A Framework and Methodology for Enterprise Process Type Configurations

Peter Bollen

School of Business and Economics
Maastricht University, Tongersestraat 53,
6200MD, Maastricht, the Netherlands
p.bollen@maastrichtuniversity.nl

Abstract. In this paper we will present the results of research into the semantics of modeling constructs for the process-oriented perspective for the conceptual modeling of enterprise subject areas. We will distinguish 3 conceptual process types that will be the building blocks for any enterprise process base. The definition of these conceptual process types will be anchored in existing process- and decision making frameworks within the fields of management information systems and the administrative sciences

Keywords: Conceptual Modeling, Process Modeling, Process Configurations, Business Process Modeling

1 Introduction

In [1, 2] we have introduced a number of conceptual process configurations in organizations. In this paper we will apply the framework, thereby introducing a modelling methodology for the process-oriented perspective and we will relate our conceptual framework to earlier decision-making frameworks from the fields of (management) information systems and administrative sciences. Furthermore, we will present the meta-model for our process modelling language; the meta-process model (see figure 1).

In this paper we will derive the semantic bridges that we need for instantiating the process modeling constructs (as defined in [1, 2]), in an enterprise subject area under the restriction that they are 'compatible' with the models for the data-oriented perspective in the fact-based approach [3, 4]. In line with the IFIP-CRIS framework [5] we will assume that an application model for the data-oriented perspective is available (see figure 1). Subsequently, we can derive a model for the process-oriented perspective that will use the model in the 'data-oriented' perspective as a starting point, thereby constraining the possible 'process-oriented' models that can exist for the application area and respecting the borders of the application that are imposed by the *Universe of Discourse* (UoD) in the 'data-oriented' perspective.

The remainder of this paper is organized as follows: in section 2 the methodology for instantiating the process modeling constructs in a specific application area will be given, in section 3 some methodological backgrounds will be provided. In section 4 a

J. Barjis, M.M. Narasipuram, G. Rabadi, J. Ralyté, and P. Plebani (Eds.):
CAiSE 2010 Workshop EOMAS'10, Hammamet, Tunisia, pp. 14-29, 2010.

process modelling procedure will be given and in section 5 the meta process model will be given, finally, in section 6 conclusions will be given.

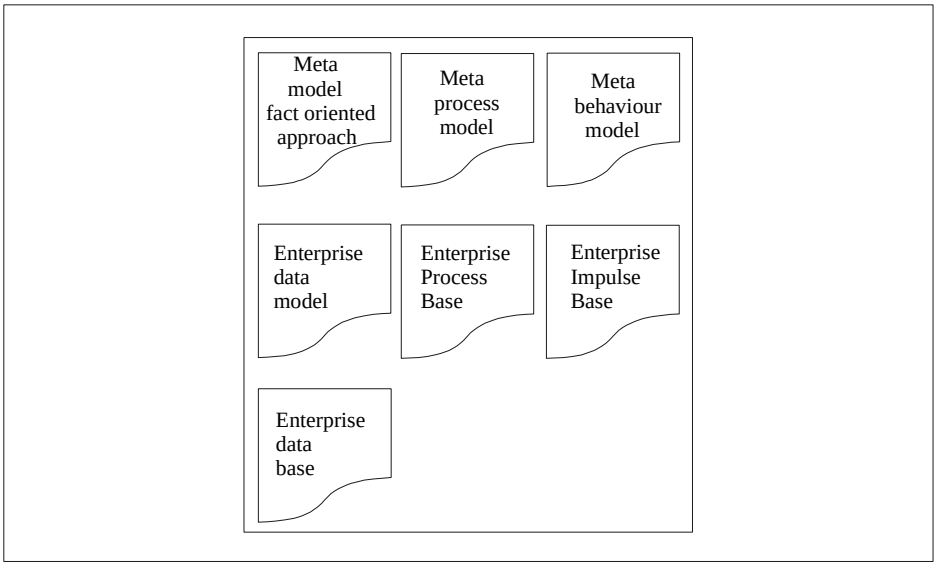

Fig. 1. Documents in the data-, process- and behaviour-oriented perspectives.

In the (information- and knowledge-) management literature different definitions of 'process' can be found. Davenport [6] defines a process as "…a structured, measured set of activities designed to produce a specified output for a particular customer or market.". Hammer and Champy [7] define a process as : "..a collection of activities that takes one or more kinds of input and creates an output that is of value to the customer." Nickols [8] addresses the issues of identification and analysis of business processes as follows: "..identification and analysis of business processes must be anchored to something concrete."

