Chapter 14
Developing Flexible Business Process Management Systems Using Modular Computing Technologies

Minhong Wang and Kuldeep Kumar

1 Introduction

Business process management (BPM) refers to activities performed by organizations to design, implement, operate, manage, and improve their business processes by using a combination of models, methods, techniques, and tools (van der Aalst and van Hee 2002; Melão and Pidd 2000). Most approaches to BPM use information technologies to support or automate business processes in whole or in part, by providing computer-based systems support. These technology-based systems help coordinate and streamline business transactions, reduce operational costs, and promote real-time visibility in business performance.

Traditional approaches to building and implementing BPM systems use workflow technologies to design and control the business process (van der Aalst and van Hee 2002). Workflow-based systems follow highly structured and predefined workflow models and are well suited to applications with standard inputs, processes, and outputs. However, contemporary business processes are complex and dynamic. They evolve and change over time as a result of complex interactions, resource competition, breakdowns and abnormal events, and other sources of uncertainty. Current

This research is supported by a UGC CERG research grant (No. RGC/HKU7169/07E) from the Hong Kong SAR Government, and a Seed Funding for Basic Research (200611159216) from The University of Hong Kong.

M. Wang (✉)
Division of Information & Technology Studies, The University of Hong Kong, Hong Kong, China
e-mail: magwang@hku.hk

K. Kumar
College of Business Administration, Florida International University, Miami, FL, USA

Rotterdam School of Management, Erasmus University, Rotterdam, Netherlands

Department of Information Systems, City University of Hong Kong, Hong Kong, China
e-mail: kumark@fiu.edu; kkumar@cityu.edu.hk

Sushil and E.A. Stohr (eds.), *The Flexible Enterprise*, Flexible Systems Management,
DOI 10.1007/978-81-322-1560-8_14,

research is attempting to support this continuously changing nature of business processes by developing flexible business process management systems using various emerging modular computing technologies, such as Agent-Oriented Computing (AOC), Service-Oriented Architectures (SOA) (Jennings et al. 2000; Leymann et al. 2002; Wang and Wang 2005), Component-Based Development (CBD), and Object-Oriented Programming (OOP) (Kammer et al. 2000; Weske 1998).

There has been a proliferation of studies about the application of agent-, service-, component-, and object-oriented computing solutions to flexible BPM. However, the fundamental questions about their use, such as why we need to introduce these solutions for BPM, how we apply them, and how we integrate them with other solutions, remain unexamined. Most research on technology support for flexible BPM is experience driven, ad hoc, and often lacks a systematic analysis of the rationale for the technology support. Sometimes, the leading edge solutions such as SOA and AOC are proposed without identifying the real rationale for their use in BPM scenarios. There is only minimal work that examines the roots of complexity of business processes, the need of effective approaches for flexible process management, and how this need affects the requirements and technology solutions for flexible process management (Kumar and Narasipuram 2006).

Moreover, as these modular computing concepts and technologies become popular, researchers often attempt to employ and sometimes integrate these modular approaches in creating business process management solutions. However, at present there is considerable ambiguity in differentiating between these overlapping terminologies and consequently their use for flexible BPM systems development. For example, we often hear people discussing their proposed solutions as agent-based systems, whereas they may just be simply using object abstraction. Furthermore, the commonsense understanding of these concepts does not easily map onto each other. Unless we have clarity on these terminologies and the way how to use them, the application and integration of these techniques is likely be problematic.

In this chapter, we first identify the underlying requirements of flexible process management that provide the business rationale for employing these modular technologies in developing BPM systems (Sect. 2). The requirements are examined by investigating the key problems with their solutions in business process management. Next, we examine the similarities and differences between these modular computing technologies to system development: OO, CBD, SOA, and AOC (Sect. 3). Finally, we match these technologies to the requirements of flexible BPM and develop a systemic approach for employing these technologies in developing flexible BPM systems (Sect. 4).

2 How to Deal with Complex Dynamic Business Processes?

A *business process* is a collection of activities that create value by transforming inputs into more valuable outputs (Hammer and Champy 1993). These activities consist of a series of steps performed by actors (either machines or humans) to

produce a product or service for the customer. These steps subdivide the business process hierarchically into modular process components (called tasks or subtasks), each component performing a part of the process. The aspiration of most modular computing technologies is to attempt to model the process architecture by a modular software-architecture (objects, components, Web services, and agents), thereby creating an analog of the business process in software.

