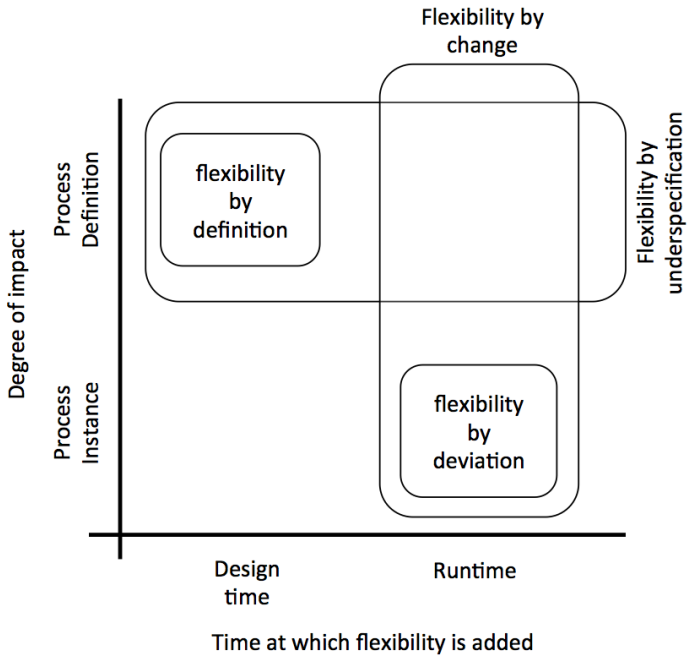


FIGURE 2.11 Degree of impact and specification of time when flexibility is added

Figure 2.11 presents the flexibility taxonomy from two perspectives : the *degree of impact* and the *time at which the flexibility is added*. The degree of impact indicates what aspect of the process is affected : the process model or only the process instance. The time perspective shows if the flexibility property occurs at design time or run-time.

The *flexibility by definition* has an impact on the process model during design time. The resulting model, having a high degree of predictability, determines the behavior during run-time. However, van der Aalst [103] states that declarative languages make it easier to defer choices to runtime, which suggests that the modeling approach could lead to an extension of the impact to the runtime environment, due to the level of implicitness of the model. Moreover, the *deferred choice* pattern described by Schonenberg et al. in [89], leaves the resolution of a choice to the enactment infrastructure at runtime even for imperative approaches.
The *flexibility by deviation* occurs at runtime and affects only process instances. In this situation, users have a certain degree of freedom to enact processes and decisions are made at run-time. The *flexibility by underspecification* has an impact on the process definition phase during both design and run-time. The modeler should define placeholders for tasks during design time and users benefit from those placeholders, replacing them by concrete task or sub-processes at runtime. Finally, the *flexibility by change* has an impact during runtime for both process models and instances.
As can be seen, all flexibility properties could have a direct impact on the runtime environment, affecting humans during the enactment of business processes. Declarative approaches such as DECLARE (Pesic [77]) and case handling (van der Aalst [107]) performed well from the control-flow perspective in the evaluation conducted by Schonenberg et al. in [89]. Declarative approaches perform well for the *flexibility by change* requirement : both process definitions and instances are automatically changed when the execution trace does not violate the constraints of the new process model.
We continue the investigation of approaches towards flexibility, more precisely analyze the level of implicitness and the potential for mediation, to propose a mediation model for the enactment of business processes.

2.5 An approach towards flexibility

2.5.1 Case handling

Case handling is an alternative BPM paradigm proposed by van der Aalst et al. [107]. This approach aims at more flexible and knowledge intensive process execution for a wide-range of scenarios for which traditional workflow management systems fail to offer an adequate solution. Case handling has its strengths changing the focus of business processes from *what should be done* to *what can be done* to achieve a business goal. The

author focuses on the following problems found in traditional workflow management systems :

— **Atomic activities**
Traditional workflow management systems consider that activities are atomic units of work. This approach is used to force the distribution of work during the design phase. However, some events are handled by users in a much more detailed and complex way.

— **Routing for distribution and authorization**
Because of the routing mechanism employed by workflow management systems only the work for which workers are authorized is revealed. Thus, a *work-item* is only available when it is in the worker's *in-tray*. The problem here is that both the distribution of work and the authorization coincide. Mixing up two important aspects of a business process in only one event considerably diminishes the level of flexibility of applications.

— **Context tunneling**
It is more a consequence than a cause. By focusing on the control flow, process data are only available when the worker receives a work-item in her in-tray. The context of the current business process handled by the workflow system is partially available, resulting in inefficiencies and errors.

— **Focus on *what should be done* instead of *what can be done***
This push-oriented perspective results in rigid and inflexible workflows.

Key properties of case handling :

— **Providing all information available**
A case handling system (CHS) presents all data about a case at any time to workers, according to their level of authorization. The *context tunneling* effect is avoided.

— **Use data to determine which activities are allowed**
A CHS use both data and flow as first-class citizens. Thus, data can also determine which activities are enabled.

— **New semantics for roles**
Workflow management systems have only the *execute* semantic for the role. A CHS provides additional possibilities like *skip* and *redo* (to undo a previously performed activity), which augments the level of flexibility during the enactment of processes.

— **Process data can always be modified**
A CHS allow workers to change data before or after the execution of activities. Data can be amended as soon as they are available.

Main concepts

The case handling paradigm makes the *case* and its *data* as the central elements, as opposed to activities and routing rules being common for workflow management systems. We refer to a case as a single process instance, meaning that a business process can handle many cases in parallel. The case could also be interpreted as the *product* that is manufactured by a group of workers that must be aware of the whole context. Examples of cases are the peer-review of project proposals, the evaluation of a job application or managing a call for bids for procurement.
The concept of *activity* is also important for case handling. Here, an *activity* is also considered a logical unit of work as occurs in traditional WfMs. Activities must be executed to enact a case. However, it carries a less rigid notion here, because an activity is a chunk of work recognized by workers that have the potential to be executed, transferred, re-executed or simply ignored (for example, a worker could skip an activity to add a supplier in a bidding process).
The use of too many precedence relations is discouraged in case handling. Here, the state and structure of a knowledge-intensive case is an important element and could be used to guide a case enactment. In case handling, state and structure are represented as *data objects*. The logistical state of the case is not determined by the control-flow status but by the presence and properties of data objects. Thus, case handling could be considered a hybrid process modeling technique, mixing control-flow and data-flow (van der Aalst [107]).
Each data object is linked to a process and could be related to one or more activities. A data process that does not have links with any activity is called a *free data object*. All other data objects are linked with one or more activities as *mandatory* and/or *restricted*. A data object is considered mandatory for an activity if it is required to complete the corresponding activity. A data object is considered restricted for an activity if it can only be entered in this activity or other activity that shares the same object. Both definitions (restricted and mandatory) are orthogonal. For example, a data could be mandatory (to be considered completed, the activity must fill this information) and restricted (the information is only changed in this activity). These objects are presented to workers using *forms*. A form is used to present different views on data objects of a particular case to workers, assuming they have the proper authorization. In CHS forms could be associated with one or more activities.

Roles

In the same manner, as occurs in traditional workflow management systems, case handling allows that one actor (user) can have multiple roles, and a role may have multiple actors. Roles can also be modeled as a graph. Using an *is-a* relation, one can define different levels of authorization. For instance, the role manager is a subtype of the role employee. The difference here is that case handling has different types of role associated with an activity ; traditional workflow management systems have only the *execute* role. For each activity, three types of roles need to be specified : the execute role, the redo role, and the skip role.

- The *execute* role is used to define who can carry out the activity.
- The *redo* role is used to re-execute or just undo an activity. For instance, the case returns to the previous state (before the execution of the activity). In this case, the undo operation is only possible of all other subsequent activities are undone as well.
- The *skip* role is used to pass over an activity.

It is a powerful mechanism for modeling flexible processes without needing to explicitly specify a broad range of exceptions. The redo operation provides an implicit (and dynamic) form of loop. Also, the skip offers an implicit form of choice. To avoid undesirable effects, the modeler just needs to configure the level of authority for each role type : for example, avoid assigning a role to the skip or redo role.
These roles allow a clear separation between work distribution and authorization. Workers are not limited to the set of activities that have been assigned to them (in-tray). Case handling allows the implementation of a flexible query mechanism that allows navigating through activities with different filters that can be applied to simulate the in-tray for actors, or ad hoc queries to monitor work.
As stated before, the focus of case handling is on the whole case, i.e., there is no context tunneling by limiting the view to specific work-items. The primary driver to determine which activities are enabled is the state of the case (i.e., the case data) and not control-flow related information such as the activities that have been executed. The basic assumption driving most workflow management systems is a strict separation between data and process. The strict separation between case data and process control simplifies things but also creates integration problems. For case handling the logistical state of a case (i.e., which activities are enabled) is derived from the data objects present. Therefore, data and process cannot be separated. Unlike workflow management, case handling allows for a separation of authorization and distribution. Moreover, it is possible to distinguish various types of roles, i.e., the mapping of activities to workers is not limited to the execute role.

2.6 Discussion

We have presented in this chapter the foundations of the Business Process Management discipline and have surveyed a range of concepts and classifications of BPM systems. The first section of this chapter focused on the BPM discipline and how BPM systems are related to this lifecycle. The second section was dedicated to the process model, a significant artifact for our approach. Next, some classification of business processes in section 3 and 4 helped for delimiting the scope of our work, for business processes. We finished by giving an overview of the case handling system, which is an approach that has interesting techniques towards process flexibility. Some methods of case handling systems that put data control at the same level of control and providing all information available, avoiding context tunneling undoubdtedly influence the design of our approach.

Chapitre 3

Personal Assistant Agents

In the previous chapter, we have presented the BPM discipline and its lifecycle as well as its core component : the business process model. A great deal of work related to process flexibility has been found. However, the majority of methods and techniques falls into the study of the flexibility for the structural aspect of processes. Few works related to the human perspective during the enactment of existing ones. Moreover, to the extent of our knowledge, we have not found any conversational interface dedicated to the subject of business process enactment and the user interface seems to be a neglected topic in this field.

Thus, we dedicate this chapter to study concepts, tools, and techniques available in the literature to implement personal assistant agents. The interaction between agents and humans have a substantial impact on our work, and a comprehensive study of existing approaches will help us delimiting the scope of our research.

This chapter is organized as follows :

- Section 3.1 concept of agent and its properties
- Section 3.2 introduces the notion of multiagent system and presents the platform developed in our laboratory.
- Section 3.3 presents the concept of personal assistant agent.
- Section 3.4 presents our vision of how personal assistants could contribute to the business process management domain, more specifically during the enactment of business processes.
- Section 3.5 presents a summary of essential elements identified to conduct our research.

Readers who are familiar with multiagent systems can skip sections 3.1 and 3.2.

3.1 Agent definition

In the context of *computer science*, the notion of *agent* [1] comes from the *artificial intelligence* domain (AI). Russel and Norvig [85] states that *a software agent is anything that can be seen as perceiving its environment through sensors and acting upon that environment through actuators*. Jennings and Wooldridge [48] highlight the autonomous aspect of agents, stating that *an agent is a computer system situated in some environment, and that is capable of autonomous action in this environment to meet its design objectives*. Both definitions from Russel [85] and Jennings [48] presents three relevant aspects under different perspectives : (i) an agent is situated in an *environment*, (ii) it has a decision-making capability, giving some autonomy, and (iii) it has one or more goals. An agent perceives its environment using its sensors and deliberately changes the environment to reach its objective using its actuators. To better understand the environment and how it influences the agent decisions, we use a classification proposed by Russel and Norvig [85] :

— *Fully observable* vs. *Partially observable :* If the agent has complete access to the state of the environment at any point in time, this environment is called *fully observable*. An environment might be partially observable because of noisy and inaccurate sensors or because parts of the state are simply missing from the sensor data. For instance, a business process management system that has an incomplete description of its participants might find difficulties when negotiating and delegating tasks.

— *Deterministic* vs. *Stochastic* If we can determine the next state of an environment based only on the actions executed by the agent, then the environment is called *deterministic*. Conversely, a stochastic environment is one in which there is uncertainty about the state resulting from performing an action in the current state. The majority of real scenarios fall into this category. For instance, each activity belonging to a business process have a predefined set of preconditions and effects. If one of these activities lacks a complete definition of preconditions and effects or if the activity produces an unforeseen result, then we have a *stochastic* environment.

— *Episodic* vs. *Sequential* In an episodic environment, the agent actions are divided into isolated episodes. More specifically, the decision made in one episode does not affect future decisions. An example of episodic environment could be a quality assurance agent that inspects new products and assigns a quality label of the

1. For the sake of readability and brevity we will often omit the word “software” before the word “agent

product without paying attention to past evaluations. Conversely, a sequential environment is more complex in nature, because each short-term decision has an impact in the medium and long term. The enactment of a business process is an example of sequential environment because each performed activity may change the environment, producing effects that influence the enactment of subsequent activities or sub-processes.

— *Discrete* vs. *Continuous* An environment is discrete of it can be modeled by finite sets. Russell and Norvig give a chess game as an example of a discrete environment, and taxi driving as an example of a continuous one. Taking into account the decisions made by the enactment engine, we can consider that this environment is discrete, but the combinations of several discrete decision variables may lead to a combinatorial explosion.

— *Static* vs. *Dynamic* If the environment can change while an agent is deliberating, then this environment is said to be *dynamic*. A dynamic environment is much more complex because each change of state is interpreted as a decision request to the agent. Depending on the architecture of information system that is used, the environment of business processes tends to be dynamic, because business processes might compete for shared resources of a company (e.g. : people allocation, database entries).

According to this classification, the hardest case is partially observable, stochastic, sequential, continuous and dynamic. Russel and Norvig also distinguish the environment between *single-agent* and *multiagent*. Needless to say, single-agent environments are simpler when compared to multiagent ones. Multiagent environments require a minimum infrastructure for communication and a well-defined interaction protocol, which increases the complexity of the overall application.

Taking into account the autonomy of agents, they could present different behaviors when added in a group. They could be *competitive* if they use one or more shared resources, or they could be *cooperative* to maximize the performance. Note that the classification of competitive or cooperative agents is not crisp. Agents may present different behaviors during their lifecycle according to their goals. Note also that from the perspective of one agent operating in a multiagent system, the environment can be regarded as being *dynamic*.

Table 3.1 presents a classification of a typical business process enactment infrastructure if managed by agents.

Now that a categorization of environments has been given with a special focus on business processes, one must explore the agent and its properties.

TABLE 3.1 Classification of an environment of a typical business process enactment infrastructure using agents

Dimension	**Value**
Observable	Partially-observable (e.g. not all departments have enough information when they are cooperating)
Deterministic/Stochastic	Stochastic (e.g. service-oriented architectures)
Episodic/Sequential	Sequential (processes are chains of activities)
Static/Dynamic	Dynamic (processes compete for resources)
Discrete/Continuous	Discrete (risk of combinatorial explosion)
Agents	Multiagent is preferable for task decomposition
Cooperative behavior	Predominantly cooperative
Competitive behavior	May compete for shared resources (e.g. people and resources such as database entries, storage devices, and processors)

3.1.1 Agent properties

As stated before, agents are computer systems capable of autonomous action to meet its design objectives. Thus, agents have distinct characteristics, and not all software components could be classified as an agent. In the nineties, Wooldridge and Jennings gave one inspiring example of an agent application :

Example 3.1.1. You are editing a file when your personal assistant requests your attention : an email message has arrived, that contains notification about a paper you sent to an important conference, and the assistant correctly predicted that you would want to see it as soon as possible. The paper has been accepted, and without prompting, the assistant begins to look into travel arrangements, by consulting some databases and other networked information sources. A short time later, you are presented with a summary of the cheapest and most convenient travel options [113].

This type of application was unusual in the nineties (the article was published in 1995), but it is quite common nowadays after the popularization of applications like *Siri* from Apple and *Cortana* from Microsoft. This generation of applications works as a *hub* of services, delegating tasks to specialized software components. This movement is aligned with the prediction of Negroponte, that stated in [71] that *the future of computing will be 100% driven by delegating rather than manipulating*. We have used some characteristics proposed by Negroponte that give more focus on human-agent interactions. To do so, agents must be :

— *Autonomous :* giving a vague and imprecise specification, an agent must determine how the problem is best solved and then solve it, without constant guidance from the user,
— *Proactive :* it should not wait to be told what to do next, rather it should make suggestions to the user,
— *Responsive :* it should take account changing user needs and changes in the task environment, and
— *Adaptive :* it should come to know user's preferences and tailor interactions to reflect them.

Of course, these characteristics could also be used in the context of agent-to-agent interaction as well. Notice that pro-activeness is an essential property to introduce another characteristic of agents : the reactivity concerning the environment. Pro-activeness means that agents must execute actions according to a predefined plan to ensure the fulfillment of its goal. However, an agent must not neglect the changes that may occur in this dynamic environment. A plan revision is necessary if these changes have an impact on the agent plan. Thus, a level of reactiveness is also required. Notice that a high degree of reactiveness may influence in the whole plan, making the agent unfocused. Actually, these two characteristics (pro-activeness and reactiveness) are difficult to balance even for human beings : a typical example is given by project managers who follow project plans without recognizing events that disrupt the pre-established course of actions (Sbodio [87]).

3.1.2 Strategies towards reasoning

An agent has an individual decision process made of three phases : perception, decision, and action. The perception phase allows collecting information about its environment. In general, an agent has a symbolic representation based on its sensors and previous experiences. During the decision phase, the agent reasons using the information collected from the environment and its goal, creating or changing its plan of actions to better reach the goal. The action phase is the enactment of such actions, transforming the environment. It is a cyclic phase, where agents are constantly perceiving, reasoning and acting.
In the literature, agents are commonly classified as *cognitive* or *reactive*. Reactive agents do not have an explicit representation of their environment. Their behavior is based on a perception/action function. The cognition phase is reduced or inexistent. The work of Brooks [16] [17] illustrates how reactive agents work. According to him, the behavior of an agent is produced by a set of behavioral rules that associate a particular stimulus with an action. The agent behavior is the result of interactions between the agents and the

environment, and agents do not explicitly communicate between them. This approach is called *principle of emergence*.
Conversely, cognitive agents have an explicit representation of the environment, of other agents and their goal. They also have a model of their social organization. The relationship between agents is done according to their degree of collaboration to solve a particular problem. Some researchers in the field use other social aspects such as emotions to guide the interaction between individuals (e.g. Lhommet et al. [61] and Rivière et al. [82]).
Notice that the boundary between these types of agents is not crisp and some agent architectures combine both techniques (also called *hybrid agents* according to Wooldridge [112]).
In our work, we propose using a personal assistant agent to help users during the enactment of business processes. This type of agent requires an explicit representation of the environment, as well as a description of other agents that collaboratively communicate during the enactment of the process. Personal assistants must also be capable of identifying their master's [2] goal based on sentences. Thus, we restrict the scope of our study to **cognitive agents**. The OMAS (Open Multiagent System) platform developed in our laboratory allows the implementation of both cognitive and reactive agents. The platform is described in Section 3.2.1.

3.2 Multiagent Systems

The idea behind multiagent systems is that a system could get better results if agents work cooperatively instead of in complete isolation. According to Sycara [97], a multiagent system benefits from two powerful modeling techniques namely modularity and abstraction. The reason for the growing success of agent technology is that the inherent distribution (modularity) allows for a natural decomposition of the system into multiple agents that interact with each other to achieve a desired global goal (Chen and Cheng [20]). The abstraction characteristic is also represented by numerous approaches used for planning and reasoning. Sycara [97] enumerate four characteristics of a multiagent system :

1. Each agent is part of the overall solution. It does not have enough information to solve the whole problem.
2. The control is not centralized.
3. Information used for problem-solving is decentralized.

2. A master is the user that owns the assistant

4. The problem-solving mechanism is asynchronous.

In multiagent systems, agents are independent individuals that share the same environment. They may present a competitive behavior since resources are shared among them (time, space and physical resources). They may cooperate to achieve common goals. Even if agents are written in different languages, they can communicate using a standard protocol, expressing their needs, their state and reaching agreements. In short, an agent that belongs to a multiagent system has two distinct objectives : performing tasks and cooperate with other agents whenever possible. As a consequence, agents should coordinate their activities and cooperate to avoid effort duplication (D'Inverno and Luck [32]).
Multiagent theories are usually complex from a computational perspective. Therefore multiagent systems often take inspiration from theories and make some pragmatic assumptions and tradeoffs to simplify the implementation. We present an overview of the platform developed in our laboratory and used to implement our MAS.

3.2.1 The OMAS platform

The OMAS platform (*Open Multiagent System)* [8] is the result of the development of research projects in several domains like engineering design (Shen and Barthès [92]). OMAS is directly derived from the open architecture of cognitive agents called OSACA developed by Scalabrin and Barthès [88].
In its latest version developed in LISP [3], the platform allows the creation of three different types of agents :

— **service agent (SA) :** it is a cognitive agent that does not have an interface dedicated to the end-user. It provides a specialized set of services according to its competence. The services could be performed by the community of agents, which forms a multiagent system.
— **personal assistant (PA) :** this agent is responsible for the interaction with humans. Its role is to understand and to serve its master. Since a PA knows what its master wants to do, it could use other agents (service, transfer agents or even other personal assistants) to develop and execute its plan.
— **transfer agents (XA) :** this agent allows the communication between OMAS and other platforms. It has a set of predefined services for connectivity.

Agents are organized in a multilevel architecture based on the concept of *coterie*. From the organizational perspective, a *coterie* is a closely related group of agents. It is particularly interesting because an agent could enter or leave a coterie at any time, avoiding registering

3. The platform and the documentation are available at http ://www.utc.fr/~barthes/OMAS/.

the agent in centralized repositories like yellow pages or directory facilitators. A consequence is that if a machine fails, no bookkeeping needs to be done on the other machines, which makes the system more robust (Barthès in [8]). From a technical perspective, the coterie is then defined as a set of agents present on a LAN loop, sharing the same port address. Thus, messages are delivered using UDP broadcast to all agents, with a single message. We could say that agents are like a group of persons located in the same room, where each person overhears what the others say. Agents can use this feature by updating their internal representation.
In OMAS, all agents are subsets of a generic agent model, presented in the next paragraphs.

Generic agent model

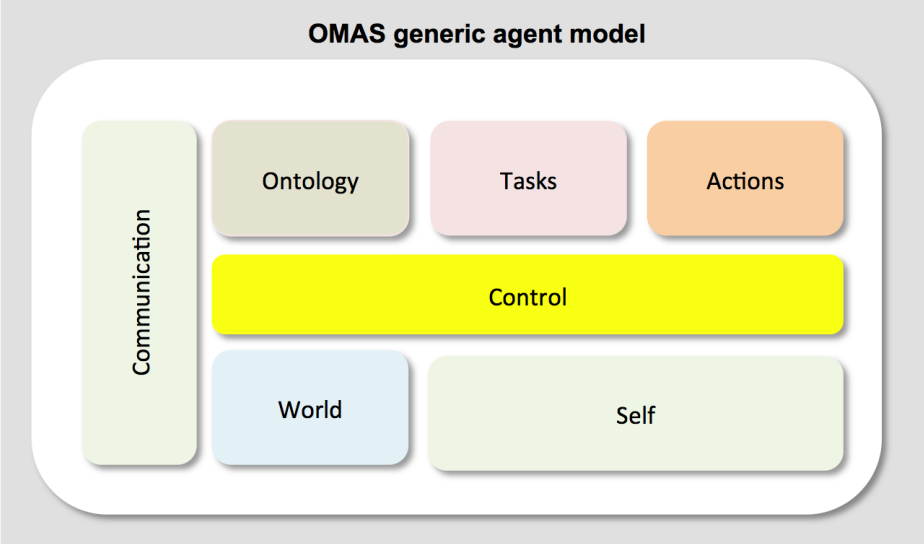


FIGURE 3.1 The OMAS generic agent model

OMAS agents are derived from a generic agent model initially proposed by Ramos [79], as depicted in Figure 3.1. An agent is composed of a set of modules that could be customized by the agent modeler when needed :

— **World :** Each agent resides in an environment composed of other agents. To interact with them, an agent should build an internal representation of the environment. Ramos suggest that this internal representation should have the competencies of other agents that belong to its group.

— **Self :** This module contains a self-description, a long-term and short-term memory. The self-description is a representation of the agent's competencies and its

long-term goals. Enembreck [37] added a learning function to this module, allowing an agent to learn the competencies of other agents iteratively by using their interaction as a source.

— **Ontology :** Ontologies are an integral part of applications involving cognitive agents. They allow a shared understanding of domains and could be used to understand the expressions of the content language, to build a knowledge base, or to interpret the utterances from the user (Barthès [8]). Thus, each agent could have its own ontology. One of the possibilities for using ontologies is using MOSS [9] as a representation language. MOSS is a framework for developing ontologies or knowledge bases. MOSS has been made part of the OMAS platform and includes multilingual and versioning features, as well as reasoning by means of a query system.

— **Skills :** Skills represent what are the agent capabilities or what it can do. Skills are realized as functions, and the platform distinguishes two types of agent skills : *atomic skills* and *complex skills*. Atomic skills execute without requiring any help from other agents, whereas complex skills require results obtained from the cooperation with other agents. Atomic skills have a reference to a static function whereas dynamic skills have two references : the first to trigger the function and the second works as a call-back function when answers from other agents are received. All skills may have preconditions (a function in charge of checking if the skill can be fired), a time-limit option limiting its execution time and an acknowledge option to return an acknowledge message to the sender if requested. Complex skills may have a timeout handler to do something when answers from sub-contracts fail to return on time, and a select-best-answer function, used to select the answer on broadcast or Contract Net protocols. Notice that the concept of skill (what the agent can do) is different from the concept of goal (what the agent *plans to do*).

— **Goals :** OMAS implements goals separately from skills. A goal can be one-shot or cyclic. Goals are enabled by an enabling function that may block the triggering of the goal (somehow analogous to the precondition function for skills). A particular mechanism of activation energy is planned : for instance, a goal can be activated when its energy becomes higher than a threshold level.

— **Tasks :** The current context of the agent is represented in the tasks part. The task that the agent is currently executing could trigger other subtasks, controlled by OMAS. Thus, a task representation is a fundamental building block.

— **Communication :** Message handling is an important feature of multiagent systems. Agents use messages for negotiation, task decomposition, and planning.

OMAS have a set of built-in functions for communication (sending and handling messages). When a message reaches an agent, it wakes up the scan process that either processes the message in special cases, or puts it into the agenda. A special process then selects the message and distributes it to the right thread, calling the proper skill. Thus, selection and distribution of messages are done automatically by the middleware, simplifying the programmer's work. Regarding protocols, OMAS implements several communication protocols : simple, broadcast and Contract Net. The OMAS agent communication language is simple and not very original, resembling KQML with the exception of a cancel-grant performative that allows granting contracts to a set of agents while at the same time canceling the task for the other ones. It reduces racing conditions. The choice of the protocol is done at the message level.

This basic structure is enough to develop a broad range of applications using the MAS metaphor. However, this generic model is not sufficient enough to implement a particular type of agent, called personal assistant. The personal assistant, an essential element in our work, is detailed in next section.

3.3 Personal Assistants

According to Barthès in [11], a personal assistant (PA) is an agent in charge of interfacing a particular person, its master. The role of the PA is to simplify the interface between users and agents in the multi-agent system. As such, a PA has very superficial technical skills and for technical problems relies on other agents with specialized skills called staff agents. In the context of information systems for the corporate environment, users could benefit from the metaphor of the personal assistant. However, building this type of application using personal assistants has an high-level of complexity. The effort of publishing legacy services by personal assistants, using traditional software engineering techniques and components could be unfeasible both in financial and time-to-market terms. From the software engineering perspective, personal assistants can be seen as *off-the-shelf* software components that could fill this gap and inaugurate a complementary software development paradigm for service-oriented information systems. Personal assistants may mediate any person-to-person interactions, resulting from business process collaborations, bringing up some interesting possibilities for reducing the usually delays frequently encountered when people try to manually make real-time contacts.

To better understand the personal assistant metaphor, we use the set of activities that a personal assistant can execute for humans according to Maes [66] :

1. by hiding the complexity of difficult tasks ;
2. by performing tasks on behalf of the user ;
3. by training the user ;
4. by helping different users to collaborate ;
5. by monitoring events and procedures.

Thus, we build a definition of a personal assistant in the context of business processes :

Definition 3.3.1. A *personal assistant* is an agent that helps his master reduce the workload during the enactment of collaborative business processes.

Next section presents the metaphor of the personal assistant implemented in the OMAS platform and contributions received from previous research.

3.3.1 The evolution of the PA in the OMAS platform

Since we want to create mixed environments with artificial and human agents, it is necessary to provide human-computer interfaces. Doing so from scratch is a laborious task. To simplify the programmer's task, OMAS provides a default interface managed by the personal assistant, an agent in charge of letting humans be part of the environment. The PA has received several contributions from the past years, and it has been used in different domains. Figure 3.2 presents a timeline summarizing recent contributions. A brief overview of contributions in given in the next paragraphs.

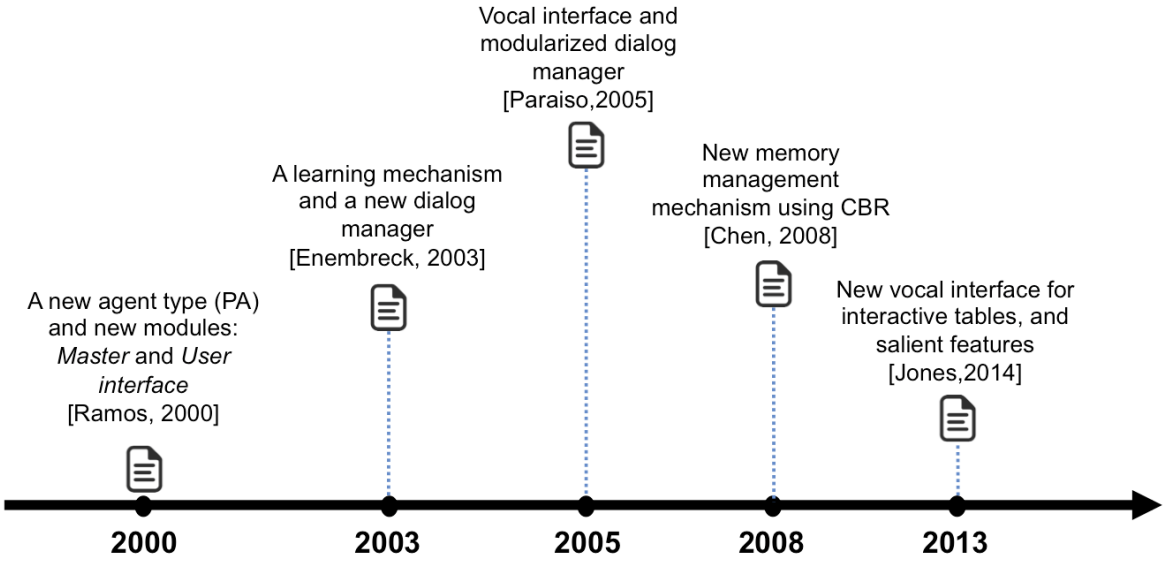


FIGURE 3.2 Timeline with improvements on the personal assistant in the platform

Ramos [79] has changed the generic agent model presented in Section 3.2.1, adding two new modules that aim at improving the communication with humans, namely *master* and *user interface*. The *master* module describes the user profile, containing his preferences and habits. It may also include personal information such as e-mail address or social network credentials. The *user interface* is the module that coordinates the interaction between humans and the PA. In fact, the *user interface* does not use conventional techniques found in classic dialog management systems. Instead, it used an intermediate solution that exploits a library of predefined actions and triggered by the successful combination of keywords given by the end-user. Using domain ontologies, the system tries to find a service agent capable of answering the user question. For instance, the sentence "Please give me the number of Kassel", *number* is a synonym of *telephone* (a property of the *Contact* concept), and *Kassel* is an entry point. An entry point is a kind of indexed property with direct access to the individual in the MOSS system.
The interpretation of sentences does not use any linguistic technique to evaluate the request. It could cause potential problems when the user types complex expressions like : "Please pay the bill and send a copy to Mary". Another situation is the usage of modifiers applicable to the object of the sentence : "Please give me the number of my preferred credit card". The dialog management system also has some limitations. It does now allow concurrent dialog contexts and user support is limited. The technique uses a finite-state graph predefined in the source-code. It is difficult to add new features, since it implies changing the graph directly into the source code, adding new tasks.
Enembreck [37] proposed a new configuration of modules initially developed by Ramos. The main objective of this architecture is to allow an adaptive agent model, with an embedded learning mechanism and a new dialog management system. The proposed approach has also reduced the complexity of the agent, by distributing the competencies among other staff agents. Ontologies, tasks, and master are now managed by the dialog system. The self and world modules remain the same. This new approach is capable of handling more complex expressions in natural language and brings a task-oriented dialog management model. The control module was removed, and its competencies have been transferred to the dialog manager. The drawback of this approach is mixing the competences of two different components (control and dialog manager), which augments the risks when new changes are applied in the framework.
Paraiso [74] dedicated his research to the development of a new dialog management architecture, returning back the control model and treating ontologies, tasks, and skills as first-class citizens. He proposed a conversational interface for personal assistants. The proposed architecture contains a set of modular building blocks dealing with a particular

problem of dialog management. Chen [21] has used the same architecture and incorporated a new memory model based on *Memory Organization Packets* (MOPs). These memory packets are organized using a case-based reasoning (CBR) approach. Several tests have shown that the learning mechanism performed well, reducing the overall time to identify and perform tasks.
Finally, Jones [49] used the personal assistant for improving the interaction of a collaborative platform that uses a large multi-touch, multi-user interactive table, coupled with an interactive board display. A new vocal interface was developed as well as a mechanism of salient features, used to quickly access recent operations.

3.3.2 Dialog management

The dialog manager is a critical component of dialog systems. Not only critical for its function of controlling a computer-human conversation but also critical in the sense that it should use techniques from a variety of fields (linguistics, statistics, AI, knowledge management). Generally speaking, the dialog manager has two distinct roles : the first role is to manage the representation of the current state by adding new information when it becomes available (interaction with users and other systems). This representation should contain relevant information for the system like the dialog history and the current state of the task. Its second role is to make decisions based on its state. It should always observe its state and select the next action to perform by the system. For instance, the dialog manager could ask a clarifying question or ask for confirmation when it does not understand what the user said. Action selection can take many forms, ranging from a direct mapping between states and actions to the application of logical rules or the use of planning techniques (Lison [63]).
A dialog system is decomposed into a layered structure, each one having components for handling a particular aspect of the interaction. For instance, Allen [5] divides the dialog management procedure into several agents (e.g. an agent to manage the discourse, an agent to control the plan). The dialog manager developed by Bohus and Rudnicky (RavenClaw) follows the same separation of concerns principle, but at the logical level. The dialog manager isolates the domain-specific aspects of the dialog control logic from domain-independent conversational skills and establish a boundary between decision layers. The approach proposed by Paraiso follows the same principle [75].

Dialog management architectures

Various approaches have been proposed to formalize the dialog management problem. A wide range of strategies has been proposed to represent, update and act upon this dialog state. We describe a summary of must used approaches.

Finite-state automata : This approach is defined by a collection of states and transitions (Larsen and Baekgaard[56]). Decision-making is made by transferring the control from one state to another state. That is to say, each state is labeled with a condition on the user input. When this condition is satisfied, then the control is moved from to the target state associated with the edge. The finite-state automata is useful for modeling simple scenarios. However, it is not practical for some complex situations, due the increasing complexity of the resulting graph.

Frame-based : It is an alternative to overcome the limitations of the finite-state automata (Seneff and Polifroni [91]). A frame-based approach encodes the dialog state as a frame constituted of a set of slot-value pairs. When the dialog starts, the frame is empty and is gradually filled with information. After each user move, a set of production rules determines next actions. For example, the dialog manager could ask for incomplete values based on the current frame. A frame-based approach relies heavily on using domain-specific information in the linguistic structure of the current and previous utterances and simple heuristics for filling the values of required variables and matching dialog control rules. Despite the advantages from the finite-state automata, the design and maintenance of frame-based approaches requires a significative effort. Since, there is a high level of cohesion between frames, rules and domain-specific information, the modularity is also affected.

chan : This approach views the interactions of a dialog as a sequence of information state updates (Larsson and Traum [57]). During the dialog lifecycle, the information state is continuously monitored by the dialog manager and represents the full contextual knowledge available to the agent. Upon reception of a relevant input, the dialog manager modifies this information state using a collection of update rules. In addition to state-internal operations that change particular variables of the information state, the update rules are also employed to derive the actions to execute by the agent. Given a collection of rules and a generic strategy for applying them, the dialog manager can both update its state and select the next action to perform by way of logical inference. This action selection can notably be grounded on the set of open questions raised and not yet

answered during the interaction. Generally speaking, the information state approach is a more general version of the frame-based method, which allows more complex representations of the information state, like modeling complex mental states such as obligations and commitments.

Plan-based : It relies on complex representations of the dialog state that encompass the beliefs, desires and intentions (BDI) of each agent (Cohen and Perrault [28], Allen and Perrault [6]). In a plan-based approach, the goal is a central element, and both users and the agent are assumed to act towards a common goal. The problem of interpretation the user's utterance is transformed into a plan recognition problem. The agent seeks to derive the belief, desires and intentions that best explain the observed conversational behavior of the speaker. Similarly, the selection of system actions is derived from the (task-specific) long-term objectives of the system. This search for the best action is an instance of a classical planning task, which can be solved using off-the-shelf planning algorithms. These algorithms require the declaration of a planning domain that specifies the pre-conditions and effects of every action.

3.4 How a PA could contribute to the BPM domain ?

This section is dedicated to the identification of the most suitable interface between humans and personal assistants to be used in the business process enactment scenario. Our work is based on the assumption that our target user could be a *subject-matter expert* (SME) of an organization or any other stakeholder involved in a process that may collaborate with other users to reach a business goal (a customer, a supplier, a retailer and so on). Our hypothesis is that a PA could support users during the enactment, finding the most suitable process to execute, controlling user's tasks. The question that arises immediately is : Do these stakeholders share the same vision ?
We have conducted a survey to better understand users needs and their expectations about the capacities of a personal assistant in a corporate environment. To be more precise, we aim at discovering what could be the role of the personal assistant and, if it has such potential, what are the most appropriate modalities of interaction with humans. We recruited 50 employees from different organizations to answer a survey. Respondents belong to various sectors (e.g. finance, manufacturing, and health), organization sizes (small, medium and big organizations). All respondents are users of our target information system, which includes Internet banking, ERPs, CRMs and typical workflow management

systems. Figure 3.3a presents a summary of the distribution of respondents by sector and Figure 3.3b by organization's size.

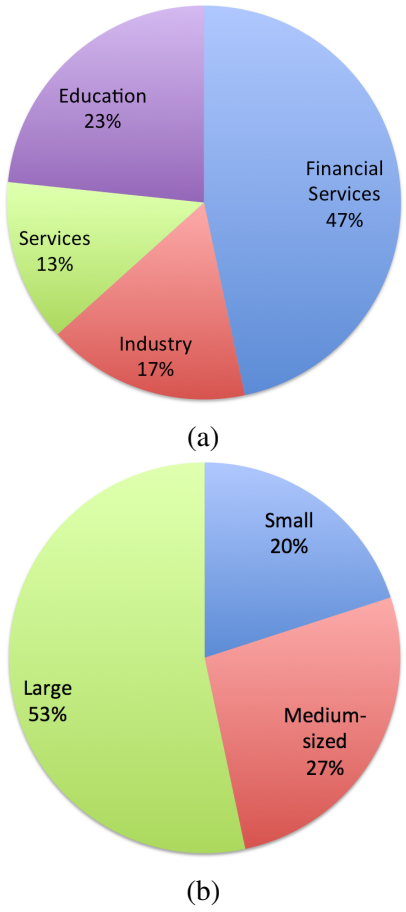


FIGURE 3.3 Distribution of respondents by sector (a), and size (b)

The first topic of the survey asks how organizations present information systems to their employees. More specifically, we asked what are the available modalities of communication between humans and information systems. As can be seen in Figure 3.4, respondents confirmed our first assumption. The majority of applications provide traditional user interfaces with buttons, menus, and other graphical controls. Only 10% of respondents assigned at least one system that provide a natural language interface (written or spoken). As can be seen, even present in several domains of applications, natural interfaces have not gained much attention in organizations until now.
When the company does not provide such interfaces, we asked participants to point out the main reasons for this phenomenon. 60 % of respondents assigned that the organization is too conservative when talking about user interfaces of enterprise systems, and this type of technology is relatively new. Another potential cause, indicated by 30% of respondents is

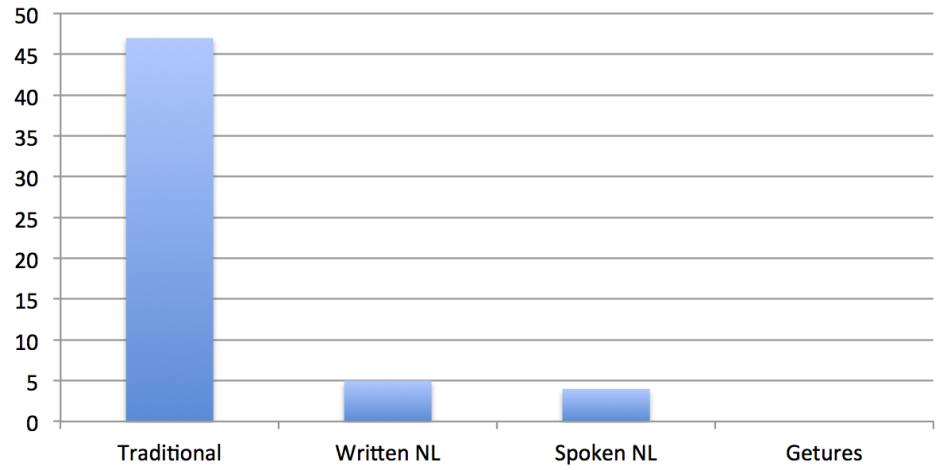


FIGURE 3.4 Current modalities of interaction present in information systems

the impression that the current technology employed for natural language recognition is not mature enough to be used at large scale. Another group (25 %) of respondents believe that a conversational interface may be a cause of distraction and reduction of productivity or people may feel intimidated when talking with an assistant or even chatting with a bot. Given this feedback, we investigated how UI is handled in business process and workflow management systems. A set of platforms has been selected, with different types of licenses, from open source to commercial versions (see Table 3.2). All investigated platforms accept standardized process modeling notations such as BPMN and BPEL and advanced features to improve portability and interoperability. They provide tools for modeling, enactment and monitoring of business processes, as well as integration facilities with web services, ESBs, and legacy applications. However, user interface and usability appears to be is a neglected requirement in these type application. Developers should build interfaces using proprietary UI design tools based on traditional forms to be rendered on the web, in a desktop app or mobile devices. This result in applications with a low degree of flexibility and poor user experience rates.

