Moving Across Paradigms between the Process Design and Enactment Phase in Enterprise Information Systems

Filip Caron and Jan Vanthienen
Leuven Institute for Research on Information Systems (LIRIS), Faculty of Business and Economics, KU Leuven,
Naamsestraat 69, 3000 Leuven, Belgium

Keywords: Process-aware Information Systems, Business Process Management, Process Modeling, Process Enactment, Transitions, Enterprise Information Systems.

Abstract: While the business process management literature often assumes a single approach (e.g. procedural or event-driven) over the process lifecycle, a transition between approaches at different phases in the process lifecycle may significantly reduce the impact of intrinsic trade-offs between process characteristics. This position paper explores several business process strategies by analyzing the approaches at different phases in the process lifecycle as well as the various transitions.

1 INTRODUCTION

Organizations face a continuous pressure to improve process compliance, flexibility, efficiency and effectiveness. While responding to these pressures individually can be demanding, the real challenge is dealing with the intrinsic tradeoffs (e.g. between compliance, efficiency and flexibility). A wide spectrum of business process management paradigms has been presented in the literature, each with its specific tradeoffs. Additionally, most of these contributions consider the selection of the optimal approach given the business environment as a one-time choice at the process design phase. However, business processes may also require different tradeoffs at different phases of the process lifecycle.

The contribution of this position paper will be the exploration of various business process strategies that combine the selection of a design-time paradigm and a run-time paradigm (i.e. position selection) with a transition path, resulting in a better fit between the business processes and the business environment. Rather than making a value judgement of the different strategies, we will focus on a discussion of the impact on the process characteristics.

The remainder of this paper is structured as follows: first we introduce and assess the different positions (section 2 & 3), followed by the specification and analysis of the transition strategies (section 4) and section 5 concludes the paper.

2 THE BUSINESS PROCESS PARADIGMS AND LIFECYCLE

Each position in the process lifecycle is defined by the combination of the related lifecycle phase and process paradigm.

Traditionally, the traditional business process lifecycle consists of four phases with distinct roles (Weske, 2007), i.e. process design phase, process implementation phase, process enactment phase and process evaluation phase. The design phase and the enactment phase straightforwardly belong to respectively the design-time and run-time. As the process analysis and evaluation phase often results in specific recommendations/requirements for a process redesign, it can be considered as part of the next cycle's design-time. The implementation phase can be associated with run-time as it is the implemented process that will be executed.

The business process management literature has proposed a wide spectrum of process paradigms with at the extremes the procedural (Zisman, 1977; OMG, 2006; Ellis and Nutt, 1993) and declarative (Goedertier and Vanthienen, 2009; van der Aalst et al., 2009b; Swenson, 2010) process paradigm. In between there exists a wide variety of hybrid paradigms that combine aspects of both extremes (Sadiq et al., 2005; van der Aalst et al., 2009a; Schonenberg et al., 2008; Kumar and Yao, 2009; Hallerbach et al., 2010).

Caron F. and Vanthienen J..
Moving Across Paradigms between the Process Design and Enactment Phase in Enterprise Information Systems.
DOI: 10.5220/0003999202180223
In *Proceedings of the 14th International Conference on Enterprise Information Systems* (ICEIS-2012), pages 218-223
ISBN: 978-989-8565-12-9

3 ASSESSING DESIGN-TIME AND RUN-TIME POSITIONS

Since every position has its own design or execution principles, all positions have different characteristics and consequently appeal to different business requirements. In (Davenport, 1993) Davenport identifies four desirable qualities for business processes; process flexibility, compliance, effectiveness and efficiency. This section analyzes the possible impact of each position's principles on these characteristics.

- **Process flexibility** is the extent to which an organization can deal with business process change, the ability to accommodate the special needs of particular business process instances (i.e. run-time flexibility) as well as to accommodate process model evolutions (i.e. maintainability).
- **Process compliance** is the extent to which a process is in correspondence with the internally defined business rules and the externally imposed business regulations.
- **Process effectiveness** is the extent to which a business process realizes its business goals.
- **Process efficiency** is the extent to which the organization of the business process is capable of minimizing the amount of utilized resources such as personnel, materials, time, machine capacity.

