Fractal Modeling Approach for Supporting Business Process Flexibility

Julija Stecjuka, Marite Kirikova, and Erika Asnina

Institute of Applied Computer Systems,
Riga Technical University,
1 Kalku, Riga, LV-1658 Latvia
{julija.stecjuka,marite.kirikova,erika.asnina}@cs.rtu.lv

Abstract. Ability to support various business models has been recognized as one of the essential competitive advantages of companies operating in global networked business environment. The use of several business models simultaneously, requires availability of flexible business process models. Flexibility of business process models, in turn, depends on appropriate information systems support. One of the ways how to support business process flexibility is to use a fractal paradigm in information systems development. The fractal paradigm can be applied at two levels of abstraction: the level of business process system and the level of software system. Applications of the fractal paradigm at two abstraction levels correspond to two different opportunities of supporting flexible business processes.

Keywords: business process, flexibility, information system.

1 Introduction

Ability to support various business models has been recognized as one of the essential competitive advantages of companies operating in global networked business environment [1]. The use of several business models simultaneously requires flexible business process models. Flexibility of business process models, in turn, depends on appropriate information systems support inside and outside the business organization. One of the important problems to be resolved in information system development for supporting business process flexibility, is the possibility to adapt an information system's functional model to business processes needs on a high level of detail, so that both domains (business domain and computer system domain) become a whole in the context of business process implementation.

The goal of this paper is to discuss applicability of a fractal approach in business process modeling and information systems design. The approach is based on the idea of a fractal enterprise that has been presented in [2] as well as on the research of business process flexibility on different levels of abstraction [3]. Fractality of an enterprise implies goal adaptability and similarity at different levels of scale (or granularity) of the enterprise. Flexibility of a business process implies that two different constituents of the business process are recognized: (1) the core of the process that changes relatively slowly and (2) the "rural" parts of the process that may be changed relatively frequently. In this paper we conceptually distinguish between (1) a fractal business process system and (2) a fractal software system. The first concept is used for supporting business process flexibility at the abstraction level of enterprise business processes; the second concept is used for achieving flexibility of software design that supports the business processes of the enterprise. The concepts are supported by corresponding fractal information systems development approaches, which are briefly presented in the paper.

The paper is structured as follows. Basic concepts and related work are discussed in Section 2. Information systems design approach for fractal process systems is presented in Section 3. Object-oriented approach for designing fractal information systems is presented in Section 4. Discussion of applicability of the approaches and conclusions are given in Section 5.

2 Basic Concepts and Related Work

The use of fractal approach in systems development has been applied in a number of different contexts [4]. Theoretically, fractal systems are self-organizing, cooperative, self-similar at different levels of scale systems that can adapt to changing systems goals and environment [5, 6 and 7]. Vitality and dynamics of such systems seem to be attractive features of enterprises operating in turbulent global environment.

One of the central notions in fractal systems is the *scale* of the system. In a fractal enterprise the scale is identical to the structure of enterprise decomposition [2], where the largest scale refers to the whole enterprise and the smallest to the smallest organizational unit exposing fractal properties. From the point of view of processes, self-similarity can be considered in at least three different dimensions, namely: agent (enterprise organizational units), process (functionality), and information architecture (Fig. 1). While each organizational unit (including an individual employee) performs particular processes and uses particular information structures, there is no simultaneous scaling commonality between organizational, process and information architectures, i.e., a small scale organizational unit can perform processes of different scales

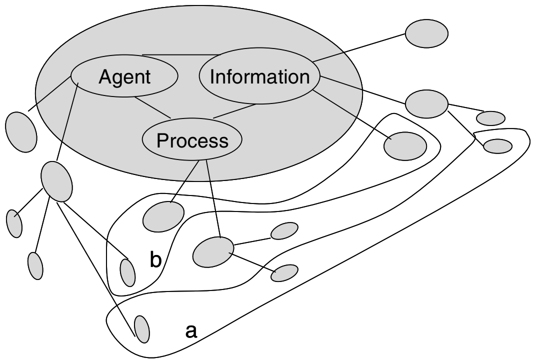

Fig. 1. Fractal architecture of organizational units, processes and information

and use information structures of different scales (see a and b in Fig. 1). Each process at different levels of scale may be performed by different agents and use different information architectures. In general, fractality of the process can be considered from several viewpoints or according to several dimensions simultaneously, i.e., it could be reflected by a multi-fractal model. Models and approaches described in Sections 3 and 4 deal with multi-fractality, however, this aspect of the models is not emphasized in the paper in order not to overcomplicate the discussion about the possibilities of supporting business process flexibility.