Back to the survey results, we asked how assistants could help improving the business process environment. Figure 3.5 presents the distribution of selected answers. 68 % of respondents believe that a personal assistant may simplify daily activities by providing a more natural interface and respecting its user preferences. 44 % believe that a personal assistant may improve the collaboration with other users. 15 % adds training and the execution of simple activities on behalf of its user as potential tasks. However, only 9 % assigned a personal assistant for performing monitoring activities. One of the reasons for the rejection of such type of activity is related to the association with the monitoring activity with the degree of autonomy of the personal assistant. According to the study of

TABLE 3.2 List of BPM platforms

Application	**License**	**BPM Standard**	**User Interface**
jBPM	Open Source	BPMN 2.0	Traditional (Form-based)
Enhydra Shark	LGPL	BPMN 2.0 and XPDL	Traditional (Form-based)
FLOWer	Commercial	Case Handling	Traditional (Form-based)
COSA	Commercial	BPMN 2.0 and XPDL	Traditional (Form-based)
SAP Workflow	Commercial	EPC	Traditional (Form-based)
Bonita BPM	Commercial	BPMN 2.0	Traditional (Form-based)
IBM Business Process Manager	Commercial	BPMN 2.0 and BPEL	Traditional (Form-based)
Oracle BPM	Commercial	BPMN 2.0 and BPEL	Traditional (Form-based)

Clavel et al. [25], users prefer assistants with a small level of autonomy, and they do not want assistants to be dominant or intrusive.
Another question made : *do business users want personal assistants with personality ?*
Clavel et al. [25] conducted a study related to social roles and personality of a particular type of agent, called companion agent. They separated the artificial companion in three social categories : (i) *personal assistant*, which manages its user appointments and helps in everyday life, (ii) *intimate friend*, which listen to your problems and helps you find solutions and, (iii) *guardian*, which ensures your safety, protects you and warns you if necessary. Of course, the scope of our work fits well with the first category of *personal assistance* and other categories seems to be out-of-scope of typical tasks in organizations. The study revealed that the majority of users would like their virtual assistant to be cooperative, not hostile, not distrustful and not cynical. The *warmth* aspect has a great importance for *intimate friends*. However, this facet is not much important for *personal assistants* and *guardians*.
Returning back to the modalities of interaction between humans and assistants, we asked respondents to enumerate the most comfortable modalities in three different environments : (i) the typical working environment like a office building or a manufacturing plant, (ii) in a home office context, where employees work at home but have all the infrastructure needed to conduct their activities, and (iii) in mobility situations like waiting for a connection in

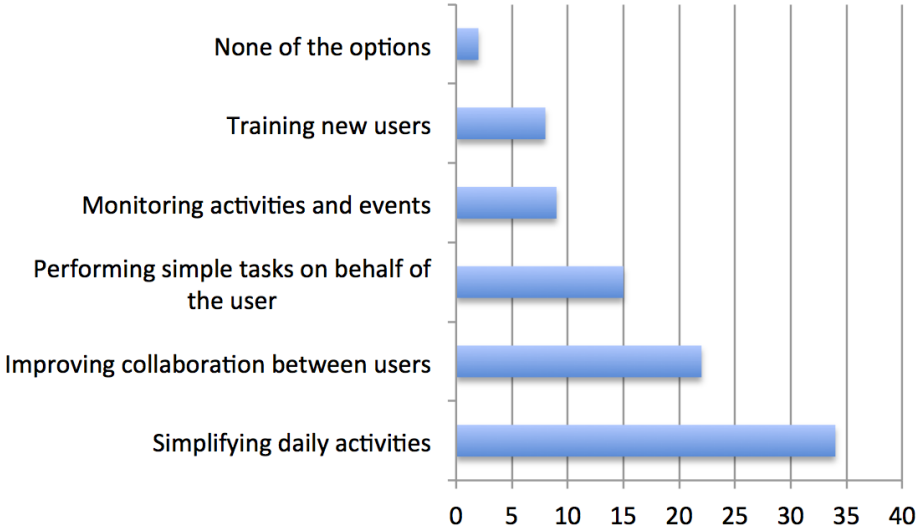


FIGURE 3.5 How personal assistants could improve the business process environment

the airport, or in a train station. We separated the modality topic in two different questions (input and output).

The first question is related to the input channel. Figure 3.6a presents results separated by type of environment. As expected, the number of respondents that selected voice and gestures for the working environment is small (respectively 16 % and 8%) if compared to other modalities. One of the reasons is that people may feel uncomfortable when speaking with the PA in this type of place. Not surprisingly, the traditional interface is highly accepted in working environments. This kind of interface is predominant in organizations and employees are used to work with tables, charts, maps and other graphical controls. Moreover, even in mobility situations, smartphones and their multitouch interface are rather comfortable to select visual items and write short sentences. Free text (sentences written in natural language) have been well accepted in all situations. Again, employees are used to talking with colleagues using groupware tools, instant messaging, and devices used in mobility situations have comfortable interfaces to write small sentences. Point out that we have excluded the situation where the employee is unable to handle a device (e.g. driving a car or working with machine that requires attention) : This situation is out of the scope of our work.

The second question is related to the output channel. Figure 3.7a presents results separated by type of environment. In the same way, the number of respondents that selected gestures is small in all environments. Sentences in natural language written by the personal assistant have been well accepted in all situations. As expected, the usage of gestures and facial expressions have been selected by few participants. It could be related to the low

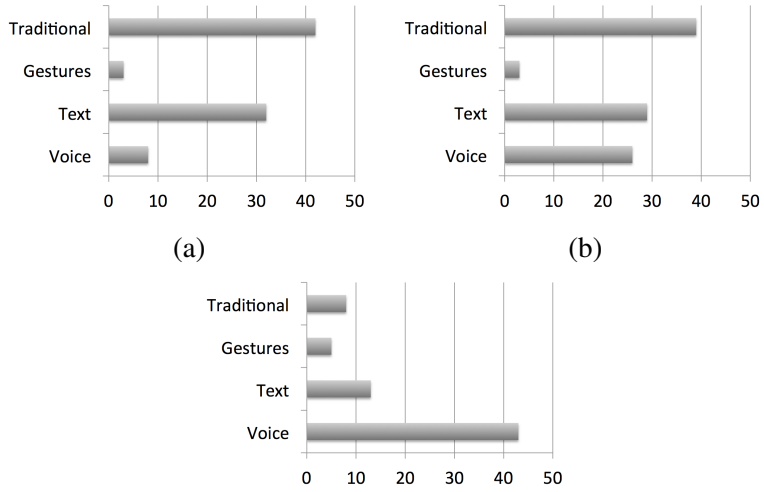


FIGURE 3.6 How humans should communicate with personal assistants in the working environment (a), home-office (b) and in mobility situations (c)

degree of interest in the personality aspect of personal assistants in the corporate environment. More specifically, people do not feel emotionally attached to the assistant, and they consider it as a tool or a "smarter" software component.
In our work, we have chosen a conversational interface to be embedded in our personal assistant. It would be valuable presenting a broad range of interfaces to users such as voice, gesture and facial expression recognition as well as materialize agents as avatars and evaluate their degree of acceptability in the corporate environment. However, this scope is rather significant, and these requirements could be subject to another work, or even an extension of this work. The capabilities of our personal assistant will be limited to the recognition of sentences written in natural language to interpret the user goals, building *ad hoc* graphical interfaces when needed. As a response, the personal assistant might generate natural language sentences or *ad hoc* graphical interfaces such as tables, graphs and charts as needed. Table 3.3 presents a summary with the scope of our work, regarding the personal assistant subject.

3.5 Summary

We have presented in this chapter the foundations of software agents, their properties, how they communicate and how they form a community of agents (MAS) to solve a particular problem. The central element of our work, the personal assistant have been presented, as well as a brief summary of projects related to PAs in our laboratory. We have also explored

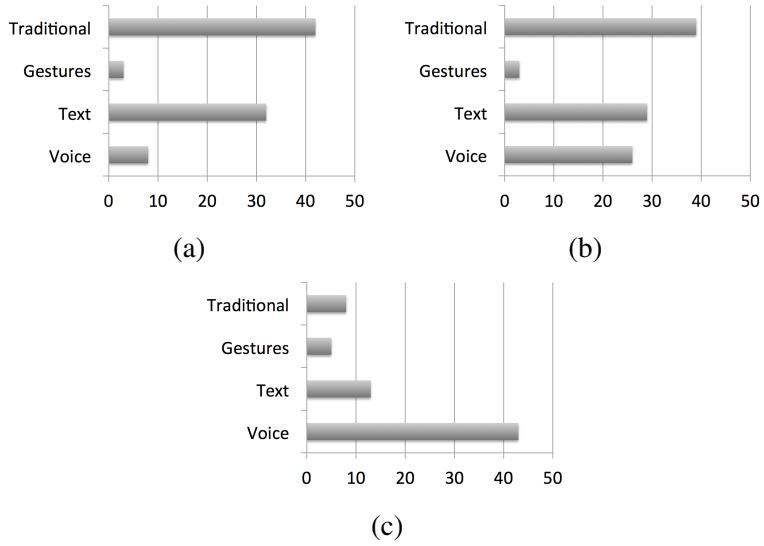


FIGURE 3.7 How personal assistants should communicate with humans in the working environment (a), home-office (b) and in mobility situations (c)

different types of interfaces used in personal assistance (from intimate friends and companions to personal assistants and guardians). A study was conducted to identify how organizations present their information system's interfaces to their end-users. As expected, the traditional interface is predominant. However, we still hypothesize that the personal assistance metaphor could improve the performance during the enactment of business processes. We have identified that a conversational interface that combines the modality of natural language expression, and graphical interfaces could be well accepted by users. Table 3.3 presents a summary of the scope of our work, related to this section.
This scope brings several new requirements that must be investigated to be embedded in our personal assistant. To be eligible to manage a conversational interface, a personal assistant must have at minimum : (i) a robust syntax analysis mechanism, (ii) a robust semantic analysis mechanism that uses one or more domain ontologies for reasoning, (iii) a dialog management mechanism, capable of manage conversations at different levels, and (iv) an explicit memory model used for context management and reference resolution.

TABLE 3.3 The scope of our work related to this section

Perspective	**Approach**
Agent	cognitive agents multiagent system
Knowledge representation	ontologies
Personal assistant interface	conversational
Personal assistant traits	cooperative not hostile not distrustful not cynical
input interface	multimodal written NL traditional interface (when needed)
output interface	multimodal NL generation (text or voice) traditional interface (when needed)

Chapitre 4

Characterizing business processes

The target of our work is the family of BPM systems. A BPM system is a generic software system that is driven by explicit process designs to enact and manage operational business processes [106]. The process model is an artifact of knowledge for an organization. It aims at capturing the different conditions in which a process can be handled, also working as a blueprint. A process is described in terms of tasks, and the ordering of the tasks is modeled by specifying causal dependencies. Also, the process model may specify temporal properties, stipulate how data are used, and designate the way that resources interact with the process [103]. As stated by Dumas [33], the *control flow* is often the backbone of a process model for this type of system. Although other perspectives such as *resource perspective* (modeling roles and authorizations), *data perspective* (flow of data, modeling decisions, data creation and usage), the *function perspective* (describing tasks and related applications) are also essential for comprehensive process models.
To help humans finding and enacting business processes, personal assistants must have access to explicit descriptions of the capabilities of business processes in a machine-readable form. Therefore, this chapter deals with the problem of characterization of business process models, to be used by personal assistants during the enactment of processes.

4.1 Introduction

4.1.1 Problem statement

As stated by Hepp and Roman [46], both the BPM and Web Services communities have yielded a wealth of languages and standardization approaches that aim at describing business processes, especially from the perspective of Web Services orchestration. The

most prominent examples are WS-BPEL [114] and BPMN [73] that focus on the choreography and the orchestration of processes. Models that comply with these standards are used by BPM enactment systems that create process instances, control their flow and combine multiple Web services to perform tasks. A personal assistant in this scenario can be seen as a user representative logged into a BPM enactment system, that selects and triggers processes and receives notifications from the system (e.g. new tasks, finished processes and so on). Thus, the integration between personal assistants and BPM enactment systems does not appear to be a real challenge.
As state before, to be flexible enough, personal assistants must have access to unambiguous descriptions of the capabilities of business processes in a machine-readable form. However, the most significant problems preventing BPM from being successfully implemented are the difficulty in querying processes. The principal reason for these problems is the insufficient semantic information stored in business process space, generally described using languages like WS-BPEL and BPMN. Thus, the problem addressed in this chapter is "the lack of semantic information in business process models". Apart from a wealth of contributions from the semantic web services field (e.g. Klusch et al. [55], Li and Horrocks [62], Paolucci et al. [68], and Trastour et al. [98]), few authors have proposed approaches for adding semantics to business process models. Hepp et al. [45] [44] [46] create the concept of SBPM (Semantic Business Process Management). They propose the combination of BPM and WSMO (Web Service Modeling Ontology). WSMO is used by them to semantically describe each atomic or composite process inside an organization as a semantic web service in a process repository. An SWS (Semantic Web Service) execution environment is proposed for the mediation between business goals and queries, and the actual process space. Leymann [60] introduces the usage of web services to model business processes. Kim [52] [53] proposes the creation of a semantic business process space using the Methontology framework. Business processes are annotated by a set of reusable upper ontologies of business process and a set of ontologies created specifically for the domain. A specialization of a Semantic Web Service matchmaking algorithm is used to query business processes.
We believe that there is a risk of strictly using the semantic descriptors of web services for deriving the capability of business processes. Even if enriched with semantic annotations, the service layer still relies on implementation aspects, which have a different level of granularity and purpose when compared to the business process level. Second, not all tasks are Web services : The outcome of some tasks are exclusively produced by human decisions (e.g. a person that assesses a risk of a grant). Third, even if a "servicification" of all tasks is possible, there is a cardinality issue between tasks and services : A task can

have N services linked with, and a service may be linked with N tasks. Thus, we argue that processes must be treated as first-class citizens in terms of capability description.

4.1.2 Contributions of this chapter

The main contributions present in this chapter are the following ones :

- An approach to describe the capabilities of business processes. Business processes are described in terms of preconditions and effects as inseparable parts. We allow the description of both deterministic and non-deterministic effects, the latter being quite useful for describing human tasks and services in non-deterministic scenarios.
- An approach that allows building a safe approximation to the whole process capability by aggregating all task capabilities present in a business process model.
- A method for querying the business process space.

4.1.3 Organization

This chapter is organized as follows :

- Section 4.2 we first present an illustrative example from the banking domain that will be used in the rest of the chapter.
- Section 4.3 presents a canonical business process model, describing the basic structure of business process models. Readers who are familiar with the family of process modeling languages derived from Petri Nets like BPMN and EPC can skip this section.
- Section 4.4 presents the function perspective of business processes and tasks.
- Section 4.5 presents an approach to modeling the capability model of a business process model, using structural elements such as events, activities, and gateways.
- Section 4.6 improves the representation of preconditions and effects using a variation of the MOSS [9] query language.
- Section 4.7 presents an approach for querying the business process space.

4.2 Illustrative example

Throughout this chapter we use an example to illustrate our approach. This example is a simplification of a real business process in the field of e-banking. It is used to process a customer's credit request (see Figure 4.1). The process starts when the customer contacts the call center or enters into the Internet banking system and chooses this functionality. The first task retrieves the customer profile (task t_1). Having the profile, the system asks for

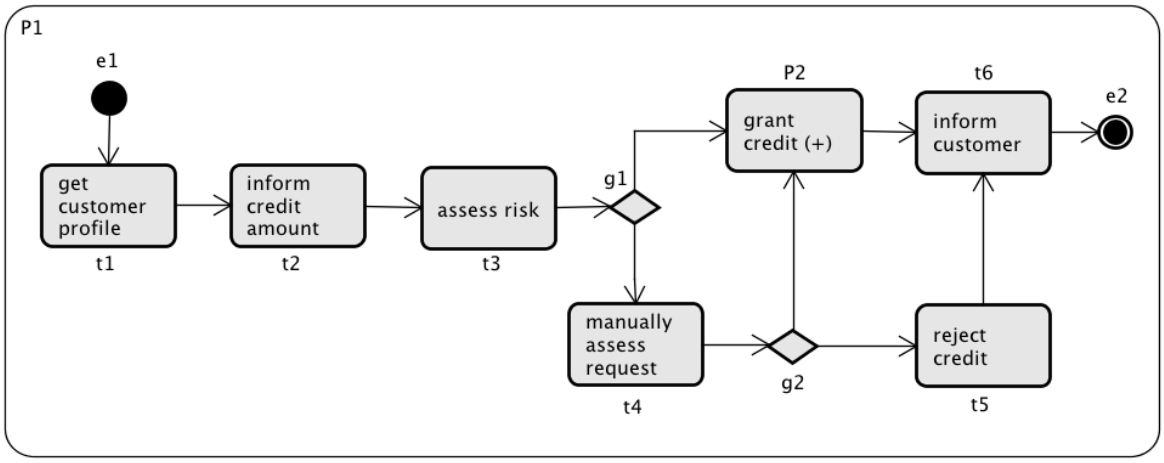


FIGURE 4.1 Illustrative example : a business process model used to process a credit request (P_1) including several tasks and a subprocess P_2

the credit amount (task t_2), followed by an assessment of the credit risk (task t_3). Then either the credit is accepted (sub-process P_2) or a request to manually evaluate the request is started (t_4). The task that manually assesses the credit request is performed by a credit manager who takes the final decision. The manager either approves the request (sub-process P_2) or rejects the request (task t_5). For both cases, the customer is informed of the decision (t_6).

This example uses a simplified variant of the Business Process Modeling Notation (BPMN) [73]. In this notation, tasks are represented by rounded rectangles, marked with the name of the task. Directed arrows express the order of the execution of tasks. Events are shown as circles, and gateways are represented by the diamond symbol, used to split and join the control flow.

This business process model forms a directed graph with focus on the control perspective of the process. The data flow perspective is equally important, but it will be covered in a dedicated section. Despite its simplicity, this process allows three different execution paths from the initial event (e_1) to the final event (e_2) : (i) The customer receives an automatic grant. A risk manager assesses the customer's request and (ii) approves, or (iii) rejects the request.

4.3 Control flow perspective : a canonical process format

As stated before, business process models can be described with a variety of modeling languages. To create a common vocabulary for the rest of the chapter, this section describes a canonical format, adapted from the work of Leopold [59]. An unambiguous

formalization of business processes is required, and this formalization covers the *control flow perspective* and is compliant, at least, with basic control flow patterns identified by van der Aalst et al. in [105]. For the sake of simplicity, this section is dedicated to the definition of tasks and other flow elements, leaving data flow perspective and resource allocation perspective to the following sections.
We start by defining the syntax of a process model as follows :

Definition 4.3.1. (Process model). A process model is a tuple $P = (T, E, G, F, \gamma)$. It consists of three finite sets T, E and G, a binary relation $F \subseteq (T \cup E \cup G) \times (T \cup E \cup G)$, and a function γ such that :

- T is a finite non-empty list of tasks or subprocesses.
- E is a finite list of events.
- G is a finite list of gateways used to describe parallel and conditional flows.
- We write $N = T \cup E \cup G$ for all *nodes of the process model*.
- F is a finite set of sequence flows of P. Each sequence flow $f \in F$ represents a directed edge between two nodes.
- $\gamma : E \rightarrow \{start, end\}$ is a surjective function that maps an event to a type.

Note that typical organizations have hundreds of business process models. A set of such process definitions configures a *process space* [44]. The example below presents our schematic example using the notation.

Example 4.3.1. The example of Figure 4.1 is formalized as $P_1 = (T_1, E_1, G_1, F_1, \gamma)$, such that :

- $T_1 = \{t_1, t_2, t_3, t_4, t_5, t_6, P_2\}$ where t_n are atomic tasks and P_2 is a sub-process,
- $E_1 = \{e_1, e_2\}$,
- $G_1 = \{g_1, g_2\}$,
- $N_1 = \{t_1, t_2, t_3, t_4, t_5, t_6, P_2, e_1, e_2, g_1, g_2\}$
- $F_1 = \{(e_1, t_1), (t_1, t_2), (t_2, t_3), (t_3, g_1), (g_1, t_4), (g_1, P_2), ..., (P_2, t_6), (t_5, t_6), (t_6, e_2)\}$

To give a precise characterization of a process model, we define the set of predecessors and successors of nodes.

Definition 4.3.2. (Predecessors and successors of nodes). Let N be a set of nodes and $F \subseteq N \times N$ a binary representation over N representing the sequence flows. For each node $n \in N$, we define the set of preceding nodes $\bullet n$ with $\{x \in N \mid (x, n) \in F\}$, and the set of successor nodes $n\bullet$ accordingly with $\{x \in N \mid (n, x) \in F\}$.

Example 4.3.2. (Figure 4.1). To discover what task is the predecessor of the gateway g_1 that decides if the credit is automatically granted, we get the content of $\bullet g_4$, which is t_3

(assess risk). To discover what is the successor of the gateway g_2 used to check whether the credit risk manager has approved or not the credit to the customer, we get the content of $g_2\bullet$, which returns P_2 (grant credit) and t_5 (reject credit).

During the validation or the execution of a business process, one might want to discover incoming and outgoing flows.

Definition 4.3.3. (Incoming and outgoing flows). Let N be a set of nodes and $F \subseteq N \times N$ a binary relation over N representing the sequence flows. For each node $n \in N$, we define the set of incoming flows $n_{in} = \{(x,n) \mid x \in N \wedge (x,n) \in F\}$, and the set of outgoing flows $n_{out} = \{(n,x) \mid x \in N \wedge (n,x) \in F\}$.

Example 4.3.3. (Figure 4.1). The set of incoming nodes of the task $t_{2_{in}}$ is $\{(t_1,t_2)\}$. The set of outgoing nodes of the join gateway $g_{2_{out}}$ is $\{(g_2,t_4),(g_2,g_3)\}$.

As a graph of tasks, the business process model must offer a simple approach to query paths.

Definition 4.3.4. (Path). Let $P = (T,E,G,F,\gamma)$ be a process model, and N a set of nodes. There is a path between two nodes $x \in N$ and $y \in N$, denoted with $x \rightsquigarrow y$, if there is a sequence of nodes $n_1,...,n_k \in N$ with $x = n_1$ and $y = n_k$ such that for all $i \in 1,...k-1$ holds $(n_i,n_{i+1}) \in F$.

Example 4.3.4. (Figure 4.1). There are three paths between nodes e_1 (starting event) and e_2 (ending event) :

— Each node in the sequence $\{e_1,t_1,t_2,t_3,P_2,t_6,e_2\}$ has a correspondent in F : $\{(e_1,t_1),(t_1,t_2),(t_2,t_3),(t_3,P_2),(P_2,t_6),(t_6,e_2)\}$

— Each node in the sequence $\{e_1,t_1,t_2,t_3,t_4,P_2,t_6,e_2\}$ has a correspondent in F : $\{(e_1,t_1),(t_1,t_2),(t_2,t_3),(t_3,t_4),(t_4,P_2),(P_2,t_6),(t_6,e_2)\}$

— Each node in the sequence $\{e_1,t_1,t_2,t_3,t_4,t_5,e_2\}$ has a correspondent in F : $\{(e_1,t_1),(t_1,t_2),(t_2,t_3),(t_3,t_4),(t_4,t_5),(t_5,e_2)\}$

It must also offer a simple approach to identify starting and ending events.

Definition 4.3.5. (Start and end events). For a process model $P = (T,E,G,F,\gamma)$ we specify the function $\gamma : E \rightarrow \{start,end\}$, to map an event to a type. An event is a starting event if it does not have predecessors : $\{\exists e \cup E \mid \bullet e = \emptyset\}$. The same rule is valid for an ending event that does not have successors : $\{\exists e \cup E \mid e\bullet = \emptyset\}$

Example 4.3.5. (Figure 4.1).
$\gamma(e_1) = start$, provided that $\bullet e_1 = \emptyset$.
$\gamma(e_2)$ = end, provided that $e_2\bullet = \emptyset$.

As mentioned before, gateways are graphically described by the diamond symbol. They have different semantics depending on their position in the graph and their purpose. For the perspective of position, a gateway may *split* the execution flow or *join* the execution flow. For the perspective of purpose, a gateway could be : (i) *Parallel*, generally identified by the plus (+) sign. It indicates that two or more flows should be performed independently when the gateway is a split gateway, or indicating that one or more flows should be joined in only one execution flow when it is a join gateway. (ii) *Conditional*, stating that the execution of the specified flow must hold a condition to split. The conditional split may be reunited using the conditional join.

Definition 4.3.6. (Gateway subsets). For a process model $P = (T, E, G, F, \gamma)$ we define the gateway subset G as a union of parallel and conditional gateways :

$$G = AND_{split} \cup AND_{join} \cup OR_{split} \cup OR_{join} \tag{4.1}$$

Conditional gateways that split or join the control flow have a condition that must hold to proceed. Hence, we define that each conditional gateway of the process must have a condition that must hold for each outgoing flow.

$$\forall or \in OR_{split} \cup OR_{join} : \forall or_{out} : \exists or_{out_{condition}} \tag{4.2}$$

The example below describes how to represent conditional gateways .

Example 4.3.6. (Figure 4.1)
The gateway g_1 has the following output flows : $\{(g_1, P_2), (g_1, t_4)\}$. Each output flow has a condition that must hold to continue. For example, the flow (g_1, P_2) that automatically approves the request must have a customer classified as low risk or requesting an amount lower than 5.000 €. Conversely, the flow (g_1, t_4) that guides the customer to the manual assessment, must have a customer classified as high risk or requesting an amount greater than 5.000 €.

To be syntactically correct, a process model must adhere to a set of specific requirements. The following definition present a set of rules used to check if a model is syntactically correct.

Definition 4.3.7. (Syntactically correct process model).

1. A process model contains, at least, one task. $|T_n| \geq 1$.
2. Each node not being an event is on a path from a start to an end event.
   $\forall n \in (N_n - E_n)) : (\exists e_1 \in E_n \mid \gamma(e_1) = start), (\exists e_2 \in E_n \mid \gamma(e_2) = end)$, such that $e_1 \rightsquigarrow n \rightsquigarrow e_2$.

3. The latter rule implies that the sets of start and end events are never empty.
   $\{e_1 \in E_n \mid \gamma(e_1) = start\} \neq \emptyset \wedge \{e_2 \in E_n \mid \gamma(e_2) = end\} \neq \emptyset$.
4. Tasks have exactly one incoming and one outgoing flow.
   $\forall t \in T_n : |\bullet t| = 1 \wedge |t \bullet| = 1$.
5. Gateways either have one incoming and multiple outgoing flows or multiple incoming and one outgoing flow.
   $\forall g \in G_n : (|\bullet g| = 1 \wedge |g \bullet| > 1) \vee (\bullet g| > 1 \wedge |g \bullet| = 1)$.
6. There are one or more ending events for the whole process :
   $|\{e \in E_n \mid \gamma(e) = end\}| \geq 1$
7. There is only one start event for the entire process :
   $|\{e \in E_n \mid \gamma(e) = start\}| = 1$

This section provided a definition with emphasis on the control flow perspective and will be used in our work as a common vocabulary. Next section discusses the task model in more details, adding semantics for data and roles.

4.4 Function and resource perspective : The task model

The term *task* has already been mentioned extensively until now. We use this section to provide more precise definitions of task and to define the task model. A task is a logical unit of work and not the performance of the work itself. It can be seen as a precise work to accomplish, with a structure and a well-defined objective. A task can be a system task (e.g. performing services) or a user interaction task (e.g. displaying information to- or requesting information from users). Tasks are reusable building blocks that are organized in a logical order by business processes. A *process model* is a blueprint, specifying which tasks need to be executed to reach the business goal. This section presents (i) how tasks are related to resources, (ii) the process space, and (iii) a model for describing the task and process capabilities.

4.4.1 Task resources

Giving a process model, several process instances can be handled by following the same definition. Thus, the same task definition is concurrently performed by different representations of process instances in a BPM enactment system.
A task that is ready to be executed for a particular process instance is called *work item* (process instance + task). For example, the approval or the rejection of the credit request is

a task that is usually performed by someone that has enough rights to approve or reject the request as soon as the form is submitted. When a task is ready to be performed for a process instance, we have a *work item*. Work items are performed by resources (either a machine or a person) with a required role (or capability). For example, the right role to perform the assessment of a credit request is the *risk manager* role. When a specific resource is allocated to perform this work item, it becomes an *activity* (process instance + task + resource).

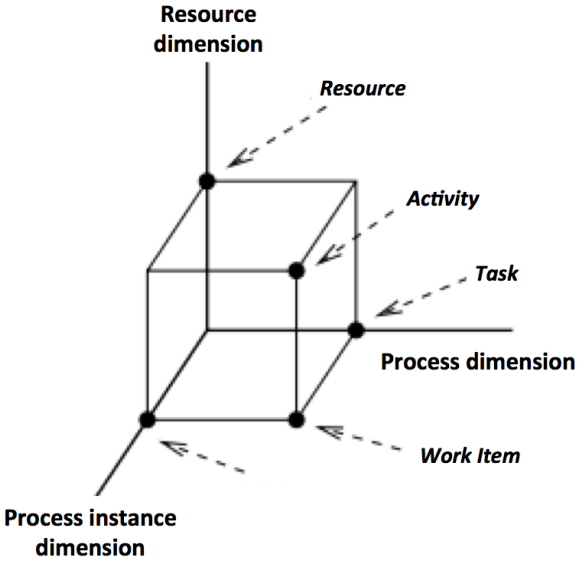


FIGURE 4.2 A three-dimensional view of a business process ([100])

Thus, the association of three elements characterizes an activity : the process, the task and an actor with the authorized role to perform the task. Based on this taxonomy, we define a set representing reusable roles a function that assign a role to an atomic task. During the enactment of a process, an actor that has the required role to perform the task is allocated and the *activity* is performed.

Definition 4.4.1. (Roles and Tasks).

- R is a finite non-empty set of roles.
- $\theta : T \rightarrow R$ is a surjective function that specifies the assignment of a role to an atomic task $t \in T$

Example 4.4.1. (Figure 4.1).

- $R = \{customer, call\ centre\ operator, relationship\ manager\}$
- The required role to perform task t_1 : $\theta(t_1) \in T_1 = customer$
- The required role to perform task t_4 : $\theta(t_4) \in T_2 = relationship\ manager$

4.4.2 Process space

As stated before, we have defined two types of tasks : processes and atomic tasks. A process can be seen as a graph of nodes organized by causal and conditional dependencies. These symbols are events, gateways, and tasks, including subprocesses, which implement a recursive relation. An atomic task has a well-defined scope, and it is linked with an appropriate role to perform it. We strongly support the idea that tasks should be independent and reusable units of work. Hence, a task might be present in different process models. For instance, a task used to *make a credit in a current account* like t_8, could be reused in other business processes (e.g. receiving a compensation from an insurance claim).
Although different in their internal structure, tasks and processes need an abstraction to be equally selected, reused and aggregated. Thus, we define a process space, putting complex (processes) and atomic tasks at the same level. The objective of the unification of atomic tasks and processes is twofold : (i) allow reusing tasks independently of their type (atomic or complex). (ii) Leverage the standardized description, allowing the selection of tasks and processes during the enactment of processes to reach the user's goal.

Definition 4.4.2. (Process space). A process space is a set $PS = P \cup T$, such that :

- P is a finite non-empty set of processes $\{P_1, P_2, ..., P_n\}$.
- T is a finite set of atomic tasks $\{t_1, t_2, t_3, ..., t_n\}$.

Example 4.4.2. The example of Figure 4.1 for the hypothetical ACME company is formalized as $PS_{acme} = P_{acme} \cup T_{acme}$, such that :

- $P_{acme} = \{P_1, P_2\}$,
- $T_{acme} = \{t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8\}$

4.4.3 Task capabilities

According to OASIS [72], a *capability* could be defined as the potential of an action to deliver the world effects or produced information. A semantic specification of tasks can be useful to improve machine readability. We have adopted the same approach that is typically used in standard semantic web service specifications such as OWL-S and WSMO, with additional features for representing both deterministic and non-deterministic effects. Our tasks have capabilities, described in terms of input, output, preconditions and effects (aka. IOPE paradigm).

Definition 4.4.3. (Task capability). For each task present in the process space ($t \in PS$), we define a corresponding capability tuple $t_{cap} = (I, O, PE)$, such that :

- I is a finite set of input parameters, provided that each $i_n \in I$ is a pair $(name, type)$ with the name and the type of the input parameter.
- O is a finite set of output parameters, provided that each $o_n \in O$ is a pair $(name, type)$ with the name and the type of the output parameter.
- PE is a finite set of pairs (p, e), interpreted as $p \implies e$, where p is a logical expression that contains the preconditions that must hold before the performance of the task and e is a logical expression describing the expected effects after performing the task.

Our approach allows two possible forms of specification of preconditions and effects :

— **Deterministic capability ($p \implies e$)** : The majority of approaches that describe preconditions and effects fall into this category. There is a direct implication between what is stated in the set of preconditions p, the execution of the task and the specified effect e. A disjunction is not allowed in the effect expression. This is because the use of expressions containing *or clauses* might cause an ambiguous situation.

— **Non-deterministic capability ($p \implies \Diamond e$)** : The execution of the task under the stated precondition may lead to different effects, expressed as logical disjunctions (with or clauses). In certain situations, business modelers know *how* to specify what are the outcomes of a task and their variations, but they don't know *why* these variations happen. Note that we make use of the diamond symbol ($\Diamond$), used in modal logic expressions to quickly recognize a non-deterministic clause in the effect.

Example 4.4.3. (Deterministic capability). Consider an example of a service that requests an international money transfer, and uses the account type as a criterion to determine if fees will incur or not. A set containing two statements in the format *precondition* $\implies$ *effect*) is defined accordingly :

— $Standard(account) \implies NewIntlTransfer(transfer) \wedge IncurringFees(account, transfer)$

— $Premium(account) \implies$
$NewIntlTransfer(transfer) \wedge NoIncurringFees(account, transfer)$

As can be seen, a deterministic capability is straightforward. The first expression in the example above states : if the customer has a standard account, after performing the task, the outcome is an international transfer with incurring fees. The second expression states : if the customer has a premium account, after performing the task, the outcome is an international transfer without incurring fees.
Non-deterministic capabilities have a different semantic, as can be illustrated in the example below :

Example 4.4.4. (Non-deterministic capability example). Going back to our illustrative example presented in Section 4.2. Consider the task t_3 that automatically assesses the risk of a credit request based on the customer profile and the required amount. After the performance of this task, the outcome is the level of risk for the application (high risk or low risk). Let's say that t_3 is a service that uses a neural network classifier, trained with real-world situations. Each tuple of the training set is labeled as low risk or high risk. The specification of a *precondition* $\Longrightarrow$ *effect* statement that specifies in what conditions a request could be labeled as *low risk* or *high risk* is too complex and maybe unfeasible. That is to say, the decision depends on the classification made by the neural network classifier. Thus, we propose a relaxed syntax of *precondition* $\Longrightarrow$ *effect* defined accordingly :

$$Active(profile) \wedge Valid(amount) \implies \Diamond LowRisk(profile, amount) \vee \Diamond HighRisk(profile, amount) \tag{4.3}$$

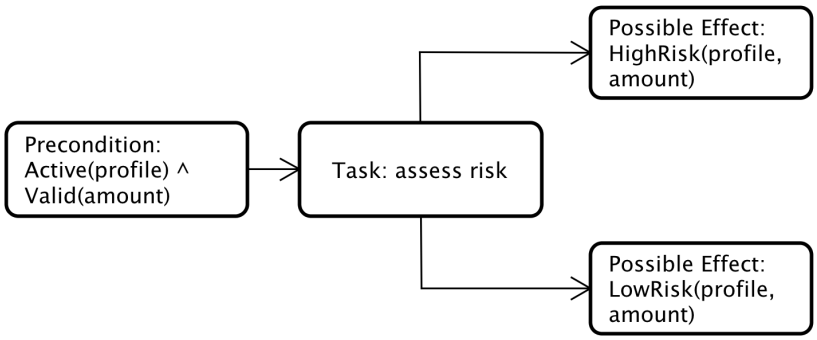


FIGURE 4.3 Graphical interpretation of the preconditions and effects of task t_3

The expression stated in the example above could be read as "*Having an active customer profile and a valid amount, after performing the task, it is possible that this request is classified as* ***low risk****. But it is also possible that this request is classified as* ***high risk***". The precondition $Active(profile) \wedge Valid(amount)$ is a logical formula resulting in only one conjunction. However, the presented effect is a disjunction : $LowRisk(request) \vee HighRisk(request)$, which is interpreted as two possible effects (non-deterministic disjunction). Note that we use the $\Diamond$ symbol used in modal logics for each identified conjunction on the effect expression. This symbol denotes the possibility and can be read as "it is possible that". Figure 4.3 presents a graphical interpretation of the preconditions and effects of the task.
Both preconditions and effects are written in agreement with the disjunctive normal form (DNF) [42]. A disjunctive normal form is a logical formula that is a disjunction of

conjunctive clauses. A logical formula is considered to be in the disjunctive normal form if and only if it is a disjunction of one or more conjunctions of one or more literals.

Example 4.4.5. (Using the disjunctive normal form for specifying both preconditions and effects). Consider a modification in our international money transfer service : When performing money transfer between two international accounts, the source account (who sends the money) must be valid and belong to the European Union or the United States. The destination account must also be valid, but the owner does not have to reside in EU or US. Two possible effects were identified : A successful money transfer with incurring fees or a successful money transfer without incurring fees. The bank in question determines if the customer will be charged with fees depending on the amount of money transferred, the exchange rate of the day, the frequency that the customer makes transfers and the customer profile (standard or premium account for example). Figure 4.4 presents a graphical interpretation of the preconditions and effects of the example below. Note that the precondition is also written according to the DNF specification.

- Input : *sourceAcc*, *destAcc* and *amount*
- Output : $moneyTransf$
- PE : { (EUAcc(source) $\wedge$ Valid(source) $\wedge$ Valid(dest)) $\vee$
  (USAcc(source) $\wedge$ Valid(source) $\wedge$ Valid(dest)) $\Longrightarrow$ $\Diamond$
  TransferWithFee(moneyTransf) $\vee$ $\Diamond$ TransferWithoutFee(moneyTransf) }

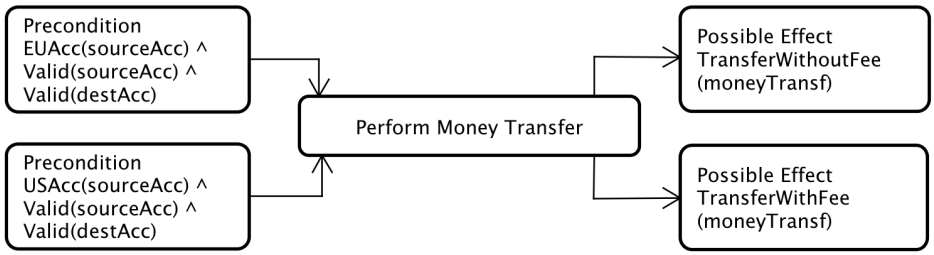


FIGURE 4.4 Graphical interpretation of the preconditions and effects of the international money transfer task

A *flexible* syntax for specifying capabilities is a must in several business process situations. It is because the use of sophisticated statistical approaches and machine learning tools for decision-making is quite common nowadays in organizations, and describing capabilities of this type of task is not trivial. Not only this kind of application requires soft approaches for effect specification : Tasks that produces outcomes based on human decisions like approving a request is an example of a non-deterministic outcome. Other examples are legacy services that are too complex to be described in terms of preconditions and effects

using a declarative approach. We believe that the existing standards used for the specification of preconditions and effects are too constrained. To the extent of our knowledge, no capability model allows the specification of a possible implication.