Real-world processes are often much messier than the typical input-transformation-output view suggests; they are best viewed as networks, in which a number of actors collaborate and interact to achieve a business goal. A business process displays complexity because of multiple interactions of its internal components and interaction of the process with its environment (Melão and Pidd 2000). In this section, we investigate the roots of complexity of business processes as a result of complex structure, interacting components, dynamic environment, and resource coordination. Based on this investigation, we identify the business requirements on technology solutions for BPM in terms of key problems with their solutions in business process management. The requirements include decomposition of complex processes; coordination of interactive activities; an increased awareness of dynamic business environments; and resource selection, integration, and coordination.

2.1 Decomposition of Complex Processes

Business processes are complex systems that are made up of a number of interacting objects with dynamic behavior. To design a complex structure, one powerful technique is to hierarchically decompose it into semi-independent but interrelated set of components (Simon 1981). Thus, a process is decomposed into tasks, task into subtasks, and so on, through many layers in a hierarchy. To reduce complexity, interactions between subtasks within a task are often encapsulated within the task; interactions between tasks are encapsulated within their higher-level process or task.

This raises the issue on how we decompose complex processes. Traditional workflow approaches have selected "task" as the basic module for building process management systems. A business process can be decomposed into a number of semi-dependent, interrelated tasks within an organization. These tasks are then linked to each other in a preestablished, transactional, usually sequential interrelationship, or dependency. With the extension of business processes from intra-organizational to inter-organizational scope, we need to deal with interactions within an organization as well as interactions across different organizations. Moreover, the complexity of business processes is increased by the interweaving of inter- and intra-organizational interactions.

To manage the complexity, we need to distinguish between inter- and intra-organizational interactions and deal with them by isolating one type of interaction from another. We propose "service" as a high-level view of the building block of a process, where a process is composed of a set of services; each service is provided

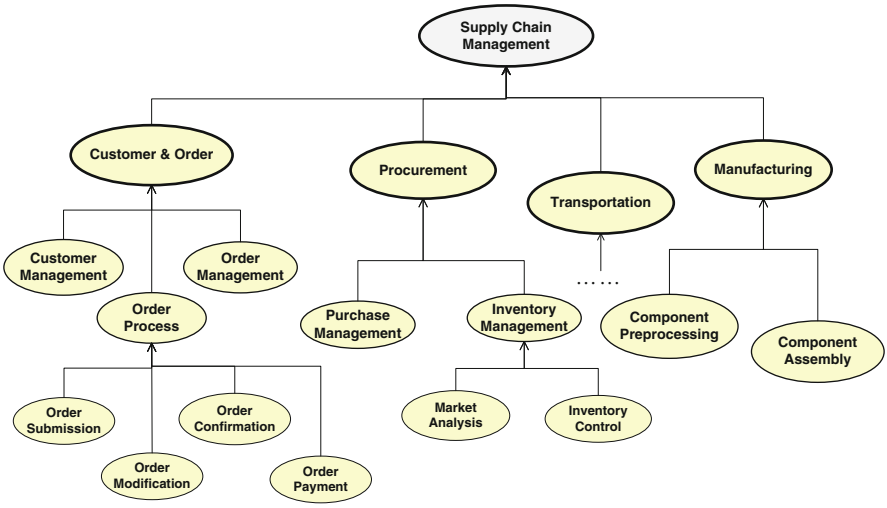

Fig. 14.1 Decomposition of a supply chain process

by a corresponding actor (organization, individual, or computer program) and can be further decomposed into subtasks. For example, a complex supply chain management process is decomposed into customer and order service, procurement service, manufacturing service, and transportation service; each individual service is provided by a corresponding organizational actor and can be further decomposed (see Fig. 14.1).

2.2 Flexible Coordination of Interactive Activities

To manage complex interactions in complex processes multiple actors, activities, resources, and goals need to be coordinated. Mintzberg (1979) suggests that every organized human activity gives rise to two fundamental requirements: differentiation, or the division of work into tasks to be performed by various actors, and integration, that is, the coordination of these tasks to accomplish the goals of the activity. After decomposing a complex process into a number of task components, we need to coordinate various interactions between the components at different levels in a network hierarchy. In the context of a hierarchy, a component can be involved in vertical interactions with its subordinates and superordinates and in horizontal interactions with its peers. A component in a complex system, no matter how large or small, may interact with a limited set of superiors, inferiors, and coordinate peers (Simon 1981).