Usually the processes in fractal systems are addressed via functions of agents. The following basic process taxonomies may be found in fractal systems development approaches:

- sensing, observing, analyzing, resolving, organizing, reporting, actuating [8]
- sensing, requirements definition and planning, execution, delivery, responding [7]
- monitoring, analyzing, reporting, planning, executing [9]
- goal dissociation, partner selection, business coordination, task execution, schedule-progress monitoring [10].

Above-mentioned taxonomies are applicable in many cases, however, their generality is not always helpful for identifying information systems requirements for business process support (see more details in Section 5).

As regards the agent perspective the following information flows are to be taken into consideration [8, 11]:

- information flows inside a fractal entity
- information flows between the same scale level fractals
- information flows between different scale level fractals
- information flows between external environment and fractal entities.

Information flows between external environment and fractal entities are used for the assessment of the change against specific levels of work environment such as cultural, strategic, socio-informal, financial, informational and technological [11, 12].

In fractal systems, information architecture has at least two orthogonal dimensions. The existence of those dimensions depends on (1) different modes of information processing (human brain, software) and (2) the particular fractal hierarchy under consideration. Fractal hierarchy relates to the living systems theory [13] where hierarchical levels emerge to achieve a higher communication efficiency. With respect to computer systems supported information transfer and processing, in a fractal level software system, part of information is to be processed by software of a higher fractal entity and, at the same time, each fractal entity may have independent information processing functionalities.

There are different definitions of business process flexibility; however, most of them focus on the ability to respond to external changes in the appropriate period of time using a reasonable amount of resources [14, 15 and 16]. Flexibility, however, possesses some degree of stability because whenever a part of the system is made flexible, some other part is made inflexible [17]. From the information systems viewpoint different approaches could be used to support relatively rigid and relatively flexible (or more rapidly changing) business process constituents.

Fractal approach may seem to be somewhat similar to aspect-oriented methods of software development [18]. However, aspect-oriented approaches, in general, do not consider multiple scales of processes and goal adaptability, which are essential features of fractal systems.

3 Business Process Fractality Oriented Software Requirements for Flexibility Support: A Process-Oriented Approach

The approach discussed in this section focuses on fractal properties of the business process system. The notion of the fractal business process system is discussed in Subsection 3.1. The approach of identification of software services for fractal process system support is briefly presented in Subsection 3.2. The approach takes into consideration commonalities and differences of requirements at different levels of process fractals and tries to identify those functionalities that are least dependent on possible changes affecting the fractal system.

3.1 The Notion of the Fractal Process System

We define the fractal process system by modifying definitions given in [8] and [9]: the fractal is a functioning systems component (enterprise business process and/or software service), whose goals and performance can be precisely described, which has unique objectives, achieves concrete results and acts autonomously in a self-optimizing way whilst interacting with other system's fractals. The fractal processes system may be described as follows [19]:

- It consists of fractals (enterprise business processes and/or software services), where each of the fractals has unique objectives, which correspond to system's common objectives.
- Each fractal has its own structure in the fractal processes system, i.e., a unique set of interconnected activities.
- Each single fractal acts separately to meet the system's common objective and interacts with other fractals in the case of the lack of resources (e.g., information stored by other fractal component).
- The fractal processes system is organized as a multilevel structure, where the level is the scale at which the fractal is examined. In a sense, a large-scale fractal includes self-similar small-scale fractals. So the system's common objective could be achieved by implementing an appropriate fractal for each scale.
- Information flows exist between fractals in the fractal processes system.

The main fractals properties in the fractal processes system are implemented in the following way [19]:

- Self-similarity manifests when processes of different scale have a common objective and similar inputs and outputs.