4.4.4 Deriving the capability matrix

The capability matrix is a derivation of the pair of preconditions and effects. In the end, a matrix preconditions and effects without disjunctions is generated. The objective of this data structure is twofold : (ii) allow an easy query mechanism. Query engines could easily verify if a task can help to accomplish (even partially) a user goal ; (ii) allow determining the capability of a business process based on the causal and conditional ordering of attached tasks. The process that derives the capability matrix is straightforward as can be seen in Algorithm 1. The function τ receives the capability representation of the task (t_{cap}) which contains the set of pairs of preconditions and effects and returns an expanded version containing only conjunctions. As can be seen in lines 3 and 5, the function *buildConjunctions* receives a logical expression compatible with DNF and splits the expression using disjunctions (*or clauses*) as delimiters. We do this for both preconditions and effects.

Algorithm 1 Deriving the capability matrix of tasks (τ)

Require: t_{cap} : task capability
1: $t_{capmat} \leftarrow \emptyset$
2: **for** each $pe \in$ PE(t_{cap}) **do**
3: // get all conjunctions of the precondition
4: precond$_{set}$ $\leftarrow$ buildConjunctions(precondition(pe))
5: // get all conjunctions of the effect
6: effect$_{set}$ $\leftarrow$ buildConjunctions(effect(pe))
7: cap $\leftarrow$ precond$_{set}$ $\times$ effect$_{set}$
8: $t_{capmat} \leftarrow t_{capmat} \cup cap$
9: **end for**
10: **return** t_{capmat}

Hence, each task $t \in PS$ has a capability matrix associated with. The capability matrix of t_n, where the subscript $_n$ is the task identifier, can be obtained accordingly :

$$t_{n_{capmat}} = \tau(t_{n_{cap}}) \tag{4.4}$$

Table 4.1 is the capability matrix $t_{3_{capmat}}$ of task t_3. As can be seen, having an active profile and a valid account, risk profile could be low or high without further explanation

(non-deterministic disjunction). Table 4.2 also presents the capability matrix of the task shown in the Example 4.4.5.

TABLE 4.1 Example 4.4.5 with inputs, outputs, and the capability matrix with preconditions and effects of *credit grant*

input	**output**	**preconditions**	**effects**
profile amount	riskProfile	Active(profile) ∧ Valid(amount)	◊ LowRisk(profile,amount)
		Active(profile) ∧ Valid(amount)	◊ HighRisk(profile,amount)

TABLE 4.2 Example 4.4.5 with inputs, outputs, and the capability matrix with preconditions and effects of *international money transfer*

input	**output**	**preconditions**	**effects**
sourceAcc destAcc amount	moneyTransf	EUAcc(source) ∧ Valid(source) ∧ Valid(dest)	◊ TransferWithFee (moneyTransf)
		USAcc(source) ∧ Valid(source) ∧ Valid(dest)	◊ TransferWithFee (moneyTransf)
		EUAcc(source) ∧ Valid(source) ∧ Valid(dest)	◊ TransferWithoutFee (moneyTransf)
		USAcc(source) ∧ Valid(source) ∧ Valid(dest)	◊ TransferWithoutFee (moneyTransf)

Business process users have different ways to express their needs and formulate their goals. Their statements will depend on the degree of knowledge of the domain, their previous experiences, and current context. The capability matrix could be used to help both types of users finding useful tasks or processes that might assist them to reach their goals. The preconditions of a capability matrix could be interpreted as the description of the user's current situation (e.g. "My credit card was lost" or "I have a European account"). Conversely, effects could be interpreted as the description of the expected situation (e.g. "I want to suspend my credit card" or "I want to make an international money transfer without fees"). As a result, a query mechanism could find a set of processes that fulfill (even partially) the user's objective and also indicate to the user if there is a deterministic implication between the precondition and the effect or a non-deterministic implication between the precondition and the effect.

Until now, we have detailed the flow control perspective of business process models and how to describe the capability of atomic tasks and how to model them. Next sections cover the data flow perspective in business processes and how the business process capability could be derived using all the elements we have so far : control flow model, task model, and data flow model.

4.5 Determining the capability of business processes

In this section, we address the problem of automatically determining the capability of a business process. The resulting capability should explicitly state all preconditions that must hold and their respective effects if the process is performed. While we have predefined preconditions and effects for each task, there is no obvious way of deciding which part of them will be included in the composite specification.
Before explaining how the capability of a business process model is derived, we need first to cover the data flow perspective of a business process.

4.5.1 Data flow perspective

Business processes operate on data. Explicitly representing data, data types, and data dependencies between tasks of a business process put a business process management system in a position to control the transfer of relevant data as generated and processed during process enactment. Now, we fall into the subject of data flow, with works in synchronization with the control flow. To illustrate the dynamics of control flow and data flow, we present in Figure 4.5 our illustrative example. Control flow is represented by solid arcs, and data flow is represented by dotted arcs. Data flow, for instance, between tasks t_1 (get customer profile) and t_3 (assess risk) indicates that t_3 risk requires data that has been created or modified by t_1.

Static versus dynamic data flow

Data is covered by parameters of tasks. Each task parameter has a data type. Each task has a capability with a set of input and output parameters. Whenever the parameter of one activity is used as an input parameter of another task, a data flow occurs. A data flow could be specified *a priori* in the process model or dynamically during the enactment of the process. Both ways of linking data have pros and cons. The definition of an *a priori* data flow is a laborious process since the model must always be up-to-date. The positive aspect is that the data flow is not ambiguous and consequently does not produce undesired effects.

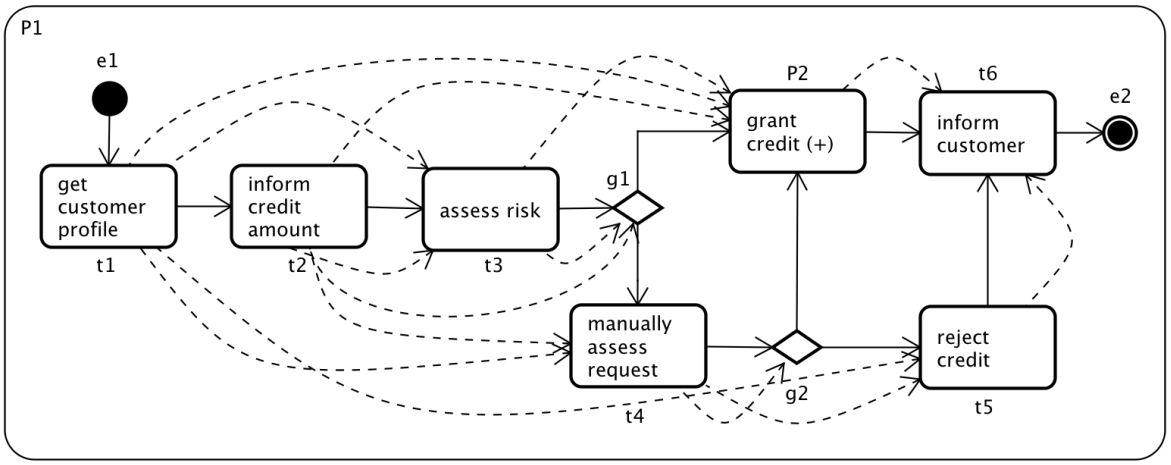


FIGURE 4.5 Control flow (solid arcs) and data flow (dotted arcs) of the illustrative example

For example, a process that transfers money from one account to another account describes without any ambiguity what is the account that will send the amount and what is the account that will receive the amount.
Dynamic data flow between tasks is frequently used in the domain of web service composition. A set of heuristics for matching input and output parameters can be used to link data. One of the advantages is the automatic discovery of data flows based on a set of heuristics. The drawback is that there is no guarantee that the data will be correctly used. It is an issue for critical tasks. Although we do not cover the dynamic aspect in this section, we present how it could be used in our approach in the chapter about results and experiments.

Data flow elements

We extend our canonical model to include the data flow perspective.

Definition 4.5.1. (Dataflow Model). We add the data flow element (D) in our process model, which yields a tuple $P = (T, E, G, F, D, \gamma)$ such that :

- We write an auxiliary set IN as a set of all input parameters of the business process
- We write an auxiliary set OUT as a set of all output parameters of the business process
- D is a binary relation $D \subseteq (IN \cup OUT) \times IN)$ between two parameters

Example 4.5.1. (Data flow example).
Figure 4.6 presents a fragment of our illustrative example P_1'. The process starts by performing task t_1 that expects an account number and returns the customer profile. Task t_2

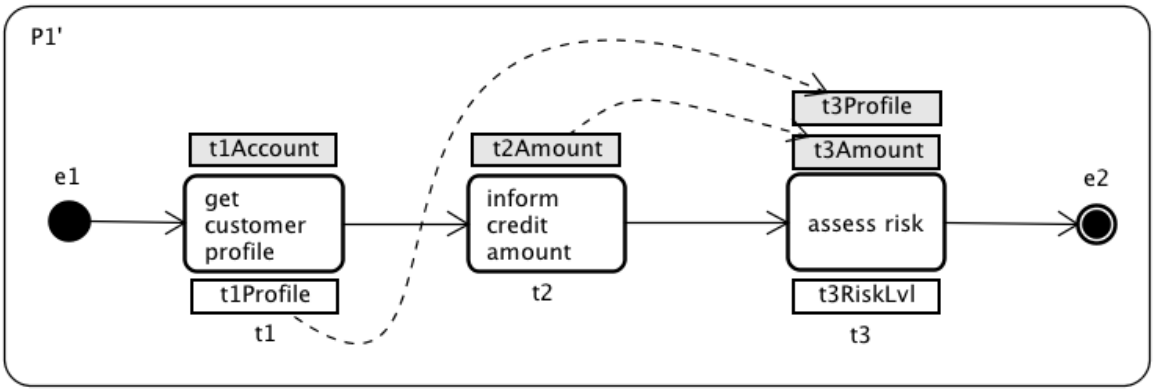


FIGURE 4.6 Control flow (solid arcs) and data flow (dotted arcs) of the illustrative example P_1'

retrieves the credit amount, followed by the risk assessment in t_3. Note that there is a data flow between tasks t_1 and t_3 and also for t_2 and t_3. The risk assessment (t_3) expects the customer profile parameter that is linked to the output parameter of t_1. The same is true for the amount parameter that is connected to the input parameter of the task t_2. Note that, for the sake of clarity, we have given different names for each parameter, adding the task number as a prefix. Of course, parameter names can be reused ; their uniqueness could be preserved since each parameter is linked to the task capability.
The example of Figure 4.6 is formalized as $P_1' = (T_1, E_1, G_1, F_1, D_1, \gamma)$, such that :

- $T_1 = \{t_1, t_2, t_3\}$,
- $t_{1_{cap}} = (\{t1Account\}, \{t1Profile\}, \emptyset, \emptyset)$
- $t_{2_{cap}} = (\{t2Amount\}, \emptyset, \emptyset, \emptyset)$
- $t_{3_{cap}} = (\{t3Profile, t3Amount\}, \{t3RiskLvl\}, \emptyset, \emptyset)$
- $E_1 = \{e_1, e_2\}$,
- $G_1 = \emptyset$,
- $N_1 = \{t_1, t_2, t_3, e_1, e_2\}$
- $F_1 = \{(e_1, t_1), (t_1, t_2), (t_2, t_3), (t_3, e_2)\}$
- $IN_1 = \{t1Account, t2Amount, t3Profile, t3Amount\}$
- $OUT_1 = \{t1Profile, t3RiskLvl\}$
- $D_1 = \{(t1Profile, t2Profile), (t2Amount, t3Amount)\}$

Preconditions and effects have no influence on the explanation. So they were omitted for the sake of simplicity.

Not only tasks but *conditional gateways* (or splits) also depend on data to evaluate their flow conditions. The condition is evaluated as soon as the incoming task is finished. The flow continues if the condition of the corresponding arc evaluates to true. Thus, each

outgoing flow of a conditional or-split has a set of incoming parameters that can be used to test conditions.

$$\forall or \in OR_{split} \cup OR_{join} : \forall or_{out} : \exists or_{out_{data}} \subset IN \cup OUT \tag{4.5}$$

Influences of data flow on the control flow

Until now, we have mentioned only the conditional split as a means of influencing the control flow based on data. However, preconditions of tasks, which are dependent on data, also have a direct influence on the control flow. Is assumed that each control flow edge has an associated condition inherited by the task precondition. This condition is evaluated after the task from which the control flow originates has terminated. If the condition evaluates to true, the follow-up task can be started. However if the condition evaluates to false, the process enactment should be canceled and a rollback signal triggered.

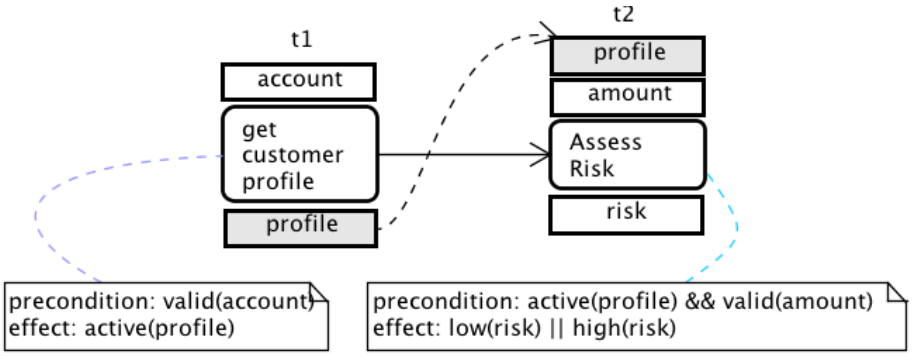


FIGURE 4.7 Fragment of the process model illustrating the relation between data flow, control flow, and task capabilities

Figure 4.7 presents an example of a control flow between two tasks. Task t_1 retrieves the customer profile given the customer account. The output profile of t_1 is wired with the input profile of t_2. Note that t_1 specifies a deterministic effect : it is expected that the task returns an *active(profile)*. The task t_2 has a precondition that is compatible with the produced effect of the previous task : *(active(profile) and valid(amount)*. Thus, the control flow will always be valid for this sequence of tasks.
However, there may even be conflicts such that the specified execution order and the pre-conditions do not match. However, this is something that needs to be dealt with at the operational level (Hepp and Roman [46]). If these inconsistencies occur during runtime the transaction must be rolled back, the user must be informed, and a comprehensive trace log must be generated for further verification.

4.5.2 An algorithm for deriving the business process model capability

We propose an algorithm that receives a business process model as input and returns its capability. In general terms our approach is divided into two steps : the first step finds a set of paths from the starting event, so each path is interpreted as a conjunction. The second step uses the identified set of paths and iterates over its nodes to determine the path capability. As a result, the process capability is obtained from the union of all path capabilities.

Figure 4.8 presents a simplified version of our credit request example, used to explain the algorithm. The process consists of three tasks : The first task (t_1) assesses the credit risk. If the risk is low, task t_2 is performed granting the credit (deterministic effect). If the risk is high, task t_3 is performed, and the credit may or may not be granted (non-deterministic effect). This illustrative example makes use of simple predicates.

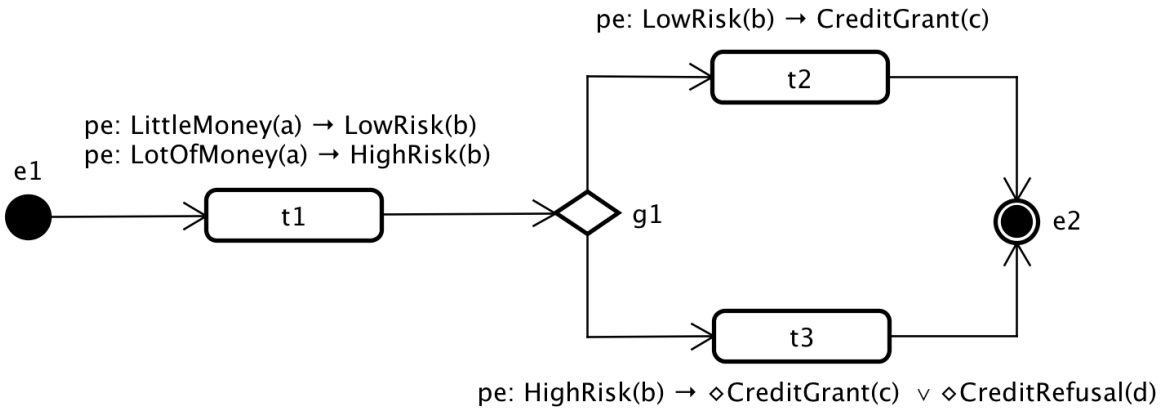


FIGURE 4.8 Illustrative example used to explain the derivation of business process capabilities

Table 4.3 presents the capabilities of each task, including input, output, preconditions and effects.

TABLE 4.3 Capabilities of tasks involved in the process model

Node	Input	Output	Precondition $\Longrightarrow$ Effect
t_1	$\{a\}$	$\{b\}$	$LittleMoney(a) \Longrightarrow LowRisk(b)$
			$LotOfMoney(a) \Longrightarrow HighRisk(b)$
g_1	$\{b\}$	none	$(g_1, t_2, LowRisk(b)) \Longrightarrow true$
			$(g_1, t_3, HighRisk(b)) \Longrightarrow true$
t_2	b	$\{c\}$	$LowRisk(b) \Longrightarrow CreditGrant(c)$
t_3	b	$\{c,d\}$	$HighRisk(b) \Longrightarrow \Diamond CreditGrant(c) \vee \Diamond CreditRefusal(d)$

Finding paths from the initial event

We use an algorithm to find all sequences of execution from the initial event to a final event, forming a set of conjunctions. The output of this algorithm is a set of paths, where the first element of the path is an initial event, and the last element is a final event. The algorithm uses a typical breadth-first search (BFS), starting from the initial node and exploring the neighbor nodes first, before moving to the next level neighbors. The only difference, if compared to the original BFS is that we first need to *flatten* multiple parallel paths (*and splits*) using an arbitrary order of parallel flows.

Example 4.5.2. (Finding paths of the illustrative example (Figure 4.8)).

- $<e_1,t_1,g_1,t_2,e_2>$
- $<e_1,t_1,g_1,t_3,e_2>$

Determining the capabilities of a business process

Now we present the algorithm that determines the capabilities of a business process. At this point, all paths of the business process have been identified. The general idea of the approach is to iterate over each path, getting the capability of the node and propagating its capabilities to the path capability, as defined by the following equation :

$$\forall P_{path_n} \in P_{path} : \exists P_{pathcap_n} = \biguplus \forall n_{cap} \in P_{path_n} \tag{4.6}$$

where P_{path_n} represents each path of the set P_{path}. Each path has an associated capability $P_{pathcap_n}$. This capability is obtained by iterating over the sequence of nodes of the path and propagating each node capability (n_{cap}) to the path capability ($P_{pathcap_n}$). The propagation function is represented by the symbol $\uplus$.
In the end, the union of all path capabilities represents the whole business process capability.

$$P_{cap} = \bigcup \forall P_{pathcap_n} \in P_{pathcap} \tag{4.7}$$

where P_{cap} is the set of capabilities of the process P.
Algorithm 2 presents another alternative to explain how the path capability is determined : The algorithm receives a set of paths as an input parameter. It starts by iterating over each path (line 1). Then, for each path, it iterates over the sequence of nodes (line 3). Inside the loop, the function $\uplus$ is called. As can be seen in line 6, the capability of the process is obtained from the union of all path capabilities.

Algorithm 2 Deriving the capability matrix of business processes

Require: P_{path} : set of paths of the process P
1: **for** each $P_{path_n} \in P_{path}$ **do**
2: var $P_{pathcap_n} \leftarrow \emptyset$
3: **for** each $n \in P_{path_n}$ **do**
4: $P_{pathcap_n} \leftarrow P_{pathcap_n} \uplus n_{cap}$
5: **end for**
6: $P_{cap} \leftarrow P_{cap} \cup P_{pathcap_n}$
7: **end for**
8: **return** P_{cap}

Function $\uplus$ is detailed in Algorithm 3. The algorithm has two distinct behaviors : When the capability is empty, the function simply extracts the capability of the follow-up task and puts a label for each element, using the sequence n as a label. For example, when processing task t_1 with the expression $LittleMoney(a) \implies LowRisk(b)$, the propagated expression will be $(1, LittleMoney(a)) \implies (1, LowRisk(b))$, where 1 denotes the sequence of actions.

When the set of capabilities of the path is not empty, the propagation function must act accordingly :

- Collect all inputs of the task and identify data flows. If a data flow has been found, the input parameter will not be propagated to the path (lines 5 to 9).
- Collect all outputs generated by the task and propagate each one to the path capability (lines 10)
- Collect, **test** and propagate all task preconditions (lines 14 to 27)
- Collect and propagate all task effects (lines 28 to 32)

The process that test preconditions is more complex, since additional tests are necessary as follows :

- The precondition of the task **is not propagated** if it is attended by previous effects already present in the path.
- The precondition of the task **is propagated** if it is attended by at least one non-deterministic effect already present in the path.
- The precondition of the task **is propagated** if the expression contains a reference to at least one parameter that must be filled by the requester.
- The precondition of the task **is propagated** if the algorithm does not have enough information to evaluate the expression. For example, the algorithm cannot evaluate the expression *salary > 1000* when the only information available is a predicate *valid(salary)*.

Algorithm 3 $\uplus$ - Propagating the node capability to the path capability

Require: n :node, $P_{pathcap_n}$
1: $P_{newpathcap_n} \leftarrow \emptyset$
2: **for** each $pc \in P_{pathcap_n}$ **do**
3: **for** each $cap \in n_{cap}$ **do**
4: var $line \leftarrow pc$
5: **for** each $input \in inputSet(cap)$ **do**
6: **if** $input \notin dataFlow(P)$ **then**
7: push (n,input) into inputSet(line)
8: **end if**
9: **end for**
10: **for** each $output \in outputSet(cap)$ **do**
11: push (n,output) into outputSet(line)
12: **end for**
13: var $pathPE \leftarrow \emptyset$
14: **for** each $pe \in preconditionEffectSet(cap)$ **do**
15: **for** each $p \in preconditionSet(pe)$ **do**
16: **if** $p \notin preconditionSet(line)$ **then**
17: **if** $p \in inputSet(line)$ **then**
18: push (seq,p) into preconditionSet(pe)
19: **else**
20: **if** precondition does not hold in previous effects or preconditions **then**
21: mark line as invalid and break the innermost enclosing loop
22: **else if** precondition may hold in previous effects (non-determinism) or precondition cannot be evaluated **then**
23: push (seq,p) into preconditionSet(pe)
24: **end if**
25: **end if**
26: **end if**
27: **end for**
28: **if** valid(line) **then**
29: **for** each e $\in$ effectSet(pe) **do**
30: push (n,e) into effectSet(pe)
31: **end for**
32: **end if**
33: **end for**
34: push pathPE into line
35: push line into $P_{newpathcap_n}$
36: **end for**
37: **end for**
38: $P_{pathcap_n} \leftarrow P_{newpathcap_n}$
39: **return** $P_{pathcap_n}$

Table 4.4 presents the generated capability matrix of the business process. Each produced line is a capability tuple. The whole set of capability tuples could be interpreted as a

conjunction of disjunctions. More precisely, each capability tuple is a conjunction since they do not have *or clauses*. As expected, each tuple will differ because each one may contain a different precondition or a different effect. It is as if there exists an *or clause* separating each capability tuple.

TABLE 4.4 The resulting capability of the business process

p	Input	Output	Precondition $\Longrightarrow$ Effect
1	$(1,\{a\})$	$\{(1,\{b\}),(2,\{c\})\}$	$(1,LittleMoney(a)) \Longrightarrow (1_{t1},LowRisk(b)) \wedge (2_{t2},CreditGrant(c)$
2	$(1,\{a\})$	$\{(1,\{b\}),(2,\{c,d\})\}$	$(1,LofOfMoney(a) \Longrightarrow (1,HighRisk(b)) \wedge (2,\Diamond CreditGrant(c))$
3	$(1,\{a\})$	$\{(1,\{b\}),(2,\{c,d\})\}$	$(1,LofOfMoney(a) \Longrightarrow (1,HighRisk(b)) \wedge (2,\Diamond CreditRefusal(d))$

The first column of the table (**p**) indicates the id of the path. The second column (**Input**) contains the set of input parameters that are required to produce the capability. Note that in this example, all tuples require the same set of input parameters. But this is not always the case : depending on the path, the set of required inputs and their order is extracted from each node in the sequence, thus they may vary. The third column (**Output**) presents the set of generated outputs for each tuple. As it happens with input parameters, the set of produced outputs and their order may vary. The **Precondition** $\Longrightarrow$ **Effect** column presents the conditions that must hold before the execution of the business process and the corresponding effects that are produced by the execution of the business process.
Note that each element in the tuple is written as a pair $(sequence, element)$. The sequence information is useful to reproduce the capability in a temporal order. For example, the first tuple indicates that output parameter b is generated by task $t1$ in the first step and c in the second step by task t_2. The second tuple has a precondition stating that condition $LotOfMoney(a)$ must hold. Note that the third tuple holds the same precondition as the second tuple. Both tuples share the same effects during the first step, stating that a will present a high risk profile $HighRisk(b)$ at the end. But in the second step they differ : The former states that a credit grant may be produced ($\Diamond CreditGrant(c)$) Conversely, the latter states that a credit refusal may be produced ($\Diamond CreditRefusal(d)$). The reason for this dichotomy is the original $(precondition \Longrightarrow effect)$ statement of t_3. The expression $HighRisk(b) \Longrightarrow CreditGrant(c) \vee CreditRefusal(d)$ have a non-deterministic disjunction in the effect part. Thus, if $HighRisk(b)$ is true, $CreditGrant(c)$ or $CreditRefusal(c)$ could be produced. The first capability tuple has a similarity with the second tuple : both produce a credit grant. The difference resides in the quality of the expression, because the effect of the first tuple is deterministic.
The number of tuples may vary according to the number of paths, the number of tasks and the degree of complexity of preconditions and effects. The resulting set could be useful for answering queries. For example, a requestor may ask : I want a credit grant. The answer

could be : if you request little money you will **certainly have** a credit grant. But if you request a lot of money you **may have** a credit grant.

TABLE 4.5 Simplified capabilities of the business process

Input :	$\{a\}$
Output :	$\{b, c, (d, optional)\}$
Precondition $\Longrightarrow$ Effect	$LittleMoney(a) \Longrightarrow LowRisk(b) \wedge CreditGrant(c)$
	$LotOfMonet(a) \Longrightarrow \Diamond CreditGrant(c) \vee \Diamond CreditRefusal(d)$

Of course, the capability matrix is machine readable, it is not appropriate for humans. A very simple step used to simplify the capability of business processes, removing temporal labels, aggregating effects, could improve readability as can be seen in Table 4.5.
For more information about the business process capability derivation, we have prepared a step-by-step example of the credit grant scenario, available in Appendix **??**.

4.6 Improving the data representation perspective

In previous sections, we have shown how to build the capability model of a business process model by making explicit three important aspects that could be used to query and select processes : control flow, resource allocation, and data flow. Until now we have used simple predicates, but real-world situations demand a richer data representation mechanism, describing concepts, properties and relations.
We propose to specify both preconditions and effects in terms of ontology queries. It is assumed that each domain has one or more ontologies describing concepts and properties and how they relate. We use a simplified subset of the MOSS query language [9]. Next sections give more details on how our query system can be used to improve the description of data.

4.6.1 Improving the description of preconditions

A *precondition* is a logical expression that returns a Boolean value. Variables that have simple types (e.g. numbers and strings) can be easily tested with common relational operators ($=$,$\neq$,<,>,$\leq$, $\geq$). Variables that have complex types, those associated with concepts could be tested by using a predicate that checks the truth of a given query. For instance, let us assume a task that is used to to buy stocks from the NYSE (New York Stock Exchange). The task capability below could be used to describe that in order to buy, the market must be opened. Note that we have omitted the output and effect part, for the sake of simplicity :

```
(deftask buy-stock
   (input @we:world-exchange ... )
   (output ... )
   ((:precondition (@we
             (has-acronym = "NYSE")
             (has-opening-time <= @current-time)
             (has-closing-time >= @current-time)))
    (:effect ... ))
)
```

@*we* is a variable, more specifically an input parameter of the task *buy-stock*, with type *world-exchange*. This constraint states that in order to be performed, the requester must pass an individual of type *world-exchange* as a parameter. The precondition states that the individual referenced by the parameter @*we* must have the attribute *acronym* equal to "NYSE" and the current-time within the @*opening-time* and @*closing-time* ranges.

4.6.2 Improving the description of effects

The execution of a task produces one or more effects, causing a world state transition. A description of an effect is not the real effect itself, but a pattern or a generalization indicating what the world will look like after the service execution. Using our previous example of stock orders, we could write the effects of the service processing the order as follows :

```
(deftask buy-stock
  (input @we:world-exchange @identifier:string @price:real)
  (output @order:trading-order @bill:commission-bill)
  ((:precondition
     (@we (has-acronym = "NYSE")
               (has-opening-time <= @current-time)
               (has-closing-time >= @current-time))
  (:effect
     (or (and (:action "create" @order
                 (has-status     = "active")
                 (has-identifier = @identifier)
                 (has-quantity   = @quantity)
                 (has-price      = @price)
                 (has-daily-lim >= @lim-inf)
                 (has-daily-lim <= @lim-sup))
```

```
        (:action "create" @bill
           (has-type = "house") (has-order = @order})))
    (:action "create" "error-log" (has-source = "trading-order")))))))
```

Here, two different effects may occur : The first effect is the creation of an *active trading order*, followed by a commission bill. Note that designers are not limited to use only the equal operator. They are free to use negations and relational operators. The second effect occurs if something wrong happens (for example, an expired credential or even a technical error). The logical expression of the effect gives a description of different possibilities, configuring a non-deterministic disjunction.

4.6.3 Action taxonomy

An effect always has an action verb associated with a concept of the domain ontology, sometimes accompanied by a criterion. We have proposed an action taxonomy to improve the expressiveness of effects, following a similar approach of Paraiso et al. [75] and Derguech [31]. The main difference between these approaches is that our action is fine-grained. That is to say, actions characterize what is the basic operation that is going to perform by the business process. This is why an action must directly or indirectly inherit from one of four basic actions namely *create*, *modify*, *get* and *remove*.
To illustrate how an action verb is inserted in this context, we present an example in Figure 4.9. It presents an action ontology containing a hierarchy of actions. Three leaves are linked with a concept of the domain ontology. In this particular scenario, a bank customer could *apply* for an investment, make an early *withdrawal* or *check the performance* of an investment.
This action taxonomy has the following characteristics : Actions are part of an action ontology and can be organized in a hierarchical fashion, so subsumption matching is allowed. It is easy to infer if the action is *world-altering* or *knowledge-providing*. A *world-altering* action changes the state of the world. More precisely, an action that is subsumed by one of the following actions : *create*, *modify* or *remove*. A *knowledge-providing* action makes some information of the concept available to the requester.

4.6.4 Condition format

Both preconditions and effects follow the BNF format described below :

```
<condition> ::= ({<action>}} <concept> | <concept-var> {<clause>}*)
```

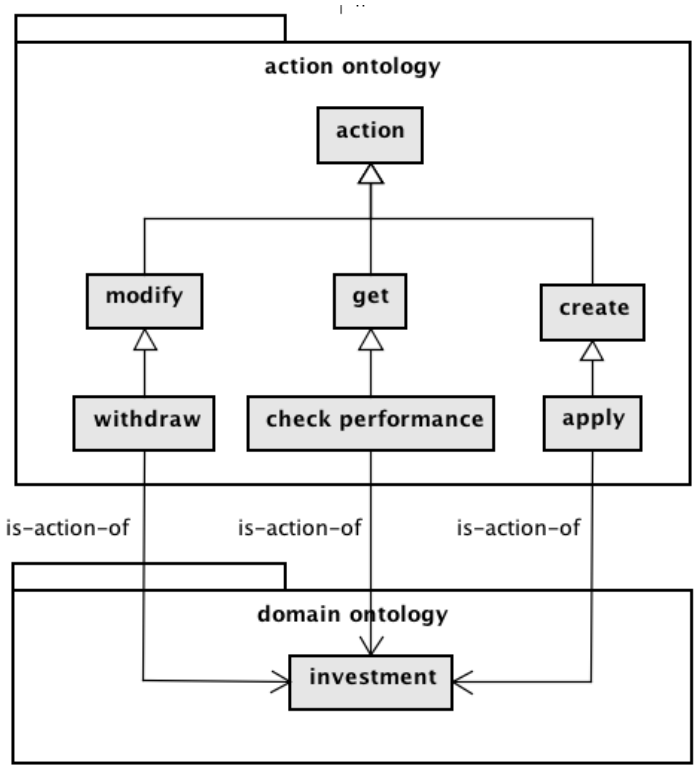


FIGURE 4.9 The action ontology containing a hierarchy of actions

```
        <action> ::= <action-keyword> <action-concept>
<action-keyword> ::= :action
<action-concept> ::= the name of the action
       <concept> ::= the name of the concept
   <concept-var> ::= <variable> having a concept as a type
      <variable> ::= the name of the variable starting with @
        <clause> ::= <attr-clause> | <sub-query>
   <attr-clause> ::= (<attribute> <attr-op> <value>) |
                     (<attribute> <attr-op> <variable>)
     <sub-query> ::= (<relation> <concept> {<clause>}*)
       <attr-op> ::= < | <= | = | != | >= | >
      <variable> ::= symbol starting with @
     <attribute> ::= the name of the attribute
         <value> ::= constant value
```

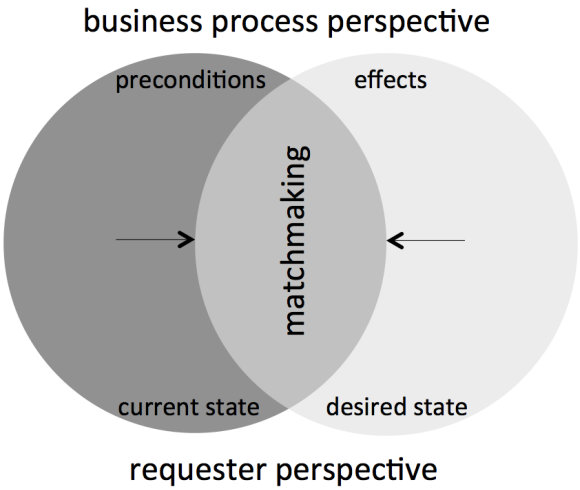


FIGURE 4.10 Semantic matching using the process space

4.7 Semantic matching using the process space

This section illustrates how semantic matching can be used for querying business processes aligning the requester perspective and the business process capability perspective (see Figure 4.10. Semantic matching is typically used to discover exact or similar concepts between one party and another. In the context of SOA, *matchmaking* has been used to find proper semantic web services that satisfy the requirements of a requester among a number of advertised services (e.g. Klusch et al.[55], Li and Horrocks [62], Paolucci et al. [68], and Trastour et al. [98]). In our approach, we look for business processes using both preconditions and effects. The matchmaking could be summarized as follows : The agent first matches the desired state of the requester with the effects of the advertised business process (i.e. both of them in the form of individuals). Next, the algorithm matches the preconditions of the advertised business process with the information provided by the requester. Finally, it computes the degree of semantic matching between them.

4.7.1 Matching concepts and their actions

We have chosen an approach for matching concepts and actions that compute a degree of match between parties. The concept of degree proposed by Paolucci et al. [68] has been used. They propose four degrees of match, described below :

— **Exact match :** the requester's concept matches exactly with the provider's concept.
— **Plug-in match :** the provider's concept subsumes the requester's concept (In this case, the former could be plugged in by the latter).

— **Subsumes match :** the requester's concept subsumes the provider's concept.
— **Fail :** no subsumption exists between the two concepts.

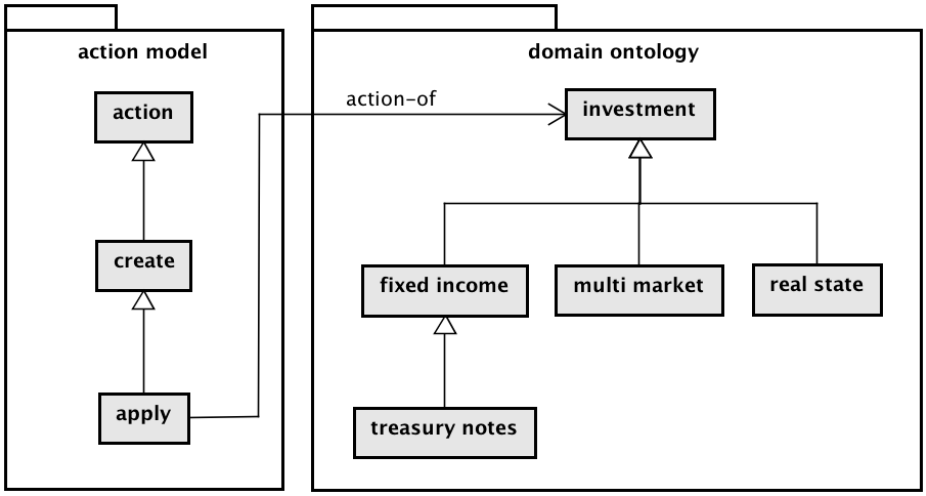


FIGURE 4.11 A fragment of the investment ontology used to illustrate the concept of semantic matching

To illustrate the concept of semantic matching degree, consider the investment ontology depicted in Figure 4.11. Suppose a requester wants a concept of type *fixed income*. If a provider's output concept is also *fixed income*, this case corresponds to an *exact match*. If the provider produces an *investment*, then *investment* can be *plugged in* by *fixed income* and this case corresponds to a *plug-in match*. Instead, however, if the concept described in the effect of the provider is *treasury notes*, that is, if *fixed income* subsumes *treasury notes*, this corresponds to a *subsumes match*. The matching degree progresses in decreasing order as follows : *exact match*, *plug-in match*, *subsumes match*, and *fail*. The matching degree for each concept is collected to compute the total matching degree of a provider's preconditions and effects against the requester's current and desired state. The same strategy for matching is used for actions. For instance, the *apply* action is a concept that has a *action-of* relation with *investment*. If the requester wants to *apply* for an *investment* and the provider produces an *apply* for an *investment*, then we have an *exact match*. Instead, however if the provider produces a *create* for an *investment*, then we have a *plug-in match*.

4.7.2 Matching attribute clauses

If there is no match between concepts and actions, then the advertised process is discarded. Instead, however, if there is a minimum match (i.e. subsumes for both action and concept

are the worst cases), the algorithm compares each attribute clause present in the requester's concept with the attribute clause present in the provider's concept. Note that our approach allows not only equality operators but also other relational operators (i.e. $>$,$\geq$,$<$,$\leq$,$=$, and $\neq$). This functionality augments the expressiveness power of the query, consequently improving the results of the matchmaking. As a consequence, it augments the degree of complexity, since not all relational operators are comparable. For instance, the clauses $a > 10$ and $a < 50$ are not comparable, as well as the clauses $a > 10$ and $a \neq 50$. When operators are not comparable, the algorithm simply ignores them.

