Process Flexibility: A Design View and Specification Schema

Udo Kannengiesser

NICTA, Locked Bag 9013, Alexandria NSW 1435, Australia, and
School of Computer Science and Engineering, University of New South Wales, Sydney, Australia
udo.kannengiesser@nicta.com.au

Abstract: This paper proposes a framework of process flexibility based on a view of processes as design objects. It is represented using the function-behaviour-structure (FBS) ontology of designing. The paper shows how the FBS ontology allows extending and generalising recent work on flexibility in engineering design, and how it allows applying this work to processes. The resulting framework provides a comprehensive account of process flexibility that subsumes existing approaches. Finally, the paper presents a specification schema for process flexibility, illustrated using examples of a property valuation process in the Australian lending industry.

1 Introduction

Flexible modelling of processes is a key issue for the effective use of process-aware information systems (PAIS) in dynamic business environments [WSR09]. Factors such as market or strategy changes, technological innovations and new regulations often require modifications of a process. Furthermore, unforeseen events in the immediate environment of the process need to be handled flexibly, such as resource bottlenecks or effects of unexpected human or system errors. PAIS using process models that are too rigidly specified are poorly applicable in real-world contexts and are ultimately rejected by their users.

Research has been concerned with understanding, modelling and implementing the notion of process flexibility. The taxonomies and methods resulting from these efforts address a wide range of aspects of flexibility [PV06, RSS06, DN07, Re07]. However, there is no coherent, comprehensive framework of process flexibility, as most approaches have been developed independently of each other. An attempt to providing such a framework has recently been undertaken by [Sc08], proposing a general taxonomy of process flexibility associated with different realisation approaches.

Flexibility in PAIS is often viewed as a balance between the freedom to change and the need for stability [Re07]. This balance is also an inherent characteristic of designing: Designers aim to change parts of the world through their designs, balancing the use of their individual perception and creativity, and the need to comply with requirements and constraints. A view of process performers as process re-designers has been well described by [Va07]. This paper explores this design view of flexibility, aiming to establish broader, interdisciplinary foundations for understanding and specifying process flexibility. This provides the basis for augmenting rather than replacing existing frameworks of process flexibility.

Section 2 introduces an ontology of designing, the function-behaviour-structure (FBS) ontology, that can be applied to any object of designing, no matter whether this object is a physical product, a software product, or a process. The FBS ontology is then used to extend and generalise recent work on flexibility in engineering design. This provides the basis for a mapping between the design view of flexibility and existing frameworks of process flexibility. Section 3 derives a schema for specifying process flexibility based on the FBS ontology. Examples from the domain of real estate valuation illustrate the use of this schema. Section 4 concludes the paper.

2 A Design View of Process Flexibility

2.1 The Function-Behaviour-Structure View of Designing

Design objects can be modelled using the FBS ontology [GK04, GK07] that has been applied to various instances of design objects, including physical products [GK04], software [Kr05] and processes [GK07].

Structure (S) is defined as a design object's components and their relationships. It can be viewed as the final outcome of a design process. In the domain of physical products, structure comprises the geometry, topology and material of individual components or assemblies. The structure of software consists of abstract constructs such as classes, components and pieces of code. In the domain of processes, structure includes three general classes of components: input, transformation and output [GK07]. The transformation often consists of a set of sub-transformations, some of which can be viewed as "micro-level" mechanisms that represent the "materials" of the transformation. For example, a sequence of activities concerned with logging into an online banking system, and filling out and submitting a funds transfer form can be viewed as a process-centred "material" of a payment transformation [Ka08]. Other "materials" are object-centred; they refer to the agent performing the transformation. Process-centred and object-centred "materials" map onto Dietz' notions of realisation and implementation of a process, respectively [Di06]. Structure encompasses control-flow, data, resource, and task views of a process [Ka08].

Behaviour (B) is defined as the attributes that can be derived from a design object's structure. They provide criteria for comparing and evaluating different design objects. An example of a physical product's behaviour is "weight", which can be derived (or measured) from the product's structure properties of material and spatial dimensions. Behaviour of software (e.g., a text editor) includes its response time for visualising user input. It can be derived from software structure and its interaction with the operating environment. Typical behaviours of processes include speed, cost, precision and accuracy. They can be derived from process structure; for example, speed can be derived from (time-stamped) input and output.