- Self-organization in the fractal processes system could be expressed as follows:
  - Appropriate alternative selection during certain fractal task processing.
  - Modification and optimization of interacting fractal relationships.
  - Creation of a new fractal or a new alternative for accomplishing tasks of a fractal process.
  - Development of a new software service.
  - Adapting to changes in external requirements.
- Systems common objectives are achieved iteratively, by developing each single fractal's individual objective taking into account feedbacks.
- Dynamics and vitality means that coordination and cooperation between self-organizing fractals are characterized by individual dynamics and the ability to adapt to dynamical environment. The information system plays a vital role in achieving this property of the fractal business process system.

3.2 The Main Steps for the Identification of Requirements for Fractal Business Processes

The main idea of the suggested method is to create a relatively simple approach for describing the process system from the point of view of information processing. The process-oriented method for fractal information system development is based on the assumption that software services can be defined as soon as the fractal sub-system of enterprise business processes is discovered.

The process-oriented method includes the following main stages [19]:

- Stage 1 – Fractal processes system development. This stage includes the following three activities:
  - Perform analysis of existing enterprise business processes with the aim of identifying a business sub-process with common or similar goals, similar inputs and outputs.
  - Identify the dimensions of the scale of selected business sub-processes.
  - Define appropriate fractal hierarchy according to previously defined scales.
- Stage 2 – Software model development taking into account the fractal hierarchy. To develop software model, the following activities are needed:
  - Define software functional and non-functional requirements for each "fractal" in fractals' structure.
  - Identify common and specific requirements for all identified "fractals" in the fractal hierarchy.
  - Identify appropriate software services for "fractals" taking into account common and specific requirements identified in the previous step.
  - Define interactions between identified software services.
  - Define relationships between software services and process "fractals". Relationships have a very important role in "fractals" self-organization, thanks to which "fractal" system is able to restructure and tolerates replacement of some of its parts. Relationships help to track changes in fractals organization and introduce necessary changes in software services.

- Stage 3 – Detailed business processes model development for obtaining common vision of the situation in fractal information system. At this stage correspondence between business process decomposition and software services is defined by gradual business process decomposition up to the granularity where it is possible to identify, which software service directly corresponds to which business sub-process.

The approach is discussed in greater detail in [19].

Figure 2 illustrates to some extent the approach presented in this subsection by a simplified fragment of travel agency processes.

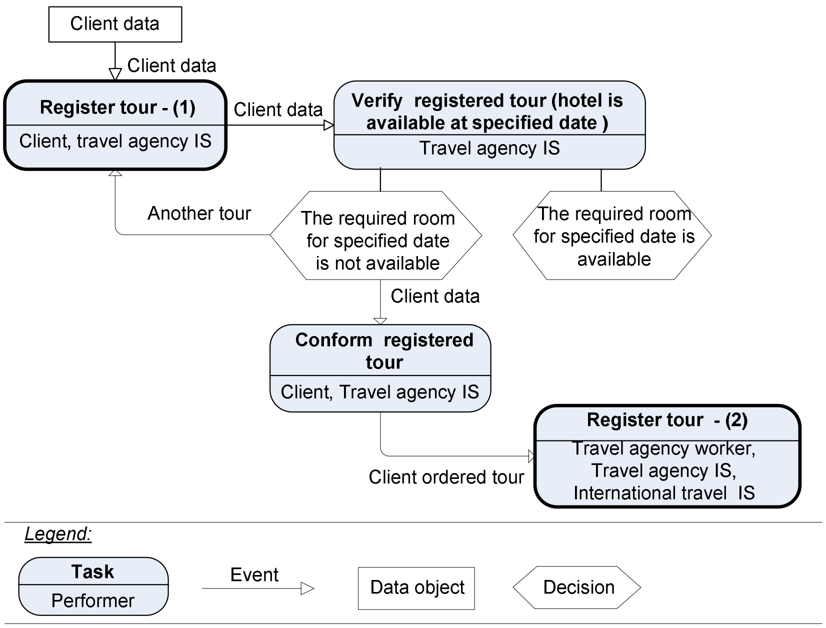

Fig. 2. A simplified fragment of a travel agency business process model

The main goal of the agency is to receive and process client orders, which contain the required tour code, hotel name and tour date [20] Analyzing the travel agency's business processes, a fractal business process "Register tour" was identified. It has common or similar goals, similar inputs and outputs on two scales: inter-organizational scale (Register tour 1) and travel agency scale (Register tour 2). According to similarities and differences in functional and non-functional requirements [20], software service "Register tour" corresponds to fractals identified in business process model Figure 3. The service supports business process flexibility in the sense that, if there are changes in the registration process, both business processes are supported by corresponding changes in one particular software unit.