TABLE 4.6 Comparison matrix for matching relational operators of the requester and the provider

x \ y	$at > val$	$at \geq val$	$at < val$	$at \leq val$	$at = val$	$at \neq val$
$at > val$	$val_y > val_x - 1$	$val_y > val_x$	n/a	n/a	$val_y > val_x$	n/a
$at \geq val$	$val_y + 1 \geq val_x$	$val_y \geq val_x$	n/a	n/a	$val_y \geq val_x$	n/a
$at < val$	n/c	n/c	$val_y - 1 < val_x$	$val_y < val_x$	$val_y < val_x$	n/a
$at \leq val$	n/a	n/a	$val_y - 1 \leq val_x$	$val_y \leq val_x$	$val_y \leq val_x$	n/a
$at = val$	n/a	n/a	n/a	n/a	$val_y = val_x$	n/a
$at \neq val$	n/a	n/a	n/a	n/a	$val_y \neq val_x$	$val_y = val_x$

Table 4.6 presents a compatibility matrix between two parties using the expression $at\ op\ val$, where at is the attribute, op is the relational operator and val is the value. The first column of the table contains the operators of the interested party, represented by the var x, and the first line of the table contains operators of the evaluated party, represented by the var y. Note that the roles *interested party* and *evaluated party* may vary during the matchmaking. More precisely, when the algorithm is evaluating the effects of a business process, the interested party is the requester and the business process capability is the evaluated party. The information provided by the requester is used to evaluate the matching degree of the business process. For instance, the requester is interested in processes that make investments with administration fees lower than 1% per year. Conversely, during the evaluation of preconditions, the roles are inverted. The interested party becomes the business process and the evaluated party becomes the requester. For instance, the business process that applies to an investment checks if the amount informed by the user is greater than 10,000 €.

4.7.3 Matching effects

We provide an example to explain how the matchmaking of effects works. Suppose that a customer is interested in investing 10,000.00 €. He expects a return on investment greater than 8%, an administrative fee lower than 2% a year, and he accepts a medium risk level.

The algorithms starts by traversing the set of advertised business processes, comparing the effects produced by each advertised process with the desired configuration of the customer. Table 4.7 presents the result of the matchmaking of effects for the given example. Three processes that are used to apply for an investment have been found : All of them produces a compatible concept using a compatible action. The first business process used to apply for a *multi market fund* reached a score of 92%. From the effect perspective, It is the process that best fits with the customer requirements (e.g. level of risk, fees, and return on investment). The remaining processes have a more modest score since some criteria are not compatible with the customer requirements.

TABLE 4.7 Result of the matchmaking of effects for the investment example

involved party	**action**	**concept**	**average ROI** [1]		**fee**	**risk**	**amount**	
requester	apply	investment	>8		<2	medium	= 10,000.00 €	
involved party	**action**	**concept**	**average ROI**		**fee**	**risk**	**amount**	**effect score**
apply for multi market	apply	multi market	>9	<11	= 1.5	medium	>= 100000.00	4.6/5 = 0.92
score :	1	0.6	1	n/a	1	1		
apply for treasury notes	apply	fixed income	= 4		= 0.7	low	>= 20000	1.6/5 = 0.32
score :	1	0.6	0		1	-1		
apply for real state	apply	real state	>3	<6	= 1.0	low	>= 5000	0.6/5 = 0.12
score :	1	0.6	-1	n/a	1	-1	n/a	

We have followed a strategy similar to the one proposed by Kim in [52] for calculating the effect score. The degree of match is used to calculate the score for the concept and the action (1.0 to exact match, 0.8 to plug-in match and 0.6 to subsumes match). In our illustrative example, all concepts have received the score 0.6, since none of them matched exactly with what the customer specified : *investment* subsumes *multi market*, *fixed income* and *real state*.

Non-deterministic effects should have a score lower than a deterministic effect. We have chosen an empirical method for penalizing such occurrence. In the case of a non-deterministic effect, the score is multiplied by 0.8.

For clauses, we compute the score using the following criteria : if the clause of the requester is compatible with the clause of the provider, then we assign the score 1.0. If the operators of the requester and the provider are not compatible (e.g. $>$ and $<$), we ignore it, assigning a score of zero. If the clauses do not match, we assign a score equal to -1.0 to penalize the overall score of the line. After computing all column scores, the score is computed for the line : the sum of all column scores divided by the number of criteria of the requester.

4.7.4 Matching preconditions

The semantics of matchmaking for preconditions is the same of the matchmaking of effects. However, an inversion of roles occurs. That is to say, during the evaluation of preconditions, it is the business process that evaluates if the requester content is compatible with the set of preconditions.
The same approach is used for computing the score of clauses, as shown in the previous section. A score of 1.0 for compatible clauses, a score of 0 for clauses that are not comparable and a score of -1 for incompatible clauses. After computing all scores, the average score is computed for the line. The average score is the sum of all scores divided by the number of criteria present in the precondition.
Table 4.8 presents the result of the matchmaking. As can be seen, each business process contains a precondition using the *amount*. The difference is that each process specifies a different criteria of acceptance. For example, the *apply for real state* requires an amount greater of equal than 5.000 €. This is compatible with the amount that the customer is expecting to invest. Thus, the score of the column is computed with 1.0. However, the remaining business processes require an amount that is greater than the customer amount. Thus, the calculated score is -1.0 for each business process.
In the end, the overall score is computed as an average of the score of the effect and the score of the precondition. Note that the process that applies for a *real state* investment is the most suitable for the customer. If we only use the effect criteria for ranking processes, the *multi marked fund* is the most suitable one. Figure 4.12 illustrates the matchmaking using the requester perspective.

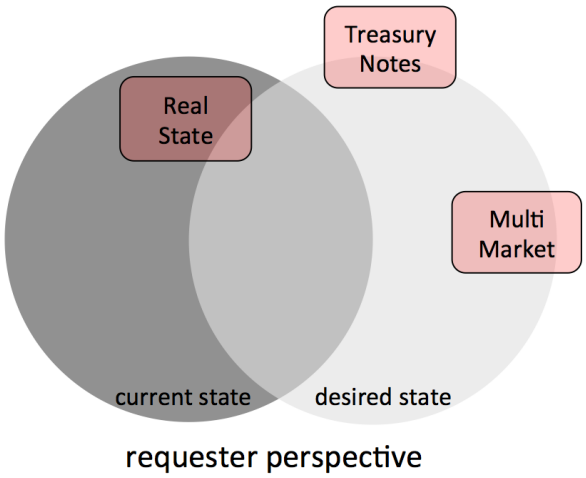


FIGURE 4.12 Result of the overall matchmaking for the investment example

TABLE 4.8 Result of the overall matchmaking for the investment example

involved party	action	concept	amount
requester	apply	investment	10,000.00 €

involved party	action	concept	amount	effect score	precond score	overall score
apply for real state	apply	real state	>= 5000	0.16	1	0.58
score :	1	0.6	1			
apply for multi market fund	apply	multi market	>= 100000.00	0.92	-1	-0,08
score :	1	0.6	-1			
apply for treasury notes	apply	fixed income	>= 20000	0.36	-1	-0,64
score :	1	0.6	-1			

4.7.5 Sending results to the requester

The score varies from -1 to 1. To send results to the requester, we first filter the list of process capabilities that have a score greater than zero. Since two or more process capabilities may appear in the list, capabilities are grouped into a single line and the score of the process is the average of all capability lines. Finally, the summarized list of processes with the average score greater than zero is sent to the requester.

4.8 Discussion

We started this chapter by discussing the problem of a lack of machine-readable representation of business processes. Since they do not provide a process space with sufficient semantics for querying business processes, it is impossible for stakeholders (e.g. personal assistants) to search useful processes directly from the process space. At the same time we draw a scenario where personal assistant have access to a unified view of business process capabilities, so assistants could benefit from a more expressive query mechanism. As a second step, we proposed an approach to describe the capabilities of business processes. The approach allows describing processes in terms of preconditions and effects. The novelty presented here is the ability to describe both deterministic and non-deterministic effects, which is useful for modeling tasks whose outcomes are based on human decisions and also for non-deterministic services. The second contribution is a derivation or safe approximation of a business process model based on the capabilities of tasks. We then showed how to search for appropriate business processes using a semantic matching technique in the process space. The overall approach can be seen as an attempt to improve the expressiveness power of business processes.
This research contributes to the field in two ways : (i) building an approach for describing business process capabilities and derive a process capability based on its building blocks

(tasks, events, and gateways). (ii) Demonstrating how semantic matching can be utilized to find the advertised business processes the most similar to a requested business process using a semantic matching approach that attempts to match functional properties.
Although annotating business processes with semantic information appears to be time-consuming and complex, the resulting knowledge can substantially improve the usage of the process space, and accelerate the development of new functionalities, even those that require a high level of knowledge representation like dialog-based interfaces.
The mechanism is at the time of writing at an early stage of development, and a number of issues and possible improvements have been detected during the designing and preliminary tests of the approach, as follows :

- We have validated the approach using a small subset of examples from the banking domain. A comprehensive evaluation using different domains is necessary.
- The algorithm that derives the business process capabilities does not allow cycles in this version.
- Even if the final representation in the form of capability matrix is machine-readable, the final format is not easy to read for humans.

The next chapter describes a conversational interface that uses this model as a source for conducting a task-oriented dialog for the discovery and the enactment of business processes.

Chapitre 5

A conversational interface for enabling the enactment of business processes

5.1 Introduction

We have studied in Chapter 2 how BPM systems influences organizations and how processes are enacted in organizations. As stated by van der Aalst [103], BPM systems are strongly dependent on process models. These models are used as a source for the creation of process instances, the generation of user interfaces, and the orchestration of process. We present in this chapter the architecture of the conversational interface that will be used to realize our approach for business process enactment using personal assistants and business process models. This chapter describes all the steps and techniques used to analyze and design the architecture, with particular attention to the interpretation of user's sentences to discover, select and execute business processes.

5.1.1 Contributions of this chapter

The main contribution of this chapter can be summarized as follows :

1. A process space query mechanism based on the interpretation of user sentences.
2. A dialog manager adapted to the business process enactment problem.

5.1.2 Organization

This chapter is organized as follows :

— Section 5.2 presents our baseline for dialog management. The OMAS dialog system is presented, as well as some illustrative examples of applications developed using this platform.
— Section 5.3 explains the dialog management mechanism and an example of implementation of our illustrative example of credit grant.
— Section 5.4 presents our approach for the enactment of business processes using personal assistants called PA4Biz, organized in the following order :
  — The overall *architecture*
  — The *top-level conversation* graph
  — The *syntactic annotation* process.
  — The *semantic annotation* process.
  — The *business process selection and triggering* mechanism.
  — The *business process enactment* mechanism.

5.2 Our baseline : The OMAS dialog system

OMAS [8] is a multi-agent platform that implements a number of mechanisms found in other multi-agent platforms, including our main interest : It allows the design and implementation of conversational interfaces using personal assistants. OMAS is a research platform and as such undergoes a number of changes, extensions, and improvements continuously. The dialog system has been used in several research and industrial projects during the last few years and some practical examples are shown in the beginning of this section. Thus, we benefit from the research effort and its outcomes, adopting the OMAS conversational interface as a baseline for building our approach targeted to the enactment of business processes.
As mentioned before, a personal assistant (PA) in the OMAS platform is a special type of agent that provides a human interface to facilitate the interaction with complex systems. The interaction takes place through a natural language dialog using written or spoken input. Each person has a personal assistant (PA) that hosts its user profile and eventually learns the user's habits and interests from continuing interaction.
One important component of the PA is the *dialog manager*. In general terms, the dialog manager is responsible for managing the dialog flow and maintaining a representation of the dialog state for decision-making purposes. It is a rather complex process, and the whole mechanism will be explained in details in a dedicated section. From now, some practical examples of conversational interfaces built with OMAS are shown.

5.2.1 Examples of conversational interfaces built with OMAS

Some examples of projects that use the conversational interface have been selected, each one having different modalities of interaction. For the sake of simplicity we selected only a few. More examples of conversational interfaces using OMAS can be found in [8] [96], [80] [75] [50].

The TATIN project

Here a description extracted from the TATIN project website [1] : *TATIN is a collaborative platform that uses a large multi-touch, multi-user interactive table, coupled with an interactive board display. These vertical and horizontal surfaces allow for complementary styles of collaboration. Users primarily interact with simple multi-touch gestures, but for more complex commands, advanced users also have the possibility of using voice commands. The voice interaction is facilitated by cognitive software agents, which can give confirmations and ask questions in the case of ambiguous commands. All the devices in the room are connected using a custom-designed multi-agent system, which manages the multi-device user interaction. In our implementation, we blend agents from two different toolkits JADE and OMAS that play complementary roles in the infrastructure.* Figure 5.1 presents a brainstorming session using the interactive table.

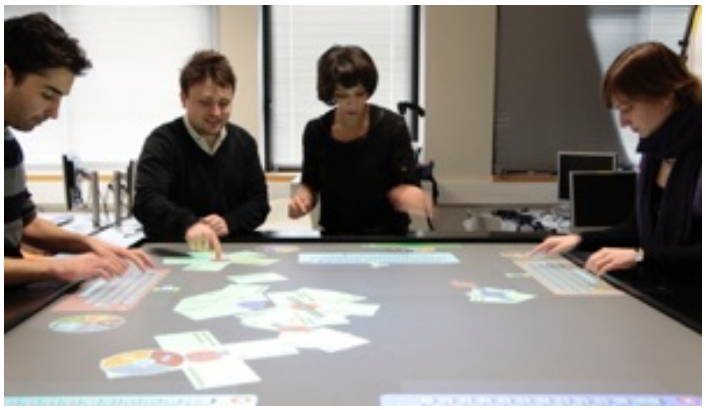


FIGURE 5.1 A brainstorming session using the interactive table of TATIN

During a collaborative session using TATIN, each user has his own personal assistant. The interaction may be multi-modal with input from a microphone and from designation on the graphics table (Barthès [10], Jones et at. [49]). The fragment below presents an example of a conversation taken from a brainstorming session when users share and organize ideas using *Post-it* notes :

1. Available at http ://www.utc.fr/tatin/TATIN/PROJECT.html

Master :	*(1) Nouveau Post-it intitulé cahier des charges.*
	(1) New Post-it entitled requirement specification.
Assistant :	*(2) Le Post-it P-23 a été crée.*
	(2) The Post-it P-23 has been created.
Master :	*[User selects the Post-it P-23 on the interactive table.]*
Master :	*(3) Supprimer ce Post-it.*
	(3) Remove this Post-it.
Table :	*[Post-it P-23 is highlighted.]*
Assistant :	*(4) Celui-là ?*
	(4) This one ?
Master :	*(5) Oui.*
	(5) Yes.
Table :	*[Selected Post-it disappears from the interactive table.]*
Assistant :	*(6) Le Post-it P-23 a été supprimé.*
	(6) The Post-it P-23 has been removed.

One of the highlights of this dialog manager is the usage of the concept of *salient features*. This technique consists in keeping a list of important objects and events, stacking them as they occur. Salient features could be used to *undo* operations and for reference resolution. An example of reference resolution can be seen in utterance (3), where the user asks for removing a *Post-it* by using the *it* pronoun. The assistant in turn, reads the salient feature stack, finds the *Post-it* and asks for confirmation before removing it from the interactive table.

The TerreGov Project

Another example of dialog using the OMAS platform comes from the European TerreGov project. This project uses ontologies, semantic web technologies, and personal assistants to provide better e-government services to citizens [14]. We have selected a conversational interface used to request direct income support [74]. The fragment below shows an example of a conversation between the user and the personal assistant using a vocal interface :

Master :	*Can you list all conditions to receive a RMI ?*
Assistant :	*Yes. The candidate must live in France, be older than 25, have income not higher than 417,88 € per month*
Master :	*OK, could you start an RMI application ?*
Assistant :	*What is your age ?*
Master :	*26*
Assistant :	*What is your monthly income ?*
Master :	*300*
Assistant :	*Where do you live ?*
Master :	*...*

The highlight of this application is the usage of a domain ontologies with minimum adaptation to be used by the dialog manager. Additional information such as synonyms and lexical information has been added to the domain ontology.

The HDSRI Project

The HDSRI project is a system developed in our laboratory to help researchers keeping track of international projects [11]. In addition to basic functions to manage the projects, the system has a dialog-based functionality that can be used for information extraction based on natural language sentences.
We show some examples of conversations taken from the system. Figure 5.2 presents a dialog session between the user *Barthès* and his assistant, which only understands expressions written in French. The statement "*Montrez-moi les projets en cours sous forme d'un camembert*" is equivalent to "*Show me a pie chart of active projects*". The system interprets the user's request by communicating with its staff agents, finding the most suitable one to attend the request. In this specific case, a *project* agent was chosen to interpret the user sentence using its domain ontology and the corresponding knowledge base to produce a proper query and gives back the result to the assistant. Then, the PA retrieves the response of the service agent and formats the result to its master using a pie chart.
The dialog system of HDSRI is not limited to text. It allows the presentation of results using a variety of components (scatter diagrams, column charts, pie charts, radial graphs, world maps and so on). For instance, Figure 5.3 presents an example of a dialog with some sort of multi-modality. First, the user asks for all active projects of the ASER research group. Then, the assistant informs that it found 3 active projects that can be displayed

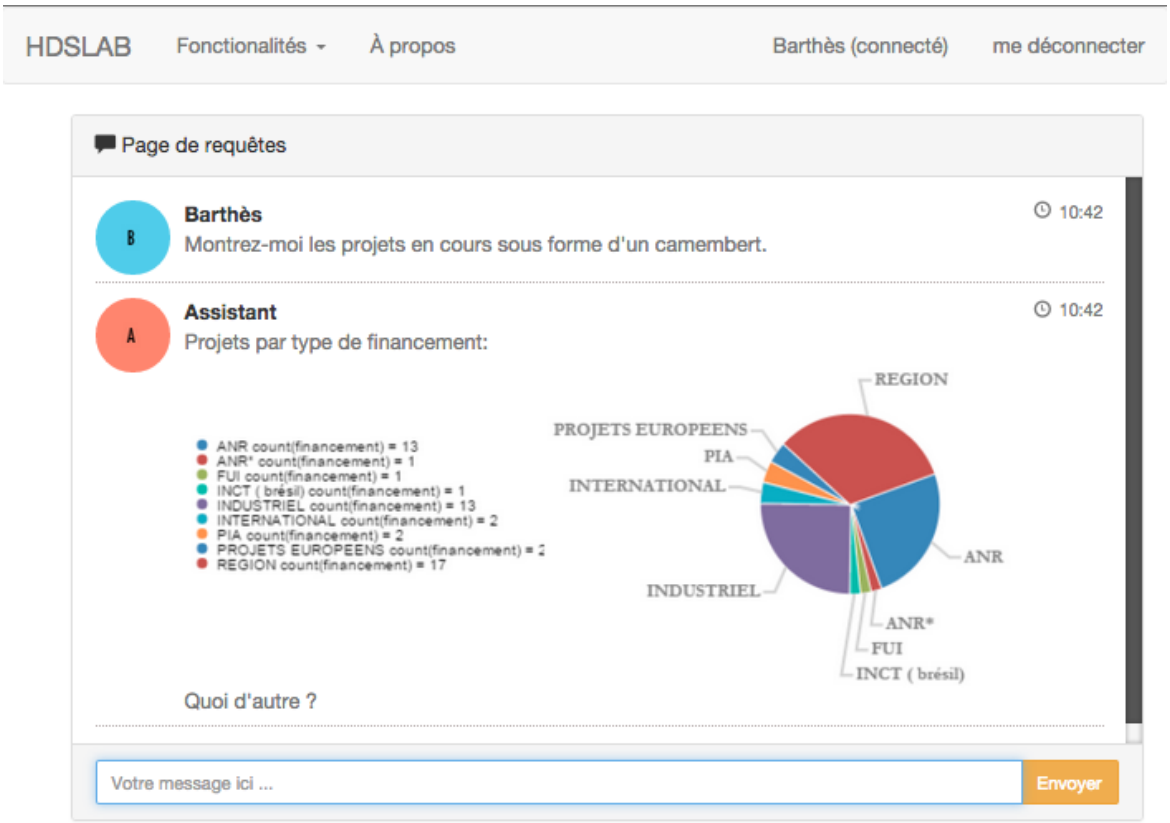


FIGURE 5.2 Asking the personal assistant for a pie chart of active projects

using three available components (table, chart or graph). Note that, even if the user can simply write his or her preference in the input (e.g. "I prefer a graph"), it is also possible to select from one of the alternatives provided by the assistant in the dialog window. After the user's choice, the system presents an interactive graph with active projects, where triangles are projects, circles are persons and squares are research groups. The component provides an interactive radial graph visualization, allowing users to navigate, search and explore the resulting graph.

5.3 Dialog approach

As presented in Chapter 3, there are several possible approaches to design and build dialog managers. The OMAS dialog approach can be seen as a hybrid solution combining the following techniques :

— *Information states* : dialogs are modeled as a conversation graph.

— *Script-based* : each conversation state contains a set of rules containing patterns influencing the whole dialog.

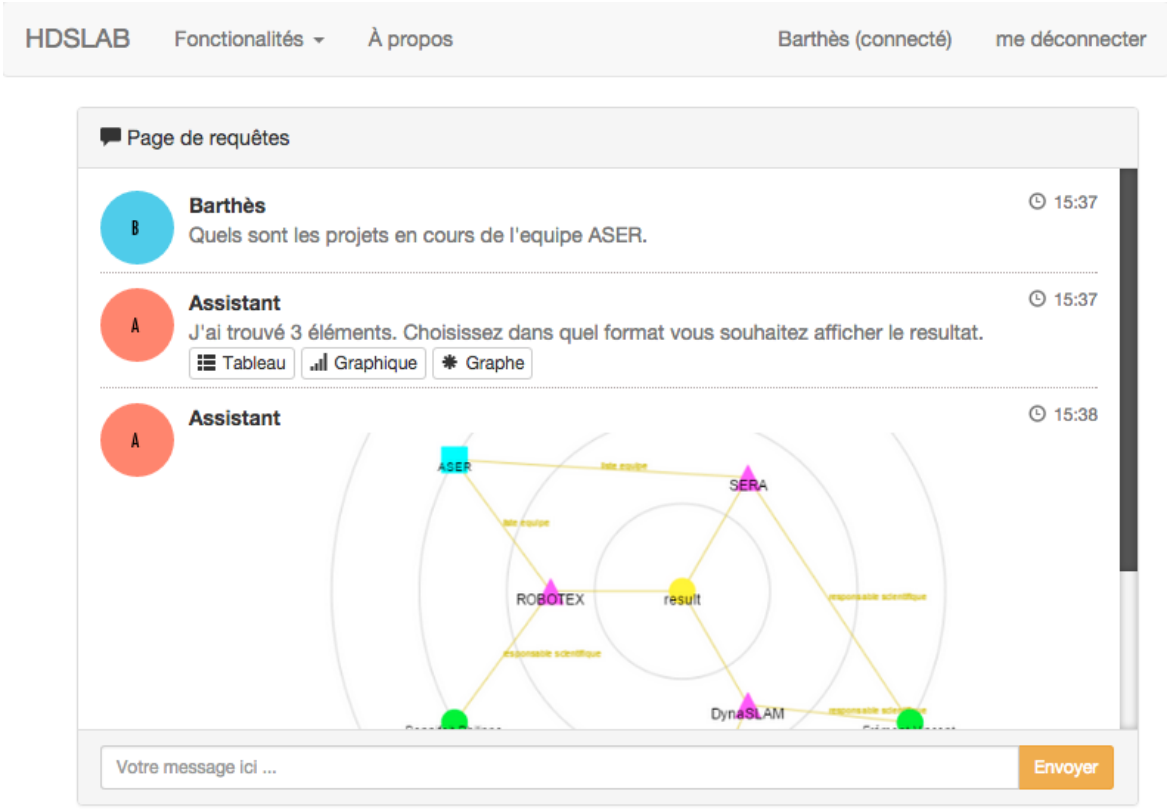


FIGURE 5.3 Asking the personal assistant for active projects without specifying the output format

— *Frame-based* : a task and its elements are modeled as frames.
— *Task-oriented dialog* : dialogs are reusable components, linked with a library of tasks.

This approach requires an environment restricted to a professional context or a focused context in which the number of possible actions is limited. Natural language can be used to find and execute a task. The dialog is conducted with a minimum of recognized phrases, and there is no need for a grammatically correct input.

5.3.1 Dialog mechanism

OMAS provides a task-oriented dialog manager. It means that the user must have a goal in mind when requesting something to the personal assistant. During the dialog usage, users might want to retrieve something from an information system or perform one or more tasks. The problem is thus simplified and can be tackled by a two-step approach : (i) recognizing what action is requested ; and (ii) extracting the necessary parameters from a

noisy input during the dialog [11]. The first step is solved by using a very general *top-level dialog* that listens to the user request and tries to find available tasks to execute, based on a *library of tasks*. The second step can be solved by triggering the dialog associated with the selected task or letting staff agents deal with the raw input. The sequence of actions performed by the top-level dialog can be summarized as follows :

1. The user asks something to the PA using the keyboard or a vocal interface (e.g., *I want to report a stolen card*). The input is then segmented into a set of words, and the performative is identified.
2. The PA tries to infer the targeted task from some of the words used as indexes. If a single task is found, the corresponding task sub-dialog is started. If more than a task is found, the one with the highest valuation is assumed to be the one that is intended. If no task is found, then a failure is declared.
3. We then go back to the beginning of the dialog.

Example 5.3.1. (An example of dialog)

1	*Master :*	*I want a credit grant.*
2	*Assistant :*	*[It identifies that the performative is a request.]*
3	*Assistant :*	*[It selects one task.]*
4	*Assistant :*	*[It starts the corresponding sub-dialog used for loan request.]*
5	*Assistant :*	*What is the amount ?*
6	*Master :*	*3000 EUR.*
		...
7	*Assistant :*	*Congratulations, you got your credit.*
8	*Assistant :*	*[It returns to the top-level dialog.]*
9	*Assistant :*	*Would you like to do something else ?*

The example above shows a fragment of the communication between a bank customer and his or her personal assistant. The user starts by asking for a credit loan in (1). Then, the personal assistant identifies that the message type is a request in (2) (could also be an assertion, a question or a confirmation). It finds a compatible task from the library of tasks (3) and starts a very specific dialog to attend the request in (4). The dialog is conducted until step (7). Note that in (8), the control is given back to the top-level dialog, so in (9) the assistant is free to attend other types of requests. This example shows a conversation with states that are causally linked and are hierarchically organized. Now, we analyze how they are modeled.

5.3.2 Modeling dialogs using conversation graphs

The dialog representation in OMAS has its roots in the approach developed by Chan Lam [19] that models a *conversation graph*. A conversation graph is an oriented graph modeling a finite-state machine. Nodes are called *conversation states* and arcs are called *transitions*. Each conversation state contains a set of rules that apply to a fact base containing information obtained from the master's input or results from the analysis of previous states. The fact base is similar to the fact base of a rule-based system. At a given node applying the rules triggers either a transition to a new state or an action (e.g. sending a message to other staff agents). A complete dialog is represented by a path or a succession of conversation states.

Example 5.3.2.

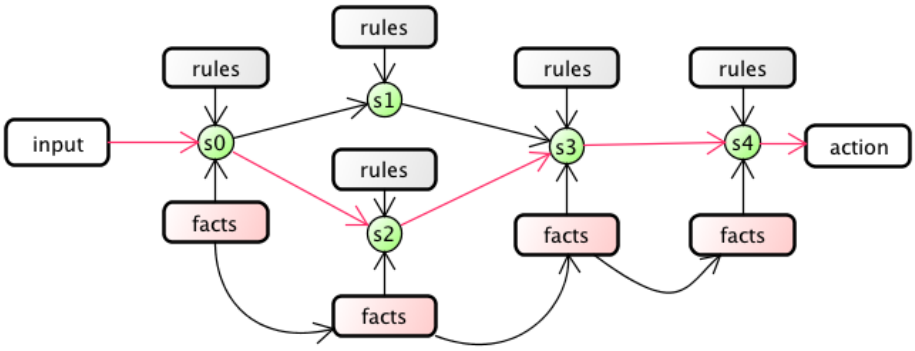


FIGURE 5.4 Dialog mechanism using information states

Figure 5.4 presents a conversation graph with five states. Each state has its own set of rules and is linked with the facts base, which is enriched as the graph is traversed. A complete dialog is represented by the sequence $\{S_0, S_2, S_3, S_4\}$.

Note that the set of states is not physically implemented as a graph structure. Indeed, there are no explicit arcs between two states. Transitions are computed dynamically. Thus, a set of states represents an ensemble of virtual dialog graphs.
The state object has two important methods that are called when the state is triggered :

- =*execute* is performed when entering a state after a transition has occurred. It usually produces a statement or prints a question to the user. Note that a question blocks the state until an answer is given. In other words, the state enters in *waiting mode*.
- =*resume* is executed when the user provides some information or when some information is already available in the facts base. The dialog manager then analyzes

the text content, updates the conversation object and computes a transition to another state.

The dialog has a conversation object. It keeps track of where we are in the dialog (which state is the current state), of the I/O channels, of the level of recursion in the sub-dialog, and contains the facts base accumulating the information during graph traversal.

The top-level dialog

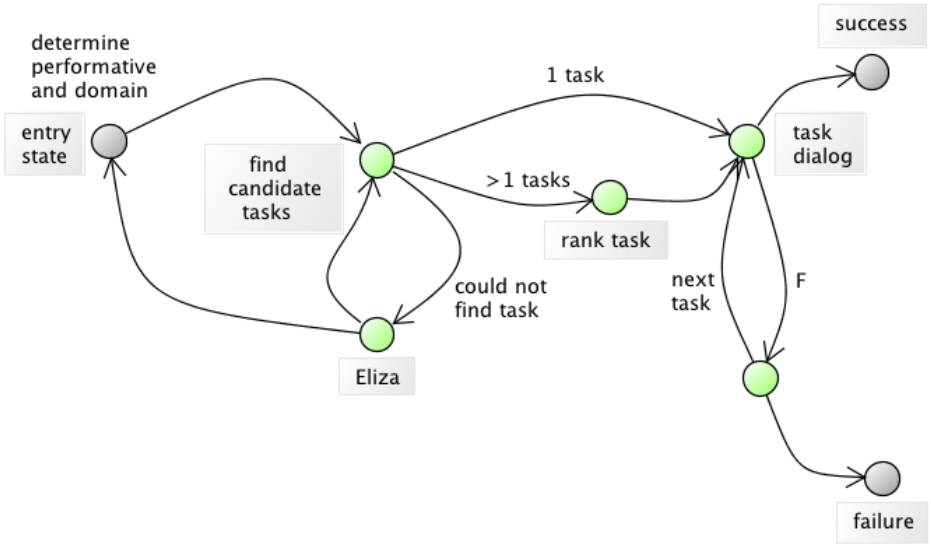


FIGURE 5.5 Conversation graph of the top-level dialog

The top-level dialog is an interesting example to explain how the conversation graph mechanism works. During its usage, the user interacts with the assistant to find an appropriate task to execute. Figure 5.5 presents the top-level dialog modeled as a conversation graph. The first state, called *entry state* is used to collect the user input and analyze the message type (e.g. a request or an assertion). When its =*execute* method is triggered, the dialog is prepared to be used for the first time (i.e. initialization of the conversation object, sending a welcome message, and waiting for input). The =*resume* method takes place as soon as the user says something. The dialog manager then analyzes the message type, stores both the performative and the raw input into the facts base. Finally, the control is transferred to the *find candidates task*. If a task cannot be determined, the PA transfer the control to *Eliza* to analyze the input and either produces an adequate answer or tells the master that it did not have enough information to process the input. The *Eliza* sub-dialog is a special simple dialog with predefined questions and answers inspired by the work of Weizenbaum in [109]. When one or more tasks can be

determined, the task with the highest score is launched by triggering a sub-dialog attached to this task. The conversation state called *task dialog* waits for the execution of the sub-dialog. At the end of the execution of the sub-dialog, the dialog manager tests if the execution was either a success or a failure, showing an appropriate message to the user.

5.3.3 The library of tasks

The set of tasks (or possible actions) is modeled and gathered into a library of tasks. These tasks are defined as individuals of the concept of *task*. Table 5.1 shows how a task for obtaining a credit grant is defined, using a *deftask* macro that produces a task individual. The *:indexes* parameter specifies a list of linguistic cues. Each phrase (cue) has a weight between -1 and 1. A value of -1 should be used when the task may not be selected if the input contains the corresponding phrase. A value of 1 should be used when the task certainly applies if the input contains the corresponding phrase.

TABLE 5.1 Defining a task

(*deftask*	“credit request”
	:doc “Task for asking a bank for a loan”
	:performative :request
	:dialog _credit-request-conversation
	:indexes (“credit” .5 “loan” .5 "grant" .3))

The *:performative* parameter acts as a filter specifying whether the task applies to a request or to an assertion. The *:dialog* parameter points to the representation of a conversation graph when interacting via the PA interface.

5.3.4 The selection mechanism in details

One of the states present in the top-level dialog is the *find candidate tasks* as can be seen in Figure 5.5. At this state, the dialog manager does not wait for the user input and directly analyzes the facts base to find an appropriate task. The sequence of actions is done as follows :

1. The user requests something to the PA (e.g., “I want a credit grant of 3000 €without fees").
2. The phrase is segmented into a set of words, removing spaces, words separators, and empty words.

3. For each task in the library, the PA checks the sentence for phrases specified in the index pattern describing the task, and computes a score by using a MYCIN-like formula (i.e. if 2 cues a and b are present, the combined score is computed by the formula $a+b-a\times b$) ; for example using the user input of step one and the cues for the task described in Table 5.1, the two words "credit" and "grant" will give a score of $(.5+.3-.5\times.3)=.65$ for the task. This is a simple but efficient approach since each match of a sub-expression reinforces the overall match score.
4. Tasks are then ordered by decreasing scores.
5. The task with the highest score is selected, and all tasks with a score under a specific threshold are discarded.

Note that the choice of the indices and the choice of the weights is an empirical task. Selecting the right linguistic cues and its corresponding weight should be the result of experiments using, for example, training sets of Wizard-of-Oz interactions.

5.3.5 Building a conversation graph from a business process model

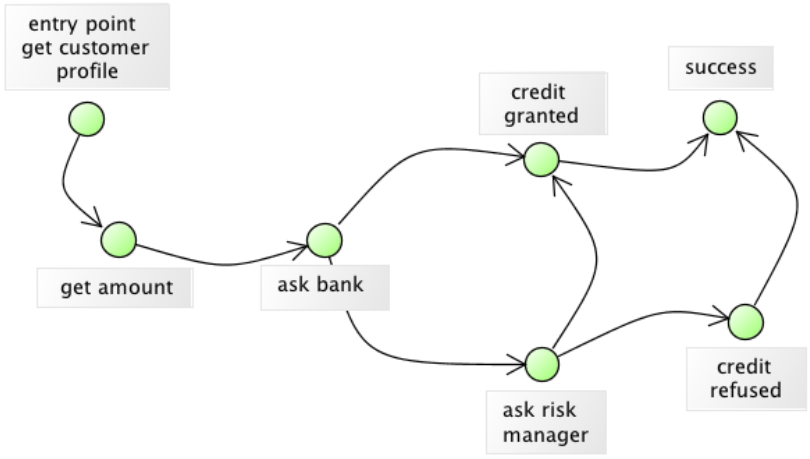


FIGURE 5.6 The credit grant conversation graph

We have built a conversation graph based on the business process model of our illustrative example presented in Chapter 3. The corresponding dialog is used for granting credit to banking customers, as can be seen in Figure 5.6. The first step is used to retrieve the customer profile. When activated, a request message is sent to the bank agent, that in turn returns an individual representing the risk profile. The individual is then stored into the facts base for further usage. Then, the assistant asks its master for the credit amount. As soon as the user answers with a valid amount, the assistant sends a message to the bank

agent to request the grant. If the credit is granted, the assistant informs the user, and the sub-dialog is finished. If the credit is not automatically granted, a message requesting the manual evaluation is sent to the personal assistant of some risk manager. As soon as the risk manager's assistant answers the request, the credit is granted or refused depending on the risk manager's response. Figure 5.7a presents the interface of a session with the customer, and Figure 5.7b presents the interface of a session with the risk manager. During a typical credit evaluation session, this environment has, at least, three agents : A personal assistant for the customer, an agent that has the skills to execute bank services, and a personal assistant for the risk manager. The bank agent is responsible for the execution of various services of information systems (e.g. querying and updating relational databases). As customers do not know risk managers *a priori* (indeed this is not recommended for credit evaluation), a resource allocation mechanism was developed. We have used the *contract-net* protocol with a simple winning criterion : When the PA of the customer needs a manual assessment, it sends a broadcast to the coterie, requesting a risk manager. The first risk manager that answers the request is the one that will assess the request. This approach is similar to the *push pattern* used for resource allocation in business processes (see Russel [83]).

5.3.6 Discussion

This section presented an overview of the OMAS dialog manager. Based on the architecture definition and the experience of developing a dialog from an existing business process definition, we focus now our discussion on the following issues :

Task selection

The task selection is a critical element of our approach since business process users have different ways of requesting a service. The reasons for this variation is rather complex and out of the scope of this work. They may come from cultural aspects, from the level of experience or personal traits. But, independently of the cause, we observe three distinct types of requests made by users :

— *Request based on the description of the current state* : Inexperienced users may not be capable of describing what they want, but rather they can easily describe their current state. We thus can assume that the user does not know what the system is capable of offering to solve his problem. For example, the user may write to the assistant "I have bought a new car". The assistant could search for tasks that are

compatible with this description, offering, for example, a service to “request a car insurance” and a service to “buy an extended warranty”.

— *Request based on one or more intermediate outcomes that may be produced* : Once users are more comfortable with the whole process, they might formulate a request using “how” elements instead of “what” elements. For instance, a user may ask the assistant : “I want to make a quotation for a new computer”. In fact, the quotation is part of the whole procurement process for buying the computer. Thus, the system may suggest starting a procurement process, but warn him that the process does much more than the user is requesting.

— *Request based on one or more final outcomes* : Users know exactly what they want. For instance, “I want to renew my driver’s license”.

The approach used for task selection uses keywords for interpreting what the user wants to do. There is no grammar for interpreting the user input. As a consequence, requests must not be too complex, and neither negations nor disjunctions are allowed. More detailed expressions like “I want a credit request without fees for next Monday” could present some problems to the dialog manager. The task selection is capable of recognizing a task based on the keywords but may lead the user to an inappropriate task.

Resource allocation

The role of *risk manager* of our illustrative example shows that the distribution and coordination of work amongst the group of human resources associated with a process is a critical task. Dialogs intended to help users during the enactment of business processes must be prepared to work in conjunction with the BPM enactment system to receive notifications of task allocations of notifications of task offers. For the sake of simplicity, we have adopted a solution based on the contract-net protocol for the example shown above. Of course, the chosen strategy has some potential problems, and a number of questions arise immediately :

— What if there are no risk managers to attend the request ?

— What if risk managers are too busy to answer the request ?

— What if a manager wants to delegate the task to another risk manager ?

— What if a manager wishes to suspend a task momentarily ?

Invoking other processes during the execution of a task

The hierarchical characteristic of the conversation graph does not allow the master to deviate from the predefined plan and execute a different sub-dialog during the

conversation, except when manually programmed for that [63, p. 25]. To clarify, we present a fragment of a conversation using our implemented example :

1 *Master :* *I want a credit grant.*
2 *Assistant :* *What is the amount ?*
3 *Master :* *Show my balance.*
4 *Assistant :* *Could you please inform a numeric value ?*
5 *Assistant :* *What is the amount ?*
6 *Master :* *4500 €.*
7 *Assistant :* *Congratulations, you got your credit.*
8 *Assistant :* *[It returns back to the top-level dialog.]*
9 *Assistant :* *Would you like to do something else ?*

In (3) the customer is not sure about how much money he or she needs to borrow. Thus, the user tried to check his or her balance without success.

Control and data flow

The credit grant example has shown a great similarity between the conversation graph and a typical business process model based on Petri nets (BPMN [73], EPC). Both provide specifications that are used by their engines that in turn, control the sequence of events, the flow of data, and involve actors when needed. For the implementation perspective, we believe that the development of dialogs for the enactment of a business process requires a great effort of development, since all process rules must be also present in the conversation graph. Thus, we believe that the effort to manually port a business process model to a conversation graph is a good reason for trying to find more feasible alternatives.

5.4 PA4Biz : A personal assistant for the enactment of business processes

In this section, we present our conversational interface called PA4Biz (Personal Assistants for Business Processes). The resulting architecture provides a new baseline if our dialog manager, allowing users to select and trigger, as well as manage activities allocated from a BPM enactment system. During this section, we describe the overall architecture as well as our personal assistant architecture in details.