Function (F) is defined as a design object's teleology ("what it is for"). This notion is independent of the common distinction between "functional" and "non-functional" properties; it comprises both as they describe the design object's usefulness for a stakeholder (or "using system" [Di06]). It should not be confused with the concept of "transfer function". Function is ascribed to behaviour by establishing a teleological connection between a human's goals and measurable effects of the design object. There is no direct connection between function and structure. The particular functions of a design object are ontologically independent of whether the design object's structure is conceptualised as a physical product, a software product or a process. For example, the functions "wake people up", "be reliable" and "be punctual" may be ascribed to relevant behaviours of a mechanical alarm clock (i.e., a physical product), a virtual alarm clock (i.e., software), or a sequence of activities (i.e., a process).

From a high-level perspective, designing can be viewed as decision making. This view implies the existence of choices [Ge94] that can be represented as alternative values for the variables of the design. The set of all design variables and their ranges of values form what is called the design state space, i.e. the space of all possible designs. The design state space is partitioned into three subspaces: function state space, behaviour state space, and structure state space, as shown in Figure 1. The three subspaces are interconnected through the designer's compiled knowledge of qualitative and quantitative relationships between function, behaviour and structure. These relationships are the basis for modelling designing as an activity that aims to produce structure that exhibits suitable behaviour to which desired function can be ascribed.

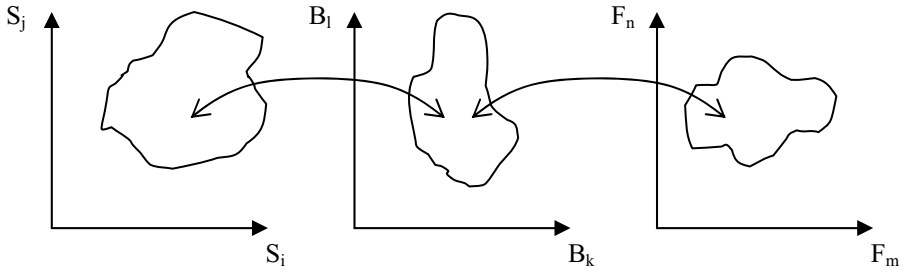

Figure 1: Function, behaviour and structure state spaces, and their interconnections

The notion of a design state space allows understanding designing through the use of spatial metaphors. Selecting values for a set of design variables can be described as a process of moving through the design state space. This model of designing is often referred to as *search* [Ge94]. However, most designing involves more than moving through a well-defined space of known design alternatives. It also involves generating the space in terms of design variables and their ranges of values. This can involve discarding some previously expected variables or ranges of values, leading to a shift of the design state space. The notion that addresses these changes is called *exploration* [Ge94]. Changes may affect all three subspaces, and changes of one subspace may lead to changes of a subspace connected to it. For example, a change of the structure state space may lead to changes of the behaviour state space. This, in turn, may lead to subsequent changes of the function state space and/or the structure state space.

Expectations about the design problem are fundamental in distinguishing exploration from search. Exploration reflects changed expectations as the designer learns more about the design problem by interacting with it [Sc83]. In contrast, the notion of search reflects unchanged expectations of the design (state space). Some of the initial design expectations are formulated through explicitly stated requirements. Others arise from the designer's understanding of the design object's socio-technical environment across the life cycle. The possible change of a design state space from its inception to a later point in time is presented conceptually in Figure 2. The increasing size of the design state space is to indicate that a great deal of the knowledge required to produce a design is constructed during designing [LS93].

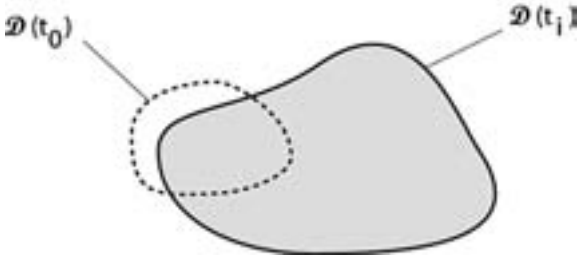

Figure 2: A design state space ($\mathcal{D}$) changes between the initial time t_0 and a later time t_i

The application of the state space concept in PAIS has commonly had a more narrow focus. It is usually understood only as a representation of the set of possible changes in the world that may occur during the execution of a process instance, not as a representation of the set of all possible process designs. Recent work on business rules in process models [Di08] can be viewed as expanding the scope of state space models of processes. However, these approaches are limited to the notion of a structure state space, and do not cover behaviour and function state spaces.

Design state spaces can be represented in many ways. The most common representations include enumerations of states (e.g., a set of alternative product modules or process fragments), generative representations (e.g., grammars), constraints, and abstraction (e.g., using types defined in domain ontologies).