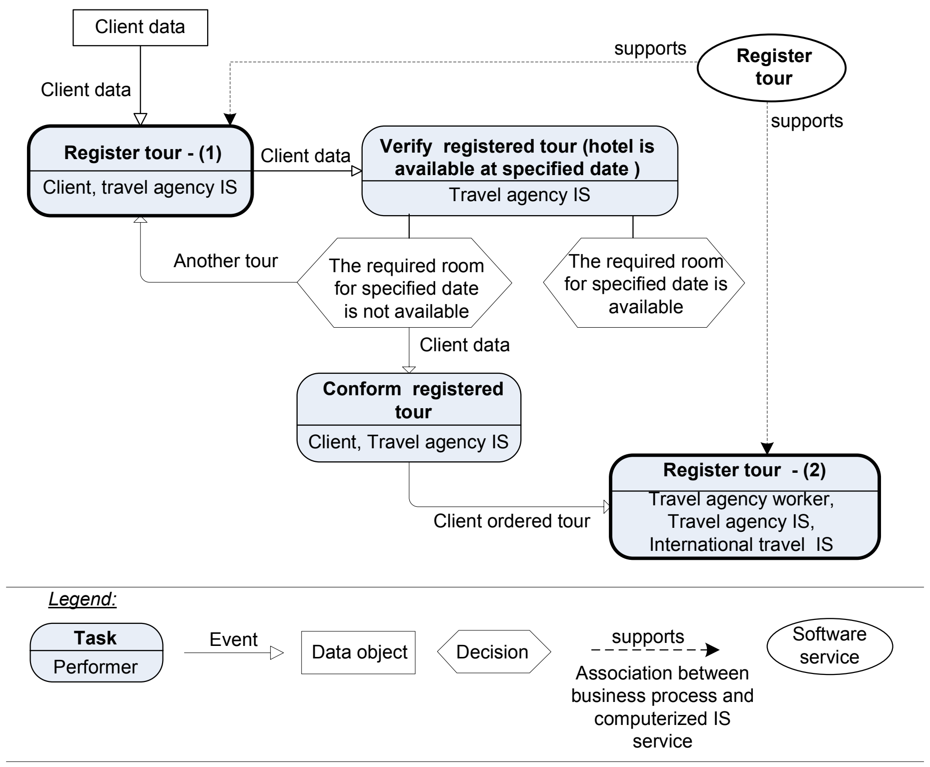

Fig. 3. Correspondence between business processes and software services

4 Fractal Software for Supporting Process Flexibility: An Object-Oriented Approach

The main idea of an object-oriented method is to create architecture of the system as polymorphic as possible. The suggested method includes the following three stages [21, 22]:

- Stage 1 – analysis of the system organization and behavior. The following are the main activities of Stage 1:
  - Define organizational structure of the system. Analysis of organizational structure facilitates the identification of actors and use cases. This relates to that in object-oriented analysis and design, system analysis starts with the identification of system's actors and use cases.
  - Define actors and their goals, use cases that are necessary for the achievement of those goals, and operation contracts, i.e., define the behavior of the information system under consideration. Use cases characterize the behavior of the software system that produces a measurable result of value to an actor. Use cases are goal-oriented. Additionally, one or several use cases can be used for the same