5.4.1 Overall architecture

This section is dedicated to present a big picture of the proposed architecture. Four types of cognitive agents have been designed : a group of personal assistants and BPM representatives, the BPM system agent and the authentication agent.
We start by explaining the architecture from the agent in the lower part of the diagram : the personal assistant. Each user has his own personal assistant that is committed to help its master during the enactment of business processes. More specifically, it helps finding the most suitable process based on their conversation history. It also manages the flow of incoming tasks that arrives asynchronously (both by allocation or by offer), as well as managing the fulfillment of input parameters to perform tasks.
As it happens with the PA, each user has his own *BPM representative*. A *representative* also serves the user, but behind the scenes. It is responsible for keeping an active session with the BPM system agent and executing all process-related tasks (e.g. search for processes, start a process, execute tasks and so on). As can be seen in the diagram, the representative does not have direct access to the user, but only to the PA. Each time the master wants to trigger a process, he or she asks the PA, which in turn delegates the request to its *representative*. In the same way, when the BPM system agent wants to notify an event that involves the master, it sends a notification to the *representative*, so the *representative* forwards the message to the PA (e.g. the BPM system wants to notify that a process has just finished).
The lifecycle for both personal assistants and *representatives* is relatively short, more specifically during their master's session. It is the *authentication agent* who is responsible for the authentication of the master into the BPM system and the creation of the duo (i.e. the PA and the *representative*). As the BPM system already have a persistency model for the user session, the authentication agent recreates the *representative* session each time the master logs in, allowing both agents receiving past notifications in case of a failure. For this reason, both the PA and the representative agent operate in memory, and they do not have a persistence model.
Finally, the BPM system agent acts on behalf of a BPM system. Of course, the scope is very restricted, limited to some basic functionalities related to the lifecycle of process instances (e.g. starting processes, performing tasks, creating sessions, feeding the BPM system with data). The novelty here is the skill to provide a comprehensive representation of business process capabilities and semantic search functionalities as well as a query mechanism. This agent implements these two capabilities that have been detailed in Chapter 4.

Note that a separation of concerns was used during the design of the architecture, avoiding strong dependencies between the PA and the BPM system. There is a thin layer in our architecture. For example, the PA does not have access to the BPM ontology. Only the representative and the BPM system have access to it. In the same way, only the personal assistant have access to the dialog ontology. However, all agents have access to the domain ontology (and the action taxonomy).
This clear separation could be helpful to the evolution of our research on BPM systems and personal assistants.

Personal assistant architecture

A great part of this chapter is dedicated to the personal assistant. Thus, it is important to give an overview of the PA's architecture before detailing each component. Figure 5.9 presents an illustration containing all elements of the PA, grouped by functionality.
The communication between the PA and its master is made through a written dialog interface mixed with graphical controls such as buttons, graphics and input controls that are generated by the dialog manager. The OMAS platform [8], developed at our laboratory allows two types of interface (both desktop and web) with minimum effort of adaptation to port from one environment to another.
As mentioned before, the scope of our work is not strongly related to natural language processing. Thus, a set of techniques for shallow semantic parsing is used. The package of NL modules contains the following components :

- A syntactic annotation function that is used to identify syntactic elements of a sentence, identify named entities (e.g. names, locations, people) and the dialog act.
- A semantic annotation function that captures and annotates the sentence with domain ontology elements.
- An information extraction function that contains a set of handcrafted linguistic rules used to extract information from a sentence and finally update the dialog context memory.

A set of predefined dialogs have been created for the business process enactment problem : A top-level dialog, a sub-dialog that takes notes during the conversation, a sub-dialog used for selecting and triggering business processes, as well as performing tasks. Note that the dialog used to perform tasks is asynchronously added during the conversation, which augments the level of complexity of the application.
The group of BPM modules contains two components : The first is the *BPM event handler*, implemented as a set of skills. It is responsible for notifying the dialog manager when a process event arrives (e.g. a new task, a process that have just finished and so on). The

second contains a set of functions used for querying the BPM system agent (e.g. check pending tasks, check pending processes).
The PA has access to two ontologies and a taxonomy : the domain ontology, the dialog ontology, and the action taxonomy are used to interpret user sentences and manage the dialog memory. The master group contains basic functions to retrieve the user profile (e.g. name, organization, role).
Now that a brief overview of the architecture was provided, we start by giving more details about the top-level dialog : the starting point for all users.

5.4.2 The top-level dialog

We have proposed a top-level conversation graph for the problem of business process selection. The top-level dialog is rather simple, having only four states as can be seen in Figure 5.10. Its main objective is to interpret the master's sentence and make a decision if : (i) the user is only describing something, the PA will only take notes ; (ii) if the user is asking or requesting something, the PA will take notes and propose the execution of one or more business processes. For instance, if the user says : "I am 26 years old, married, and my salary is between 4,000 and 5,000 €", the PA identifies that this is an assertion and it will extract information and store it in the dialog context memory for further usage. If the same user continues : "What do you recommend ?", the PA will take notes and also search for business processes based on the information provided since the last sentence is a question. The PA could suggest an investment funds application or a treasury notes application.
The dialog starts in the *entry state* that is used to prepare the conversation object and display an opening message. Then the user sentence is retrieved in *get input* state, which transfer the control to the *take notes* sub-dialog (the plus sign denotes a sub-dialog). As it name suggests, this sub-dialog is responsible for interpreting the user's sentence, extract information and store it for further usage. If the dialog act is a directive or a question, the control is transferred to the *business process selection and trigger* sub-dialog that tries to find the most suitable process or set of processes for its master.

5.4.3 Taking notes

The *take notes* sub-dialog is a generic sub-dialog that is used by both the *top-level* dialog and the *task execution* sub-dialog. It makes use of all shallow natural language modules presented in the PA architecture as can be seen in Figure 5.11.

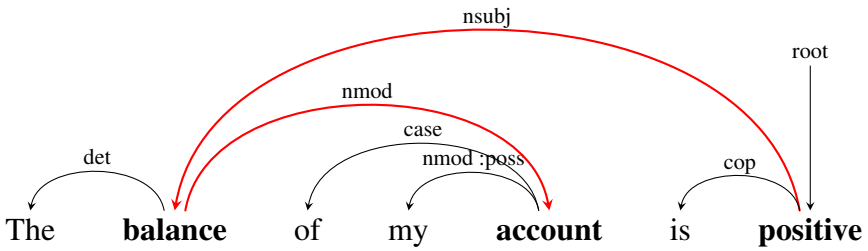
It starts by making the *syntactic annotation* of the sentence, retrieving, among other information, the dialog act of the sentence. Next, the semantic annotation uses the domain ontology, the action taxonomy and the dialog ontology as a source to annotate the sentence for further information extraction. Note that this sub-dialog is only prepared to handle assertive, directive and interrogative acts. All other types of dialog acts are transferred to *Eliza*, a safety net used to avoid letting user without any answer. As can be seen in Figure 5.11, both the semantic annotation and the information extraction are invoked as sub-dialogs. Both sub-dialogs allow the user to give up at any time, aborting the dialog. We continue by explaining the syntactic annotation sub-dialog.

5.4.4 Syntactic annotation

The objective of this process is threefold : (i) Capture grammatical relations of the sentence. These relations will be used for further processing. It is desirable that the whole approach could be easily ported to other languages such as French and Portuguese for further evolution of the framework, although our preliminary effort is dedicated to the English language. (ii) Annotate the sentence with named entities (people, location, and organizations) as well as temporal and numeric content. (iii) Identify the dialog act for further processing. Figure 5.12 presents a diagram with the steps taken during the syntactic annotation, producing additional information over time.

Capturing grammatical relations

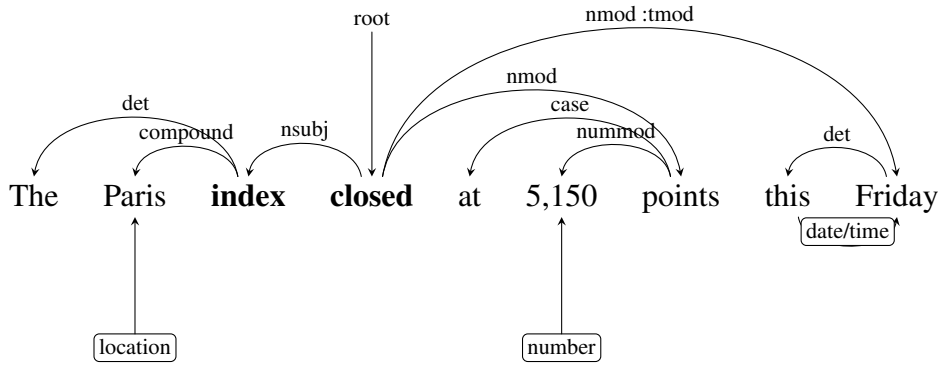
We have chosen a typed dependency approach developed by the Stanford Natural Language Processing Group, called *Universal Dependency Representation* [30]. The resulting structure provides a description of grammatical relationships in a sentence as typed dependency relations. The outcome is a set of triples of a relation between pairs of words. For instance, the sentence “The balance of my account is positive” has a triple stating that *positive* refers to the *balance*. The illustration below shows the sentence with labeled arcs. The outgoing word is the *governor* of the dependency and the incoming word is the *dependent* word :



The outcome of this process is a tree, which is generated from the set of dependency triples. Our information extraction procedure will use this tree based on handcrafted roles, described in details in the next section.

Annotating the sentence with named entities

Needless to say, the interpretation of sentences is a notoriously difficult endeavor. Noisy sentences and ambiguities may appear during both the syntactic or semantic analysis. Thus, to help reduce ambiguities, we have added a step during the syntactic annotation, the objective of which is to recognize entities, temporal expressions, and number expressions. Such expressions can be seen as unique identifiers of entities (organizations, persons, and locations), times (date, time) and quantities (monetary values, numbers, and percentages). As argued by Kiryakov et al. [54], *named entity* is an alternative to the traditional notion of token (word stems) for classical information retrieval systems.
We have chosen Stanford NER as our named entity recognizer. The software provides a general implementation of (arbitrary order) linear chain Conditional Random Field (CRF) sequence models (see Finkel et al. in [38]). We provide more details about the usage of Stanford NER in Chapter 6, used to describe the implementation and tests. Models were trained on a mixture of CoNLL, MUC-6, MUC-7 [23] and ACE named entity corpora, and, as a result, the models are relatively robust across domains.
The example below presents a sentence with its corresponding grammatical relations. The analysis annotated the sentence with three labels : The word *Paris* was recognized as *location*, The word *5,150* was identified as a *number* and the fragment *this Friday* was identified as date/time.



This complementary information is particularly useful for the semantic annotation and the information extraction. It is to be used to resolve ambiguities and reduce the space search where unclassified information is found. For instance, suppose that both the words *index* and *closed* have been identified as attributes of a concept called *market history*. The value *Paris* is assigned to the attribute *index* of *market history*, and the value of *5,150* is assigned to *points* since both attributes are compatible with the extracted information. However, the fragment *this Friday* does not have yet an "owner", and it is "floating" on the sentence. There is no evident approach to properly link this information to an attribute. The *date/time* flag could be a clue to discover possible attributes present in the *market history* concept that could own this information. The algorithm might assume that *this Friday* refers to the attribute *close date* that belongs to the *market history* or asks for confirmation if it is not sure. A section dedicated to information extraction gives more details on how these annotations will be used for solving this type of problem.
From now on, the outcome of this analysis is a set of labels assigned to one or more words present in the sentence.

Dialog act identification

We have discussed the importance of dialog acts for dialog managers in Chapter 3. Dialog acts are strongly related to the turn-taking phenomenon, which means that both the personal assistant and users have a dialog act associated with each sentence that they produce. Now we are interested in the user (the master) dialog acts.
Table 5.2 presents the adopted taxonomy of dialog acts used by the human actor. For practical reasons a dialog act has two levels : the first level is more general and follows the traditional taxonomy of acts. The second level, more specific, reduces the semantic scope, useful for filtering purposes during the dialog.
We start by discussing directive acts. They are attempts by the user to get the assistant to do something, like start a task or describe a situation. A *directive/request* act is an order, a

TABLE 5.2 Dialog acts and some examples of sentences

Dialog Act		**Examples**
General	**Specific**	
directive	request	Please, create an insurance product.
		I want to evaluate the risk management plan.
	abort	Please stop !
assertive	inform	My car is broken, parked near the Chateau de Compiègne.
		The manager of the Bercy branch is Clement Martin.
	inform-fragment	The responsible of the claims department.
	answer-affirmative	Yeah.
	answer-rejection	No no.
interrogative	what	What is the e-mail of Maxime Bertrand ?
	who	Who is the manager of the Compliance department ?
	where	Where the meeting will be held ?
	when	When will the project finish ?
	conditional	Is the project managed by John Silver ?

command with an action verb that sometimes have the desired state associated with. The first example “Please, create an insurance project” indicates the desired state at the end of the action. The second example “I want to evaluate the risk management plan” is more subtle. Even different in style, both sentences are directives and are treated without distinction. Another type of directive act is the *directive/abort* that contains negative statements used to abort the current task. We use this act to detect when the user wants to abort the execution of a sub-dialog.

An assertive act is an attempt to committing the speaker to the truth of a proposition. Here, the act *assertive/inform* indicates that the user is describing something to the assistant. The interpretation of this type of dialog act is highly context-dependent. For instance, the sentence “My car is broken and parked near the Chateau de Compiègne” is an example of an *assertive/inform* act of a user that is having some problems with his or her car and is contacting the call center of an insurance company. He or she preferred to describe the problem instead of the solution (e.g. request a taxi or technical assistance to repair the vehicle). When the user freely describes something without having a previous question posed by the assistant, it means that the assistant should simply take notes and find compatible services when things get clearer.

Other specific types of assertions can appear as a result of a previous question made by the assistant. The *assertive/inform-fragment* is a fragment, also known in the spoken dialog domain as a non-sentential utterance. The sentence “The responsible of claims department” could be an answer to the question “Who is the contact in case of problems in the risk plan ?” made by the personal assistant. Sometimes users might be more verbose when they

answer questions, and the resulting act is not a fragment but a complete sentence. The sentence “The manager of the Bercy branch is Clement Martin” is an assertive/inform act, but also the answer to the question "Who is the manager of the Bercy branch ?". Thus, when assistants are expecting assertions, a useful filter could be *assertive/inform**, getting both complete assertions and fragments. Finally, the acts *assertive/answer-affirmative* and *assertive/answer-rejection* are self-explanatory and can be used for confirmation purposes or grounding.
Interrogative acts are divided into specific types of questions. The *interrogative/what* is more general and their questions are related to the state of things. The *interrogative/who* reduces the scope of questions to persons, organizations, and roles. The *interrogative/where* normally asks something about locations and places. When the expected outcome is a temporal information, the act is classified as *interrogative/when*. Finally, *interrogative/conditional* is used when the user wants to test the truth of some proposition.

Wrap up : A unified format for the syntactic annotation

Figure 5.13 presents an example of the final outcome of the syntax annotation task. It has three distinct parts, specifying the dialog act, the dependency graph and the set of labels for entities, number and date expressions. For the phrase “The Paris index closed at 5,150 points this Friday”, the task identified an *assertion/inform* dialog act and three tags : a location (Paris), a number (5150) and a temporal expression (Sep/9/2015 at morning). This structure is the input for the remaining processes used for semantic annotation and information extraction.

5.4.5 Semantic annotation

The semantic annotation is used by both the *business process selection and triggering* sub-dialog and the *task execution* sub-dialog. They have different objectives : The former is responsible for finding the most suitable business process based on one or more sentences and trigger this process in the BPM system. The latter is responsible for the execution of a specific task.
What they have in common is that both must deal with the problem of the interpretation of the user’s sentence : From the business process selection perspective, the objective is to understand what the user wants to do. From the task execution perspective, the objective is to extract input parameters. Both activities must take into account the information available in the dialog context memory. This is indeed a very difficult task and requires full

understanding of natural language for open domains. This is clearly out of the scope of this thesis. A more modest target is the detection of instances of known concepts and the intention of the user (e.g. if he or she is requesting, asserting or confirming something) in a controlled domain.
Our approach to semantic annotation and information extraction is based on the notion that the meaning of sentences can be inferred from using domain ontologies. Hopefully, the information extraction using ontologies is a very active research field and is called ontology-based information extraction (OBIE). Our scope is limited to the techniques used for ontology population (i.e. the identification of key terms in the text and then relating them to concepts of a domain ontology [69]).
The semantic annotation can be summarized in four distinct steps as can be seen in Figure 5.14. The first step annotates the sentence with ontology concepts. More specifically, it identifies words or group of words with the potential to be concepts, attributes or relations of the domain ontology. The example present in the figure shows the sentence "Please, create a project, it is sponsored by the AMF [2]". The task identified three tags in the sentence : one concept and two possible attributes. The second step annotates the sentence with references to individuals when the sentence has missing clauses (i.e. due to the ellipsis or anaphora phenomenon). The task tries to find individuals present in the dialog context memory, and if found, it annotates them for further analysis. The pronoun "it" in the sentence 'Please, create a project, it is sponsored by the AMF" has been annotated with a new individual of project. The third step annotates the sentence with an action if the dialog act is directive and has one or more action verbs. For example, the verb "create" has been annotated as an action. Finally, the fourth step checks if the sentence has overlapping tags. In the case of an overlap, the step evaluates the overlapping of tags and finds the most suitable combination.
Domain ontologies are critical elements for the semantic annotation. Thus, we discuss the role of domain ontologies in next section.

The role of domain ontologies for the semantic interpretation

As stated by Wimalasuriya and Dou in [111], OBIE systems use domain ontologies as a "guide" for the information extraction process. Ontologies provide formal and explicit specifications of conceptualizations of a given domain. We argue that this approach is suited to applications in which both the vocabulary and the scope are well known. Taking into account that organizations already have several domain ontologies implemented in

2. *Autorité des Marchés Financiers* is the French Financial Market Authority

their information systems, a question arises immediately : What is the necessary degree of adaptation of a domain ontology, so that it can be used for dialog and IE purposes ? Eriksson in [39] argues that the domain ontology used for dialog systems must incorporate the user's view of the world or part of the world, in terms of the types of entities that can be included and how they can be organized. Indeed, ontologies provide a common vocabulary that can be used to state facts and to formulate questions about the domain. She suggests a number of techniques to adapt the domain ontology to the user perspective. For example, multiple inheritances can be a way to deal with the requirement that the ontology used by a dialogue system should reflect both the users' and systems' view of the domain. It is a modification that does not globally affects the semantics of the ontology. Dzikovska et al. in [35] have a different view : Domain ontologies must remain untouched. They present a method for customizing a broad-coverage parser to different domains by maintaining an additional ontology with linguistic elements, and defining mappings between the linguistic one and the domain one.
Different modeling strategies are also proposed from the field of information extraction. For Wimalasuriya and Dou [111] who investigate OBIE techniques, this is an open question, and there is no consensus to decide if IE components such as linguistic rules should affect the domain ontology or not. They argue that, for example, it is hard to defend the idea that IE components such as linguistic rules lie in the domain ontology. Moreover, IE elements are prone to errors (they are not 100% accurate). Conversely several authors have argued that information extractors should be considered a part of an ontology when linguistic rules are used as the information extraction technique (e.g. Yildiz and Miksch [115], Maedche et al. [65], and Embley [36]).
From the dialog perspective, we follow the vision of Eriksson, who states that the domain ontology must reflect the user view of the domain. Another argument to use and align the domain ontology to the user's view of world is pointed by Kietz et al. [51] and Maynard et al. [70]. They state that an ontology-based information extraction can also be used to evaluate the quality of this ontology. If a given domain ontology can be successfully used by an OBIE system to extract the semantic contents from a set of documents related to that domain, it can be deduced that the ontology is a satisfactory representation of the domain. Furthermore, the weaknesses of the ontology can be identified by analyzing the types of semantic content it has failed to extract. From the IE perspective, our approach separates domain ontology elements from IE components. Our linguistic rules (extractors) are outside the domain ontology.
To give some examples, we use a domain ontology fragment of our illustrative example of credit grant. Figure 5.15 presents the diagram. We intentionally reduced the ontology to a

bare minimum, extracting only some elements for illustrative purposes (rectangles are concepts, white circles are attributes, and gray circles are synonyms).

Concepts and their relations Concepts and their relations (binaries) are the basic elements that constitute the ontology. The use of hyponymy and hyperonymy to link concepts (is-a relation) and meronymy to describe dependency (has-a) may help interpret incomplete or unexpected statements. For instance, the user demands to the PA : *List the balance of profiles*. According to the ontology in Figure 5.15, listing the balance of risk profiles could be easier to interpret because *balance* is an attribute of *risk profile* instead of *profile*. However, thanks to the hyperonymy relation between *profile* and *risk profile*, a formal representation would be obtained as well.

Definition of attributes To describe each attribute of a concept, one should define its type, a set of synonyms, its cardinality, and type. Type restrictions are useful for information extraction and asking a clarifying question to users. They should be one of the following :

- *string* : a typical sequence of characters with the following possible specializations :
  - *people* : for describing people.
  - *organization* : for describing organizations.
  - *location* : for describing geographic information such as cities and countries.
- *date/time* : a typical representation of temporal data
- *number* : a typical representation of numeric data with the following possible specializations :
  - *percentage*
  - *monetary value*
- *Boolean* : A Boolean value

Both dialog and IE techniques make use of language phenomena for reasoning such as hypernyms/hyponyms, holonyms/meronyms and synonyms. For example, Table 5.3 presents some examples of linguistics and knowledge management semantics used for the domain ontology of credit grant.

Definition of multiple instances Since ontologies should be able to incorporate several views of a domain, (e.g. user and system, or several different information sources), as advocated by Eriksson [39], multiple instantiation is important. Multiple instantiation is useful for defining the list of tasks in the task ontology, as well. The resultant ontology

TABLE 5.3 Linguistic and knowledge management semantics

Type	Example	KM equivalent
hypernym	*profile* is the hypernym of *risk profile*	(*profile* :type-of *risk profile*)
hyponym	*risk profile* is a hyponym of *profile*	(*risk profile* :is-a *profile*)
holonym	*credit grant* is a holonym of *amount*	(*credit-grant* :has-attr *amount*)
	credit grant is a holonym of *risk mgr*	(*credit grant* :has-mgr *risk mgr*)
meronym	*amount* is a meronym of *credit grant*	(*amount* :is-attr-of *credit grant*)
	risk msg is a meronym of *credit grant*	(*risk msg* :is-mgr-of *credit-grant*)
synonym	*sum* is synonym of *amount*	(*sum* :is-syn-of *amount*)
	loan is synonym of *credit grant*	(*loan* :is-syn-of *credit grant*)

contains domain information but also information on synonyms. To avoid any misunderstanding, it is important to highlight that the main ontology structure (concepts and their relations) construction is guided by the domain application and not by its utilization in the semantic interpretation system.
Tables 5.4 and 5.5 present a definition for the credit grant system using the MOSS ontology language. All language elements are self-explanatory, except for the notion of synonym. Each synonym is an integral part of the description of the ontology, separated by a semicolon (;).

TABLE 5.4 The domain ontology fragment written using the MOSS language (:att specifies an attribute, :rel a relation)

(*defontology*	(:title “credit grant ontology”) (:version “1.2”) (:language :en))
(*defconcept*	“profile” (:doc “Represents a customer profile”) (:att “account number” (:type number) (:unique)) (:att “customer name” (:type people) (:unique)) (:att “customer type” (:type string) (:unique)))
(*defconcept*	“risk profile” (:doc “Represents a risk profile used for credit evaluation”) (:is-a “profile”) (:att “salary ; income” (:type money)) (:att “since” (:type datetime)) (:att “balance” (:type money)) (:att “timestamp” (:type datetime)))

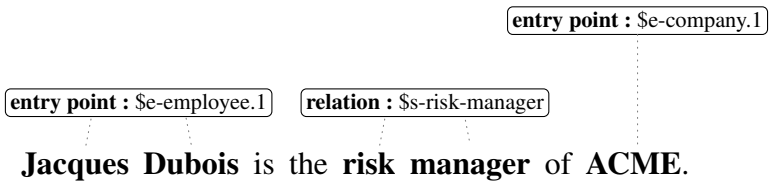
Annotating the sentence with ontology concepts

The objective of this task is, given a sentence, to recognize ontology concepts, attributes, relations and individuals from the dialog context memory. As a result, it returns the sentence with tags (i.e. words or group of words and their corresponding tags).

TABLE 5.5 The domain ontology fragment written using the MOSS language (:att specifies an attribute, :rel a relation)

(*defconcept*	“credit profile”
	(:doc “Represents an evaluation of credit risk”)
	(:rel “profile” (:to “risk profile”) (:unique))
	(:att “score” (:type string))
	(:att “timestamp” (:type datetime)))
(*defconcept*	“credit grant”
	(:doc “Represents a credit grant”)
	(:rel “risk evaluation” (:to “credit risk”) (:unique))
	(:att “amount ; sum” (:type money) (:max 1))
	(:rel “approved by” (:to “employee”) (:min 1))
	(:att “timestamp” (:type datetime)))
(*defconcept*	“employee”
	(:doc “Represents a regular employee”)
	(:att “name” (:type people) (:max 1))
	(:att “role” (:type people) (:max 5))
	(:att “created” (:type datetime) (:max 1)))

The mechanism is straightforward : It matches words and group of words present in a sentence with the set of labels of the domain ontology (synonyms of concepts, relations and attributes). *Entry points* are also included in the set of labels. As stated earlier, an *entry point* is a useful functionality of the MOSS language that allows mapping an individual of a knowledge base to a string. From the dialog perspective, an individual is generated during the interaction with the user and is stored during the conversation. The information extraction component is responsible for the creation of individuals. From now on, we consider that these individuals are already present in the dialog context memory.
The following example illustrates a tagged sentence. Tags came from both the domain ontology and the knowledge base.



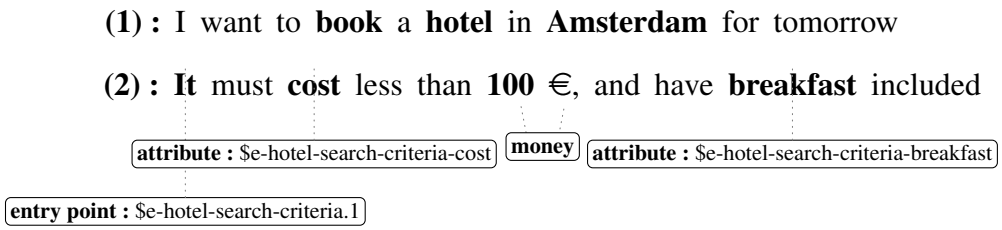
The matching algorithm has identified that the name *Jacques Dubois* is an *entry point* to an *employee*. Again, we assume that this individual is available in the dialog context memory. The phrase *risk manager* points to a *relation* tag that links a *company* to an *employee*. Finally *ACME* has been annotated as a *company*.

Tagging references with individuals

In dialogs users often use anaphora and ellipsis. This means that the dialog context is essential to identify some fragments in the sentence [39]. The dialog manager should decide if the information present in the sentence is a complement of what is already provided. It is up to the dialog manager to integrate the new content to the referenced fragment. Ellipsis and anaphora are complex phenomena studied by linguistics. Phenomena like gapping, stripping verb phrase ellipsis, answer ellipsis, sluicing ellipsis are much more profound in the sense of linguistics. The first version of our system only deals with the following situations :

— A subject pronoun (e.g. he, she, it).
— An object pronoun (e.g. him, her, its).
— A possessive adjective (e.g. its, his, her).

For the sake of balancing examples of different domains other than the banking environment, we use an example of sentence used in travel search portals. The example below presents two sentences : The first one (1) requests a hotel reservation, which triggers the creation an individual of type *hotel-search-criteria*. Then the user complements the information in (2) using the subject pronoun *It*. Note that the pronoun was annotated with a tag that points to the individual created in (1). The idea here is to provide enough information for resolving reference looking for the latest updated individual of the dialog context memory.



A note on special cases : When the entry point is marked as a tag for reference resolution purposes, it is not clear it the user is really mentioning the entry point or not. Thus, one necessary step is to complement the annotation of pronouns with the each concept present in the sentence, so it can “compete” with the referring entry point”.

Annotating the sentence with the action

As stated in the previous chapter, the business process ontology describes processes in terms of capabilities. One of the elements of the business process capability is the set of preconditions and effects. Thus, is normal and expected that there is a unidirectional

dependency between the business process ontology and the domain ontology. That is to say, when a business process is described, it has references to concepts, attributes and relations of the domain ontology as a way to describe preconditions and effects.
An effect is a formal description of the action produced after the execution of the business process. An effect always has an action verb associated with. To illustrate the presence of action verbs, we use our illustrative example of credit grant. Table 5.6 has a line with a pair of type *preconditions* $\Longrightarrow$ *effects*. For the sake of simplicity we have used an informal description of the effect using natural language, with a focus on the action itself. As it can be seen, each effect statement has an action (in bold) followed by the object. The object in this context is a concept of the domain ontology, sometimes accompanied by a criterion. In this particular example, the business process produces three different effects : it creates a grant, it approves a credit grant, and it makes a money transfer.
Just to recap from the previous chapter, actions are very simple concepts that belong to the action taxonomy. They can be organized in a hierarchical fashion, so subsumption-based matching is allowed. These actions are fine-grained. That is to say, they should characterize what is the basic operation performed on this concept by the business process. Thus, an action must directly or indirectly inherit from one of four basic actions namely *create*, *modify*, *get* and *remove*. Going back to the example present in Table 5.6, the first effect and the second effect uses the action *create*. As expected, a new individual of this concept will be generated at the end of the operation. The third effect uses the action *approve*. Here, we assume that this of action modifies the existing *credit grant*, by adding, for instance, the person who approved the request, the timestamp of the approval, and so on. Thus, the *approve* action for the *grant* concept should be a child of the *modify* action.

TABLE 5.6 The credit grant example : the set of actions produced after the execution of the business process

preconditions $\Longrightarrow$ *effects*	
have a valid account	**create** a credit request without fees
	create a money transfer from the bank to the customer account
	approve a credit grant

A dialog-related task is to supplement the action taxonomy with synonyms and verb phrases to improve the recognition of actions. This task is easily executed using some Wizard-of-Oz interactions.
As mentioned before, annotated sentences may present overlaps in some exceptional situations. We have developed an algorithm that is called each time one or more overlaps occur. The algorithm and some examples of overlaps are available in Appendix **??**.

Wrap-up : Semantic annotation as a sub-dialog

From the dialog manager perspective, the semantic annotation is a sub-dialog. More specifically, the semantic annotation is called by the *take notes* sub-dialog. Thus, semantic annotation has states and transitions as can be seen in Figure 5.16.
During the first state, the dialog manager calls the annotation function. If there are no overlaps, the control is returned to the caller with the status equal to *success* (2). In case of overlapping (the event of a tie), the sub-dialog manager checks what is the *preferable* combination of annotations. This function returns only a small set of combinations that have exactly the same semantic density (or the same overall cost). Thus, we have chosen a very simple grounding mechanism : the dialog manager retrieves the first combination in (3) and asks its user in (4) if the current representation of the overlapping tag is the preferred one (e.g. “When you say price, do you mean the price of shares ?”). If the user refuses the suggestion, the control is then return back to (3) to get the next combination, checking again with the user, until all possible alternatives have been exhausted (5) or if the user aborts the dialog (6). If the user accepts the suggestion, then the control is given to (7) that will update the facts base and the dialog returns the control to the caller with the status equals to *success* in (8).

5.4.6 Information extraction

Now we fall into a particular type of information extraction. Our target is a sentence annotated with named entities and ontological information (i.e. concepts, relations, attributes, individuals and actions). One of the methods used for IE is the production of linguistic rules [111]. The most common type of linguistic rules is based on regular expressions (e.g. Faustus and GATE). Linguistic rules based on dependency trees has become very popular since they become faster and robust on irregular texts [15]. Systems such as Kraken [1] and ClauseIE [29] are examples of information extraction that use hand-crafted rules on dependency trees, some of them outperforming traditional classification-based approaches [2].
For our task of information extraction, we have chosen a rule-based approach based on dependency trees. The main reason is the ease of customization and interpretation of the outcomes based on fired rules. We have built a handcrafted set of rules to (i) simplify a dependency tree, (ii) generate a formal representation of the extracted information and (iii) update the dialog context memory. Figure 5.17 illustrates the overall process.

Tree simplification using hand-crafted rules

This task uses a dependency tree annotated with syntactic and semantic tags as an input and generates a simplified version for further information extraction. Our method simplifies the tree by merging, transforming and removing nodes by traversing the dependency tree using a depth-first search, until the tree is stable. The main objective of this step is as follows :

- **Grouping nodes :** nodes that belong to the same tag are grouped. If one specific tag annotates two or more words, the corresponding nodes of the tag are dispersed in the tree. The objective is to merge these nodes into a single one. Some examples : The noun phrase "this morning" is a date/time named entity. Thus, the nodes *this* and *morning* becomes a single node. The same occurs with the compound noun "credit grant", where nodes *credit* and *grant* are also merged.
- **Identifying sources of information** : to fill concepts, attributes and relations a number of grammatical relations are used (e.g. nominal subjects, adjectival modifiers, adjectival complements, appositional modifiers and so on).
- **Recognizing relational operators :** A set of rules is used to recognize expressions that have the potential to become relational operations (i.e. $>$, $\geq$, $<$, $\leq$, $=$, $\neq$). Nodes that follow a pattern compatible with one of these operations will be merged into a single operation unit. Depending on the position on the tree, the node could become a left-hand operand, a right-hand operand, or the operator itself.

To illustrate the simplification process, we use the following sentence :

I want to travel from Paris to London for two days, departing tomorrow.

Figure 5.18 presents the original dependency tree on the left-hand side and the simplified tree and the right-hand side. This sentence has triggered 4 out of 38 existing rules. A brief description of the fired rules is presented in Table 5.7.
Having the simplified tree, the next step is the creation of the formal representation of the sentence.

Information extraction from the simplified tree

During this step, the remaining set of extraction rules is used to build the final representation of the extracted information. This step follows the same approach of the three simplification, by traversing the tree using a depth-first search, running handcrafted rules for each visited node. In the end, a formal representation is generated according to the following BNF form :

TABLE 5.7 Rules triggered for the sentence *"I want to travel from Paris to London for two days, departing tomorrow"*

Rule #	Description	Effect
r13	A numeric modifier of a noun that is marked as an attribute becomes an operation (*noun = nummod*)	*travel-days = 2*
r14	an object of a preposition that is assignable and the preposition is marked as an attribute becomes an operation (*prep = obj*)	*travel-to = "London :location"*
		travel-from = "Paris :location"
r8	A preposition that is in the middle of two operations is removed	*for is removed*
r2	A personal pronoun that does not have an annotation and is linked with the main verb is removed	*I is removed*

```
       <sentence> ::= <action> (<concept | <indiv>) {<clause>}* {<unknown>*})
        <concept> ::= the name of the concept
          <indiv> ::= reference of an individual (a string starting with \$)
         <action> ::= <action-keyword> <action-concept>
 <action-keyword> ::= :action
 <action-concept> ::= the name of the action
         <clause> ::= <property-clause> | <sub-query> | <ask-clause>
<property-clause> ::= (<property> {<property-op> <value>})
     <ask-clause> ::= (:ask <clause>*)
      <sub-query> ::= (<concept> <query>)
    <property-op> ::= < | <= | = | != | >= | >
       <property> ::= the name of the property
          <value> ::= constant value
        <unknown> ::= (:un {<utuple>*})
         <utuple> ::= (<moss-type> :from <offset> :to <offset> <string>)
         <offset> ::= numeric value
```

The main objective of this step is as follows :

— **Identifying indirect properties :** Sometimes users are not aware of relations between concepts. As a result, some relations are "flattened" by the user. The objective here is to rebuild the hierarchy between these concepts.

— **Identifying floating values :** Sentences may present important information floating on the sentence, without an attribute that "owns" this value. The objective

is to find one or more potential attributes so that a clarifying question can be sent to the master.

— **Identifying the head of the sentence :** Our approach only accept one "head" concept per sentence. Sometimes users do not specify the concept in the sentence. For these cases, we use attributes and relations as a source to find the head. If more than one concept that does not have direct relations is found, then an error is produced.

Tables 5.8 and 5.9 contain some test cases used to validate the extraction rules. Cases from 1 to 8 are examples of unambiguous sentences. All outcomes have an action, a concept and attributes with compatible values. Cases 4 and 5 are a bit different but still valid. The sentence did not specify that VISA the value of the attribute *type* of the *credit card*. However, one of the rules identified that VISA is a noun compound modifier of credit card. It used the type of the node to recursively look for compatible properties of the credit card, finding at the end, only the attribute named *type*. Case 9 has a well-defined outcome, but as can be seen, there are parts of the sentence that could not be recognized. For instance, the stream of tokens "because I am not using it in" have been marked as *unknown*, as well as the location "New York". From now, we simply discard this content and use grounding to confirm if the information extraction was satisfactory.
When a value can fit more than one direct or indirect attribute of a concept, the algorithm annotates the value using the tag *ask* followed by the set of potential attributes. For example, sentence 9 has one example. The user said : "Please, increase my credit card number 9500125143214433 up to 5000", and the algorithm found both *limit* and *issue-number* as potential attributes. When it happens, a sub-dialog is used to ask the user what is the most appropriate attribute. For instance, the PA asks "For the value 5000, do you mean the limit or the issue number ?".

Update memory

Here we assume that the formal representation does not have ambiguities (i.e. all attributes have been identified). This task uses the formal representation to create or update the individual in the dialog context memory. This individual also carries two essential elements used to find suitable business processes : the dialog act and the action. The dialog act is used by the search engine to infer if the user is talking about his or her current state (i.e. assertion) or desired state (i.e. a directive act).
The following sentence is used to show how the individual is created :

TABLE 5.8 Some test cases showing different types of sentences and the outcome

1	Directive act with action and concept	
	Sentence :	Cancel my credit card
	Outcome :	*(:action “cancel” "credit-card")*
2	Directive act with action, concept and direct attributes with compatible values	
	Sentence :	Please, cancel my credit card with number 9500125143214433, expiring next month
	Outcome :	*(:action “cancel” ("credit-card" ("number" = "9500125143214433") ("expiration-date" = "10/2015")))*
3	Directive act with action, concept and direct relation with compatible reference	
	Sentence :	Add John Silver as a member of the ACME project
	Outcome :	*(:action “add” "project" ("name" = "ACME") ("member" = "$e-person.1"))*
4	Directive act with action, concept and indirect known attribute without value	
	Sentence	List the quantity of points of my credit card
	Outcome	*(:action “list” "credit-card" ("reward-program" ("points")))*
5	Directive act without action verb	
	Sentence :	I want a credit card
	Outcome :	*(:action “create” "credit-card")*
6	Directive act with action, without concept and attribute with compatible value	
	Sentence :	I want to increase my limit up to 7000
	Outcome :	*(:action “increase” "credit-card" ("limit" = "7000"))*
7	Directive act with action, concept and a value that is compatible with the concept hierarchy	
	Sentence :	Please cancel my VISA credit card
	Outcome :	*(:action “cancel” ("credit-card" ("type" = "VISA")))*

Sentence (directive act) :	Fees must be less than 1%
Formal representation :	(:action “create’ "credit-grant" ("fee" < "1%"))
(defindividual	*"credit-grant"* *("fee" :lt 1)* *("act" "directive")* *("action" "create")* *("timestamp" (get-universal-time)))*

Note that the individual is stored even if the expression has relational operators (i.e. $>$, $\geq$, $<$, $\leq$, $=$, $\neq$). This is particularly useful for the business process selection task, that could use those operators to better filter and classify processes.

Wrap-up : Information extraction as a sub-dialog

As it happens with the semantic annotation, the information extraction is also implemented as a sub-dialog. It is also called by the *take notes* sub-dialog. Thus, the information extraction also has states and transitions as can be seen in Figure 5.19.

TABLE 5.9 Some test cases showing different types of sentences and the outcome (continuation)

8	Assertion using an individual and direct attributes with compatible values	
	Sentence :	It must cost less than 100 euros and have breakfast.
	Outcome :	*(:action "modify" "$e-hotel-search" ("cost" < "100") ("breakfast" = "true"))*
9	Directive act with action, concept, one direct attribute with a compatible value and one named entity that is incompatible with the concept hierarchy	
	Sentence :	Please cancel my credit card with number 9500125143214433 because I am not using it in New York.
	Outcome :	*(:action "cancel" "credit-card" ("number" = "9500125143214433")* *(:unknown (:string :from 9 :to 15 "because I am not using it in")* *(:location :from 16 :to 17 "New York"))*
10	Directive act with action, concept, one direct attribute with compatible value and another compatible value compatible with two direct attributes	
	Sentence :	Please, increase my credit card number 9500125143214433 up to 5000
	Outcome	*(:action "increase" ("credit-card" ("number" = "9500125143214433")* *(:ask ("limit" = "5000") ("issue-number" = "5000"))))*

The dialog manager starts by calling the IE function in (1). If there is no ambiguity the control is transferred to the caller with the status equals to *success* (2). In the case of ambiguity, the dialog manager asks clarifying questions to the user. For instance, user says "The ACME share has reached 34.02", and the recognized concept (*share-log* has two or more attributes that are compatible with this type like *buying price* and *selling price*. The dialog manager gets the ambiguity from the stack in (3) and asks the user in (4) to choose from a set of options what is the most appropriate type of information (including discard the information). For instance, the PA asks : "When you mention 34.02, do you mean buying price, selling price or none of them ?". The user can abort (5) or choose from one of the options. Then the choice is updated in (6). When there are no more elements in the stack, the dialog context memory is updated in (7), and the control is transferred to the caller with the status equals to *success* (8).