2.2 Generalising an Engineering Design Approach to Flexibility

Flexibility has been a popular concept in many areas within engineering design. Similar to process flexibility, it is understood here as the ability of a product or system to handle change. However, more precise definitions of this notion are often missing [SHN03]. One of the most comprehensive approaches to defining and characterising flexibility in engineering design has recently been proposed by researchers from the MIT Engineering Systems Division [RRH08]. This Section provides an overview of relevant concepts of this work, and expands and generalises them using the FBS ontology. [RRH08] propose two aspects of flexibility of a design object: change effects, and change mechanisms.

Change effects characterise the difference between the states of a design object before and after its change. States are described in terms of variables and values that may refer to any aspect of the design object, including function, behaviour and structure. There are three categories of change effects: robustness, scalability, and modifiability.

Robustness is the ability to maintain the design object's required functions without changing its structure, despite the presence of changes affecting the object's internal or external environment [RRH08]. For example, a car may achieve its function of transportation without changing its design, despite internal changes such as tire abrasion or external changes such as altered road conditions. Robustness handles change by being insensitive to it. In fact, it has been understood as a concept that is related but quite distinct from flexibility [SHN03].

Scalability is the ability to vary the design object's state in terms of the values of its variables [RRH08]. For example, varying the length, width and height of a mobile phone is a scalable change of structure. Varying the speed of a central processing unit is a scalable change of behaviour. And varying the reliability of an alarm clock is a scalable change of function. Scalability is captured in the state space representation of designing as either search (if the change remains within state space boundaries) or exploration (if the change involves crossing a state space boundary in terms of ranges of values).

Modifiability is the ability to vary the design object's state in terms of its variables [RRH08]. For example, the addition of a DVD burning module to a computer is a modifiable change of structure. Changing a car's petrol consumption rate to rapeseed oil consumption rate is a modifiable change of behaviour. And augmenting a mobile phone with the ability to play MP3 files is a modifiable change of function. Modifiability is captured in the state space representation of designing as exploration via changing the set of variables.

Variations of function, behaviour and structure rarely occur in isolation of each other. As outlined in Section 2.1, a change of one subspace often leads to changes of other subspaces. The specific ways in which a change propagates across the subspaces depends not only on given requirements and constraints but also on the individual expertise and interpretations of the designer. We can view this "design freedom" as an additional dimension of flexibility, embedded within the notion of change effects.

Change mechanisms represent different ways of achieving the desired change effects. [RRH08] propose the number of possible change mechanisms, filtered by a subjective acceptability threshold for their "cost", as a basis for quantifying flexibility. Here, "cost" is an aggregated measure for the consumption of various resources including time and money.

We can expand the notion of change mechanisms by defining three categories, Figure 3: design goal achievement, design realisation, and design assessment.

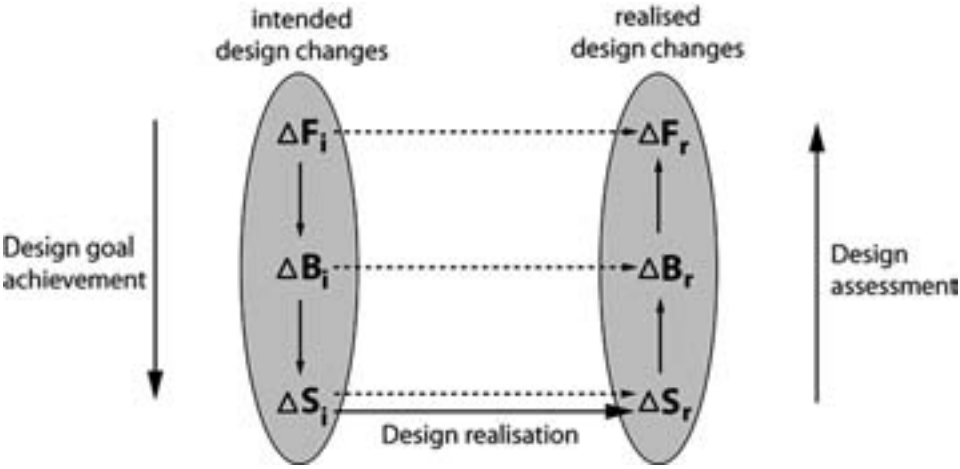

Figure 3: Three categories of change mechanisms: design goal achievement, design realisation, and design assessment.

Design goal achievement is the ability to vary the activities and resources required for transforming intended changes of function (ΔF_i) into intended changes of structure (ΔS_i) via intended changes of behaviour (ΔB_i). For example, besides generating design changes from scratch, one may have the option of reusing previous design knowledge captured in patterns, best practices, rationale, case bases, prototypes or other forms of representation. Each of these options requires different technologies and user skills.