goal. Actors are humans or roles of other computer systems in an organization. Besides that, actors activate the execution of use cases. Thus, after the identification of *actors*, their *functional goals* are determined. In accordance with the identified functional goals, *use cases* needed for the goal achievement are identified. Use cases are specified by their scenarios that are main flow and alternate flow (in case of errors or exceptions) description of each use case. Here it is important to identify the so-called *operation contracts* for each actor's request. Each contract describes one operation with the specified name, parameter list, *references to other use case that use the same operation, pre and post conditions.* The result of this activity is a use case model and use case descriptions for each organizational level "fractal".
  - Define a conceptual model. In addition the third activity is identification of concepts in the system's description. Concepts are ideas, things, or objects. Concepts and their relationships should be defined in the initial model of concepts.
- Stage 2 – definition of fractal scale invariants and organization. Scale invariants are properties of the system that do not change with the change of scale. The following are the main activities of Stage 2:
  - Define behavioral scale invariants. Initially the interactions in each identified use case should be analyzed and may be specified using UML interaction diagrams – sequence or collaboration diagrams. During the analysis of interactions, it is important to define messages that are sent and received by objects on different scales while acting towards the achievement of the same goal.
  - Define structural scale invariants. To define structural scale invariants, it is necessary to analyze the data of the structure. In fractal systems, structural data that are related to the fractal structure can be candidates to structural scale invariants. Those candidates that must be presented on all scales of the fractal system are separated as attributes of a fractal class (classes).
  - Define fractals' architecture. To support fractal's property "self-organization", a designer should define architecture (organization) of fractals by mapping it to the organizational structure of the system in the real world.
- Stage 3 – design of fractal classes and interfaces. Since the object-oriented approach supports inheritance and platform-independent modeling supports implementation of a defined behavior and structure at different platforms, it is useful to define all shared fractal related aspects as a distinct class – a fractal class. A *fractal class* is a class that specifies similar structure and functionality of fractals of the same kind. A *fractal interface* is an interface that should be implemented by a class that specifies a fractal. The interface specifies those scale invariants that are mandatory for specific kinds of fractals.
- Stage 4 – design of fractal classes' behavior on distinct scales. During Step 4 classes (and their behavior), which correspond to organizational scales and inherit fractal classes, must be defined. Here the important point is to make sure that all differences in operation implementations are taken into account.

A simplified use case model detected in Stage 1 is represented in Figure 4.

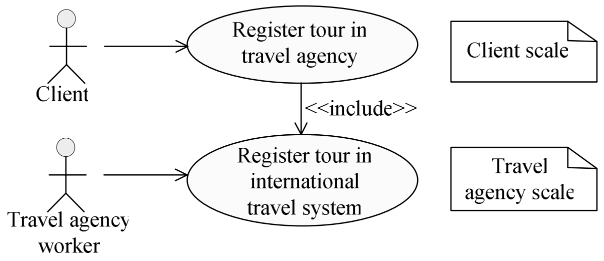
Client
Register tour in travel agency
Client scale
<<include>>
Travel agency worker
Register tour in international travel system
Travel agency scale

Fig. 4. Use case model for process "Order tour"

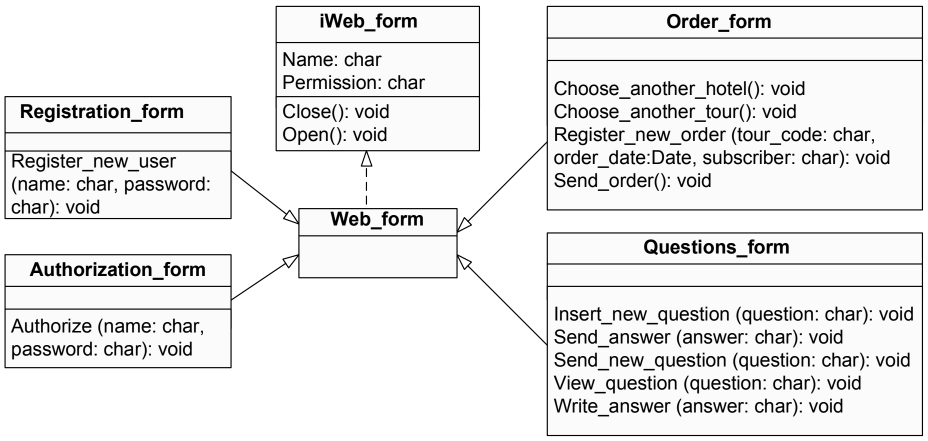
iWeb_form
Name: char
Permission: char
Close(): void
Open(): void
Registration_form
Register_new_user (name: char, password: char): void
Authorization_form
Authorize (name: char, password: char): void
Web_form
Order_form
Choose_another_hotel(): void
Choose_another_tour(): void
Register_new_order (tour_code: char, order_date:Date, subscriber: char): void
Send_order(): void
Questions_form
Insert_new_question (question: char): void
Send_answer (answer: char): void
Send_new_question (question: char): void
View_question (question: char): void
Write_answer (answer: char): void