Mintzberg further suggests that environmental uncertainty is an important determinant of the mode for interactions and coordination. The more stable and predictable the situation, the greater the reliance on coordination based on structured and specifiable schedules, such as coordination by plan and coordination by standardization. The more

variable and unpredictable the situation, the greater will be the reliance on informal and flexible communication, such as coordination by feedback and coordination by mutual adjustment (Kumar and van Dissel 1996; Kumar et al. 2007). Thus, when faced with increased uncertainties in dynamic environments, organizations need to use more flexible coordination mechanisms to coordinate their business processes. Flexible coordination is portrayed by more bottom-up initiatives and less centralization of decision-making at the top. This requires flatter hierarchies, decentralized autonomy-based units, and decision-based coordination, which in turn reduces direct hierarchical control and encourages greater mutual adjustment and coordination between the work units (Mintzberg 1979; Volberda 1999).

2.3 Awareness of Dynamism in Business Environments

As a result of complex interactions, resource competition, abnormal events, and other sources of uncertainty, business processes continuously evolve and change over time. Furthermore, a complex process is usually semi-structured or unstructured; there is an absence of routine procedures for dealing with it. In such situations, we cannot depend on providing the computer system with exact details about how to accomplish a process, but provide the system with guidelines to help it determine how to deal with the process. In other words, problem solving is regarded as an interaction between the behaving organism and the environment under the guidance of a control system. Information and data are input to this system, represented in its memory as declarative knowledge, and then used in problem solving following algorithmic or heuristic steps (Wang and Wang 2006).

A basic idea underlying this viewpoint is the control of complex dynamic systems or situations based on situation awareness. Awareness, according to biological psychology, is a human's or an animal's perception and cognitive reaction to a condition or event. Situation awareness is the perception and understanding of objects, events, people, system states, interactions, environmental conditions, and other situation-specific factors in complex and dynamic environments (Endsley 1995). Situation awareness underpins real-time reactions to environmental changes. In terms of cognitive psychology, situation awareness refers to the active content of a decision-maker's mental model, its purpose being to enable rapid and appropriate decisions and effective actions.

In a dynamic business process environment, an exact execution order of activities is impractical; the interaction or relationship between the environment and activities is more appropriate in determining how to manage and coordinate tasks (Wang and Wang 2006). The dynamicism therefore requires spontaneous decisions and coordination of processes based on situation awareness. We need to be able to coordinate the processes by sensing and comprehending the situation, determining responses to it while, at the same time, taking actions to work towards business goals. In other words, the question of which task to execute and when to execute it is dependent on the current environment and underlying business rules rather than a static process schema.

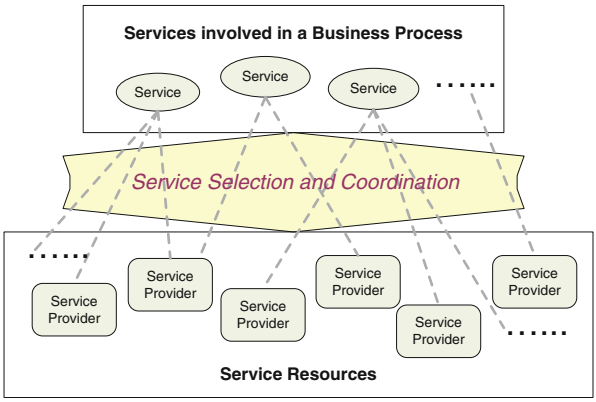

Fig. 14.2 Resource selection and coordination in business processes

2.4 Flexible Resource Selection, Integration, and Coordination

Business processes require actors and resources to perform the tasks. These actors and their associated resources may reside either within the organization or, in the case of inter-organizational processes, across a network of multiple organizations. Business networks of resources and actors can be temporarily assembled, integrated, and driven by demands that emerge and operate for the lifespan of the market opportunity (Kumar 2001). In this conception, a firm is not considered as a black box guided by the strategist, but as a bundle of firm-specific resources of use for specific tasks. Along with this conception, new business models have accordingly come into view, such as demand chain, virtual enterprise, and electronic marketplace. They allow companies to operate in dynamically changing environments by quickly and accurately evaluating new market opportunities or new products. The companies may coordinate with potential partners in demand-driven and resource-based soft connections that are made for the duration of the market opportunity.

As a result, a business process can be dynamically established at run-time by connecting or composing several services together from different organizations through alliances, partnerships, or joint ventures. In this situation, attentions on business processes should be extended to other elements in addition to task and procedure, which include resources discovery, selection, integration, and coordination.

What is new in this business process model is reliance on the idea of separating resource requirements from concrete satisfiers (Mowshowitz 1997). This separation allows for crafting process structures that enable management to switch between different resources options for implementing a process (see Fig. 14.2). It creates an environment in which the means for reaching a goal are evaluated and selected for optimized performance. The success of the model is highly dependent on the match between the requirements and satisfiers that deliver the services. One

way to ensure this balance is to model the integration or composition of business processes as a management problem which involves (1) the separation of requirements from the means for realization and (2) the dynamic selection and allocation of available resources to requirements (Mowshowitz 1997). In doing it, we model the complex structure of inter-organizational processes by using "service" as the building block, which supports flexible integration and coordination among the services or satisfiers.