1.1 The Enterprise Process Type Configurations in our Framework

In this article we will 'anchor' the concept of a conceptual (business) process to the 'tangible' result of such a business process in terms of 'knowledge'. In line with Nickols our position is that processes are not discrete sets of related activities but rather selected portions of larger streams of activity. In figure 2 we have summarized the different conceptual process configurations that we have introduced in [1, 2]. We will provide the definitions of these conceptual process types here.

Definition 1. A *derivation process type* is a conceptual process type whose process instances create fact instances by applying the same derivation rule on instances of the same ingredient fact type(s) (from the enterprise data model).

Definition 2. A *mixed determination process type* is a conceptual process type in which the fact generator uses instances of the same ingredient fact types (that are contained in the application's data model) for all process instances.

Definition 3. A *strict-determination process type* is a conceptual process type in which the fact generator does not use a known derivation rule all the time and the fact generator does not use instances of the same ingredient fact types (that are contained in the application's data model) in all process instances.

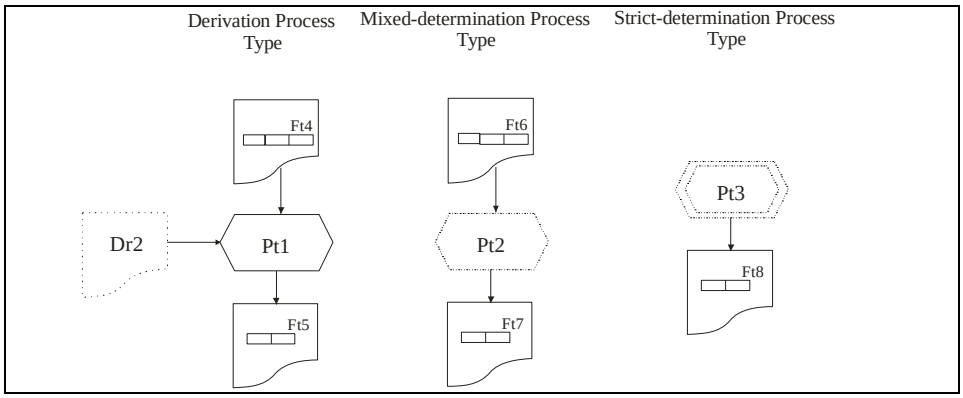

Fig. 2. Conceptual process configuration types (in[1]).

2 Typology of Process Types in Management Information Systems and Knowledge Management

2.1 The Gorry and Scott-Morton Framework

In the field of *management information systems*, Gorry and Scott-Morton [9] have introduced a typology for managerial decision making. In their framework, managerial decisions can considered to be *structured*, *semi-structured* or *unstructured*. This framework was based on the work of Simon who made a distinction into 'programmed' and 'non-programmed' decisions [10]: " The basis for these differences is that in the unstructured case the human decision maker must provide judgement and evaluation as well as insight into problem definition. In a very structured situation, much if not all of the decision-making process can be automated." [9]. Although Gorry and Scott-Morton, refer to managerial decision making in a broad sense, we will interpret their framework in the context of active users that create fact instances as 'materilization' of a decision making process.

Definition 4. A conceptual process type is *structured* if ingredient fact types are known to exist (within or outside the enterprise data model) AND if a derivation rule for the process type is known to exist (either accessible or not accessible by the active users within the SoI).

Definition 5. A conceptual process type is *semi-structured* if ingredient fact types are known to exist (within or outside the enterprise data model).

Definition 6. A conceptual process type is *unstructured* if ingredient fact types are not known to exist (within or outside the SoI) AND a derivation rule for the process type is not known to exist (either accessible or not accessible by the active users within the SoI).

Definitions 4, 5 and 6 are based upon the extent in which ingredient fact types and derivation rules are known to exist. This means that the fact instances of these ingredient fact types are recorded within the focal UoD or within at least one different UoD, and that derivation rules should be potentially accessible by active users within or outside the focal Sphere of Influence.

2.2 The (revisited) Polanyi Framework

Polanyi classifies knowledge into *tacit* knowledge and *explicit* knowledge: " Tacit knowledge is personal, context-specific, and therefore hard to formalize and communicate." 'Explicit' or 'codified' knowledge, on the other hand, refers to knowledge that is transmittable in formal, systematic language" [11]. In Den Hertog et al. [12] tacit knowledge is defined as: " (implicit) knowledge stored in the brains of human beings rather than material knowledge carriers." McBriar et al. [13] define explicit knowledge as: "knowledge that can be represented in words, drawings, plans, equations or numbers, which can easily be communicated between people".