Fig. 5. Fractal interface iWeb_form

One of the fractal classes interfaces defined after analyzing use cases in Stage 3 is represented in Figure 5.

Figure 5 illustrates the fractal interface class *iWeb_form*. The fractal class *Web_Form* implements this interface. The interface *iWeb_form* specifies that a class, which realizes it, must contain information about its name and access permission and must perform its opening and closing if necessary. Hence, classes *Authorization_form*, *Registration_form*, *Order_form* and *Questions_form* inherit this responsibility and may realize this responsibility in their specific way. Interfaces of fractals support rapid propagation of changes in the behavior of agents from the business level to information system's functionality.

5 Discussion and Conclusions

Section 3 and 4 of this paper illustrate that, using fractal approach, several ways of business process flexibility support can be identified. The solutions depend on the

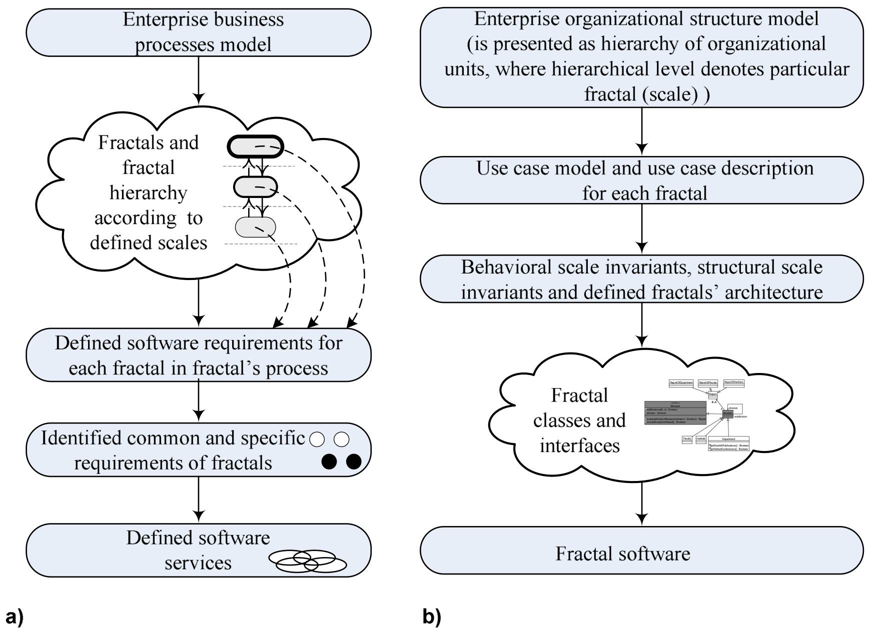
Enterprise business processes model
Fractals and fractal hierarchy according to defined scales
Defined software requirements for each fractal in fractal's process
Identified common and specific requirements of fractals
Defined software services
a)
Enterprise organizational structure model (is presented as hierarchy of organizational units, where hierarchical level denotes particular fractal (scale))
Use case model and use case description for each fractal
Behavioral scale invariants, structural scale invariants and defined fractals' architecture
Fractal classes and interfaces
Fractal software
b)