Design realisation is the ability to vary the activities and resources required for transforming an intended change of structure (ΔS_i) into a realised change of structure (ΔS_r). This includes the allocation of agents (machines or people), their coordination and any setup tasks required (for example, programming, instructing and training). It also includes strategies for preventing or mitigating issues, such as downtime and obsolete work items arising from changes in the realisation.

Design assessment is the ability to vary the activities and resources required for monitoring, analysing and validating the success (for example, the consistency, correctness and efficiency) of the change. This includes the methods and tools available for deriving changes of behaviour (ΔB_r) from a realised change of structure (ΔS_r), and ascribing changes of function (ΔF_r) to these changes of behaviour.

2.3 Applying the Design View to Process Flexibility

The three categories of change effects (robustness, scalability, and modifiability) and the three categories of change mechanisms (design goal achievement, design realisation, and design assessment) can be mapped onto existing research in flexible PAIS.

Robustness maps onto "flexibility by design" [Sc08] or "flexibility by definition" [SSO01] that is the ability to include multiple execution paths in the process model at design time. They represent different ways of dealing with anticipated variations and exceptions occurring in the execution environment. The different paths are selected at runtime for individual process instances.

Scalability maps onto two categories of process flexibility proposed by [Sc08] that imply the existence of expected choices. One is "flexibility by deviation" [Sc08] that is the ability of a process instance to deviate from the original process model (type) without altering it. It encompasses only changes in the execution sequence of tasks, not the tasks themselves. We conceptualise the ordering relationships that determine the execution sequence as a set of interrelated "ports" of the tasks [Go08]. Specifically, every task has variables for their "inflow ports" and "outflow ports", and the values of these variables are pointers to other tasks. Changing the relationships can then be viewed as varying the values of structure variables. The other category of process flexibility corresponding to scalability is "flexibility by underspecification" [Sc08] or "flexibility by templates" [SSO01] via late binding of process fragments to a placeholder. This category of process flexibility is the ability to execute an incomplete process model by completing it at runtime, via selection from a pre-defined set of process fragments. The fragments can be represented as structure variables with Boolean values. A process fragment with the value "false" means that this fragment is currently not selected. Changing the value to "true" corresponds to selecting it to instantiate the placeholder. Potential subjects of scalable change include not only control flow but also other aspects of process structure [RSS06].

Scalable changes of process function and process behaviour are not included in existing work on process flexibility. Examples include improving maintainability of a process (a scalable change of function), and reducing the cost of a process (a scalable change of behaviour).

Modifiability maps onto two categories of process flexibility proposed by [Sc08] that imply a shift of expectations. One is "flexibility by underspecification" [Sc08] via late modelling, which is the ability to construct a new process fragment for a placeholder. It is best thought of as the generation of a new variable to be introduced in the structure state space of the process. The other category mapping onto modifiability is "flexibility by change" [Sc08], which is the ability to modify a process at runtime, in response to unforeseen circumstances in the execution environment. Here, changes represent new tasks being introduced and/or removed, affecting process instances and/or process types. By representing every task as a structure variable, we can model these changes as modifications of the structure state space. Potential subjects of modifiable change include not only control flow but also other aspects of process structure [RSS06].

Modifiable changes of process function and process behaviour are not included in most existing work on process flexibility. Examples include considering the waste production of a manufacturing process in addition to other process attributes (a modifiable change of behaviour), and changing the goal of a transportation process from "people transportation" to "cargo transportation" (a modifiable change of function). There is work on using variations of quality goals to generate different configurations of process structure [e.g., LYM07].

Design goal achievement is addressed by the range of methodologies for process design. Methodologies differ in their notations, their coverage of different process views, their technological support, and their ease of use. A number of existing technologies, including ADEPT1, YAWL, FLOWer and Declare, have been evaluated by [Sc08] regarding their support for the different change effects.

Design realisation subsumes migration strategies for running process instances, and mechanisms for version, access and concurrency control, which are described in [WRR08]. Design realisation also includes methods and technologies for communication and "setup" (instructing, training, etc.).

Design assessment includes methods and technologies for analysing correctness, consistency, efficiency, traceability, usability and other process quality attributes, which are described in [WRR08].