Additionally, two important characteristics of process modeling languages are taken into account for the evaluation of the design-time positions:

- The **expressibility** of a process modeling language is determined by its ability to express specific process elements, e.g. control-flow, data, execution and temporal information (Lu and Sadiq, 2007; zur Muehlen et al., 2007)
- The level of **comprehensibility** reflects the ability of a process modeling language to define understandable process models that can be easily communicated among various stakeholders (Fahland et al., 2009a).

Tables 1 and 2 represent the assessment results of respectively the design-time and run-time positions.

4 DESIGN-TIME TO RUN-TIME TRANSITIONS: DEFINITION AND ASSESSMENT

Traditional business process management solutions are oriented towards a single process paradigm, e.g. the business processes are modeled using procedural process modeling languages (such as BPMN (OMG, 2006)) and then executed in a procedural enactment environment (such as BPEL (OASIS, 2007)). These design-time to run-time transitions within a single process paradigm are the *same paradigm transitions*. There also exist business processes that have different requirements at design-time and run-time, e.g. in terms of flexibility. In these cases we analyze design-time to run-time transitions between process paradigms, the *cross paradigm transitions*.

Different process lifecycle phases may require different tradeoffs between process characteristics. Consequently, cross paradigm transitions may in practice result in a better fit between the business process-aware information systems and the business environment.

4.1 Same Paradigm Transitions

Procedural - Procedural Transition. Different transformation strategies between procedural process modeling languages and procedural process execution languages have been proposed in the literature (Decker et al., 2008; Ouyang et al., 2009). However, due to a *conceptual mismatch* between the standard procedural process modeling and execution languages, translation techniques are generally only offered for a core subset of the procedural process modeling constructs (Recker and Mendling, 2006).

Impact on Process Characteristics. A set of research contributions on *process variant management* (based on querying process variants before run-time) falls into the scope of the procedural-procedural transition, e.g. (Lu et al., 2009). While this procedural-procedural transition might have a positive impact on the process flexibility, managing changing compliance requirements can be challenging for repositories with large collections of process variants. The impact on process efficiency and effectiveness depends on the quality of the process variants and the number of different situations captured by the process variants in the repository. In (Reichert et al., 2009) a technique is presented for adding, replacing and moving activities.

Example Process Case. The procedural-procedural transition can be recommended for business processes in a *stable environment with predictable execution paths*, e.g. processing standard and static items (such as online orders).

Declarative - Declarative Transition. The common declarative process modeling languages have their roots in formal logic. Due to this formal foundation the translation of a high-level declarative process

Table 1: Design-Time Positions Analysis.

	Procedural process modeling	**Declarative process modeling**	**Hybrid process modeling**
Description	Exact specification of all execution paths, events and exceptions.	Set of constraints, business rules, event conditions or other (logical) expressions that define mandatory properties and dependencies between activities.	Combination of procedural and declarative constructs resulting in process models with placehoder activities or procedural process models with rule-based adaption
Impact on process characteristics	*Positive*: Comprehensible models, process efficiency *Negative*: Overspecification (Sadiq et al., 2005), maintainability issues (Fahland et al., 2009b)	*Positive*: Flexibility, compliance by design, maintainability (traceability and absence of duplication), high expressibility (Lu and Sadiq, 2007) *Negative*: Limited comprehensibility (large and unstructured sets of formal rules) (Fickas, 1989)	*Moderation of impact*: Improved flexibility and reduced maintainability issues (declarative placeholder activities), Improved efficiency and reduced flexibility (procedural placehoder activities), improved flexibility (rule-based adaption)
Languages examples	Petri Net (based) modeling (Zisman, 1977; Ellis and Nutt, 1993), BPMN (OMG, 2006) and UML Activity Diagram (OMG, 2004)	ConDec (Pesic and van der Aalst, 2006), DecSerFlow (van der Aalst and Pesic, 2006) and BPCN (Lu et al., 2009)	Placeholder activities (Sadiq et al., 2005; van der Aalst et al., 2009a; Schonenberg et al., 2008), rule-based adaption (Kumar and Yao, 2009; Hallerbach et al., 2010)

Table 2: Run-Time Positions Analysis.