Fig. 6. Differences between process-oriented (a) and object-oriented (b) fractal approaches for information systems design

level of abstraction or decomposition where fractality of the system is identified or defined. Both approaches are compared in Figure 6. The process-oriented approach (discussed in Section 3) seeks to, first, identify fractal properties at the business process level and then structures software requirements according to identified fractal processes. In software systems design this allows to distinguish between more frequently and less frequently changing parts of the business process. The object-oriented approach (discussed in Section 4) analyzes goals of actors and seeks to identify fractals at the computerized information processing level thus introducing the fractal software architecture for business system support. Both object-oriented paradigm and fractal paradigm support flexibility. The contribution of a fractal paradigm in the object-oriented approach is in the utilization of self-similarity, i.e. fractals can organize themselves in order to perform some task due to their self-similarity. Flexibility in the object-oriented paradigm is effect of the realization of the "polymorphism" principle where the units that implement this principle may not be self-similar. It means that the object-oriented paradigm provides a technique to implement a certain property of fractals, namely, their self-similarity.

Process-oriented and object-oriented approaches were applied to 15 different medium complexity business systems by 15 information systems designers [23]. While theoretically it would be possible that both approaches suggest similar software for business process support, it was not so in any of 15 cases. Almost in all cases seeking fractals at the business process level led to service-oriented software architectures for business system support, while seeking fractals at the level of software processes led to object-oriented software architectures. With the process-oriented approach two different situations were observed: in some cases identified software services corresponded to the basic functionality of the business process, but in most cases these were "satellite" services such as saving data in the data base, data transfer from one location to another, etc. In addition, the depth of the analysis of processes was greater than the process taxonomies used in several sources devoted to fractal approaches in systems design (see Section 2). In the object-oriented approach, fractality of data in information flows was taken into consideration while in the process-oriented approach only intuitively perceivable similarity of information flows was analyzed. Thus the object-oriented approach utilized two and the process-oriented approach utilized one fractal hierarchy of three essential business systems dimensions represented in Figure 1.

The usefulness of the fractal process- and object-oriented approaches of system design was perceived differently by different designers. Two basic criteria they used in the evaluation of approaches were 1) compatibility of the method with their previous knowledge and 2) simplicity of defining fractal subsystems in the particular application domain. Thus practitioners of the object-oriented approach perceived the object-oriented method to be easier than the process-oriented one, and vice versa: designers used to process analyses preferred the process-oriented approach. In almost all cases it was possible to identify multi-scale "fractals" of software hierarchy. However, not all business processes exhibited this property. For instance, fractals were not found in VCOR framework [24], which initially seemed to be a very appropriate reference base for the use of process-oriented approach.

One of the expectations of the use of a fractal paradigm for supporting business processes was cutting of the time for user and software requirements definition. However, none of the designers reported shorter time for requirements acquisition when using a fractal paradigm in the systems development. Almost all pointed to the fact that the use of a fractal paradigm forces the requirements engineer to gather more detailed requirements and pay more attention to a variety of requirements with respect to different levels of scale in identified "fractals".

On the whole, the use of a fractal paradigm for support of business process flexibility has the following benefits:

- The process-oriented approach makes it possible to define software services that support fractal business processes and thus utilize flexibility support opportunities provided by service-oriented architectures.
- The object-oriented approach makes it possible to define flexible object-oriented software designs to support the user requirements of business process participants.
- Both approaches foster acquisition of detailed requirements with respect to individual differences at different scales of "fractals" in the business process or software respectively.

The application of a fractal paradigm is limited only to those business process and software subsystems in the enterprise, which exhibit fractal properties in at least one modeling dimension (agent, process or information architecture). The agent perspective was not analyzed in depth in the paper while it could to some extent contribute to both process- and object-oriented approaches. Actually, consideration of agents' fractality imposes analysis of agents' goals on a higher level of abstraction, i.e., not only with respect to software use as it was in Section 4. That suggests slightly different process- and object-oriented approaches to information systems design for support of enterprise flexibility and agility. Another direction of further research is the investigation of possibilities to integrate process- and object-oriented approaches in order to support business process flexibility simultaneously from both business and software perspectives.