Kim et al. [14] studied the existing distinction into 'tacit' and 'explicit' knowledge in the literature and concluded that a revised epistemology was necessary in order to make a distinction into the concept of 'tacit' knowledge as defined by Polanyi [11] (in which tacit knowledge cannot be expressed externally) and the concept of 'tacit' knowledge as defined by Nonaka [15] (in which tacit knowledge is defined as knowledge that is (currently) not expressed externally). They revised the existing epistemology by replacing the old concept of 'tacit' knowledge by the revised concepts of *tacit* knowledge and the new concept of *implicit* knowledge: "tacit knowledge is knowledge that cannot be expressed externally and implicit knowledge is knowledge that can be expressed externally when needed, but currently exists internally" [15] (p.3).

Definition 7. A conceptual process type is *explicit* if a derivation rule is known to the user groups within the SoI AND all ingredient fact types are contained in the enterprise data model.

Definition 8. A conceptual process is *implicit* if a derivation rule is known to exist outside the SoI but currently is not accessible to the active users of the user group(s) within the SoI OR[1] ingredient fact types are known to exist outside the enterprise data model but currently are not contained in the enterprise data model.

Definition 9. A conceptual process type is *tacit* if a derivation rule does not exist outside or within the SoI.

Definitions 7, 8 and 9 are based upon the extent in which ingredient fact types exist in the enterprise data model and the extent in which the derivation rules are known to user groups in the SoI.

The modalities in the different knowledge typologies, however, cannot be matched 1-on-1 because these three different 'knowledge typologies' are based on different domain paradigms. Gorry and Scott-Morton take managerial decision making as the foundation for their classification. Kim et al. take the extent in which knowledge can be made explicit as a starting point. Our process-typology that we have introduced in [1] has systems theory as its foundation.

3 The Modeling Methodology for the Process-Oriented Perspective

In order to be able to model the process-oriented features for fact types that are contained in the application's data model but that are created in conceptual process instances that are executed by active users *outside* the focal SoI we need to introduce a fourth conceptual process configuration to our framework from section 1.1: the *enter* process type.

Definition 10. An *enter* process type models the process-oriented characteristics for those fact instances of fact types that are contained in the enterprise data model but that are 'created' in conceptual processes by active users *outside* the SoI of the enterprise subject area.

Definition 10 implies that every 'potential' process type that can not be executed under the responsibility of one or more user groups within the SoI will be considered an enter process type when the enterprise process base is created.

If we consider all possible combinations of ingredient fact types, conceptual process types and resulting fact types in terms of whether the fact types are contained in the application's data model and/or whether the instances of the conceptual process types are performed under the responsibility of active users within the application's SoI, we yield 14 possible process types and sphere of influence/UoD combinations. If a *declarative document* or *user example* is within this border, its fact types can be considered to be part of the enterprise data model. If a *conceptual process type* lies within the rectangle it can be considered to be executed under the responsibility of the

[1] To be interpreted as an *inclusive* OR.

user groups within the SoI. If a *prescriptive document* lies within the rectangle, it means that the *derivation rule* on that prescriptive document is accessible to active users in the enterpise SoI that tells potential user groups *how* to execute a process instance.

The duality that exists regarding *declarative* versus *prescriptive* documents can be explained as follows. If a document is qualified as a prescriptive document it means that within that part of the enterprise subject area the document must be considered to specify a course of action. The same document can however, be considered as an instance of a declarative document in the process base of a different (part of the) enterprise subject area. In Anthony's hierarchy [16] three types of management control can be distinguished: *operational-*, *tactical* (or *management*)- and *strategic* control. Within the UoD and SoI of tactical control a derivation rule might be an outcome of a fact-generating activity. In this situation the derivation rule is an instance of a declarative document (for example a lot-sizing decision rule). Such an instance of a declarative document, however, can be used as a prescriptive document for operational control in another UoD and SoI when it is a derivation rule in a prescriptive document (see figure 3).