3 Clarifying the Terminologies of Technical Solutions

As mentioned above, various modular computing architectures such as *objects, components, Web services,* and *agents* have been proposed to develop systems for flexible process management. In order to use these concepts in business process management, we must first have a clear understanding of their similarities, differences, and relationships. Research employing these concepts has often borrowed from natural language; terms such as agents, autonomous agents, brokers, actors, services, and components are used without precisely differentiating between them. Thus, before we can use these concepts appropriately, we need to understand the similarities and differences between them. Moreover, we also need to understand how these concepts can be used for developing flexible process management solutions. In this section we outline four such concepts: Agents and Agent-Oriented Computing, Services and Service-Oriented Architecture, Objects and Object-Oriented Programming, and Components and Component-Based Development. It needs to be emphasized that the purpose of this section is to clarify the similarities and differences between these approaches and to make explicit some of the underlying assumptions inherent in the use of the terminology, not to redefine them.

3.1 Agent and Agent-Oriented Computing (AOC)

Recently the Agent-Oriented Computing paradigm has gained popularity among researchers attempting to develop complex systems for business process management. Terms such as "autonomous agent" and "agency" are beginning to be commonly used in computer science literature. On the other hand, a rich body of literature on the concept of Agency and the role of agents already exists in the institutional economics and business field. This section is an attempt to reconcile the various terms from the two research traditions.

Actor vs. Agent. Actor is someone who performs an act, that is, does something. An actor may be a person, an organizational unit, or a computer program. An actor may be completely autonomous, that is, it acts of its own volition. If the actor is authorized to do something on behalf of someone else, the actor is an "agent" of the other party.

Agent. Agent is an actor (performer) who acts on the behalf of a principal by performing a service. The agent provides the service when it receives a request for service from the principal. The principal-agent relationship is found in most employer/employee relationships. A classic example of agency relationship occurs when stockholders hire top executives to run the corporation on their behalf. To manage the relationship between a principle and an agent of the principle, agency theory is concerned with various mechanisms used for aligning the interests of the agent with those of the principal such as piece rates/commissions and profit sharing (Eisenhardt 1989).

Agent vs. Broker. A Broker is a special type of agent that acts on behalf of two symmetrical parties or principals – the buyer and seller. A broker mediates between the buyer (service requesting party) and the seller (service providing party). Acting as an intermediary between two or more parties in negotiating agreements, brokers use appropriate mediating techniques or processes to improve the dialogue between the parties, aiming to help them reach an agreement. Normally, all parties must view the mediator as neutral or impartial.

Autonomy. Autonomy is the power or right of self-government. It refers to the capacity of a rational individual to make an informed, uncoerced decision. An autonomous agent therefore is a system situated in, and part of, an environment, which senses that environment, and acts on it, over time, in pursuit of its agenda as derived from its principal. "Autonomous" means that the actor is independent, that is, the actor can decide what to do and how to do it. As an agent acts on behalf of the principal, the agent cannot be fully autonomous. The principal may give the agent different levels of choice in performing the task. For example, the principal can tell the agent what to do, but leave it to the agent to decide as to how to do it.

Software Agent. In computer science, the term **"**agent" is used to describe a piece of software or code that acts on behalf of a human user or another program in a relationship of agency. It may denote a software-based entity that could enjoy the some properties of autonomy (agents operate without the direct intervention of its principal humans), social ability (agents communicate with other agents), reactivity (agents perceive their environment and respond to changes in a timely fashion), and proactivity (agents do not simply act in response to their environment, but are able to exhibit goal-directed behavior by taking some initiative) (Jennings et al. 2000). The agent-based computing paradigm is devised to help computers know what to do, solve problems on behalf of human beings, and support cooperative working. The behavior of software agents is empowered by human and implemented by software.

Agent-Oriented Computing (AOC). The key idea of Agent-Oriented Computing is the delegation of tasks and responsibility of a complex problem to software agents. It emphasizes autonomy and mutual cooperation of agents in performing tasks in open and complex environments. A complex system can be viewed as network of agents acting concurrently, each finding itself in an environment produced by

its interactions with the other agents in the system. AOC is used to model and implement intelligent solutions to semi- or ill-structured problems, which are too complex to be completely characterized and precisely described. AOC offers a natural way to view and describe systems as individual problem-solving agents pursuing high-level goals defined by their principals. It represents an emerging computing paradigm that helps understand and model complex real-world problems and systems, by concentrating on high-level abstractions of autonomous entities (Wooldridge and Jennings 1999).