	Procedural process enactment	**Declarative process enactment**	**Hybrid process enactment**
Description	Straightforward execution based on precisely specified execution paths	An individual execution path that satisfies the set of mandatory constraints is dynamically created	The base process is executed according to the principles of the paradigm, whenever a placeholder activity is executed a paradigms switch takes place
Impact on process characteristics	*Positive*: high efficiency *Negative*: limited flexibility	*Positive*: high flexibility, assured compliance *Negative*: little support, limited efficiency	Characteristics of process parts are determined by their paradigm. Flexibility as a service improves flexibility
Language examples & execution environments	BPEL (OASIS, 2007) and YAWL (van der Aalst and Ter Hofstede, 2005)	LTL (Pesic et al., 2008), CTL (Yu et al., 2006), PLMflow (Zeng et al., 2002), ECA rules (Kappel et al., 1998) and event rules (Paschke and Boley, 2009)	Chameleon (Sadiq et al., 2005) and subprocesses encapsulated in a service (van der Aalst et al., 2009a)

model into enactable rules is *rather straightforward* (e.g. translation of a ConDec process model into LTL expressions (Pesic et al., 2008)).

Impact on Process Characteristics. During the declarative-declarative transition no (additional) factors that affect the business process's flexibility, compliance, efficiency or effectiveness will/can be introduced.

Example Process Case. These transitions are suitable for business processes in a *highly evolving environment* and/or business processes with *non-predictable execution paths*. As declarative process management systems might provide limited support at run-time (Weber et al., 2009), this transition type will be most suited for experts dealing with unique cases (Schmidt, 2006) (e.g. non-standardized health care processes).

Hybrid - Hybrid Transition. The hybrid-hybrid transition can only be applied on hybrid process models that contain placeholder activities. Within the context of this type of hybrid paradigm, *the business process paradigms of each process parts determine which same paradigm transition will be used for that* *process part*.

Impact on Process Characteristics. As these transitions are intrinsically similar to the previously described same strategy transitions, we argue that the impact on the process characteristics of this transition is determined by those same strategy transitions. It should be noted that process variant management approaches in this context will be easier to maintain than those presented in the context of procedural-procedural transitions, since the base structure is not duplicated.

Example Process Case. Hybrid-hybrid transitions will be used for business processes that contain *both process parts with stable and highly evolving environments* and/or that consist of *both process parts with predictable and non-predictable execution paths* (e.g. an advisory project with structured administrative activities and unstructured problem solving parts).

4.2 Cross-Paradigm Transitions

While the impact of the same paradigm transitions on the desirable characteristics of a business process is rather limited, the impact of the cross-paradigm transitions can be rather extensive.

Procedural - Declarative Transition. The procedural process model is translated into a set of event-based business rules (e.g. preconditions), which can be used for a declarative process enactment (Casati et al., 1998).

Impact on Process Characteristics. Since the implicit constraints governing the procedural process model are exactly mapped on event-based business rules, the *issue of overspecification is not dealt with*. However, *process flexibility slightly increases* compared to the procedural-procedural transition due to the possibility of a run-time replacement or addition of a service task and the ability to define extra event-based business rules to deal with temporary circumstances (Dumas et al., 2005). The process effectiveness and efficiency are determined by the quality of the process model and the declarative execution environments.

Example Process Case. The procedural-declarative transition is suitable for use within the context of *distributed processes*, for which the process environment remains relatively stable and the ability to dynamically deal with temporary circumstances is valued (e.g. order-to-cash process in virtual organizations).

Declarative - Procedural Transition. Before run-time a systematic procedure is used for the construction of an optimal control-flow with reference to a particular characteristic. This procedure is closely related to artificial intelligence planning techniques (Hendler et al., 1990; Ferreira and Ferreira, 2005).

Impact on Process Characteristics. While the declarative process specifications provide *extensive design-time flexibility, run-time flexibility remains limited* to the flexibility offered by procedural enactment. However, the declarative process model in combination with a time-efficient planning algorithm, allows for a *rapid adoption of new compliance requirements*. Moreover, when the procedural workflow engine does not support any of the run-time flexibility enhancing techniques, compliance can be easily checked against the declarative process model.

The use of an artificial planning algorithm might positively affect both the *process efficiency and effectiveness*, since an optimization criterion needs to be specified. In addition, compared to declarative process enactment the end-user will be sufficiently guided and supported.