3 Specifying Process Flexibility

3.1 A Schema for Process Flexibility

The framework presented in Section 2 can be used as the basis for a schema for flexible process specification, whose central notion is the design state space with its three subspaces for function, behaviour and structure. We have presented this notion as the set of all possible designs based on the expectations and experience of the individual designer. When we adopt a prescriptive stance, a design state space becomes the set of all "permitted" designs according to specifications given to the process designer. Here, the boundaries of the design state space, both in terms of the specified set of variables and their ranges of values, represent requirements that may be socially enforceable. In fact, the specified design state space represents a "normative restriction of design freedom" [Di06].

On the other hand, as shown in Figure 2, parts of the initial specification of a design state space may be relaxed over the course of designing, resulting in a new design state space with modified boundaries. Therefore, different degrees of "normative strength" of the state space boundaries should be made explicit to specify what parts of the space may be changed and what parts must not. This can be realised by associating individual design variables and their ranges of values with modality attributes such as "mandatory" and "optional" (or a finer-grained set of attributes such as proposed by [BS06]).

Table 1 shows the resulting specification schema. The columns represent FBS level, state space, and modality. The contents of the white boxes in this Table are suggested approaches to representing these notions in a way that is easy to comprehend for readers of this paper. This is the reason why we use abstraction for representing the structure state space, despite the lack of well-defined, formal domain ontologies for most PAIS. Abstraction is also likely to be useful in phases of requirements elicitation and process definition, where domain experts negotiate a common, high-level view of the process. More formal representations, such as declarative (constraint-based) notations of process structure, are certainly needed in the later phases of process execution and process analysis.

Table 1: A specification schema for process flexibility

FBS level	State space	Modality
F	Enumeration of functions	[mandatory, optional]
B	Constraints	[mandatory, optional]
S	Abstraction	[mandatory, optional]

The schema can be used for specifying not only change effects but also change mechanisms. This is because change mechanisms can themselves be viewed as design objects that can be described using the FBS ontology. Change effects and change mechanisms are then represented in two separate sets of specifications but using the same schema. However, in practice only few aspects of change mechanisms are specified explicitly. These are typically aspects related to design realisation, rather than design goal achievement and design assessment. As outlined in Section 2.1, these aspects can be captured as "materials" of process structure.

3.2 Examples

This Section demonstrates the use of the specification schema for a number of (sub-) processes in the context of property valuation (short: valuation) in the Australian lending industry. Figure 4 shows a simplified model of the valuation process in BPMN (Business Process Modeling Notation; see www.bpmn.org). The process starts when the valuation company receives a request from a lender (e.g., a bank) to assess the market value of a specific property. An employee (called the "valuer") is then assigned to perform the valuation by inspecting the property and preparing a valuation report that contains the estimated market value of the property. After that, the valuation report is sent to the lender, and, concurrently, an invoice is sent. Upon receipt of payment, the valuation process terminates. This process model is assumed to be fixed; we will specify flexibility only for some of the sub-processes in this model.

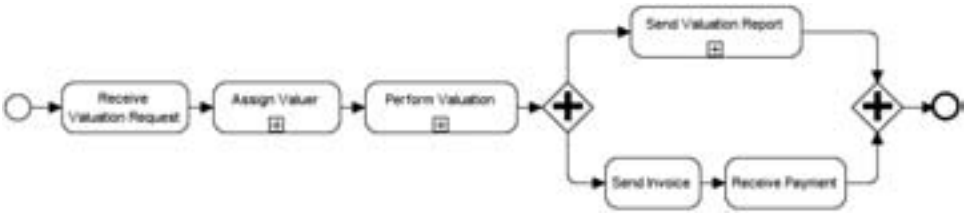

Figure 4: Top-level model of a property valuation process

Figure 5 shows the details of the sub-process "Perform Valuation". It includes multiple paths that handle cases in which valuation fees need to be renegotiated due to complicated site conditions, such as irregular building shapes or slopes. This is an example of robustness, as the designed process structure is insensitive to changes in (external) site conditions.

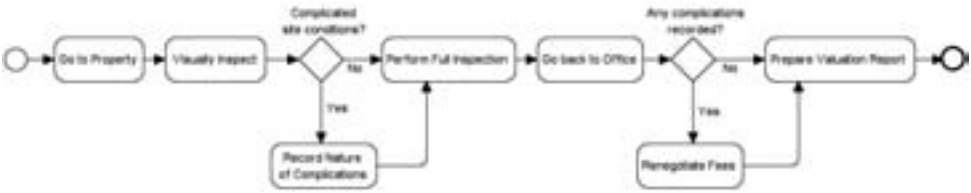

Figure 5: Sub-process "Perform Valuation"

Flexibility within the valuation process can be specified using annotations that are structured according to the schema in Table 1. Figure 6 shows three examples.