3.2 Service and Service-Oriented Architecture (SOA)

Service. A service is the work done by somebody (the agent) for someone else (the principal).

Web Service. As defined by W3C (World Wide Web Consortium), a Web service is a software application identified by a URI (Uniform Resource Identifier), whose interfaces and bindings are capable of being defined, described, and discovered by XML and which supports direct interactions with other software applications using XML-based messages via Internet-based protocols. Web services are self-contained and modular business applications based on open standards (Papazoglou 2007). They can share information using standardized communication protocols to ask each other to do something, that is, ask for service.

Service-Oriented Architecture (SOA). Service-Oriented Architecture utilizes Web services as fundamental elements for developing applications. It is an emerging paradigm for architecting and implementing business collaborations within and across organizational boundaries. SOA enables seamless and flexible integration of Web services or applications over the Internet. It supports universal interoperability and location transparency. SOA reduces the complexity of business applications in large-scale and open environments by providing flexibility through service-based abstraction of organizing applications.

AOC vs. SOA. Software agent is a software-based entity that enjoys the properties of autonomy, social ability, reactivity, and proactivity. Web service is a software application in the Web based on open standards. Though both of them are computer applications that perform tasks on behalf of principals (human beings or other programs), the focus of software agents is on their autonomous properties for solving complex problems, while Web services are characterized by their open access standards and protocols over the Internet. While a Web service may only know about itself, agents often have awareness of other agents and their capabilities as interactions among the agents occur. Agents are inherently communicative, whereas Web services are passive until invoked. Agents cooperate autonomously and flexibly and, by forming teams and coalitions, can assemble higher-level and more comprehensive services. However, current standards or languages for Web services do not

provide for flexible composing functionalities, such as brokering and negotiation in e-marketplaces (Huhns 2002). Thus, Web services are inherently less autonomous and independent than software agents.

Against this background, there is a movement towards combining the concept of Web services with software agents. W3C introduced a concept, where software agents are to be treated as the foundation for Web services architecture – "A Web service is an abstract notion that must be implemented by a concrete agent." AOC may take SOA into new dimensions to model autonomous and heterogeneous components in uncertain and dynamic environments. The integration of Web services with software agents can function as computational mechanism in their own right, thus significantly enhancing the ability to model and construct complex software systems. It will be a promising computing paradigm for efficient enterprise service selection and integration.

3.3 Object and Object-Oriented Programming (OOP)

Object-Oriented Programming (OOP). Rumbaugh defines OOP as programming in terms of a collection of discrete objects that incorporate both data and behaviors (Rumbaugh 1991). OOP is a software engineering paradigm that uses "objects" and their interactions to design applications and computer programs. It is seen as a collection of cooperating objects, as opposed to a traditional view in which a program is seen as a list of instructions to the computer. OOP was deployed as an attempt to promote greater flexibility and maintainability in programming by strongly emphasizing modularity and reusability in software.

AOC vs. OOP. From a software engineering point of view, Object-Oriented (OO) methodologies provide a solid foundation for Agent-Oriented modeling. AOC can be viewed as a specialization of OOP. OOP proposes viewing a computational system as made up of modules that are able to communicate with one another. AOC specializes the framework by representing the mental states and rich interactions of the modules (agents). While objects emphasize passive behavior (i.e., they are invoked in response to a message), agents support more autonomous behavior, which can be achieved by specifying a number of rules for interpreting the states and governing multiple degrees of freedom of activities.

3.4 Component and Component-Based Development (CBD)

Component-Based Development (CBD) is another branch of the software engineering discipline, with an emphasis on decomposition of the engineered systems into functional or logical components with well-defined interfaces used for communication across the components. CBD includes a component model and an interface model. The component model specifies for each component how the component behaves in

an arbitrary environment; an interface model specifies for each component how the component interacts with its environment (Szyperski 2002).

OOP vs. CBD. A component is a small group of objects working together to provide a system function. It can be viewed as a black box at the level of a large system function. At a fine level of granularity, we use objects to hide behavior and data. At a coarser level of granularity, we use components to do the same.

Components inherit much of the characteristics of objects in the OO paradigm. But the component notion goes further by separating the interface from the component model. OO reuse usually means reuse of class libraries in a particular OO programming language or environment. For example, you have to be conversant with SmallTalk or Java to be able to reuse a SmallTalk or Java class. A component, by using public interface, can be reused without even knowing which programming language or platform it uses internally.

SOA vs. OOP and CBD. At a conceptual level, SOA is an extension of earlier OOP and CBD concepts. OOP focuses on the encapsulation of both data and behavior of an object, while SOA also focus on the user's view of a computing object or application, that is, the interface. An interface specifies the services that are provided and contains metadata defining how they behave.