Example Process Case. The declarative-procedural transition is useful for processes that require *far-reaching redesigns* at regular intervals and/or for processes that benefit from an *optimization* with reference to a certain criterium. These process, however, are at the same time relatively *stable in the periods between those redesign phases*.

Hybrid - Procedural Transition. Within the context of this transition the focus is primarily placed on hybrid models of the second type, the process models that combine a full procedural specification with a set of business rules. Before run-time the procedural reference model is *customized to the specific needs* of a particular case by applying the set of customizing business rules (Kumar and Yao, 2009; Hallerbach et al., 2010).

Impact on Process Characteristics. Due to the hybrid process model as a starting position, a neat approach to process variant management is provided. Compared to the process variant management approach introduced in the procedural-procedural transition, *maintenance of requirements is not needlessly complicated* since there is no duplication of the base process. However, the customization must be performed correctly and completely in order not to affect the process effectiveness and compliance.

Example Process Case. This type of hybrid - procedural transition will most likely be used for supporting a set of business processes that all only slightly differ from a specific reference process. We also expect the business processes to have predictable execution paths (e.g. claim handling for different insurance products).

Translating Declarative Placeholders. Naturally, hybrid process models with placeholder activities can be transformed into procedural execution models as described in the declarative-procedural transition, which also results in a hybrid-procedural transition. Characteristics of the transition in this context are comparable with those of the declarative-procedural transition.

5 CONCLUSIONS

Designing information systems that provide support for operational business processes with the right level of process flexibility, compliance, efficiency and effectiveness can be a challenging task. This position paper promotes a clear distinction between the business process strategies and their differences at distinct points in the process life cycle. Furthermore, the paper elaborated on the transitions between design-time and run-time; in addition to the same paradigm transitions three interesting cross-paradigm transitions were presented.

REFERENCES

Casati, F., Ceri, S., Pernici, B., and Pozzi, G. (1998). Deriving active rules for workflow enactment. In *Database and Expert Systems Applications*, pages 94–115. Springer.

Davenport, T. (1993). *Process innovation: reengineering work through information technology*. Harvard Business Press.

Decker, G., Dijkman, R., Dumas, M., and García-Bañuelos, L. (2008). Transforming BPMN diagrams into YAWL nets. *Business Process Management*, pages 386–389.

Dumas, M., Fjellheim, T., Milliner, S., and Vayssière, J. (2005). Event-Based Coordination of Process-Oriented Composite Applications. *Business Process Management*, pages 236–251.

Ellis, C. and Nutt, G. (1993). Modeling and enactment of workflow systems. *Application and Theory of Petri Nets 1993*, pages 1–16.

Fahland, D., Lubke, D., Mendling, J., Reijers, H., Weber, B., Weidlich, M., and Zugal, S. (2009a). Declarative versus imperative process modeling languages: The issue of understandability. *Enterprise, Business-Process and Information Systems Modeling*, pages 353–366.

Fahland, D., Mendling, J., Reijers, H., Weber, B., Weidlich, M., and Zugal, S. (2009b). Declarative vs. Imperative Process Modeling Languages: The Issue of Maintainability. In *1st International Workshop on Empirical Research in Business Process Management*, pages 65–76, Ulm, Germany.

Ferreira, D. and Ferreira, H. (2005). Learning, planning, and the life cycle of workflow management. In *EDOC Enterprise Computing Conference, 2005 Ninth IEEE International*, pages 39–45. IEEE.

Fickas, S. (1989). Design issues in a rule-based system. *Journal of Systems and Software*, 10(2):113–123.

Goedertier, S. and Vanthienen, J. (2009). *An overview of declarative process modeling principles and languages*, volume 6, pages 51–58. Communications of systemics and informatics world network.

Hallerbach, A., Bauer, T., and Reichert, M. (2010). Capturing variability in business process models: the provop approach. *Journal of Software Maintenance and Evolution: Research and Practice*, 22(6-7):519–546.

Hendler, J., Tate, A., and Drummond, M. (1990). AI planning: Systems and techniques. *AI magazine*, 11(2):61.

Kappel, G., Rausch-Schott, S., and Retschitzegger, W. (1998). Coordination in workflow management systems a rule-based approach. *Coordination Technology for Collaborative Applications*, pages 99–119.