5.4.7 Business process selection and triggering

The business process selection and triggering is one important feature of our conversation interface. It aims at capturing potential business processes based on what the master said (i.e., what is stored in the dialog context memory).

The business process conversation dialog

In practical terms, the business process selection is modeled as a conversation graph with conversation states linked with a facts base and edges representing the decision made during the transition. Figure 5.20 presents an illustration of the conversation graph.
At this point, the *take notes* sub-dialog was already executed. Thus the formal representation of the previous sentence is available in the memory. The sub-dialog starts in (1), preparing the query to be sent to the BPM agent. That is to say, the PA collects the set of individuals created by the PA during the *take notes* activity. In (2) a request containing these individuals is sent to the BPM agent. The BPM agent answers with a set of compatible business processes. Then, the list of compatible processes is presented to the user in (3). The list of processes by decreasing order of overall score is presented to the user. Each line contains a description of the process, the score of preconditions, the score of effects and the overall score. Just to recap, the precondition score measures the level of conformity of the set of assertions made by the user with preconditions of the business process. The effect score measures the level of conformity of the effects of the business process with the user's expectation. Finally the overall score is a ratio between the precondition and the effect score. Finally the user can start one of the processes (4) or simply get information about what the business process offers and continues the dialog.

5.4.8 Business process enactment

In our approach, the enactment of a business process is not managed by the dialog manager, but by the BPM system. The BPM system is responsible for the whole orchestration of the business process and the dialog manager for selecting and triggering processes and executes human tasks when required. It means that the dialog manager must be aware if several asynchronous events are generated during the conversation with its master. Some examples : a task allocation that arrives during the selection of a process ; a process that finishes the execution of a task. Thus, a special mechanism for dialog management is necessary.
This section explains how our system works in connection with a typical BPM system. The lifecycle of a process instance is first analyzed, assuming that the master is authenticated and has active agents.

Creation of a process instance

The main outcome of the previous section was the act of triggering a business process from a dialog session. We recapitulate this functionality in more details by using a

sequence diagram present in Figure 5.21. The solid arrowhead denotes synchronous messages. The corresponding return messages are drawn as dotted lines with an open arrowhead back to the originating lifeline. Asynchronous messages are drawn as solid lines with an open arrowhead to the receiver.
First, each element of the sequence diagram is presented. <*usr*> is our master, a human. We have used the placeholder <*usr*> to denote the user id (e.g. Barthes, Moulin). The agent <*usr*>-*pa* is a personal assistant in charge of the dialog management and <*usr*>-*rep* represents a logged user in the BPM system. The fourth component of the sequence diagram is the *BPM system agent* representing a generic software that is driven by explicit process designs to enact and manage operational business processes (van der Aalst [103] [106]. This agent has access both to the domain ontology, the business process ontology and its knowledge base, as well as a skill to execute the matchmaking.
The example starts when the user asks something to the PA using a directive act (e.g. "I want a credit grant of 5000 euros without fees") in (1). Then, the PA interprets the user's directive using both the domain ontology and the dialog ontology. It makes a request to its partner (*bpm-rep*) to find processes or services that are compatible with the given information (2). This agent, however, does not have a specific skill for making a semantic match between the set of individuals sent by the PA and the set of business process capabilities present in the knowledge base. Thus, it delegates this task to the *BPM system agent* in (3). The matchmaking is made, and the agent returns the result to *bpm-rep* that, in turns return the result to the PA. Finally the PA formats the result to its master.
The user accepts the suggestion of the PA in (4). The PA asks its partner to create and run a process instance in (5). Then, *bpm-rep* requests the creation of the process instance to the BPM system. A new instance of the process is created (7), the *representative* is added as a listener for all lifecycle events of the process (8) and the process starts (9). Note that the message sent in (9) is asynchronous, which means that the BPM system agent does not wait for the process to finish and other agents are free to execute other activities.
The act of listening for BPM events is one of the most important roles of the *bpm-rep* agent. It listens not only to events of processes that have been created by its user, but it also listens to the arrival of tasks that have been allocated or offered to the user. Each event will be explained in the remaining of this section following the typical lifecycle sequence. Now, we examine how tasks are executed.

Task notifications

Just to recap : a process is a sequence of tasks and each one is a *work item* that has been allocated to a resource. It can be executed automatically if it does not require a human

resource. Conversely, a *human task* requires a human to provide information, to see results or both. The master of a PA could be the *initiator* of a process (i.e. he or she has started the process), or the master could be a *participant* in a process. For instance, the customer that requests a credit grant is the *initiator* of the process ; the risk manager that evaluates the credit request is a *participant*.
Figure 5.22 illustrates the lifecycle of a *work item*, adapted from Russel in [84]. A *work item* is first created and then allocated by offer or direct allocation. When allocated, a *work item* becomes a task and the human starts the task. During its execution, the task can be suspended to be further resumed, can fail due to technical problems or can complete successfully.
We start by describing the moment when the BPM system notifies the PA that its master has a task (i.e. task at the *allocated* state). The PA should avoid interrupting its master if she is already involved in a conversation. For instance, the master could be providing information to another task or the master could be selecting a business process. In both situations, the PA assumes that its master is *busy*. Thus, it adds the notification of new task to its *todo list* slot. The user interface component listens to modifications of this slot, so in the case of new elements, it displays a discrete hint indicating that a new task has arrived for further verification. At any time, the master can access the *todo list* and start the task it contains. If the master is free (i.e. running only the top-level dialog), then the task conversation is started immediately.
The second type of notification is used to offer a work item to the master. The master is not yet the *owner* nor *committed* with the work item. It is up to the master to accept the task offer or to refuse the task offer. This type of message is directly added to the todo slot of the PA. At any time, the master can access the todo panel and accept or refuse the task offer.
The third type of notification is the cancelation of a task offer, meaning that a previous task offer has been canceled (the sender has given up or has found another participant). This type of message is added to the todo slot of the PA. At any time, the master can access the todo panel and get informed. No follow-up action is required.

Starting a task

As stated earlier, a human task is not started automatically by the system, since it requires a human action : the fulfillment of input parameters, the visualization of content or both. Thus, a sequence of messages is exchanged between agents to properly start the task (e.g. filling information, validating preconditions, displaying error messages). We present a sequence diagram in Figure 5.23 illustrating the communication between agents.

We assume here that a process instance is active, and a human task has been allocated to our master. Our master could be the initiator of the process (e.g. the customer asking for credit) or a participant (e.g. the risk manager in charge of approving or refusing the credit). The sequence diagram begins with the *BPM system agent* sending a notification to the *bpm-rep* agent (1) informing that its user has a pending task. The message is then forwarded to the PA in (2). Again, if the PA is busy, then the notification is added to its *todo list* slot. If it is idle, then a dialog dedicated to executing a task is started. The dialog procedure that follows is the same in both cases. In this example, we assume that the PA was idle, and it automatically started a dialog.
Suppose that this task requires the fulfillment of the *amount* to borrow. The PA follows the message instructions and asks its master to specify the amount in (3). Having all parameters filled, the PA sends a message to the *bpm-rep* agent, requesting to start the task in (4). It then delegates the request to the BPM system agent that detects a violation of the precondition (e.g. an amount superior to a limit). It then answers the request containing details about the violation. The PA informs the user about the violation and asks again for the value in (7). This process continues until all information is correctly fulfilled and the task starts as can be seen in (10). Finally, when the task finishes, a notification message is sent to the PA (12), that displays the results to the user (13).

Task dialog

To provide information for the task, a task sub-dialog has been created. The purpose is to help users filling input parameters. Notice that in general, business process systems make use of traditional forms as a communication channel with the user. These forms are dispersed among several tasks of the process. This model has been used during decades in typical information systems, and users are familiar with it. Our approach is slightly different, having a personal assistant, asking questions, taking notes and using previous notes as a source for automatically filling information. When developing the task sub-dialog we have obeyed the following requirements :

- *Help users during the fulfillment of complex parameters :* Input parameters may be atomic (strings, numbers) or complex (a concept with attributes and relations). Complex parameters have cardinality constraints that must be observed.
- *Allow users to make general assertions at any time :* Users should be free to provide information without needing to wait for the “right time” to give the information.
- *Reuse notes taken by the assistant while filling parameters.*
- *Allow users to select and trigger business processes during the conversation.*

We present here some highlights of the algorithm. It transforms the current task and the list of parameters into a directed acyclic graph (DAG). It is not a tree because the corresponding types of each parameter are linked with a common and reusable type (e.g. string, number and domain ontology concepts). This DAG is traversed using breadth-first, visiting each node and expanding them when needed. More specifically, an expansion is not necessary for simple ranges (e.g. string, number) does not need expansion and the corresponding action when reaching the node is just asking the user for the value. When the type is a concept, then an expansion is executed, taking into account the concept and its corresponding properties (attributes and relations). Note that when the property is a relation, a recursive call is performed. Of course, a list of visited nodes is used to avoid infinite recursive calls when the concept has a relationship with itself (e.g. a person with an attribute *conjoint*).
Figure 5.24 presents an example of sub-dialog for task parameter fulfillment. In this example, the user is interested in buying some shares. The user sends a directive act to the PA that in turns takes notes and creates an individual to store the information (symbol and price) into the dialog memory (1). The PA continues the dialog selecting and triggering a business process in (2). As soon after the process is started, the PA receives a notification of a pending task from the BPM system. The user must provide information to register the order. Next, the PA matches all parameters with its context memory, finding two compatible individuals : The current account is matched with the first parameter, and the order details created from the *take notes sub-dialog* is also used. Of course, this individual is not complete. The PA starts by asking for the due date in (4). Note that users Are free to make assertions, directives and so on. In (5) the user changes his mind, altering the symbol and the price. The PA take notes in (6) but continues from the previous request, asking for the due date. Note that *take notes* sub-dialog is used in all points where user data is necessary.

Finishing a task

The PA is notified when the following events occurs :

- A task was finished with success.
- A task was canceled with success.
- A general error occurred.
- The process instance to which the task belongs was canceled.

For all the above scenarios, the PA does not take any special action, only displaying the information attached to the message. The most common and expected event is when a task finishes with success, having a description after completion. Other events (cancelation and

error) have an attached message with the error reason that will be simply shown to the user.

Finishing a process

The PA is notified when the following events occurs :

- A process was finished with success.
- A process was canceled.
- A general error occurred during the execution of the process.

The behavior is the same than for the task function. For all the above scenarios, the PA does not take any special action, only displaying the information attached to the message. The most common and expected event is when the process finishes with success.Other events (cancelation and error) have an attached message with the error reason that will be simply displayed to the user.

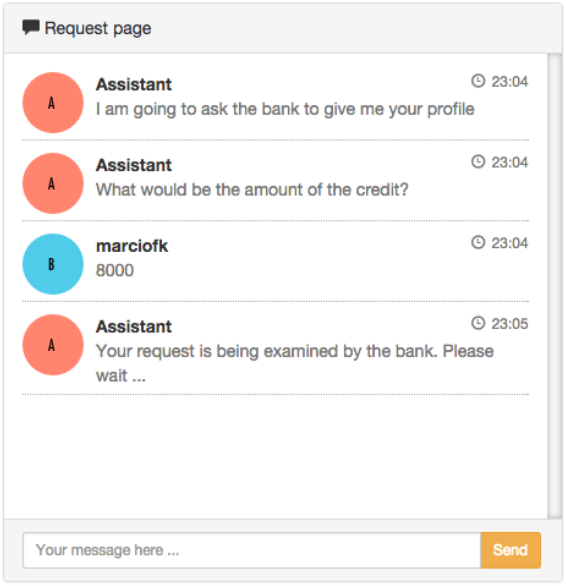
5.5 Discussion

We started this chapter by discussing how the control structure of a process has a direct influence on the user experience and how it affects the productivity of users. The expressiveness power of traditional interfaces makes virtually impossible for stakeholders (e.g. personal assistants) to search useful processes directly from the process space. Another aspect discussed is more a motivation than a problem, but affects directly the IT budget of organizations : the growing acceptance of natural language (written or spoken) as a common interface for applications and the prohibitive effort to port those applications to more natural and expressive interfaces.
Then, an overview of the OMAS dialog manager was given, developing at the end a simple dialog and drawing some basic requirements for developing our approach. The PA4Biz conversational interface for enacting process was presented in three distinct parts : working with controlled natural language sentences and using ontologies for information extraction, selecting and triggering processes and enacting processes. The PA4Biz can be viewed as an attempt to provide a modern enactment interface, combining the flexibility of dialog systems with the robustness (but also rigidity) of business processes.
Since this topic (assistants for processes) uses concepts from several domains (e.g. BPM, MAS, PA, shallow NLU, KM and so on.) we have developed a dialog manager architecture in separated layers, with a low degree of cohesion. Unfortunately, the time to make a comprehensive test on each component and algorithms developed is incompatible with the timeline or a PhD research.

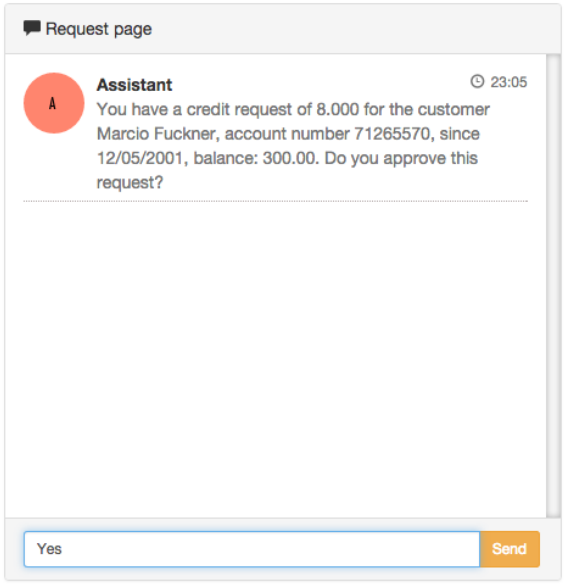
The idea of combining personal assistants and business processes must be evaluated in different scenarios. PA4Biz is at the time of writing at an early stage of development and does not benefit from the years of incremental refactoring and testing shown by more mature architectures. A number of issues and possible improvements have been detected during the designing of PA4Biz, like :

- Hand-crafted rules may not cover all cases and are too general to be universally applicable.
- The semantic interpretation only analyzes factual information and does not make a differentiation between space and time.
- Personal assistants could “learn” new rules from the BPM system but they do not learn from the user.
- Semantic lexicons could dramatically improve the interpretation of sentences

Further development work is certainly needed to move PA4Biz from being a research prototype to a platform capable of being deployed in any application domain.



(a)



(b)

FIGURE 5.7 Two dialog sessions : (a) with the customer, and (b) with the risk manager

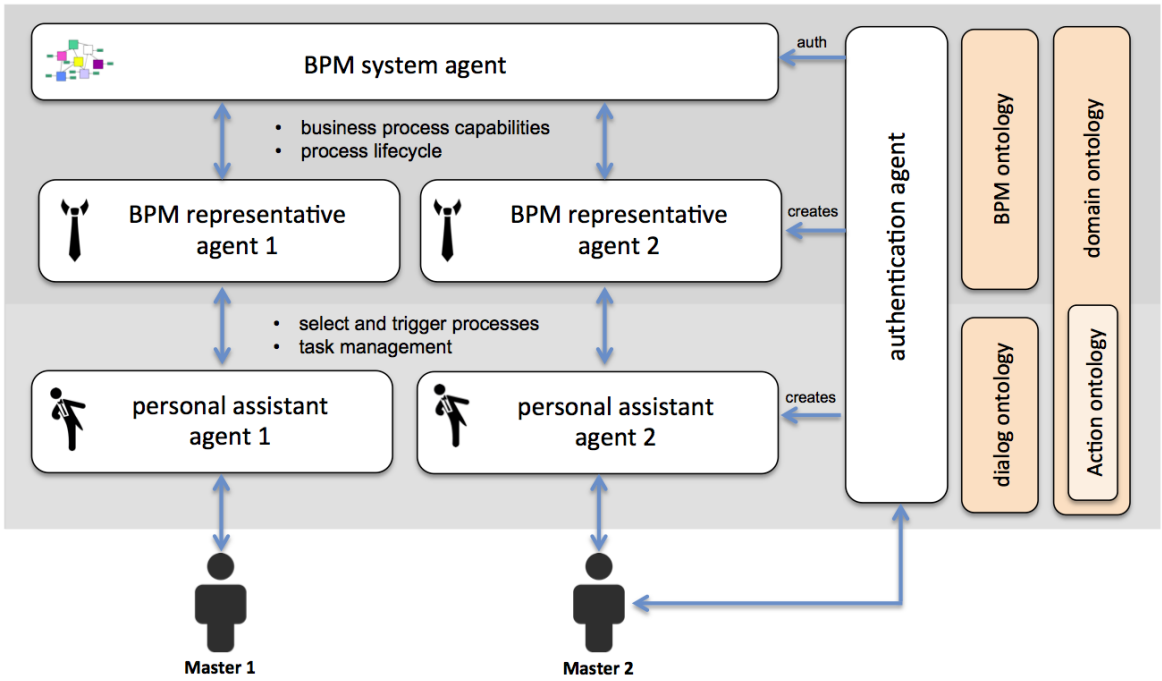


FIGURE 5.8 Overall architecture of PA4Biz

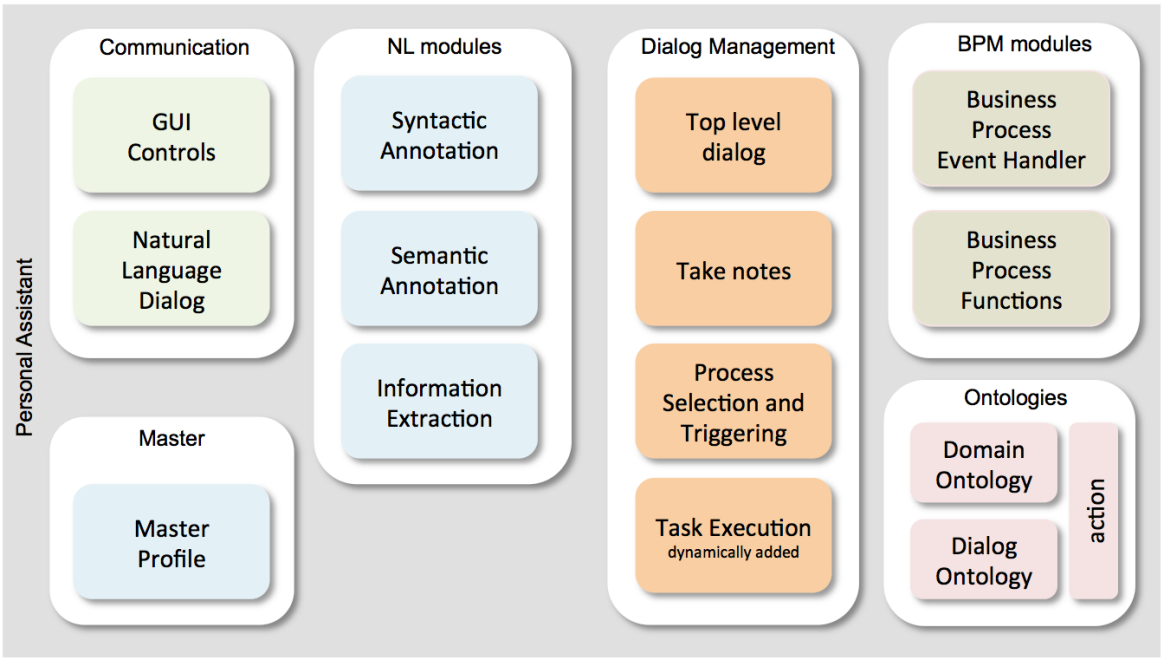


FIGURE 5.9 Personal assistant architecture

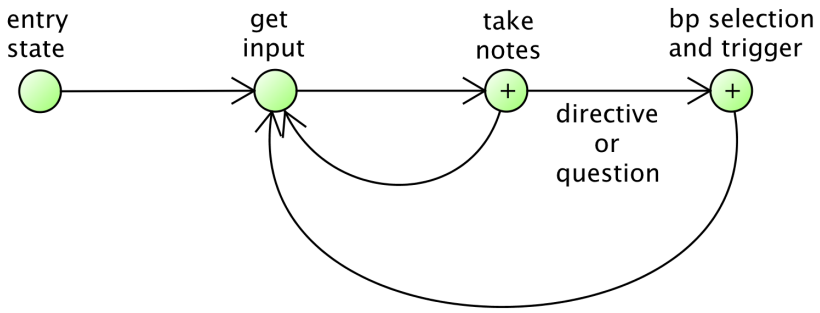


FIGURE 5.10 Top-level dialog for the business process selection problem

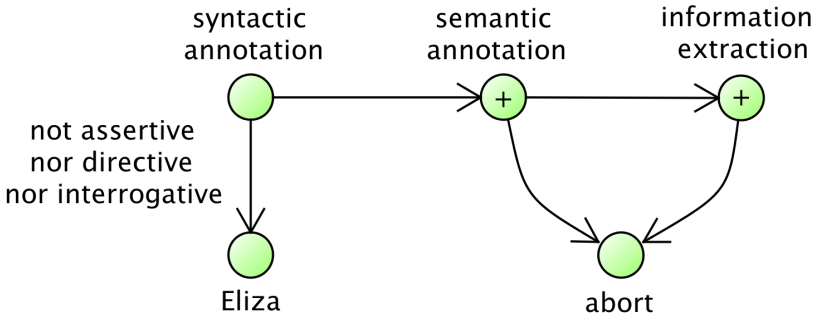


FIGURE 5.11 The sub-dialog used for taking notes

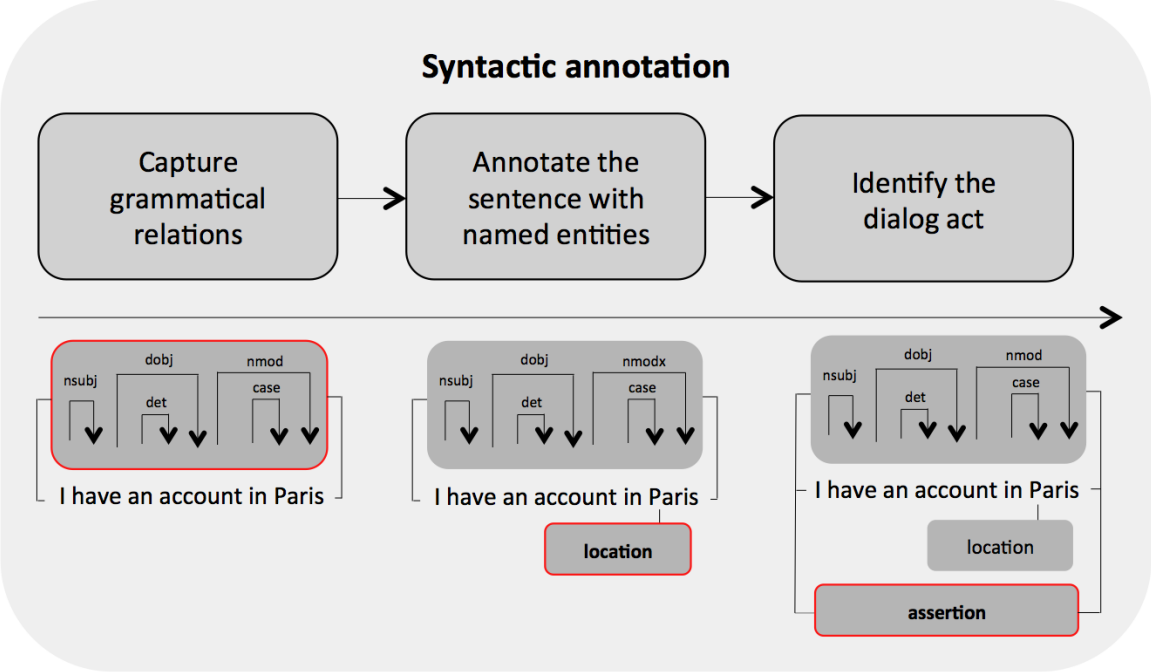


FIGURE 5.12 The syntactic annotation process with an example of sentence been annotated

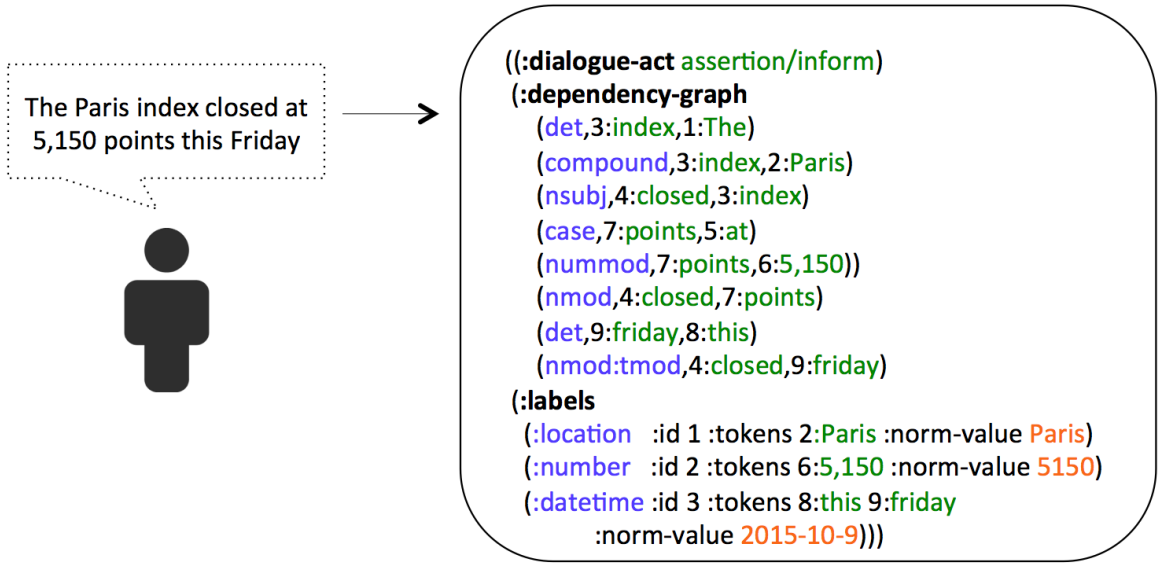


FIGURE 5.13 From the sentence to the unified format

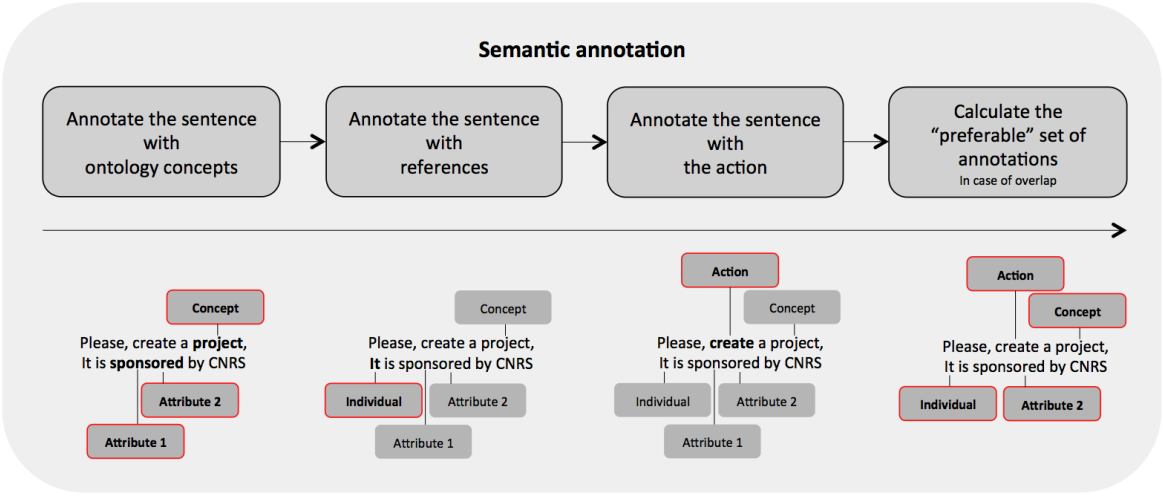


FIGURE 5.14 The semantic annotation process with an example of sentence been annotated

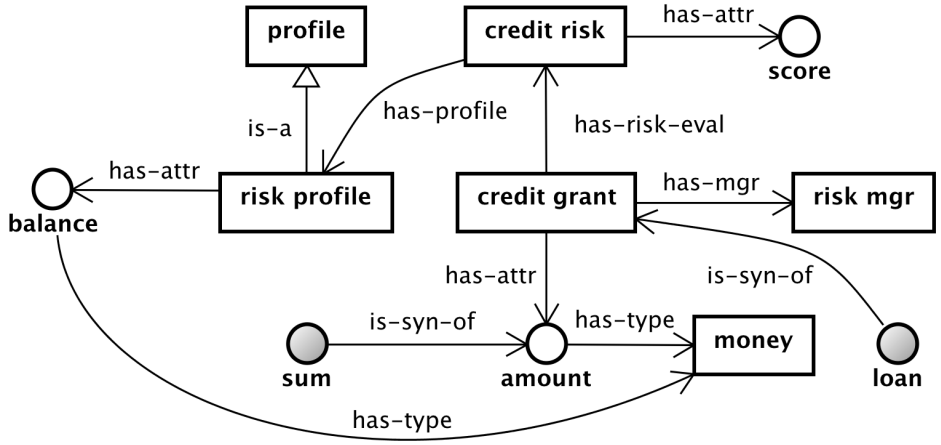


FIGURE 5.15 A fragment of a domain ontology for our illustrative example of credit grant

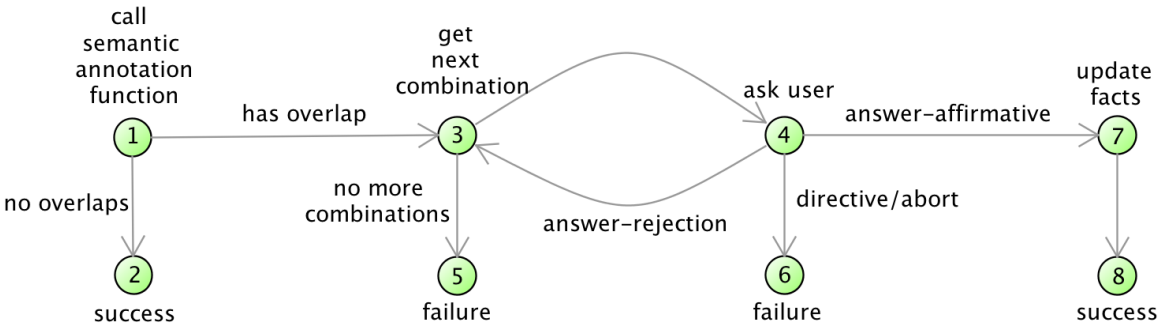


FIGURE 5.16 The semantic annotation sub-dialog

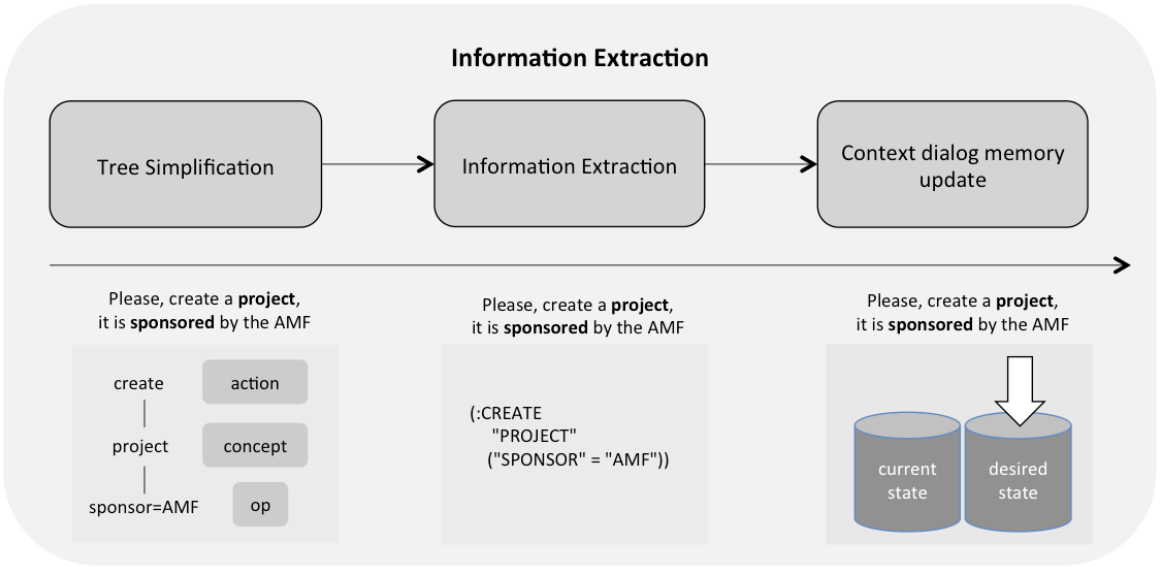


FIGURE 5.17 Three steps for information extraction

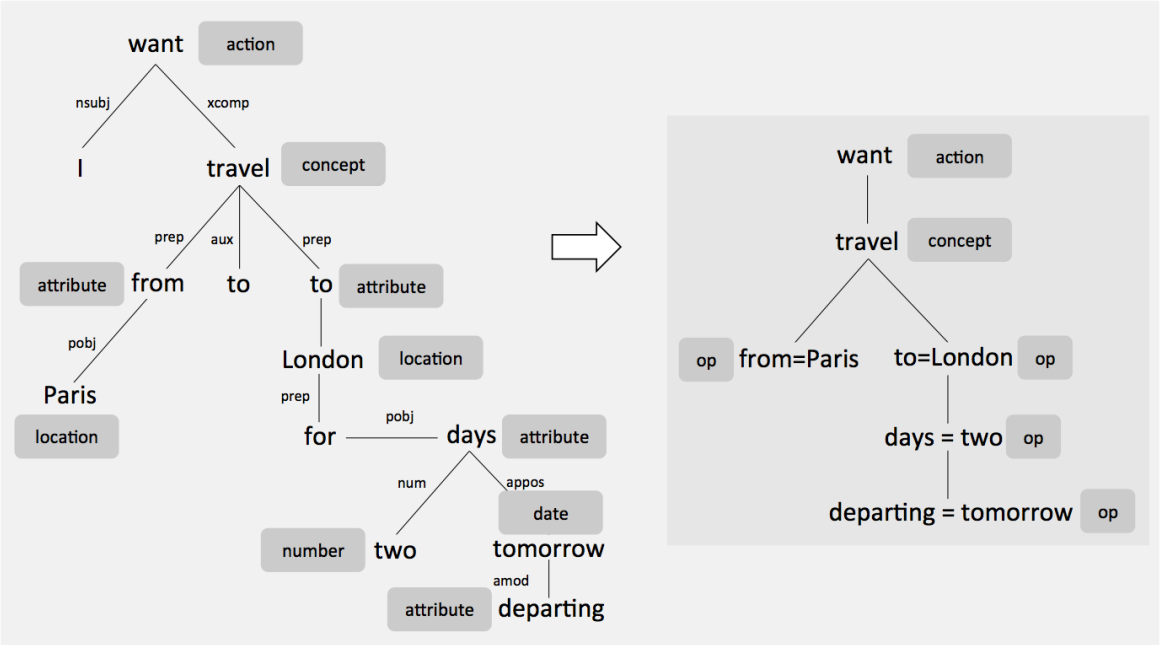


FIGURE 5.18 An example of tree simplification using hand-crafted rules

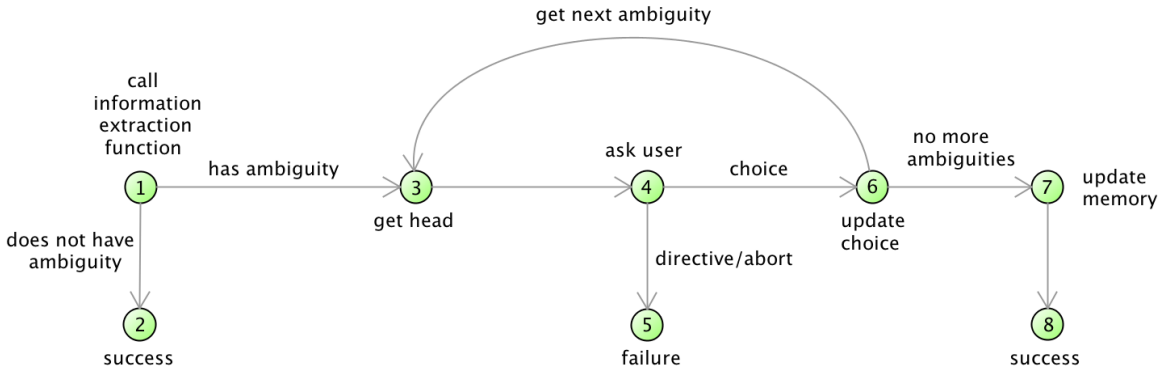


FIGURE 5.19 The information extraction as a sub-dialog

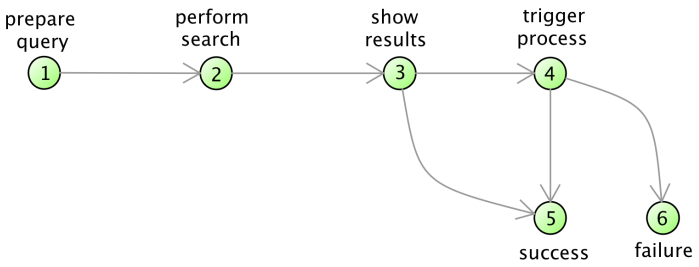


FIGURE 5.20 The conversation graph of the business process selection

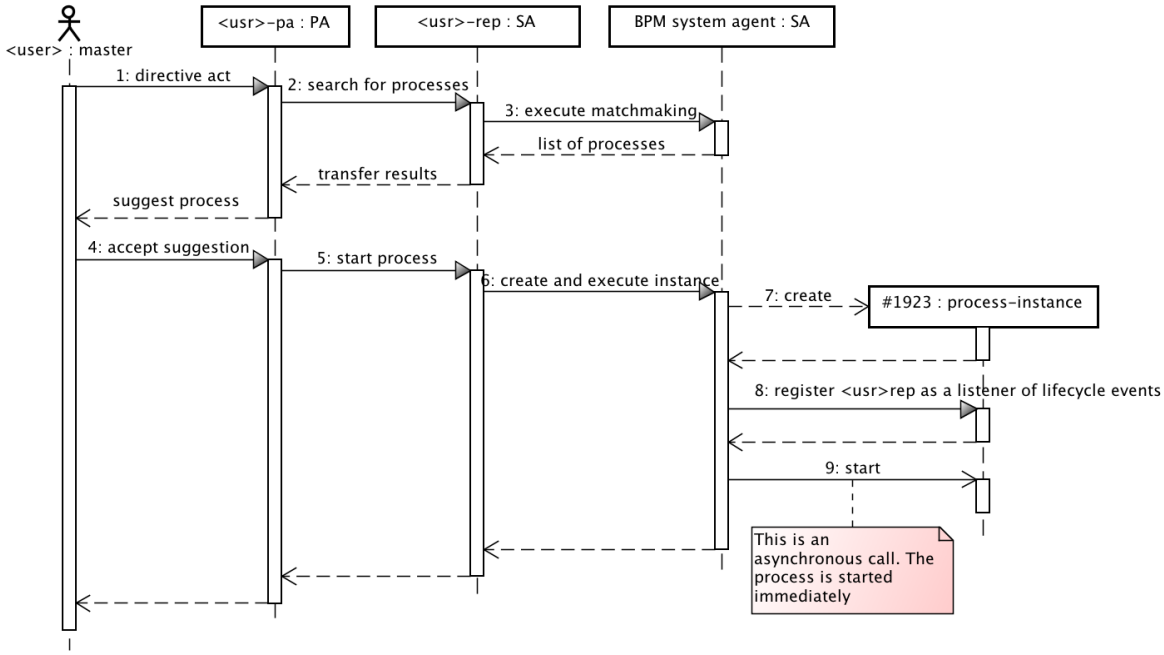


FIGURE 5.21 A sequence diagram showing the interaction between the MAS and the BPM system to instantiate and start a process