Though CBD goes further than OOP by supporting public interfaces used for communication across the components, the interfaces of a component are easier to change because they are only used by the known clients. In SOA, a service has a published network-addressable interface. A published interface is one that is exposed to the network and may not be changed so easily, because the clients of the published interface are not known. The difference is analogous to an intranet-based site only accessible by employees of the company and an Internet site accessible by anyone.

3.5 Reconciling OOP, CBD, SOA, and AOC

From OOP and CBD to SOA and AOC, the practice of software programming has evolved through different development paradigms. At the conceptual level, these concepts and approaches build upon each other are complementary, and all have a role to play in designing and managing software systems. Each method shift came about in part to deal with greater levels of software complexity. In all cases, the way we manage complexity is by decomposing a complex system or process into smaller modules that can be designed independently, that is, modularity. Modularity ensures easy maintenance and updates of complex systems by separating the high-frequency intra-module linkages from the low-frequency inter-module linkages and limiting the scope of interactions between the modules by hiding the intra-module relations inside a module box (Baldwin and Clark 1997). Based on the idea of modularity, constructs such as objects, components, software agents, and Web services have been continuously invented and evolved for developing software applications.

Object-Oriented (OO) methodologies provide a foundation for software engineering that uses objects and their interactions to design applications and computer programs. CBD provides a coarser grained construct for larger systems and separates interface from the behavior of the construct for supporting public communication between the components which know about each other. SOA goes further by using XML-based and network-addressable interface as well as XML-based messages and standard protocols for open communication among all software applications in Internet. In SOA, a Web service can find and talk with another Web service which is unknown a priori. Compared with OOP, CBD, and SOA, AOC is used to model and implement solutions to semi- or ill-structured problems, which are too complex to be completely characterized and precisely described. In addition to passive behavior, agent is used to perform more autonomous activities in solving complex problems. To achieve this, knowledge or rules for governing the behavior are separated from the behavior of the agent.

In computer science, the terms object, component, software agent, and Web service describe a piece of software that performs some action on behalf of human beings, like an agent or actor. In addition to actor, agent can also be a broker, which mediates between the buyer (service requesting party) and the seller (service providing party). In terms of broker, software agent can be used to search appropriate applications, for example, Web service in the Internet, to perform requested services. This special type of agent works as an intermediary between service requester and service provider, coordinating on behalf of two parties regarding service requirements, qualities, costs, constraints, etc.

4 Applying Technical Solutions to BPM Problems

In Sect. 2, we identified four business requirements on flexible process management (decomposition of complex processes, task coordination, dynamism of the environment, and the resource acquisition and assembly) that provide the business rationale for employing appropriate technologies in developing flexible BPM systems. In Sect. 3, we clarify and explicitly define the similarities and differences between four modular technologies: OOP, CBD, SOA, and AOC. In this section we will show how these technologies can be used to address the challenges outlined in Sect. 2. Table 14.1 below summarizes the relationship between the problem aspects identified in Sect. 2 and solution technologies defined in Sect. 3. Sections 4.1, 4.2, 4.3 and 4.4 expand on this table.

4.1 Decomposition of Complex Processes

Business processes display complexity as a result of interactions of their internal components and interaction of the process with its environment. A process can be

Table 14.1 Technical solutions applied to flexible BPM

Technical solutions	Flexible BPM
OOP, OBD, SOA, and AOC for decomposing complex processes at different level of granularity	Decomposition of complex processes
SOA for decomposing and integrating inter-organizational processes over the Web	
AOC for decomposing and delegating ill-structured tasks to autonomous software entities	
OOP, OBD, and SOA for structured communications among tasks or task components	Flexible task coordination
SOA for open communication among all software applications over the Internet	
AOC for flexible coordination by supporting flatter hierarchies, loosely coupled autonomy-based units and decision-based coordination mechanisms	
Object, component, and service unable to behave in dynamic environments	Awareness of dynamic environments
Agents for reaction to changes in dynamic environments through continuous perception of and interaction with the environment	
Agents for proactive behavior by making prediction of future state of dynamic environments	
Object and component not used for resource coordination	Flexible resource coordination
SOA for seamless and flexible integration of resources across different organizations over the Web	
Agents for coordination among resources	

decomposed into a set of tasks, task into subtasks, and so on, through several layers in a hierarchy. Tasks or subtask components can be delegated to software objects, components, agents, and services, as actors of the tasks, which interact and communicate in performing the process.