Kumar, A. and Yao, W. (2009). Process Materialization Using Templates and Rules to Design Flexible Process Models. In *Proceedings of the 2009 International Symposium on Rule Interchange and Applications*, pages 122–136. Springer-Verlag.

Lu, R. and Sadiq, S. (2007). A survey of comparative business process modeling approaches. In *Business Information Systems*, pages 82–94. Springer.

Lu, R., Sadiq, S., and Governatori, G. (2009). On managing business processes variants. *Data & Knowledge Engineering*, 68(7):642–664.

OASIS (2007). Web services business process execution language 2.0.

OMG (2004). Uml 2.0 superstructure specification.

OMG (2006). Business process modeling notation (bpmn) 1.1.

Ouyang, C., Dumas, M., Aalst, W., Hofstede, A., and Mendling, J. (2009). From business process models to process-oriented software systems. *ACM transactions on software engineering and methodology (TOSEM)*, 19(1):1–37.

Paschke, A. and Boley, H. (2009). *Handbook of Research on Emerging Rule-Based Languages and Technologies: Open Solutions and Approaches*, chapter Rules capturing events and reactivity. IGI Publishing.

Pesic, M., Aalst, W., and Eijnatten, F. (2008). *Constraint-based workflow management systems: shifting control to users*. PhD thesis, Technische Universiteit Eindhoven.

Pesic, M. and van der Aalst, W. (2006). A declarative approach for flexible business processes management. In *Business Process Management Workshops*, pages 169–180. Springer.

Recker, J. and Mendling, J. (2006). On the translation between bpmn and bpel: Conceptual mismatch between process modeling languages. In *The 18th International Conference on Advanced Information Systems Engineering. Proceedings of Workshops and Doctoral Consortium*, pages 521–532.

Reichert, M., Rinderle-Ma, S., and Dadam, P. (2009). Flexibility in process-aware information systems. *Transactions on Petri Nets and Other Models of Concurrency II*, pages 115–135.

Sadiq, S., Orlowska, M., and Sadiq, W. (2005). Specification and validation of process constraints for flexible workflows. *Information Systems*, 30(5):349–378.

Schmidt, R. (2006). Flexibility in service processes. In *Proceedings of the CAISE 2006 Workshop on Business Process Modelling, Development and Support, BPMDS*.

Schonenberg, H., Mans, R., Russell, N., Mulyar, N., and Aalst, W. (2008). Process flexibility: A survey of contemporary approaches. *Advances in Enterprise Engineering I*, pages 16–30.

Swenson, K. (2010). Mastering the unpredictable: How adaptive case management will revolutionize the way that knowledge workers get things do.

van der Aalst, W., Adams, M., Hofstede, A., Pesic, M., and Schonenberg, H. (2009a). Flexibility as a Service. In *Database Systems for Advanced Applications*, pages 319–333. Springer-Verlag.

van der Aalst, W. and Pesic, M. (2006). DecSerFlow: Towards a truly declarative service flow language. *Web Services and Formal Methods*, pages 1–23.

van der Aalst, W., Pesic, M., and Schonenberg, H. (2009b). Declarative workflows: Balancing between flexibility and support. *Computer Science-Research and Development*, 23(2):99–113.

van der Aalst, W. and Ter Hofstede, A. (2005). YAWL: yet another workflow language. *Information Systems*, 30(4):245–275.

Weber, B., Sadiq, S., and Reichert, M. (2009). Beyond rigidity–dynamic process lifecycle support. *Computer Science-Research and Development*, 23(2):47–65.

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

Yu, J., Manh, T., Han, J., Jin, Y., Han, Y., and Wang, J. (2006). Pattern based property specification and verification for service composition. *Web Information Systems–WISE 2006*, pages 156–168.

Zeng, L., Flaxer, D., Chang, H., and Jeng, J. (2002). PLM flowDynamic Business Process Composition and Execution by Rule Inference. *Technologies for E-Services*, pages 51–95.

Zisman, M. (1977). *Representation, specification and automation of office procedures*. PhD thesis, Wharton School.

zur Muehlen, M., Indulska, M., and Kamp, G. (2007). Business process and business rule modeling: A representational analysis. In *EDOC Conference Workshop, 2007. EDOC'07. Eleventh International IEEE*, pages 189–196. IEEE.