References

1. Chapman, M., Berman, S., Blitz, A.: Foreword by Tushman M.: Rethinking innovation. Fast Thinking (2008)
2. Warneke, H.J.: The Fractal Company. Springer, Heidelberg (1993)
3. Kirikova, M., Strazdina, R., Grundspenkis, J., Osis, J.: Analysis of business process flexibility at different levels of abstraction. In: The 9th International Conference on Enterprise Information Systems: Software Agents and Internet Computing, Funchal, Madeira, INSTICC, Portugal, pp. 389–396 (2007) ISBN 978-972-8865-91-7
4. Kirikova, M.: Toward multi-fractal approach in IS development. In: 16th International Conference on Information Systems Development Challenges in Practice, Theory and Education (ISD 2007), Galway, Ireland (in print, 2008)
5. Bruneton, E., Coupaye, T., Stefani, J.B.: The Fractal Component Model. Specification, The ObjectWeb Consortium (2004)
6. Fryer, P., Ruis, J.: What are fractal systems: A brief description of complex adaptive and emerging systems (2006) (accessed April 14, 2007), http://www.fractal.org
7. Ramanathan, Y.: Fractal architecture for the adaptive complex enterprise. Communications of ACM 48(5), 51–67 (2005)
8. Ryu, K., Jung, M.: Fractal approach to managing intelligent enterprises: Creating Knowledge Based Organizations. In: Gupta, J.N.D., Sharma, S.K. (eds.), pp. 312–348. Idea Group Publishers (2003)
9. Canavesio, M.M., Martinez, E.: Enterprise modeling of a project-oriented fractal company for SMEs networking. Computers in Industry (2007), http://www.sciencedirect.com, doi:10.1016/j.compind.2007.02.-05
10. Hongzhao, D., Dongxu, L., Yanwei, Z., Chen, Y.: A novel approach of networked manufacturing collaboration: fractal web based enterprise. Int. Journal on Advanced Manufacturing Technology 26, 1436–1442 (2005)
11. Tharumarajah, A., Wells, A.J., Nemes, L.: Comparison of emerging manufacturing concepts. Systems, Man and Cybernetics 1, 325–331 (1998)
12. Sihn, W.: Re-engineering through fractal structures. In: IFIPWG5.7 working conference Reengineering the Enterprise, Ireland, pp. 21–30 (1995)
13. Cottam, R., Ranson, W., Vounckx, R.: Life and simple systems. Systems Research and Behavioral Science 22(5), 413–430 (2005)
14. Regev, G., Soffer, P., Schmidt, R.: Taxonomy of Flexibility in Business Processes (2006), http://lamswww.epfl.ch/conference/bpmds06/taxbpflex
15. Daoudi, F., Nurcan, S.: A Benchmarking Framework for Methods to Design Flexible Business Processes. In: Software Process Improvement and Practice, pp. 51–63 (2007)
16. Snowdon, R.A.: On the Architecture and Form of Flexible Process Support. In: Software Process Improvement and Practice. John Wiley & Sons, Chichester (2006)

17. Regev, G., Wegmann, A.: Business Process Flexibility: Weick's Organizational Theory to the Rescue (2006), http://lamswww.epfl.ch/conference/bpmds06/program/Regev_13.pdf
18. Chitchyan, R., Rashid, A., Sawyer, P.: Survey of Analysis and Design Approaches, OSD-Europe EU Network of Excellence (2005), http://www.comp.lancs.ac.uk/computing/aop/papers/d11.pdf
19. Stecjuka, J., Kirikova, M.: The process-oriented fractal information system development method. In: 14th International Conference on Information and Software Technologies (IT 2008), pp. 171–181. Kaunas University of Technology (2008)
20. Astahova, T.: Course work in Requirements Engineering. Riga Technical University, Riga (2007) (in Latvian)
21. Asnina, E., Osis, J., Kirikova, M.: Design of fractal-based systems within MDA: platform independent modeling. In: 3rd EuroSIGSAND Symposium 2008, Marburg/Lahn. GI-LNI P-129, pp. 39–53. Koellen-Verlag (2008)
22. Asnina, E., Osis, J.: Analyzing of multifractal system properties in object-oriented software development. In: 48th International Riga Technical University Conference, Scientific Proceedings of RTU, Series — Computer Science (5), Applied Computer Systems. RTU, Riga (2007)
23. Deliverables of project Nr. R7199 Development of fractal information systems design methodologies, Riga, RTU (2007) (in Latvian)
24. VCOR framework (2007), http://www.value-chain.org/