To deal with interactions across different organizations, SOA proposes "service" as a high-level view of the building block of a process. A process is composed of a set of services, each of which is provided by an individual organization. By using SOA, the interservice interactions are separated from intraservice interactions; the complexity of both maintained at different layers. Moreover, we can take advantage of reusability, interoperability and extensibility of Web services on the basis of open standards to cater for business process integration and interoperation over the Web.

The highly dynamic and unpredictable nature of business processes makes agent-based approach appealing. AOC assigns business applications' main activities to autonomous agents. Such agents are flexible problem solvers that have specific goals to achieve and interact with one another to manage their autonomy and interdependencies in business processes. AOC is well suited for complex process situations that are not all known a priori, cannot be assumed to be fully controllable in their behaviors, and must interact on a sophisticated level of communication and coordination (Wang and Wang 2005).

4.2 Flexible Task Coordination

A business processes is made up of a number of task components that interact with dynamic behavior. OOP uses objects to hide behavior and data, supporting communications among small objects, for example, functions of tasks. CBD extends OOP by supporting interaction among components, that is, coarser grained constructs, using public communication interface. SOC goes further by using XML-based and network-addressable interface as well as XML-based messages and standard protocols for open communication among all software applications over the Internet.

While OOP, OBD, and SOA mainly support structured communications among tasks or task components, AOC are able to support ill-structured interactions among tasks. To coordinate the interactions in dynamic situations, flatter hierarchies, decentralized autonomy-based units, and decision-based coordination mechanisms are required, where AOC is directly applicable. AOC supports decentralized control and asynchronous operations by a group of autonomous software entities, which are able to perform decision-based coordination of their activities.

In AOC, after decomposing a complex process into a number of loosely coupled tasks in a flat hierarchy, we delegate the tasks to a number of autonomous agents, each working both autonomously and collaboratively throughout the whole process. In complex process management, it is impossible to predefine all activities and interactions at design time. Instead, we define the goal or role of each agent and specify a set of rules for governing the behavior of the agent. Agents operate asynchronously and in parallel. This also results in an increase in overall speed and robustness in BPM. The failure of one agent does not necessarily make the overall system useless, where other agents may adjust and coordinate their behavior reactively and proactively to the change.

4.3 Awareness of Dynamic Environments

The complexity of business processes comes not only from interactions of their internal components but also from interaction of the process with its environment. To manage business processes in a dynamic environment, we need to be able to continuously perceive the environment and make real-time decisions on the process. Objects, components, and services are normally unable to behave in dynamic environments. Agent-based software entity is able to sense and recognize the situation and determine appropriate actions upon the situations. Information about the environment (e.g., events, state of activities, and resources) is sensed and interpreted by the agent on the basis of predefined scheme and rules. In case that information is unanticipated or comes as complete and total surprise, it will be sent to human manager for manual processing.

Unlike the ECA (event-condition-action) rules in workflow systems that make reaction to certain events, AOC goes further by incorporating all environmental information into a mental state that watches over the whole environment. Individual

events are put together for a comprehensive understanding; ambiguous information is understood after appropriate interpretation and reasoning (Wang and Wang 2006).

Moreover, AOC supports prediction of future state of the environment for purpose of proactive actions. Different from passive response to current events, proactive behavior has an orientation to the future, anticipating problems and taking affirmative steps to deal with them rather than reacting after a situation has already occurred. It refers to the exhibition of goal-oriented behaviors by taking initiatives.

4.4 Flexible Resource Coordination

As discussed, the rise of Internet-mediated e-Business brings the era of demand-driven and resource-based soft connections of business organizations. A business process can be dynamically established by connecting or composing services provided by different organizations. Against this background, SOA provides a real platform of resource selection and allocation in implementing seamless and flexible integration of business processes over the Web.

However, it is a complex problem to search appropriated services from a large number of resources as well as schedule and coordinate them under various constraints. The complexity arises from the unpredictability of solutions from service providers (e.g., availability, capacity, and price), the constraints on the services (e.g., time and cost constraint), and interdependencies among the services. A service solution to an individual service involved in an integrated process does not have a view of the whole service, very often resulting in incoherent and contradictory hypotheses and actions (Wang et al. 2006).

To deal with the problem, AOC can be used for distributed decision-making and coordination. In process integration, decision-making and coordination among services can be modeled as a distributed constraint satisfaction problem, in which solutions and constraints are distributed into a set of services and to be solved by a group of agents (brokers) on behalf of service requesters and providers. In this context, service-based process integration is mapped as an agent-based distributed constraint satisfaction or optimization problem. Individual services are mapped to variables, and solutions of individual services are mapped to values. A distributed constraint optimization problem consists of a set of variables, each assigned to an agent, where the values of the variables are taken from finite and discrete domains. Finding a global solution to an integrated process requires that all agents find the solutions that satisfy not only their own constraints but also interagent constraints (Wang et al. 2008).