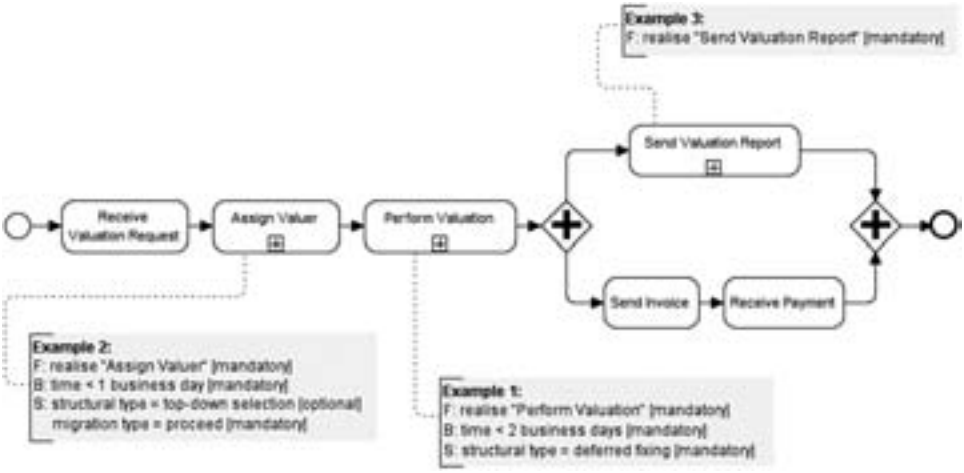

Figure 6: Examples of specifications of sub-process flexibility within the valuation process

Example 1 in Figure 6 is a flexible specification of the sub-process "Perform Valuation". Here, scalable changes of process behaviour are allowed within the mandatory range of values specified for "time". The set of process structures that can produce these variations are specified as a mandatory "structural type" referencing "deferred fixing", which is an exception-handling pattern proposed by [Le08]. The basic idea behind this pattern is that an exceptional situation (here, the complicated site conditions) is identified and recorded, but dealt with (here, by renegotiating fees) later in the process. This abstract description leaves room for various changes in the execution sequence, all of which are scalable changes of process structure. One instance of process structure consistent with this specification was shown in Figure 5. Another instance is shown in Figure 7; here, the position of the fee renegotiation activity within the sub-process is altered. The specification in this example does not constrain the change mechanisms that can be chosen to realise the change effect.

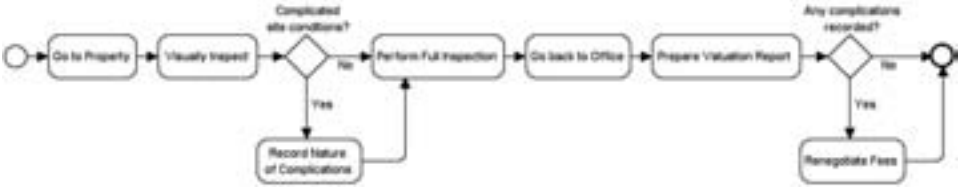

Figure 7: Sub-process "Perform Valuation", alternative process structure

Example 2 in Figure 6 is a flexible specification of the sub-process "Assign Valuer". Behaviour is again specified as a mandatory range of values for "time". Structure includes an optional "structural type" referencing a "top-down selection" procedure such as shown in Figure 8. Suppose we want to drastically reduce the time behaviour to, say, 1 hour instead of 1 business day. This scalable change of behaviour requires a change of structure that is likely to exceed the limits defined in the specification of structure. Since the modality attribute of the structure is "optional" rather than "mandatory", modifiable changes can be made resulting in a process structure such as shown in Figure 9. Here, the "top-down selection"-type is substituted with a "bidding"-type process structure that can more quickly identify those potential valuers that are currently located near the property to be valuated. This process is inspired by the way taxi companies dynamically assign incoming customer requests to specific drivers. This change of process structure requires further changes such as the development and implementation of appropriate information and communication technologies (not shown in the BPMN model). The mechanism for design realisation is specified as a "migration type" that refers to a mandatory "proceed" strategy [Sc08] for migrating existing process instances.

Figure 8: Sub-process "Assign Valuer", using a "top-down selection"-type process structure

Figure 9: Sub-process "Assign Valuer", using a "bidding"-type process structure

Example 3 in Figure 6 is a flexible specification of the sub-process "Send Valuation Report". Here, only the principal function is specified, leaving a great deal of freedom for designing the sub-process. Suppose the designer wants to make a modifiable change of function by including an additional function "provide interoperable data". This is a very realistic scenario as the Australian lending industry is moving towards interoperable, straight-through processing based on the standard Credit Application Language (CAL) currently being defined by the LIXI consortium (Lending Industry XML Initiative; see www.lixi.org.au). The additional function may lead to a new behaviour variable termed "LIXI compliance", and new structure variables including "output file type = XML" and "output vocabulary type = CAL". These are modifiable changes of behaviour and structure. The example does not specify any constraints on how these changes are to be realised in terms of change mechanisms.