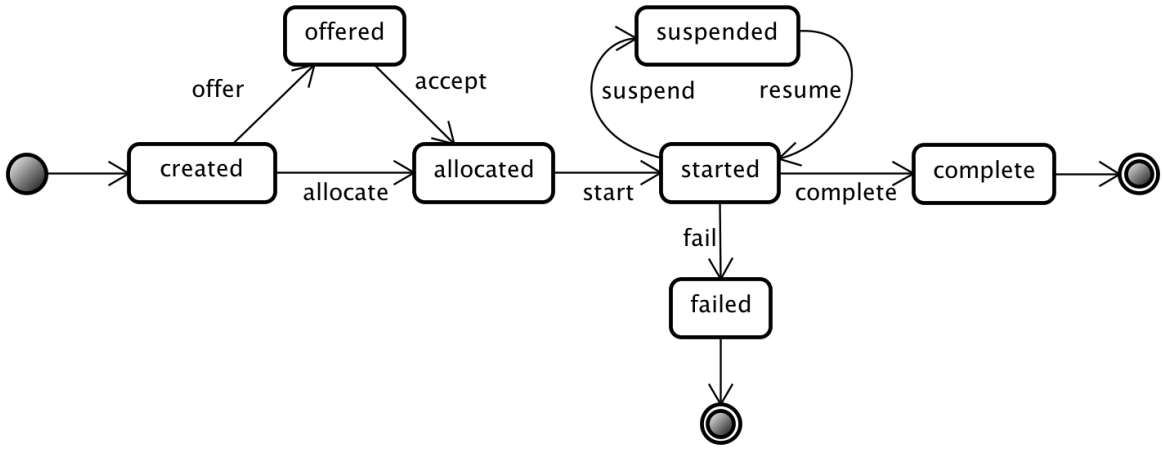


FIGURE 5.22 The lifecycle of a work item that becomes a task [84]

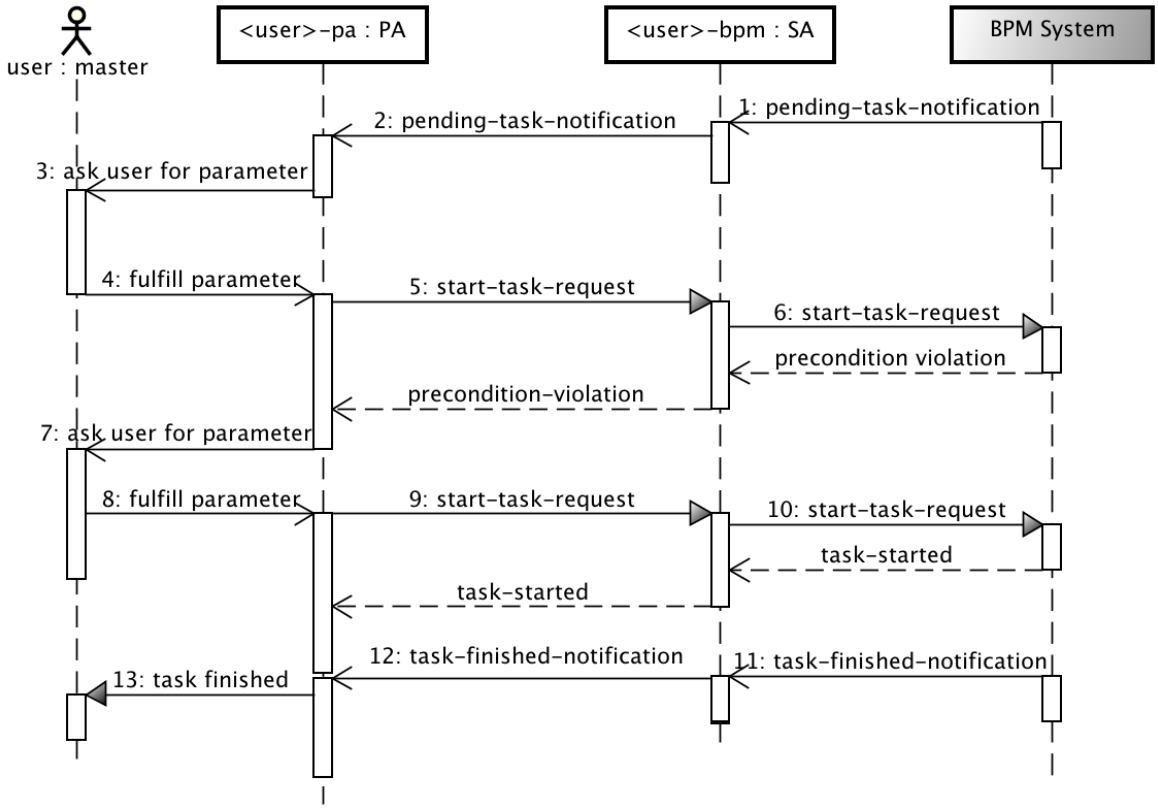


FIGURE 5.23 Receiving a notification of a pending task, providing information and starting the task

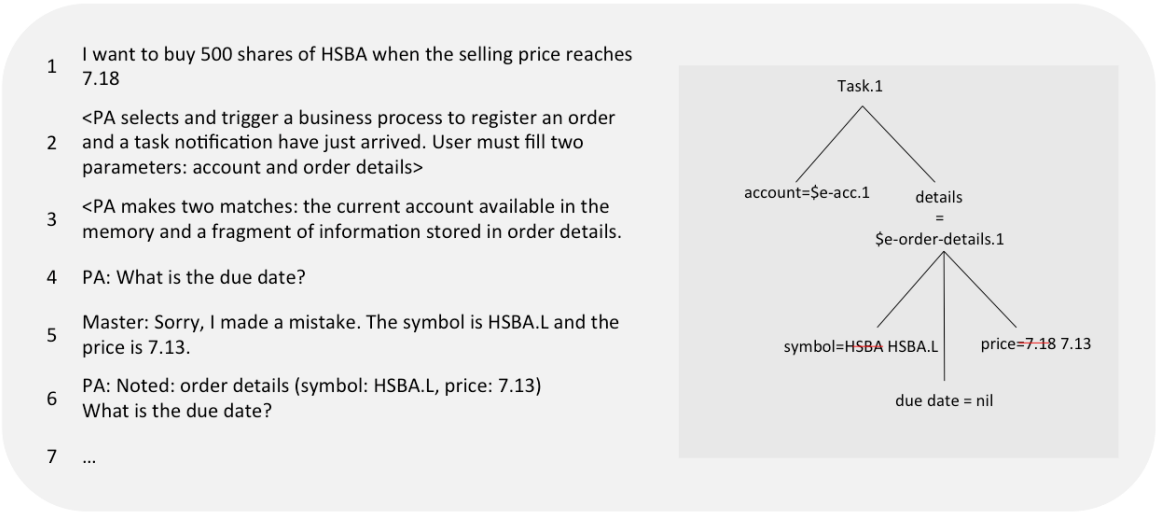


FIGURE 5.24 An example of dialog used to fulfill parameters

Chapitre 6

Realization and experiments

In Chapter 4, we introduced an approach to describe and build the capabilities of business processes using a business process model as a source, as well as an approach to query the business process space. To endow agents with these capabilities, we described in Chapter 5 a query mechanism based on the interpretation of user sentences and a dialog manager adapted to the business process enactment problem.
In this chapter, we first describe the implementation of a functional prototype, PA4Biz, in Section 6.1. Then, we present the evaluation of our functional prototype in Section 6.2, which includes an investment application. Section 6.3 summarizes the chapter.

6.1 Technical Architecture

We have built a functional prototype based on previous specifications. Figure 6.1 presents the technical architecture of our functional prototype. The system was developed progressively, and its components are detailed in the following sections from the bottom to the top, from the user interface to the enactment engine.

Web agent

We used the OMAS framework allowing the creation of conversational interfaces using both client applications and web applications. A client application requires the installation of a Lisp environment and the personal assistant on the user machine. A web application requires not only these components but also a web server to handle HTTP requests. The first version of our functional prototype is a web application that uses the Allegro Lisp Environment and AllegroServe, an open-source Web server [3].

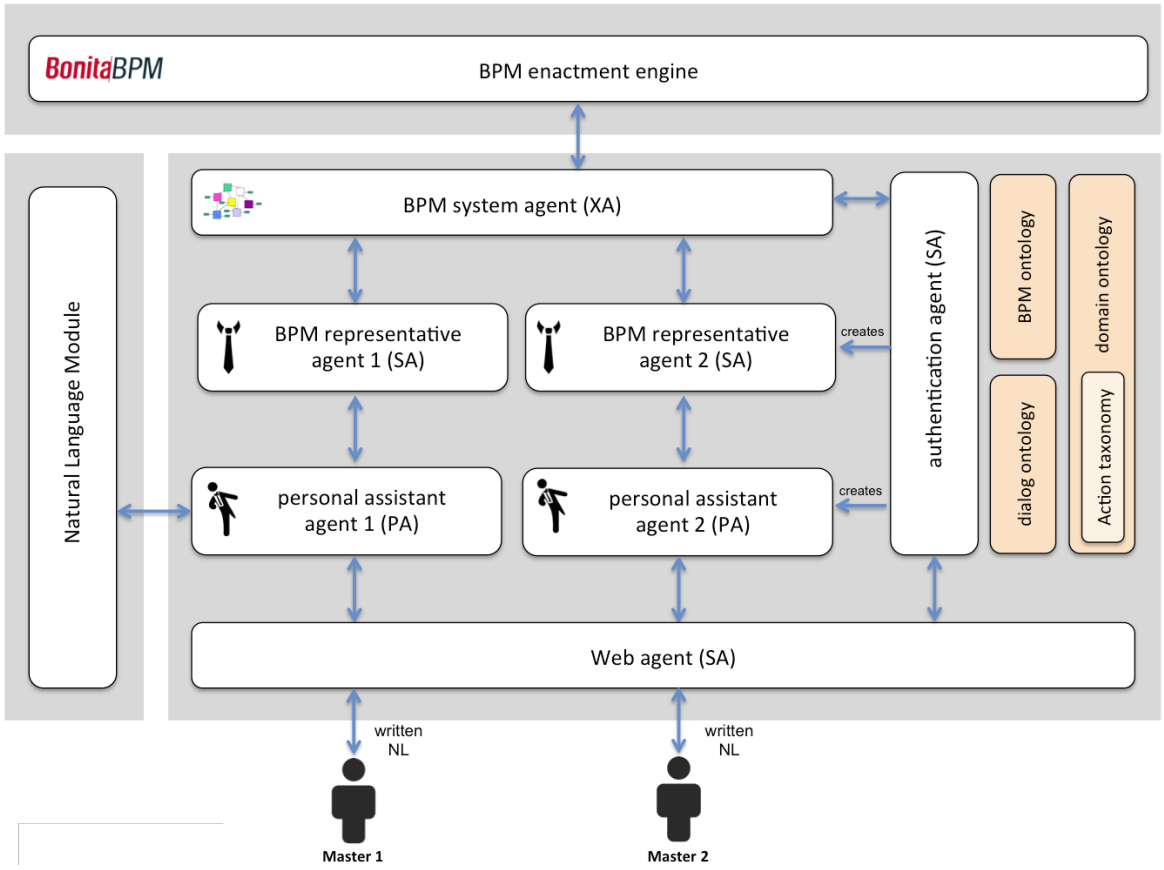


FIGURE 6.1 Technical architecture of the PA4Biz functional prototype

Note that part of the multi-agent system becomes client/server in nature when exposing the user interface to users through a web server, losing the P2P characteristic of multi-agent systems. Thus, the direct exposition of the personal assistant to the end user using a single web entry point could result in a lack of scalability and portability. For this reason, we have designed the *web agent* that is responsible for (i) receiving requests coming from the user, (ii) identifying and delegating the most proper service to execute, and (iii) building the response to the user at the end.

The *web agent* can be considered a hybrid *service agent* in the sense that it must deal not only with its pairs using the internal protocol, but also receive requests coming from a web server. To allow a clear separation from the presentation logic to the application logic, we use the *Front Controller* pattern used to separate the application in logical layers (presentation, control, and logic). To accelerate the development process, we use the open-source web framework called *WebActions* [4] that already implements these features. To better understand how it works, Figure 6.2 presents a UML deployment diagram. Let's assume that the user called Sarah is using a web browser and is already authenticated into

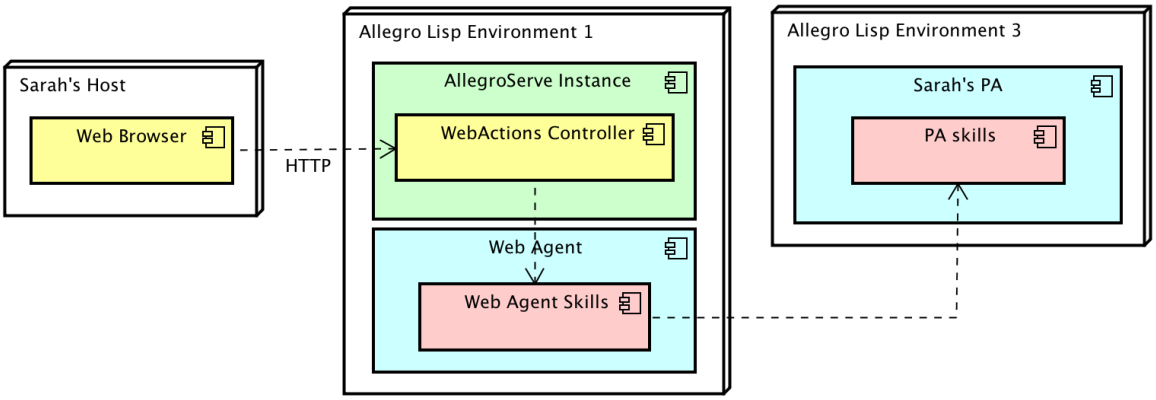


FIGURE 6.2 The role of the web agent into the application

the system. Using the dialog interface, she writes the sentence "What is my balance ?" and sends the request to the web server. A particular thread of the *AllegroServe* instance receives the HTTP message. Next, the *WebActions* component takes control and chooses the most suitable action to execute from a set of predefined actions. In this particular case, the controller invoked the action that sends a message to the personal assistant. The agent delivers the message to the personal assistant of Sarah and waits for the assistant's response. Having the response, it forwards the assistant's answer to the controller. In the end, the controller invokes the presentation logic function to build the result for the user.

Authentication agent

The purpose of the *authentication agent* is twofold : Its first role is to check the credentials of users, and its second role is to prepare the working session when used for the first time. That is to say, this agent is responsible for the creation of user agents during the authentication process and destroying them during the logout process. It is a *service agent* that attends requests coming from the *web agent*. The *web agent* receives an HTTP request from the user during the login and logout processes, forwarding it to the *authentication agent*.

Figure 6.3 shows an example of authentication using a UML sequence diagram. The authentication process starts when the user fulfill her credentials (id and password) using a web form, followed by the submission of the request to the *web agent*. Next, the *web agent* forwards the request to the *authentication agent*. Note that the *authentication agent* does not have the skill to communicate with the *BPM enactment system*, so it sends the request the authentication to the *BPM system agent*. Having a valid credential, the *BPM system agent* answers with a session token, that will be used by the user during the BPM

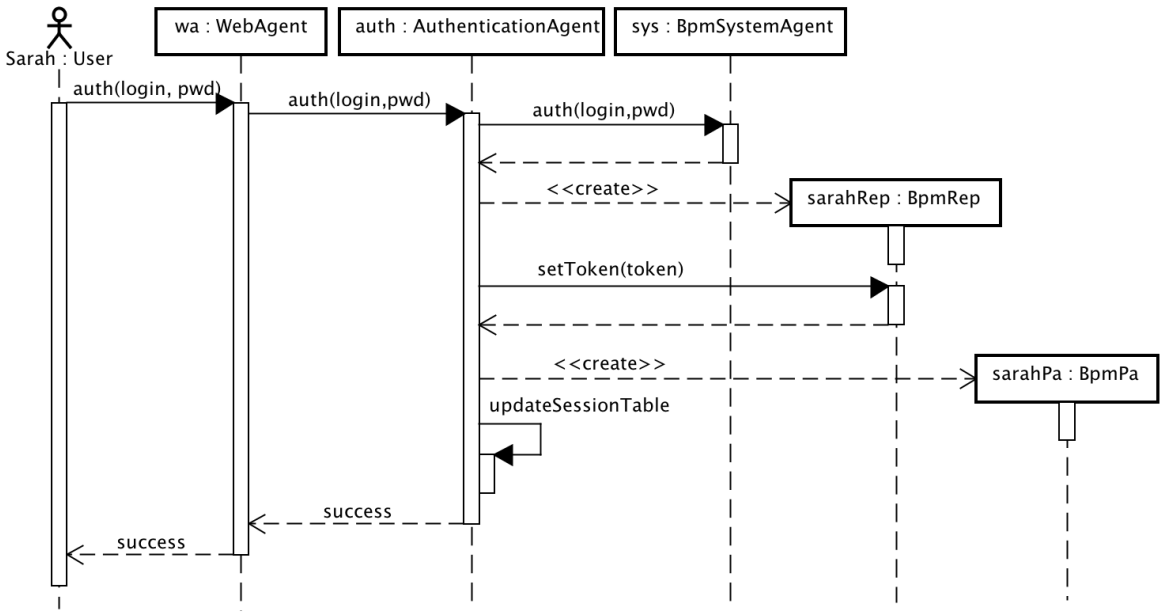


FIGURE 6.3 An example of authentication

enactment system usage. Next, the agent creates two dedicated agents to handle the subsequent requests from the user : the *BPM representative* and the *personal assistant*. Both the authentication token and the user profile are *injected* into the *BPM representative agent*, used in subsequent requests.
Note that the authentication process may vary from system to system. The authentication agent has been designed to be independent of the enactment engine, so it forwards the final authentication request to the *BPM system agent*. The latter has the skills of transformation, marshaling and unmarshaling messages, according to the implementation of the enactment engine.

A note on the chosen BPM engine. For our purposes we selected the Bonita BPM engine. The authentication agent has been designed to be scalable. This agent is stateless, which means that it does not store context information. Thus, if the number of users increases substantially, it is easy to implement a pool of sparse agents to attend concurrent requests. However for this first version of our prototype, we provided only one authentication agent. Since Bonita runs on a JEE web server, the authentication is made by sending an HTTP message to the login endpoint, as can be seen in the following example :

```
POST http://samos.gi.utc:8080/bonita/loginservice HTTP/1.1
username=walter.bates&password=bpm&redirect=false
```

If the user credentials are valid, an HTTP response is sent to the caller containing a header, more accurately a cookie called *JSESSIONID*, as can be seen below :

```
HTTP/1.1 200 OK
Cookie: JSESSIONID=75B64F4991EBD57746B8AAE58DDC7A8B
```

The standard installation of Bonita is configured to invalidate inactive sessions that reach 30 minutes of inactivity. In such cases, the system redirects the user to the login page to perform a new authentication.

Personal Assistant

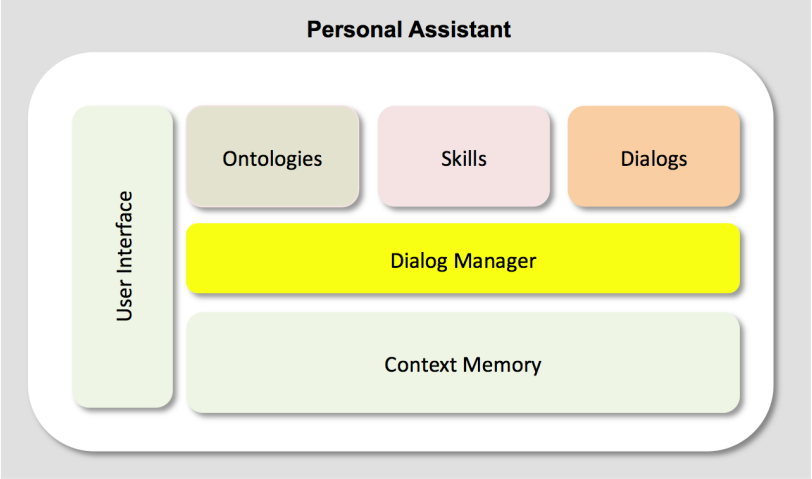


FIGURE 6.4 The personal assistant architecture

As mentioned earlier, each user has his own personal assistant, created by the authentication agent during login. Figure 6.4 presents the personal assistant and its components. This agent has full access to the domain ontology, the dialog ontology, and the action taxonomy. It has a set of skills for triggering business process actions (start a process, start a task), A standard dialog manager that controls the context memory and uses predefined dialogs (i.e., conversation graphs) for the selection and the enactment of processes. These skills are triggered from a particular user interface developed for the web environment.

User Interface

As can be seen in Figure 6.5, the user interface is purposefully simple, having only two panels. The left panel contains the traditional dialog history between the user and the assistant. The right panel has multiple functions, and its content may vary depending on the context.

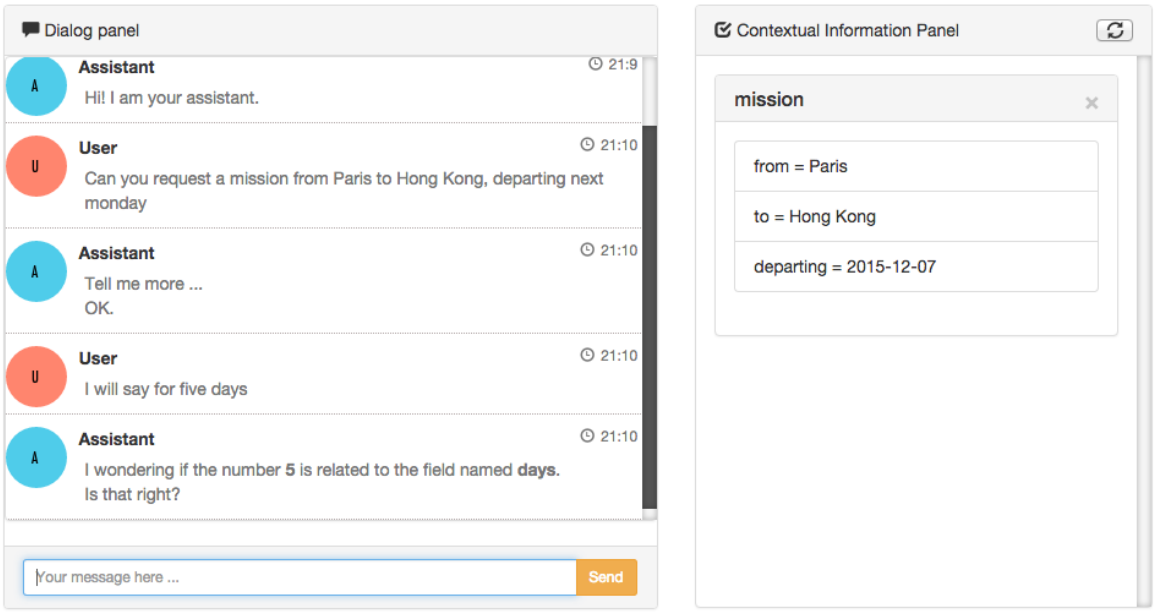


FIGURE 6.5 Conversational Interface

The first role of the panel is to display extracted information. When the user makes an assertion, a question or a directive sentence, the *information extraction* process collects the information and stores it into the dialog context memory in the form of individuals. The dialog manager listens for changes in the dialog context memory and displays new or modified information to the user. The user in turn, can see the information in real time, make corrections or even discard the information provided at any time.
This approach can be seen as a *demonstration grounding act*. As described by Clark and Schaeffer [24], positive evidence of understanding can be expressed by demonstrating the evidence, by reformulating the speaker sentence. Displaying the individual *as-is* to the master produces an implicit feedback, without the need to ask for confirmations that could bother the user. In the example shown in Figure 6.5, the user starts by asking the assistant to request a mission from Paris to Hong Kong and the interpretation process is executed. The dialog manager detects a change in the context memory and displays the individual of type *mission* in the left-panel. The user continues the dialog, making an assertion, stating that he will stay five days. Note that sometimes the assistant must ask clarifying questions when the semantic interpreter is not sure whether a value belongs or not to a property. In this particular case, it asks the user to confirm the number of days.
The second function of the panel is to provide information about the pending tasks of its master. Just to recap, BPM systems have different strategies for task allocation. According to Russel [83], there are three distinct groups of resource allocation : (i) creation patterns,

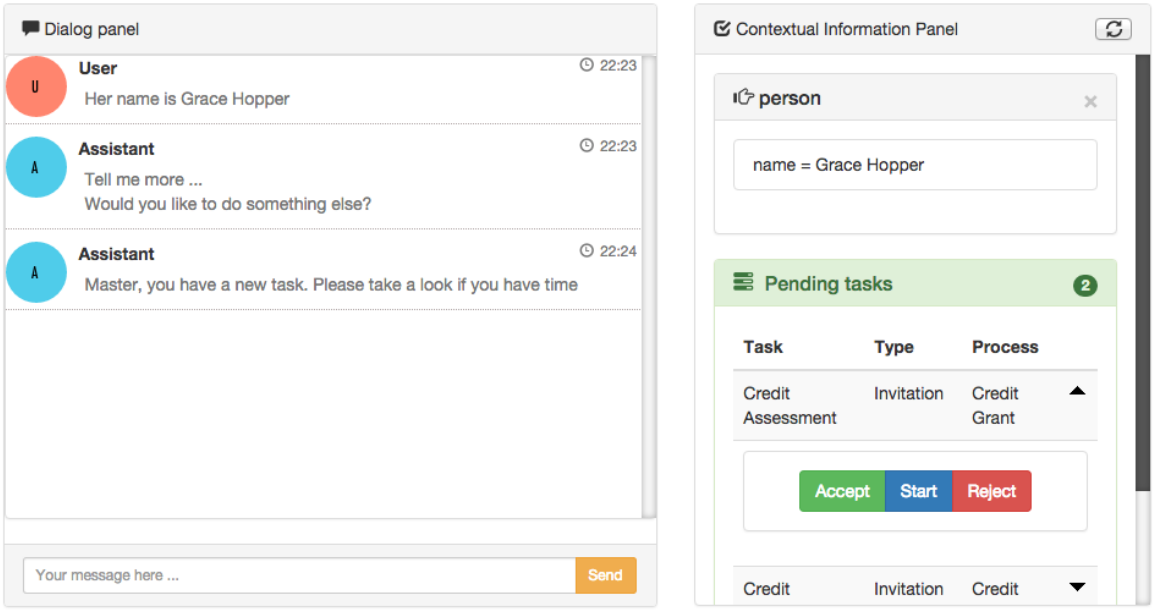


FIGURE 6.6 Task Information

which allocates people at design time, (ii) push patterns, for which new items are proactively offered to workers by the BPM system, and (iii) pull patterns, for which workers take the initiative in committing to and undertaking the work item. In this first version of the prototype, we detect all types of allocations made by the BPM system, whether the assignment was made at design time or runtime. It includes variations of create patterns and push patterns. Other variations of the pull pattern have not been implemented yet and are reserved for future work.

In Figure 6.6, the assistant alerts its master that a new task has arrived. Note that an internal window appears in the right panel, containing details about tasks. The master can (i) accept the task so that the task will become an assigned pending task ; (ii) start, so the task is allocated and immediately started ; or (iii) refuse the offer.

The third function of the panel is to provide information about the search in the process space. Each time the master gives new information to the assistant, it sends a request to the BPM system that in turn, answers its master with a set of advertised services. Figure 6.7 shows an example : The user asks the assistant for an investment with a return on investment greater than 8%. The assistant answers and adds a set of services in the left panel, allowing the user to get more information and trigger each process on demand.

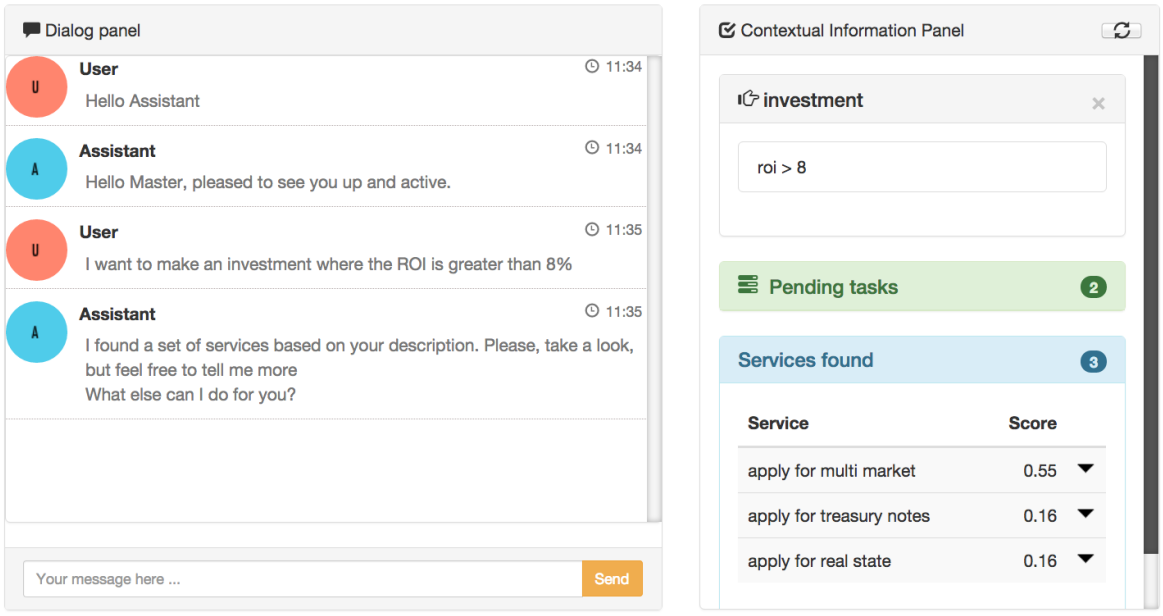


FIGURE 6.7 Searching the process space

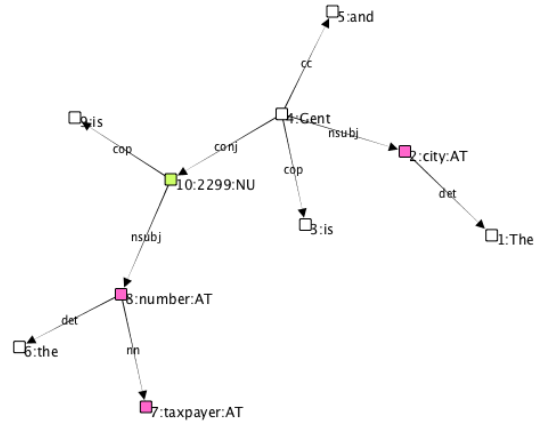
Natural Language Module

We have implemented a natural language module that is responsible for the syntactic and semantic annotation of sentences and information extraction. This module was developed in Java and uses the Stanford Core NLP framework. The function have been exposed as a REST endpoint. The module is accessed by the personal assistant during the conversation. Just to summarize, the syntactic annotation is responsible for generating a dependency tree. This tree is then processed by an algorithm that simplifies this tree, until all nodes are sources of information extraction. The information extraction procedure traverses all remaining nodes and builds a representation used to update the dialog context memory. Figure 6.8 presents an example of the tree simplification process for the phrase “The city is Gent and the taxpayer number is 2299”. The figure at the right-hand side is the original dependency tree, and the figure at the left-hand side is the simplified tree. The algorithm, described in Chapter 5 use a set of handcrafted rules that are used to merge, add and discard nodes until the tree is stable.

Memory Management

The memory management module is responsible for updating the information extracted during the interpretation of sentences into memory. It uses MOSS to manage all domain ontology individuals.

FIGURE 6.8 Distribution of respondents by sector (a), and size (b)



(a)

(b)

Dialog manager

The dialog manager is responsible for managing the dialog flow and maintaining a representation of the dialog state for decision-making purposes. As detailed in Chapter 5, this component has a predefined set of dialogs used for taking notes of user sentences, querying process, controlling the enactment of processes and managing parallel conversations.

BPM user representative

The BPM user representative, as the name suggests, is a service agent that represents the user into the enactment engine. Both the user representative and the personal assistant are

created during the authentication. It stores the session token that will be used during the dialog.

BPM system agent

The BPM system agent is the mediator between the multi-agent system and the enactment engine. It has a standard agent written according to the specification provided in Chapter 4. This implementation contains the following elements :

- A BPM ontology : It is a blueprint on how business processes should be supplemented with semantic information. More precisely, how input and output parameters are linked with the domain ontology and how preconditions and effects are described (concept level) ;
- A BPM knowledge base : it is a set of individuals of the BPM ontology, supplementing each business process with semantic information (instance level) ;
- A query mechanism : It is a skill used to query business processes based on notes taken by the personal assistant ;
- A set of abstract skills : They are used as a source of communication between the multi-agent system and the enactment engine to manage processes.
- A set of concrete event-based skills : They are used to receive notifications of events from the BPM enactment engine. These skills can be seen as callback functions.

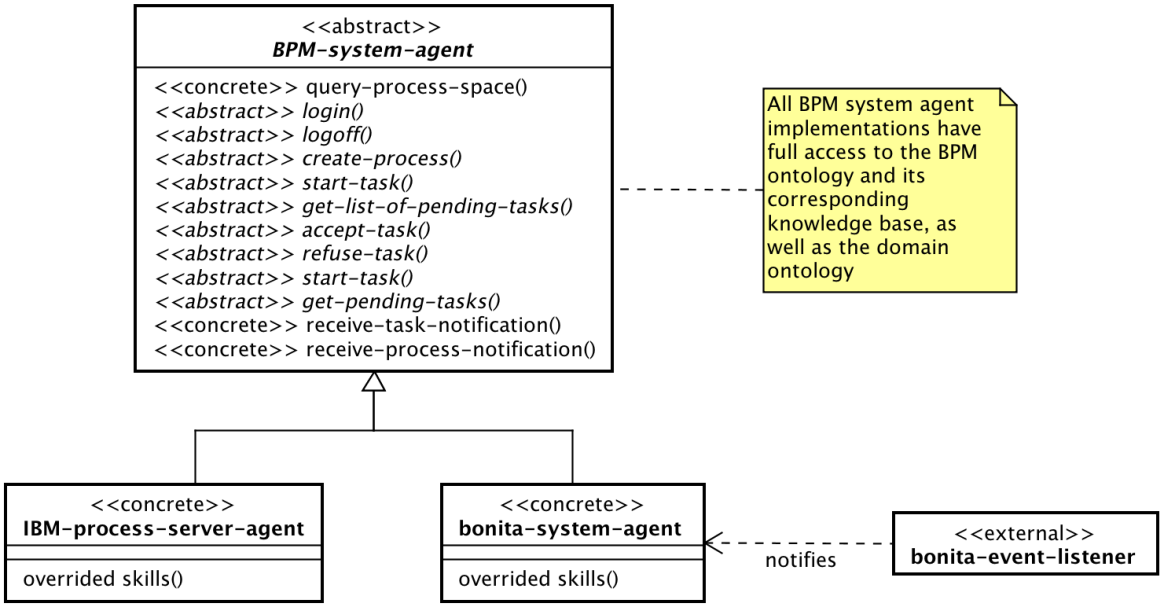


FIGURE 6.9 The class diagram of the BPM system agent

Figure 6.9 presents a UML class diagram to illustrate the BPM system agent and its properties. This agent has full access to the domain ontology, to the BPM ontology as well as its corresponding knowledge base. It also has a concrete skill called *query-process-space*, used to query processes based on a set of notes taken by the personal assistant. We refer to notes taken as a set of individuals of the domain ontology created during the conversation between the master and the assistant. This skill returns a set of pairs of business process and their corresponding scores.
Note that the BPM system agent is represented as an abstract class. Due to the lack of standards for BPM enactment engines, the communication varies from implementation to implementation. As a consequence, each enactment engine must have its own BPM system agent adapted to its communication standards. As can be seen in the diagram, we have designed a set of abstract skills used for the authentication and the management of process instances. Thus, all concrete classes that inherit from the base class must provide a concrete implementation of these skills. For example, the Bonita BPM engine allows three different ways of integration : It can be locally by using a Java library or remotely via Enterprise Java Beans (EJB) or REST API. Since the core of our functional prototype is written in Lisp, the REST API is the most convenient form of access. Thus, our concrete implementation of the BPM system agent for Bonita transforms all requests internal into HTTP requests. The majority of operations use the JSON format for message interchange. The example below is an HTTP response of the enactment engine after requesting the creation of a process instance.

```
HTTP/1.1 200 OK
Cookie: JSESSIONID=75B64F4991EBD57746B8AAE58DDC7A8B

{
    "processDefinitionId": "6606043060015038213",
    "start": "2015-12-02 20:02:23.859",
    "rootCaseId": "77",
    "id": "77",
    "state": "started",
    "started_by": "23",
    "last_update_date": "2015-12-02 20:02:23.859",
    "startedBySubstitute": "23"
}
```

Needless to say, enactment engines are multi-user and multi-process. As a consequence, this type of system generates several asynchronous events. For instance, a single user starts

a collaborative business process could have one or more participants during the process instance lifecycle. These users should be notified when a task is assigned to them so they can contribute to the execution. They should also be notified when the task finishes or some problem has been found. Thus, a conversational interface for business process must be aware of these events, providing mechanisms to deal with asynchronous events and alert policies, without boring users with alerts during the conversation.
The standard agent has two skills that should be called when an event occurs : The first skill is the *receive-task-notification*, called when an event of task is generated (e.g. an assignment of a task to a user, a validation error, an invitation to perform a task, the finalization of a task). The second skill called *receive-process-notification* has the semantics of the previous one, but for processes. When a process is finished, canceled or an error has been generated, this skill should be executed. These skills are programmed to forward the notification to the agent that represents the target user. These skills solve part of the problem. It still remains an actor that is responsible for handling these events and notifying the agent.
Our implementation uses a functionality of the Bonita BPM engine that allows the registration of event handlers written in Java. The event handler is represented by the *bonita-event-listener* component in the class diagram. It is an external component that has a strong cohesion with the enactment engine. When an event arrives, the listener sends an inform message to the BPM system agent, invoking the skill that correspond to the event.

BPM enactment engine

We have chosen Bonita BPM Community to provide the reference implementation of a BPM enactment engine [18]. Bonita BPM is an open-source business process management framework created in 2001. It started at the ECOO team of the France National Institute for Research in Computer Science (INRIA) [64]. The project has received several contributions from the research and development department of Bull, a French software house. Since 2009, the development of Bonita is supported by a company dedicated to this activity called Bonitasoft [1]. The framework comprises the following modules :

- A graphic modeling tool based on the BPMN 2.0 standard. We used this modeling tool to design our processes, define flow control elements, data elements and resource allocation policies.
- A portal that allows users and administrators manage organization information such as workers, roles as well as business process information (instances, tasks and so on).

1. http ://www.bonitasoft.com

— An engine that allows the enactment of business processes. We use this engine to access organization information (e.g. roles and users) manage process instances and tasks.

The BPM engine is an essential component of the framework. It executes processes, handling actions related to tasks, such as database access, and housekeeping operations such as logging. The engine is compatible with any Java Enterprise Edition (JEE) server. For our prototype, the engine runs remotely on an Apache Tomcat server.

6.2 Experimental validations

In order to validate the proposed approach of business process enactment using personal assistants, we implemented a functional prototype. The chosen scenario for the prototype was the investment application introduced in Section 4.7. In this scenario, a customer wants to invest an amount of money but he or she is not sure about what is the most suitable type of investment, among a list of different options. The PA must help its master, taking notes of relevant information like risk level, amount to invest and expectations on the return on investment, ranking the options according to a composite score. Then, the user triggers the business process with the highest score, been notified by the PA when a task arrives. During the execution of the task, the PA assists his master fulfilling information and reusing previous data whenever possible.

6.2.1 The investment scenario

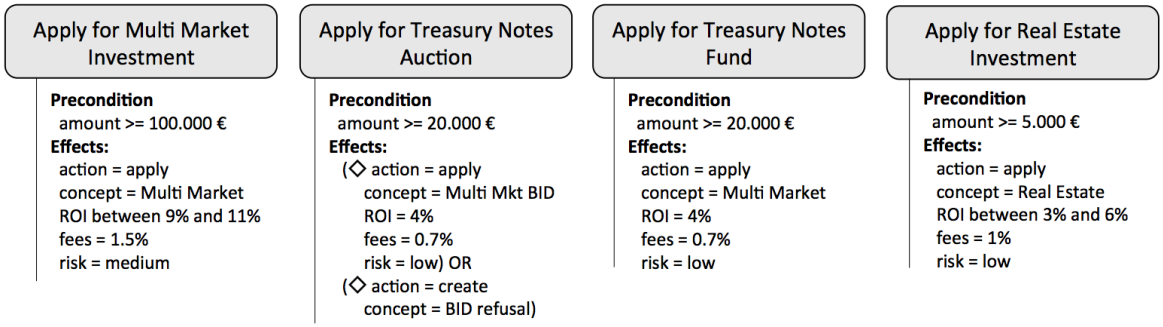


FIGURE 6.10 Business processes used for investment applications and their preconditions and effects

The scenario is composed of a set of banking business processes, among which four types of investment applications, as can be seen in Figure 6.10. Besides their similarity in terms of preconditions and effects, each business process is an independent unit of work, making

integration with different information systems (multi market funds, treasury notes bidding, real estate).
Note that all effects are deterministic, except the *Treasury Notes Auction* that is non-deterministic. That is to say, this process applies for a special type of investment, where customers make a bid in order to buy notes. The success of the investment depends on the availability of the treasury notes. Thus, customers may have a valid investment in the best case or a bid refusal in the worst case, as can be seen in the effect expression using the diamond symbol ($\Diamond$).