5 Conclusion

In this chapter, we have investigated how relevant modular programming technologies can be applied and integrated in developing flexible BPM solutions. On the one hand, we examine the main problems to be solved in flexible process management.

Based on a theoretical understanding on business processes and the roots of their complexity, we identify and address the main problem with their solutions in flexible BPM. On the other hand, we analyze the overlapping technical concepts and solutions used for flexible BPM systems development. We clarify the differences and relationships between these terminologies and techniques in the context of BPM. Based on the examination of the both sides (BPM requirements and supporting techniques), we have made a clear picture with a systemic approach on how these concepts and technologies can be applied and integrated in developing flexible process management systems. This chapter will benefit professionals, researchers, and practitioners by advanced analysis and theoretical investigations of problems and solutions in developing flexible BPM solutions.

References

Baldwin CY, Clark KB (1997) Managing in an age of modularity. Harv Bus Rev 75(5):84–93
Eisenhardt KM (1989) Agency theory: an assessment and review. Acad Manage Rev 14(1):57–74
Endsley MR (1995) Toward a theory of situation awareness in dynamic systems. Hum Factors 37(1):32–64
Hammer M, Champy J (1993) Reengineering the corporation: a manifesto for business revolution. Brealey, London
Huhns MN (2002) Agents as web services. IEEE Internet Comput 6(4):93–95
Jennings NR, Faratin P, Norman TJ, O'Brien P, Odgers B (2000) Autonomous agents for business process management. Int J Appl Artif Intell 14(2):145–189
Kammer PJ, Bolcer GA, Taylor RN, Hitomi AS, Bergman M (2000) Techniques for supporting dynamic and adaptive workflow. Comput Support Coop Work 9(3–4):269–292
Kumar K (2001) Technology for supporting supply chain management: introduction. Commun ACM 44(6):58–61
Kumar K, Narasipuram MM (2006) Defining requirements for business process flexibility. In: Seventh workshop on business process modeling, development, and support, CAiSE, Luxembourg
Kumar K, van Dissel H (1996) Sustainable collaboration: managing conflict and cooperation in interorganizational systems. MIS Q 20(3):279–300
Kumar K, van Fenema PC, von Glinow MA (2007) Offshoring and the global distribution of work: implications for task interdependence theory and practice. In: First annual research conference and workshop on offshoring, North Carolina
Leymann F, Roller D, Schmidt MT (2002) Web services and business process management. IBM Syst J 41(2):198–211
Melão N, Pidd M (2000) A conceptual framework for understanding business processes and business process modeling. Inform Syst J 10:105–129
Mintzberg H (1979) The structuring of organizations. Prentice Hall, Englewood Cliffs
Mowshowitz A (1997) Virtual organization. Commun ACM 40(9):30–37
Papazoglou MP (2007) Web services: principles and technology. Pearson Education Ltd., Harlow
Rumbaugh J (1991) Object-oriented modeling and design. Prentice Hall, Englewood Cliffs
Simon HA (1981) The sciences of the artificial. MIT Press, Cambridge, MA/London
Szyperski C (2002) Component software: beyond object-oriented programming. Addison-Wesley Professional, Boston
van der Aalst WMP, van Hee KM (2002) Workflow management: models, methods, and systems. MIT Press, Cambridge

Volberda HW (1999) Building the flexible firm: how to remain competitive. Oxford University Press, Oxford

Wang M, Wang H (2005) Intelligent agent supported business process management. In: Proceedings of 38th Hawaii International Conference on System Sciences (HICSS-38), IEEE Computer Society Press, Hawaii

Wang M, Wang H (2006) From process logic to business logic – a cognitive approach to business process management. Inform Manage 43(2):179–193

Wang M, Cheung WK, Liu J, Xie X, Lou Z (2006) E-service/process composition through multi-agent constraint management. In: Fourth international conference on Business Process Management (BPM), LNCS 4102, Vienna, pp 274–289

Wang M, Liu J, Wang H, Cheung W, Xie X (2008) On-demand e-supply chain integration: a multi-agent constraint-based approach. Expert Syst Appl 34(4):2683–2692

Weske M (1998) Object-oriented design of a flexible workflow management system. In: 2nd East-European symposium on advances in databases and information systems. Lecture notes in computer science, Poznan, Poland, vol 1475. pp. 119–130

Wooldridge M, Jennings NR (1999) Software engineering with agents: pitfalls and pratfalls. IEEE Internet Comput 3(3):20–27