4 Conclusion

A design view of process flexibility leverages a characteristic that is inherent in the nature of designing: the capacity to operate within a space of alternatives that is generated based on not only a set of requirements but also the designer's individual understanding of the problem. The framework presented in this paper expands and generalises a recent approach from engineering design, using a domain-independent ontology of designing. Current approaches to modelling process flexibility fit in this framework, but fall short of covering essential aspects including function and behaviour. These aspects capture "what should be done without specifying how it should be done" [PV06] in a more comprehensive way than existing declarative approaches that are limited to process structure. The design view provides a unifying framework that brings together different research streams in flexible PAIS, most of which can be categorised as focusing on either change effects or change mechanisms.

The proposed specification schema can be used to define the flexibility of a process at different design-ontological levels and with different degrees of "normative strength". It supports the view of stakeholders as designers or re-designers of the process, while constraining their design activities using a necessary and sufficient set of specifications. The examples in this paper demonstrate a range of process designs that can be generated based on different process specifications. One issue that is not addressed here is the need to move from one state space representation to another. The light-weight approach we utilised for representing structure state spaces is useful for high-level communication among domain experts; however, for process implementation a more formal approach is needed based on well-defined domain ontologies and executable process notations. Progress in this area is likely to draw on research in process patterns [Va03], configurable process models [Go08] and semantic business process management [We07].

There are opportunities to expand the design view of process flexibility, using further analogies from engineering design. One research direction may focus on the qualitative relationships between function, behaviour and structure state spaces. Capturing them can guide process designers realising desired changes of function through appropriate changes of behaviour and structure. For example, research in product family design maps different types of product families (that can be modelled as sets of scalable or modifiable behaviours and structures) onto strategies for targeting different market segments (that can be modelled as sets of scalable or modifiable functions) and onto different manufacturing paradigms (that can be modelled as design realisation options) [MF07]. Research in PAIS is likely to benefit from further investigation of these analogies, as they allow tapping into well established methodologies of flexibility from various domains.

Acknowledgements

NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Centre of Excellence program.

Bibliography

[BS06] Borch, S.E.; Stefansen, C.: On Controlled Flexibility. In (Latour, T.; Petit, M., Eds.): Proceedings of Workshops and Doctoral Consortium, The 18th International Conference on Advanced Information Systems Engineering – Trusted Information Systems, Namur University Press, Namur, 2006, pp. 121-126.

[DN07] Daoudi, F.; Nurcan, S.: A Benchmarking Framework for Methods to Design Flexible Business Processes. Software Process: Improvement and Practice, 12(1), 2007, pp. 51-63.

[Di06] Dietz, J.L.G.: Enterprise Ontology: Theory and Methodology, Springer-Verlag, Berlin, 2006.

[Di08] Dietz, J.L.G.: On the Nature of Business Rules. In (Dietz, J.L.G.; Albani, A.; Barjis, J., Eds.): Advances in Enterprise Engineering I, LNBIP 10, Springer-Verlag, Berlin, 2008, pp. 1-15.

[Ge94] Gero, J.S.: Towards a Model of Exploration in Computer-Aided Design. In (Gero, J.S.; Tyugu, E., Eds.): Formal Design Methods for CAD, North-Holland, Amsterdam, 1994, pp. 315-336.

[GK04] Gero, J.S.; Kannengiesser, U.: The Situated Function-Behaviour-Structure Framework. Design Studies, 25(4), 2004, pp. 373-391.

[GK07] Gero, J.S.; Kannengiesser, U.: A Function-Behavior-Structure Ontology of Processes. Artificial Intelligence for Engineering Design, Analysis and Manufacturing, 21(4), 2007, pp. 379-391.

[Go08] Gottschalk, F. et al.: Configurable Workflow Models. International Journal of Cooperative Information Systems, 17(2), 2008, pp. 177-221.

[Ka08] Kannengiesser, U.: Subsuming the BPM Life Cycle in an Ontological Framework of Designing. In (Dietz, J.L.G.; Albani, A.; Barjis, J., Eds.): Advances in Enterprise Engineering I, LNBIP 10, Springer-Verlag, Berlin, 2008, pp. 31-45.