6.2.2 Evaluation

We have conducted an evaluation to better understand the performance of our approach. More precisely, the central question investigated in this experiment is the following : "Does the proposed personal assistant and its conversational interface can improve the efficiency of the selection and the enactment of business processes ?" The evaluation covers two major parts : one that evaluates the performance of the business process enactment and an other one that evaluates the performance of the dialog manager.
In order to evaluate the business process perspective, we have built a traditional form-based enactment application containing exactly the same set of business processes present in PA4Biz. That is to say, business process models are shared by both PA4Biz and the form-based application without any changes. Using the counterpart application, evaluators are able to access the set of processes and perform queries using a keyword-based search, trigger processes and perform tasks using traditional form-based interfaces. User satisfaction and the overall time to execute each scenario are compared.
For the dialog manager perspective, there are standard approaches of evaluation like the PARADISE framework of Walker et al. [108] and the three-tiered model of Stibler and Denny [95]. PARADISE uses a weighted function to combine a set of individual measures like task success and user satisfaction into an unified measure. The three-tiered approach of Stibler and Denny evaluates dialog systems at three levels of abstraction namely *user satisfaction*, *task success* and *component performance*. Both approaches allow the combination of subjective measures (e.g. user opinion) and objective measures (e.g. task success and effort), producing a subject-independent assessment (Dybkjær and Bernsen [34]).
In order to evaluate the dialog, we have chosen a set of performance measurements collected while using the conversational interface (see Table 6.1). A similar principle of the PARADISE framework has been followed : the measurements of performance are used by a weighted function in order to produce a unified measure, useful for further

TABLE 6.1 Quantitative and qualitative measures

Type	Measure
Task success	Confusion matrix
Component performance	Amount of time to perform the task
	Dialog length
	Number of misinterpreted sentences
User opinion	User satisfaction (a score ranging from 1 to 6)

comparisons. The first measure is the task success obtained from a confusion matrix. If the personal assistant guided the user to the right business process, then it is considered a true positive. Conversely, if the PA guided the user to the wrong business process, then it is considered a false positive. The second measure is related to the dialog manager component, computing the overall time to execute a session, the dialog length (i.e. number of sentences sent by the user during the session) and the number of misinterpreted sentences per session. Finally, the user opinion is collected at the end of the session.

Evaluation setup

In the provided scenario, we consider customer banking services, and our target user is a typical customer of a banking system, interested in investing some amount of money. To reproduce a real environment, we have developed a small set of banking services, as can be seen in Table 6.2.

TABLE 6.2 Set of available business processes for both systems

Id	Business Process Name
p_1	Apply for multi market investment
p_2	Apply for treasury notes investment
p_3	Apply for real estate investment
p_4	Participate in a treasury notes auction
p_5	Get the account balance
p_6	Change the investment profile
p_7	Apply for credit
p_8	Apply for credit limit increase
p_9	Apply for a health insurance
p_{10}	Apply for a property insurance
p_{11}	Report a lost or stolen credit card

Three different investor profiles have been created, each one leading to a target business process, as can be seen in Table 6.3. The target business process can be reached by using

the expectations of the customer (e.g. risk awareness, ROI expectation) and the constraints imposed by the process (e.g. minimum amount to invest). For instance, the *Multi Market Investment* is compatible with the profile 1, having a ROI between 9% and 11%, medium risk and minimum amount of investment greater or equal than 100,000.00 €. The *Real Estate* investment is compatible with the profile 3, having low risk, fees equals to 0,7% and minimum amount to invest greater or equal than 5,000 €.

TABLE 6.3 Evaluation setup : Formation of profiles and their corresponding information

Profile	**Description**	**Target Business Process**
1	Aims at investing 50% of the balance (200,000.00 €)	p_1 - Multi market
	Return on investment (ROI) must be greater than 9%	
	Investment risk profile is medium	
2	Aims at investing 50% of the balance (50,000.00 €) €	p_2 - Treasury notes
	Investment risk profile is low	
	Fees must be lower than 1%	
3	Risk is low	p_3 - Real estate
	Aims at investing 50% of the balance (16,000.00 €)	
	Investment risk profile is low €	
	Fees must be lower than 1%	

A group of 12 evaluators have participated in this experiment. Each evaluator has access to all profiles and instructions on how to use the target system (Appendix **??**). We have intentionally omitted the balance information, so evaluators must find the appropriate process to get this information. The first group of 6 participants have been invited to use the conversational interface PA4Biz and the remaining group have been invited to use counterpart application. Both groups have the same objective, that is apply for a specific type of investment. Table 6.4 presents the set of expected actions, followed by how the action is accomplished using each system.
As stated before, we consider the evaluation in two different parts : the performance of the business process enactment and the performance of the conversational interface. Next sections presents the evaluation results for the business process enactment perspective and the conversational interface perspective.

Evaluation results for the business process enactment

The business process enactment performance has the objective of measuring how well the system performs using a traditional enactment interface as a baseline. The metrics used are

TABLE 6.4 Actions executed by evaluators for each profile

Action	**How the action is performed**	
	PA4Biz	**Enactment Application**
Perform the authentication	Standard form using login and password	Standard form using login and password
Get the account balance	By describing needs using natural language sentences and triggering the most appropriate process	By navigating through the list of business processes, matching the process description with their needs
Select and trigger the investment application	By describing needs using natural language sentences and triggering the most appropriate process	By navigating through the list of business processes, matching the process description with their needs
Perform tasks	By answering the personal assistant using natural language sentences	By filling traditional forms

the time spent to select, trigger and enact the business process, the false positive ratio and an indicator of satisfaction assigned by the evaluator for each round.
We start by showing some measures related to the business process enactment performance in Table 6.5.

TABLE 6.5 Summary of collected measures that are comparable to the form-based application

Measure	**System**	
	Form-based enactment	**PA4Biz**
Average time to find the business process	01 :59 (± 00 :54)	01 :31 (± 00 :49)
Average time to execute tasks	03 :34 (± 01 :39)	02 :58 (± 01 :18)
Average session time	05 :33 (± 02 :01)	04 :28 (± 01 :45)
Average user satisfaction (from 1 to 6)	4,4 (± 0.7)	5,2 (± 0.4)
False positive ratio	$\frac{4}{18} \rightsquigarrow 22\%$	$\frac{0}{18} \rightsquigarrow 0$

The first line presents the average time to find the target business process. We have noticed a slight improvement in both the time spent on finding the business process (1'59" using the counterpart system versus 1'31" using PA4Biz), and in the standard deviation (54" using the counterpart system versus 49" using PA4Biz). It represents a reduction of 24% of the time spent and 10% of standard deviation for the cases that used PA4Biz.
The same improvement occurred for the time to execute tasks (3'34" using the form-based system versus 2'58" using PA4Biz) as well as the standard deviation (1'39" using the form-based system versus 1'18" using PA4Biz). This represents a reduction of 17% of

time and 21% of variation when using PA4Biz. As a result, PA4Biz shortened the the total amount of time to perform the application in 18%, and reducied the standard deviation of 15%.

The counterpart system presented a false positive ratio of 22%. That is to say, in 4 out of 18 sessions, users selected and triggered the wrong business process, (i.e a business process that is not entirely adapted to the profile, but at the same time fulfilling the business process precondition). For instance, one of the users has selected process p_3 (real estate) instead of process p_2 (treasury notes). Since users must find processes using a keyword search and also interpret the business process description, it is easy to mistakenly choose the wrong process.

User satisfaction was obtained from a single question to the tester after the execution of a session, asking him or her to rate the experience, providing a numeric value ranging from 1 to 6, where 1 represents the the worst experience and 6 represents an excellent experience. The average user satisfaction of PA4Biz was 18% greater than the form-based system as well as presented a lower standard deviation (0,4 versus 0,7).

Overall, the evaluation shows a slight reduction of the average time to find and trigger processes. More importantly, the prototype prevented users from selecting the wrong process. However, we have identified several cases that the counterpart application performed better than the conversational interface in terms of time to complete the session. All of these cases belong to the second and the third session. We argue that the learning curve of this type of application is more evident, since the level of repetitiveness and predictability of form-based applications is high. For instance, user 1 spent 3'44" to provide information in the first session, followed by 2'43" in the second session and 1'33" in the third session. It represents a reduction of 58% of time from the first to the third session. The same improvement of learning curve was not observed in sessions using PA4Biz. That is to say, there is a reduction of time from session to session, but much more discrete than the form-based application. Thus, we believe that a conversational interface could not be considered as a substitute for a form-based interface, but a new way of enacting business processes. A conversational interface is well suited for users that do not have time or do not want to memorize sequences of steps to use the system. It is is also better suited when users have a variety of available services and different possibilities for doing the same thing. In these cases, a personal assistant could improve the overall performance by guiding his master during the enactment of the whole process. Conversely, the interaction with an assistant may slow down the use for repetitive and daily tasks. Thus, we believe that a form-based and traditional interface is better suited for this type of task.

Evaluation results for the conversational interface

In this section we present some measures to evaluate how well the system recognized the set of sentences, looking more closely at every turn taking. That is to say, we check if the system response was appropriate, and if it was not, which aspect of the dialog manager was mainly responsible for the problem.

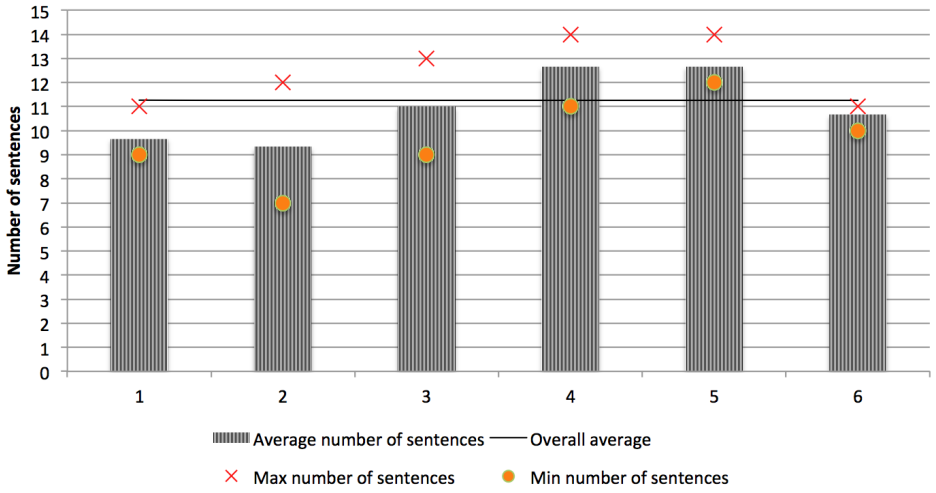


FIGURE 6.11 Average dialog length per user

Figure 6.11 shows the average dialog length (counting only the number of sentences produced by the user) for each user. Over all 18 sessions, users took 11 sentence on average to apply for the investment. The variation of the number of sentences depends on (i) the successful recognition of concepts in a sentence, (ii) the amount of information existing on each sentence, (iii) the number of clarifying questions asked by the assistant to the user.

We start by taking the example of user 2 that has the best case with a dialog length of 5 sentences. This user started the conversation with the sentence “I would like to invest the amount of 100000 euros with a ROI greater than 9% and fees less than 1%, medium risk is acceptable.” This sentence produced sufficient information to select the most suitable service, as well as provide information for the enactment. Subsequent requests or enactment will benefit from the information provided a priori. Another example taken from user 1 : During the enactment, this user informed the city, the country and taxpayer number in only one sentence “The city is Amsterdam, Netherlands, taxpayer number is 2233.” Since the concept of country was not present in the sentence, the system requested a confirmation if “Netherlands” was the country, followed by the confirmation of the user. In the majority of cases during their preliminary usage, users preferred to produce fine-grained sentences like the sequence “I want to invest an amount of money. I want fees

lower than 1%. I have a low risk profile." Other users did not notice that temporal expressions like "two days from now" and "next week" could be used in the sentence to specify the date of debit in the current account, resulting in a longer dialog length, and consequently, a longer time to finish the session. This shows that some users, perhaps able to gain more experience through some sessions with PA4Biz, could interact with the system in more flexible and effective ways than others.
The worst case needed 15 sentences to complete the session. The reason for this long dialog was that the dialog manager did not recognize the expression "I want to invest 50% of my current balance." The objective of the sentence is clear and unambiguous. However, to reach the objective, the system must build a plan, executing first the business process that retrieve the balance, and then executing the investment business process, using the balance as input. This is a typical problem of service composition that is out of the scope in this first version of our functional prototype, although it is part of our future work. As an effect, the dialogue manager interpreted the fragment *50* as a potential candidate to fulfill the *investment* individual. The dialog manager traversed all potential attributes of the *investment* individual and asked to the user, for example "I guess that 50 refers to the taxpayer number. Is that right ?" The user had to repeatedly refuse 6 suggestions and, needles to say the user experience was affected for this session. We changed the system in order to limit the number of suggestions, allowing the user to rephrase when the extraction was not possible.
Another example of misinterpretation occurred with user 4. During the conversation, he produced the sentence "balance" to the assistant. The assistant assigned the *assertive* dialog act to the sentence. When the user makes an assertion, the assistant only take notes and continue listening. In this particular case, the assistant took notes but did not start any matchmaking to find the process to get the balance. Then, the user tried again, changing the sentence to "What is my balance ?", which was properly labelled as a interrogative dialog act. As a consequence, the matchmaking was executed and the assistant suggested to proper process to start. In order to avoid such situation, we have changed the dialog manager, so it triggers the semantic match even when the user produce an assertion at the top-level dialog.

TABLE 6.6 Summary of collected measures related to the conversational interface

Measure	Value
Dialog length	11,3
Average number of words per sentence	6,8
Average number of non-recognized sentences	0,27

We finish by presenting the summary of measures in Table 6.6. The table contains the actual dialog length (i.e. the number of sentences sent by the user), the average number of words per sentence and the average number of non-recognized sentence per session. This set of measures can be used to validate the performance of the conversational interface and create a baseline for future improvements.

6.3 Discussion

The present chapter has described the architecture of our functional prototype, and our experimental evaluation in the field of e-banking systems. The outcome of the experiment corroborates one of the central claims of this thesis – namely, that a personal assistant could improve the enactment of business processes, more specifically for querying the process space using semantic descriptors (using the formalism of business process description in this thesis).
The presented approach could represent an alternative to the traditional interface between humans and enactment engines. Existing business processes can benefit from this approach with minimum effort. As shown by the investment example, the whole set of business processes was shared by the traditional application and our system without any changes in their models. Of course, the effort for building the domain ontology and the semantic annotation of processes must be considered.
Of course, there are limits to the approach. The semantic analyzer uses only the domain ontology as a source for the identification of potential values for information extraction and is unable to extract information from sentences that lacks ontological cues. This restriction could be canceled by learning from previous experiences.
Another limitation of the approach is when user tasks require the fulfillment of lots of information. Even if the personal assistant dynamically presents its notes during the fulfillment, allowing corrections at any time, a traditional form is easier to fill. An alternative for this limitation could be a hybrid interface, giving the PA the ability to build dynamically input forms, assisting the user for filling them. Another alternative is to use a vocal input, either through the native operation system speech-to-text mechanism or better using a third party software. The main problem of the speech-to-text recognition systems is that they do not recognize everything being said, which leads to noisy inputs.
The system presented here is not a substitute for business process enactment interfaces, but just supports the implementation of dialogs for existing business processes without structural changes. Thus, the quality of the overall process will depend on the level of detail used during the annotation of business process and how well the domain ontology

has been modeled. We are quite willing to accept errors and quid pro quo in the dialog, as long as it results in enough information to find business processes or provide input parameters to tasks. If the process is not what the user wants, then the user will ignore it, trigger the next process or provide information differently.
The approach presented here is both simple and efficient, in particular when the effort to adapt the interface of business processes to be used by personal assistants is prohibitive in both technical and financial terms. Furthermore, the approach can be extended and refined to improve the accuracy of the dialog manager.

Chapitre 7

Conclusion

The research in this thesis concerns dialog management in personal assistant applications that allow the selection and enactment of business processes through a written natural language dialog. The developed prototype, PA4Biz, is a multi-agent system, in which dialog management is performed by a personal assistant, acting as the central component for coordinating other agents, as well as maintaining a coherent dialog with the user. This agent-based approach in an attempt to provide a conversational interface for the selection and enactment of business processes, without the need to re-engineer the whole set of existing business process models of an organization every time there are changes.
The contribution of our work is twofold, covering two perspectives : (i) the dialog management perspective ; and (ii) the characterization of processes perspective.
For dialog management (i), most existing methods are suitable for specialized tasks, requiring a considerable effort of development (e.g. dialog management, user interface, and domain model). The originality of our approach is a dialog model that is scalable to different business process models, consequently to different domains. The dialog manager relies on domain and business process ontologies, yet necessitating a minimal effort of adaptation on domain ontologies to enable the interaction.
For characterization of business processes (ii), we proposed a formalism to describe the capabilities of business processes, allowing the definition of deterministic and non-deterministic effects. This feature is quite useful for describing human tasks and other non-deterministic scenarios, sometimes difficult to encode using declarative rules. Based on machine-readable descriptors, we have developed both a safe approximation of the whole business process based on tasks, and a querying mechanism based on preconditions and effects of processes. Note that business processes have an independent life cycle, controlled by a workflow engine that performs its orchestration. Our dialog manager can

handle both user interaction and asynchronous events of the engine without disrupting the conversation.
Our approach makes use of a cognitive agent architecture for the dialog management and the integration with a business process enactment engine. To identify user needs and select the most suitable business process, we make use of the concepts of speech acts and of the semantic interpretation of sentences. Our dialog manager can be seen as a hybrid solution combining the following techniques : (i) Information state, modeling a library of dialogs as conversation graphs ; (ii) script-based, providing a set of rules for each conversation state ; (iii) frame-based, modeling tasks and elements as frames ; and (iv) task-oriented, linking each dialog with a library of tasks. The dialog control flow is derived automatically as the result of the interpretation of the conversational graph. The conversational context, the domain-specific knowledge, and the user model are shared by the personal assistant and the user representative agent. Of course, this approach requires an environment restricted to a professional context or a focused context in which domain ontologies play a key role to provide a common vocabulary.
The potential of our agent-based approach has been demonstrated through the realization of our functional prototype and the evaluation results, which shows a high task completion rate. We have developed a functional prototype implemented using the OMAS platform, running in the Allegro Common Lisp environment. We have carried out the evaluation of PA4Biz with the intention of evaluating the performance for the selection and the enactment of business processes, compared with a traditional enactment interface. Although the recognition of some sentences was not perfect, the system was able to execute the enactment of all processes, been more assertive than the traditional interface for the selection problem. It also prevented users from filling redundant information, by using previous notes taken during the conversation. Additionally, most users were satisfied with the overall performance of the system and indicated their willingness to use such a system in future.

7.1 Future research

There are many areas of improvement for the current PA4Biz system that might be undertaken as future work.
We only use the business process model as a learning source. Our personal assistant, the dialog manager component, can be more adaptable to each user, choosing different strategies for suggesting services and filling parameters, according to the user's

preferences. Also, we can learn some language patterns employed by users, such as the use of expressions, abbreviations, and references (referring to people using nicknames, etc.).
As mentioned in Chapter 5, we have perceived that we need to give an additional skill to our agent so that it can perceive the environment and the context and decide what is the best interface to provide. For example, a form-based interface is better suited when users need to fill lots of information. The turn-taking characteristic of the dialog-based approach may slow down the whole process. There are also other situations in which users are unable to interact with a web interface. A vocal interface could be better suited in this case.
Finally, our work also suggests possibilities for future research in dialog management. Current trends show an increased use of pervasive devices as first-class citizens in business processes. These devices are used both for providing, but also receiving information produced during the enactment (e.g. fitness monitoring tools and vehicle sensors, and so on.). Both written and spoken dialog interaction would be useful for applications that deal with these devices to improve usability. The dialog model of PA4Biz could be reused and extended for use in these domains. Our model must be enhanced to allow multimodal support. The multimodal dialog management using our agent-based approach could be another future research problem.
Moreover, we support the integration with only one enactment engine. Supporting the integration with a "federation" of enactment engines could also be an interesting research area that we did not consider.

Bibliographie

[1] Akbik, A. and Alexander, L. (2012). KrakeN : N-ary facts in open information extraction. In *AKBC-WEKEX '12 Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction*, pages 52–56.

[2] Akbik, A., Konomi, O., and Melnikov, M. (2013). Propminer : A Workflow for Interactive Information Extraction and Exploration using Dependency Trees. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics : System Demonstrations*, pages 157–162.

[3] Allegro (2013). AllegroServe - A Web Application Server. Available at http://franz.com/support/documentation/current/doc/aserve/aserve.html.

[4] Allegro (2015). Allegro Webactions. Available at http://franz.com/support/documentation/current/doc/webactions.html.

[5] Allen, J., Byron, D., Dzikovska, M., Ferguson, G., Galescu, L., and Stent, A. (2000). An architecture for a generic dialogue shell. *Natural Language Engineering*, 6(3&4) :213–228.

[6] Allen, J. F. and Perrault, C. R. (1980). Artificial Intelligence. *Analyzing intention in utterances*, 15 :143–178.

[7] Atrash, A., Abel, M.-H., Moulin, C., Darène, N., Huet, F., and Bruaux, S. (2015). Note-taking as a main feature in a social networking platform for small and medium sized enterprises. *Computers in Human Behaviour*, 51(B) :705–714.

[8] Barthès, J.-P. (2011a). OMAS - A Flexible Multi-agent Environment for CSCWD. *Future Generation Computer Systems*, 27(1) :78–87.

[9] Barthès, J.-P. and Moulin, C. (2014). MOSS : A Formalism for Ontologies Including Multilingual Features. *Knowledge and Systems Engineering*, 245 :95–107.

[10] Barthès, J.-P. A. (2011b). Exchanging Information among Cognitive Agents in Collaborative Environments. In *IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 1908–1913, Anchorage, AK. IEEE.

[11] Barthès, J.-P. A. (2013). Improving Human-Agent Communication Using Linguistic and Ontological Cues. *International Journal on Electronic Business*, 10(3) :207–231.

[12] Barthès, J.-P. A. and Tacla, C. (2001). Agent-Supported Portals and Knowledge Management in Complex R & D Projects. In *Computer Supported Cooperative Work in Design*, pages 287–292. IEEE.

[13] Becker, J., Brocke, J. V., and de Marco (Eds.), M. (2015). Proceedings of the European Conference on Information Systems. In *Association for Information Systems*, Münster. AIS Electronic Library.

[14] Bettahar, F., Moulin, C., and Barthès, J.-p. (2009). Towards a Semantic Interoperability in an e-Government Application. *Electronic Journal of E-government*, 7(3) :209–226.

[15] Bohnet, B. (2010). Very high accuracy and fast dependency parsing is not a contradiction. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING '10)*, pages 89–97.

[16] Brooks, R. (1986). A robust layered control system for a mobile robot. *IEEE Journal on Robotics and Automation*, 2(1) :14–23.

[17] Brooks, R. (1991). Intelligence without Reason. In *IJCAI*, pages 569–595. Morgan Kaufmann.

[18] Chabanoles, N. and Ozil, P. (2015). Bonita BPM : an open source BPM based application development platform to build adaptable business applications. In *International Conference on Business Process Management*.

[19] Chan-Lam, V. (1979). *Conception et réalisation d'une base de données ensembliste*. PhD thesis, Université de technologie de Compiègne.

[20] Chen, B. and Cheng, H. H. (2010). A review of the applications of agent technology in traffic and transportation systems. *IEEE Transactions on Intelligent Transportation Systems*, 11(2) :485–497.

[21] Chen, K. (2008). *Contribution à la conception de la mémoire d'un agent assistant personnel*. PhD thesis, Université de technologie de Compiègne.

[22] Chen, P. P. (1976). The Entity-Relationship Model : Towards a unified view of Data. *ACM Transactions on Database Systems*, 1 :9–36.

[23] Chinchor, N. (1997). MUC-7 Named Entity Task Definition. Technical report, NIST.

[24] Clark, H. H. and Schaefer, E. F. (1989). Contributing to discourse. *Cognitive Science*, 13(2) :259–294.

[25] Clavel, C., Faur, C., Martin, J.-C., Pesty, S., and Duhaut, D. (2013). Artificial companions with personality and social role. In *CICAC*, pages 87–95.

[26] Clément, L., König, D., Mehta, V., Mueller, R., Rangaswamy, R., Rowley, M., and Trickovic, I. (2010a). Web Services - Human Task (WS-HumanTask). Technical report, OASIS.

[27] Clément, L., König, D., Mehta, V., Mueller, R., Rangaswamy, R., Rowley, M., and Trickovic, I. (2010b). WS-BPEL Extension for People (BPEL4People) Specification. Technical report, OASIS.

[28] Cohen, P. R. and Perrault., C. R. (1979). Elements of a plan-based theory of speech acts. *Cognitive Science*, 3(3) :177–212.

[29] Corro, L. D. and Gemulla, R. (2013). ClausIE : clause-based open information extraction. In *WWW '13 Proceedings of the 22nd international conference on World Wide Web*, pages 355–366.

[30] de Marneffe, M.-C., Dozat, T., Silveira, N., Haverinen, K., Ginter, F., Nivre, J., and Manning, C. D. (2014). Universal Stanford Dependencies : A cross-linguistic typology. In *International Conference on Language Resources and Evaluation (LREC'14)*.

[31] Derguech, W. and Bhiri, S. (2013). Business Process Model Overview : Determining the Capability of a Process Model Using Ontologies. *Business Information Systems*, 157 :62–74.

[32] D'Inverno, M. and Luck, M. (2004). *Understanding Agent Systems*. Springer, 2nd edition.

[33] Dumas, M., van der Aalst, W. M. P., and ter Hofstede, A. H. M. (2005). *Process-Aware Information Systems : Bridging People and Software through Process Technology*. Wiley.

[34] Dybkjær, L. and Bernsen, N. O. (2001). Usability Evaluation in Spoken Language Dialogue Systems. In *Proceedings of the workshop on Evaluation for Language and Dialogue Systems* -, volume 9, pages 1–10.

[35] Dzikovska, M. O., Allen, J. F., and Swift, M. D. (2003). Integrating linguistic and domain knowledge for spoken dialogue systems in multiple domains. In *Workshop on Knowledge and Reasoning in Practical Dialogue Systems*.

[36] Embley, D. W. (2004). Toward semantic understanding : an approach based on information extraction ontologies. In *ADC '04 Proceedings of the 15th Australasian database conference*, pages 3–12.

[37] Enembreck, F. (2003). *Contribution à la Conception d'Agents Assistants Personnels Adaptifs*. Phd thesis, Université de technologie de Compiègne.

[38] Finkel, J. R., Grenager, T., and Manning, C. (2005). Incorporating non-local information into information extraction systems by Gibbs sampling. In *Proceedings of the 43nd Annual Meeting of the Association for Computational Linguistics (ACL 2005)*, pages 363 – 370.

[39] Flycht-eriksson, A. (2003). Design of Ontologies for Dialogue Interaction and Information Extraction. In *IJCAI*.

[40] Georgakopoulos, D., Hornick, M., and Sheth, A. (1995). An Overview of Workflow Management : From Process Modeling to Workflow Automation Infrastructure. *Distributed and Parallel Databases*, 3 :119–153.

[41] Hammer, M. and Champy, J. (1993). *Reengineering the Corporation : A Manifesto for Business Revolution*. HarperBusiness.

[42] Hazewinkel, M. (2001). Disjunctive normal form. In *Encyclopedia of Mathematics*. Springer.

[43] Heinl, P., Horn, S., Jablonski, S., Neeb, J., Stein, K., and Teschke, M. (1999). A comprehensive approach to flexibility in workflow management systems. In *International Joint Conference on Work Activities Coordination and Collaboration*, pages 79–88.

[44] Hepp, M., Leymann, F., Bussler, C., Domingue, J., Wahler, A., and Fensel, D. (2005a). Semantic business process management : a vision towards using semantic web services for business process management. *IEEE International Conference on eBusiness Engineering ICEBE05*, 5(1) :535–540.

[45] Hepp, M., Leymann, F., Domingue, J., Wahler, A., and Fensel, D. (2005b). Semantic business process management : a vision towards using semantic Web services for business process management. *IEEE International Conference on e-Business Engineering (ICEBE'05)*, pages 1–6.

[46] Hepp, M. and Roman, D. (2007). An Ontology Framework for Semantic Business Process Management. *Wirtschaftsinformatik 2007*, pages 1–18.

[47] Hoque, F., Walsh, L. M., and Mirakaj, D. L. (2011). *The Power of Convergence : Linking Business Strategies and Technology Decisions to Create Sustainable Success*. Amacom.

[48] Jennings, N. R. and Wooldridge, M. J. (2002). *Agent Technology : Foundations, Applications, and Markets*. Springer, 2nd edition.

[49] Jones, A., Kendira, A., Lenne, D., Gidel, T., and Moulin, C. (2011). The TATIN-PIC project : A multi-modal collaborative work environment for preliminary design. In *International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, pages 154–161, Lausanne.

[50] Jones, A., Moulin, C., Barthès, J.-p., Lenne, D., Kendira, A., and Gidel, T. (2012). Personal Assistant Agents and Multi-agent Middleware for CSCW. In *IEEE 16th International Conference on omputer Supported Cooperative Work in Design*, pages 72–79.

[51] Kietz, J.-U., Mädche, A., Maedche, E., and Volz, R. (2000). A Method for Semi-Automatic Ontology Acquisition from a Corporate Intranet. In *EKAW-2000 Workshop "Ontologies and Text"*, pages 2–6.

[52] Kim, G. and Suhh, Y. (2010). Ontology-based semantic matching for business process management. *ACM SIGMIS Database*, 41(4) :98.

[53] Kim, G. and Suhh, Y. (2012). Building Semantic Business Process Space for Agile and Efficient Business Processes Management : Ontology-Based Approach. *Business Enterprise, Process, and Technology Management : Models and Applications*, pages 51–73.

[54] Kiryakov, A., Popov, B., Ognyanoff, D., Manov, D., and Kirilov, M. G. (2004). Semantic annotation, indexing, and retrieval. *Journal of Web Semantics*, 2 :49–79.

[55] Klusch, M., Fries, B., and Sycara, K. (2009). OWLS-MX : A hybrid Semantic Web service matchmaker for OWL-S services. *Web Semantics : Science, Services and Agents on the World Wide Web*, 7(2) :121–133.

[56] Larsen, L. B. and Baekgaard, A. (1994). Rapid Prototyping of a Dialogue System Using a Generic Dialogue Development Platform. In *Proceedings of the Third International Conference on Spoken Language Processing*, pages 919–922.

[57] Larsson, S. and Traum, D. R. (2000). Information state and dialogue management in the TRINDI dialogue move engine toolkit. *Natural Language Engineering*, 6(3&4) :323–340.

[58] Leidner, D. and Ross (Eds.), J. (2015). Proceedings of the International Conference on Information Systems. In *Association for Information Systems*. Association for Information Systems.

[59] Leopold, H. (2013). *Natural Language in Business Process Models : Theoretical Foundations, Techniques and Applications*. Springer, New York, New York, USA.

[60] Leymann, F. (2002). Web services and business process management. *IBM Systems Journal*, 41(2) :1980211.

[61] Lhommet, M., Lourdeaux, D., and Barthès, J.-P. (2012). Foule sentimentale : influence des caract{é}ristiques individuelles sur la contagion {é}motionnelle. *Revue d'Intelligence Artificielle*, 26(3) :281–308.

[62] Li, L. and Horrocks (2003). A software framework for matchmaking based on semantic web technology. In *Proceedings of the 12th international conference on World Wide Web*, pages 331–339.

[63] Lison, P. (2013). *Structured Probabilistic Modelling for Dialogue Management*. PhD thesis, University of Oslo.

[64] Loridan, C. and Valdes, M. (2004). Bonita : A Java 2 platform, Enterprise Edition (J2EE) open source cooperative workflow system. In *Java One*.

[65] Maedche, A., Neumann, G., and Staab, S. (2003). Bootstrapping an Ontology-Based Information Extraction System. *Intelligent Exploration of the Web*, 111 :345–359.

[66] Maes, P. (1994). Agents that Reduce Work and Information Overload. *Communications of the ACM*, 37(7) :30–40.

[67] Maruster, L. and Jorna, R. J. (2005). From data to knowledge : A method for modeling hospital logistic processes. *IEEE Transactions on Information Technology in Biomedicine*, 9(2) :248–255.

[68] Massimo Paolucci, Takahiro Kawamura, Terry R. Payne, K. S. (2002). Semantic Matching of Web Services Capabilities. *Lecture Notes in Computer Science*, 2342 :333–347.

[69] Maynard, D., Li, Y., and Peters, W. (2008). NLP Techniques for Term Extraction and Ontology Population. In *Proceedings of the 2008 conference on Ontology Learning and Population : Bridging the Gap between Text and Knowledge*, pages 107–127.

[70] Maynard, D., Peters, W., and Li, Y. (2006). Metrics for evaluation of ontology-based information extraction. In *international World Wide Web Conference*, volume 179, pages 1–8.

[71] Negroponte, N. (1996). *Being Digital*. Vintage.

[72] OASIS (2012). Reference Architecture Foundation for Service Oriented Architecture. Technical Report December, OASIS.

[73] OMG (2011). Business Process Model and Notation (BPMN). Technical Report January, OMG.

[74] Paraiso, E. C. (2005). *Une Interface Conversationnelle pour une Aide Intelligente*. PhD thesis, Université de technologie de Compiègne.

[75] Paraiso, E. C. and Malucelli, A. (2011). Ontologies Supporting Intelligent Agent-Based Assistance. *Computing and Informatics*, 30 :829–855.

[76] Persson, A. and Stirna (Eds.), J. (2015). International Conference on Advanced Information Systems Engineering. In *Springer*, Stockholm.

[77] Pesic, M. (2008). *Constraint-based workflow management systems : shifting control to users*. Phd thesis, Eindhoven University of Technology.

[78] Porter, M. E. (1985). *Competitive Advantage : Creating and Sustaining Superior Performance*. Free Press.

[79] Ramos, M. (2000). *Structuration et évolution conceptuelles d'un agent assistant personnel dans les domaines techniques*. PhD thesis, Université de technologie de Compiègne.

[80] Ramos, M. P., Tacla, C. A., Sato, G. Y., Paraiso, E. C., and Barthès, J.-P. A. (2011). CSCW in Software Development : Collaboration among Humans and Artificial Agents through Dialogs. *International Journal of Energy, Information and Communication*, 2(4) :31–45.

[81] Ribeiro, J., Carmona, J., Mısır, M., and Sebag, M. (2014). A Recommender System for Process Discovery. *Lecture Notes in Computer Science*, 8659 :67–83.

[82] Rivière, J., Adam, C., and Pesty, S. (2014). Un ACA sincère comme compagnon artificiel. *Revue d'Intelligence Artificielle*, 28(1) :67–99.

[83] Russell, N. and van der Aalst, W. M. P. (2007). Evaluation of the BPEL4People and WS-HumanTask extensions to WS-BPEL 2.0 using the workflow resource patterns. Technical report, BPM Center Report.

[84] Russell, N., van der Aalst, W. M. P., ter Hofstede, A. H. M., and Edmond, D. (2005). Workflow resource patterns : Identification, representation and tool support. *Advanced Information Systems Engineering*, 3520 :216–232.

[85] Russell, S. and Norvig, P. (2009). *Artificial Intelligence : A Modern Approach*. Pearson, 3rd edition.

[86] Sadiq, S., Soffer, P., and Völzer (Eds.), H. (2014). International Conference on Business Process Management. *Lecture Notes in Computer Science*, 8659 :1–434.

[87] Sbodio, M. L. (2009). *Planning Web Agents*. PhD thesis, Université de technologie de Compiègne.

[88] Scalabrin, E. E. and Barthès, J.-P. A. (1993). OSACA : une architecture ouverte d'agents cognitifs independents. In *Journée Nationale du PRC-IA sur les systèmes multi-agents*, Montpellier, France.

[89] Schonenberg, H., Mans, R., Russell, N., Mulyar, N., and van der Aalst, W. M. P. (2008). Process flexibility : A survey of contemporary approaches. *Lecture Notes in Business Information Processing*, 10 :16–30.

[90] Schonenberg, M. H., Mans, R. S., Russell, N. C., Mulyar, N. A., and Van Der Aalst, W. M. P. (2007). Towards a taxonomy of process flexibility. Technical report, BPM Center.

[91] Seneff, S. and Polifroni, J. (2000). Dialogue Management in the Mercury Flight Reservation System. In *Proceedings of the ANLP/NAACL 2000 Workshop on Conversational Systems*, pages 11–16.

[92] Shen, W. and Barthes, J.-P. a. (1996). An Experimental Multi-Agent Environment for Engineering Design. *International Journal of Cooperative Information Systems*, 5(2-3) :131–151.

[93] Smith, A. and Krueger, A. B. (1776). *The Wealth of Nations*. Bantam Classics.

[94] Soffer, P. (2005). On the notion of soft-goals in business process modeling. In *Conference on Advanced Information Systems Engineering.*

[95] Stibler, K., Denny, J., Street, F., and Nj, C. (2001). A three-tiered evaluation approach for interactive spoken dialogue systems. In *Proceedings of the first international conference on Human language technology research*, pages 1–5.

[96] Sugawara, K., Manabe, Y., Shiratori, N., Yaala, S. B., Moulin, C., and Barthès, J.-P. A. (2011). Conversation-based support for requirement definition by a Personal Design Assistant. *IEEE 10th International Conference on Cognitive Informatics and Cognitive Computing (ICCI-CC'11)*, pages 262–267.

[97] Sycara, K. P. (1998). Multiagent Systems. *AI Magazine*, 19(2) :79–92.

[98] Trastour, D., Bartolini, C., and Gonzalez-Castillo, J. (2001). A Semantic Web Approach to Service Description for Matchmaking of Services. In *Proceedings of the International Semantic Web Working Symposium*.

[99] Vallabhaneni, S. R. (2008). *Corporate Management, Governance, and Ethics Best Practices*. Wiley.

[100] van Der Aalst, W. M. P. (1998). The Application of Petri Nets To Workflow Management. *Journal of Circuits, Systems and Computers*, 08(01) :21–66.

[101] van der Aalst, W. M. P. (2009). Process-aware information systems : Lessons to be learned from process mining. *Transactions on Petri Nets and Other Models of Concurrency II*, pages 1–26.

[102] van der Aalst, W. M. P. (2011). *Process Mining : Discovery, Conformance and Enhancement of Business Processes*. Springer-Verlag, Berlin.

[103] van der Aalst, W. M. P. (2013). Business Process Management : A Comprehensive Survey. *ISRN Software Engineering*, 2013 :1–37.

[104] van der Aalst, W. M. P., Aldred, L., Dumas, M., and ter Hofstede, A. H. M. (2004). Design and Implementation of the YAWL System. *Advanced Information Systems Engineering*, 3084 :142–159.

[105] van der Aalst, W. M. P., ter Hofstede, A. H. M., Kiepuszewski, B., and Barros, a. P. (2003a). Workflow Patterns. *Distributed and Parallel Databases*, 14(1) :5–51.

[106] van der Aalst, W. M. P., ter Hofstede, A. H. M., and Weske, M. (2003b). Business Process Management : A Survey. *Lecture Notes in Computer Science*.

[107] van der Aalst, W. M. P., Weske, M., and Grünbauer, D. (2005). Case handling : A new paradigm for business process support. *Data and Knowledge Engineering*, 53(2) :129–162.

[108] Walker, M. A., Litman, D., Kamm, C. A., and Abella, A. (1997). PARADISE : A framework for evaluating spoken dialogue agents. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics, ACL/EACL 97*, pages 271–280.

[109] Weizenbaum, J. (1966). ELIZA - A Computer Program For the Study of Natural Language Communication Between Man and Machine. *Communications of the ACM*, 9(1) :36–45.

[110] Weske, M. (2007). *Business Process Management : Concepts, Languages, Architectures*. Springer-Verlag New York, Inc.

[111] Wimalasuriya, D. C. and Dou, D. (2010). Ontology-Based Information Extraction : An Introduction and a Survey of Current Approaches. *Journal of Information Science*, 36(3) :306–323.

[112] Wooldridge, M. (2002). *An Introduction to Multi-Agent Systems*. John Wiley & Sons.

[113] Wooldridge, M. and Jennings, N. R. (1995). Intelligent agents : Theory and Practice. *The Knowledge Engineering Review*, 10(2) :1–46.

[114] WS-BPEL Technical Committee, O. (2007). *Web Services Business Process Execution Language Version 2.0*. OASIS Standard.

[115] Yildiz, B. and Miksch, S. (2007). ontoX - A Method for Ontology-Driven Information Extraction. In *Computational Science and Its Applications–ICCSA 2007*, pages 660–673. Springer Berlin Heidelberg.