[Kr05] Kruchten, P.: Casting Software Design in the Function-Behavior-Structure Framework. IEEE Software, 22(2), 2005, pp. 52-58.

[Le08] Lerner, B.S. et al.: Exception Handling Patterns for Processes. In (Garcia, A. et al., Eds.): Proceedings of the 4th International Workshop on Exception Handling, Atlanta, GA, 2008, pp. 55-61.

[LS93] Logan, B.; Smithers, T.: Creativity and Design as Exploration. In (Gero, J.S.; Maher, M.L., Eds.): Modeling Creativity and Knowledge-Based Creative Design, Lawrence Erlbaum, Hillsdale, 1993, pp. 139-175.

[LYM07] Lapouchnian, A.; Yu, Y.; Mylopoulos, J.: Requirements-Driven Design and Configuration Management of Business Processes. In (Alonso, G.; Dadam, P.; Rosemann, M., Eds.): Business Process Management, LNCS 4714, Springer-Verlag, Berlin, 2007, pp. 246-261.

[MF07] Maier, J.R.A.; Fadel, G.M.: A Taxonomy and Decision Support for the Design and Manufacture of Types of Product Families. Journal of Intelligent Manufacturing, 18(1), 2007, pp. 31-45.

[PV06] Pesic, M.; Van der Aalst, W.M.P.: A Declarative Approach for Flexible Business Processes Management. In (Eder, J. et al., Eds.): BPM 2006 Workshops, LNCS 4103, Springer-Verlag, Berlin, 2006, pp. 169-180.

[Re07] Regev, G.; Bider, I.; Wegmann, A.: Defining Business Process Flexibility with the Help of Invariants. Software Process: Improvement and Practice, 12(1), 2007, pp. 65-79.

[RRH08] Ross, A.M.; Rhodes, D.H.; Hastings, D.E.: Defining Changeability: Reconciling Flexibility, Adaptability, Scalability, Modifiability, and Robustness for Maintaining System Lifecycle Value. Systems Engineering, 11(3), 2008, pp. 246-262.

[RSS06] Regev, G.; Soffer, P.; Schmidt, R.: Taxonomy of Flexibility in Business Processes. In (Latour, T.; Petit, M., Eds.): Proceedings of Workshops and Doctoral Consortium, The 18th International Conference on Advanced Information Systems Engineering – Trusted Information Systems, Namur University Press, Namur, 2006, pp. 90-93.

[Sc83] Schön, D.A.: The Reflective Practitioner: How Professionals Think in Action, Harper Collins, New York, 1983.

[Sc08] Schonenberg, H. et al.: Process Flexibility: A Survey of Contemporary Approaches. In (Dietz, J.L.G.; Albani, A.; Barjis, J., Eds.): Advances in Enterprise Engineering I, LNBIP 10, Springer-Verlag, Berlin, 2008, pp. 16-30.

[SHN03] Saleh, J.H.; Hastings, D.E.; Newman, D.J.: Flexibility in System Design and Implications for Aerospace Systems. Acta Astronautica, 53(12), 2003, pp. 927-944.

[SSO01] Sadiq, S.; Sadiq, W.; Orlowska, M.: Pockets of Flexibility in Workflow Specification. In (Kunii, H.S.; Jajodia, S.; Solvberg, A., Eds.): Conceptual Modeling – ER 2001, LNCS 2224, Springer-Verlag, Berlin, 2001, pp. 513-526.

[Va03] Van der Aalst, W.M.P. et al.: Workflow Patterns. Distributed and Parallel Databases, 14(3), 2003, pp. 5-51.

[Va07] Van Aken, J.E.: Design Science and Organization Development Interventions: Aligning Business and Humanistic Values. Journal of Applied Behavioral Science, 43(1), 2007, pp. 67-88.

[We07] Wetzstein, B. et al.: Semantic Business Process Management: A Lifecycle Based Requirements Analysis. In (Hepp, M. et al., Eds.): Semantic Business Process and Product Lifecycle Management. Proceedings of the Workshop SBPM 2007, Innsbruck, Austria, 2007, pp. 1-10.

[WRR08] Weber, B.; Reichert, M.; Rinderle-Ma, S.: Change Patterns and Change Support Features – Enhancing Flexibility in Process-Aware Information Systems. Data & Knowledge Engineering, 66(3), 2008, pp. 438-466.

[WSR09] Weber, B.; Shazia, S.; Reichert, M.: Beyond Rigidity – Dynamic Process Lifecycle Support. Computer Science – Research and Development, 23(2), 2009, pp. 47-65.