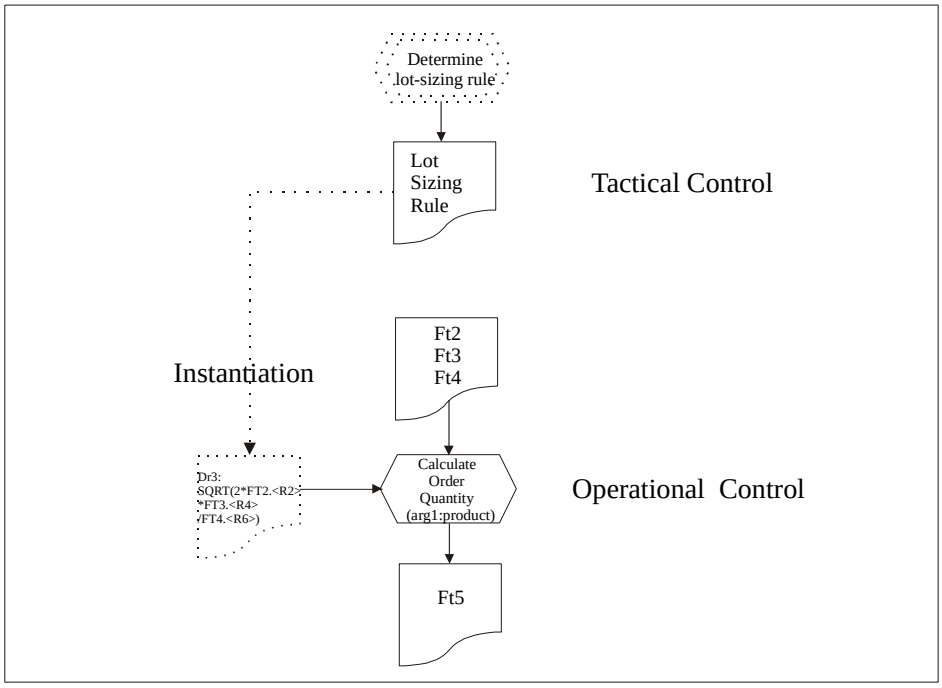

Fig. 3. Application data model for the payroll example.

As a running example for the remainder of this paper we will use the ABC payroll example.

Example: The ABC company

The example application is the payroll department of the branch X of the ABC company. The relationship between this department and the other parts of the organization are shown in figure 4.

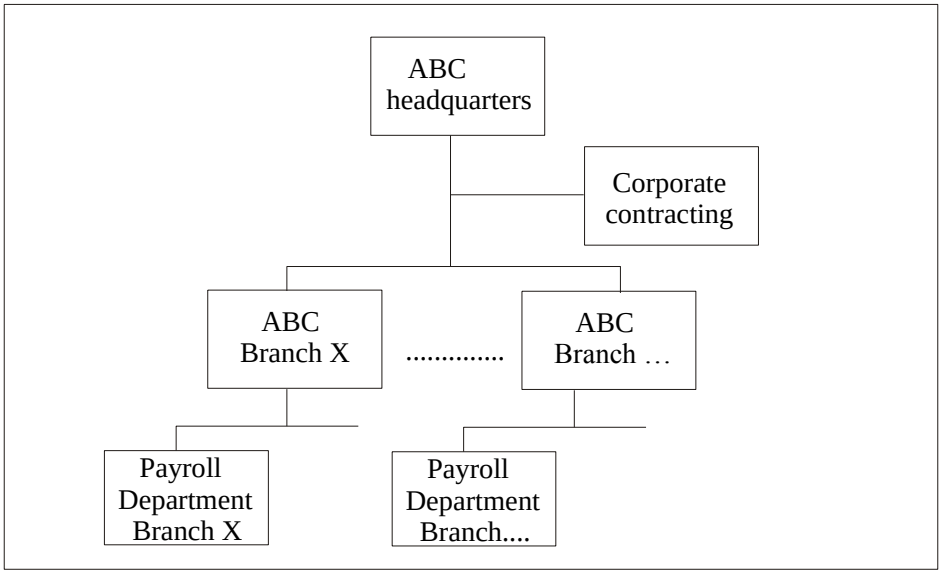

Fig. 4. Organization chart ABC company.

The users in the user groups of the payroll department of branch X, 'decide' how many hours an employee has worked in a given week by inspecting work-order documents and taking additional information into account, e.g. traveling time and information that was obtained in personal contact with the employees. For some employees no work-order documents exist, and therefore the determination of their work-hours is entirely based upon facts that are not contained in the current UoD of the ABC example. The active users in this department furthermore decide upon the gross salaries for the employees that are directly recruited. Although the criteria that determine the salary for each employee are known, the facts that are needed for applying these criteria are not available in the current UoD[2]. The net salary that will appear on the salary slip for the employees is calculated outside the payroll's enterprise area by a payroll service provider. The *gross-to-net* calculation rules are applied by this outside service-agency, and therefore are not accessible by the active users payroll department of the ABC company. Under some conditions it is possible

[2] This would normally suggest, that we will extend the UoD by those documents that contain this information. However, in order to be able to illustrate these 'border' concepts precisely, for now, we assume that we can take any UoD as a starting point and illustrate how we can derive an enterpise process base that belongs to such an arbitrary UoD.

that the working hours for contractors must be recorded although these contractors are not on the company's payroll. In addition it is possible that employees are on the payroll who are hired under the responsibility of a temping-agency. The users in the user groups of the payroll department of the branch X of the ABC company, are also responsible for knowing the highest (gross) salary for an employee at any time.

The fact-based model for the data perspective for this UoD and SoI is given in figure 5. For a brief explanation of the modelling concepts in fact-based modelling we refer to the appendix. The SoI consists of the users in the user group of the payroll department of branch X.

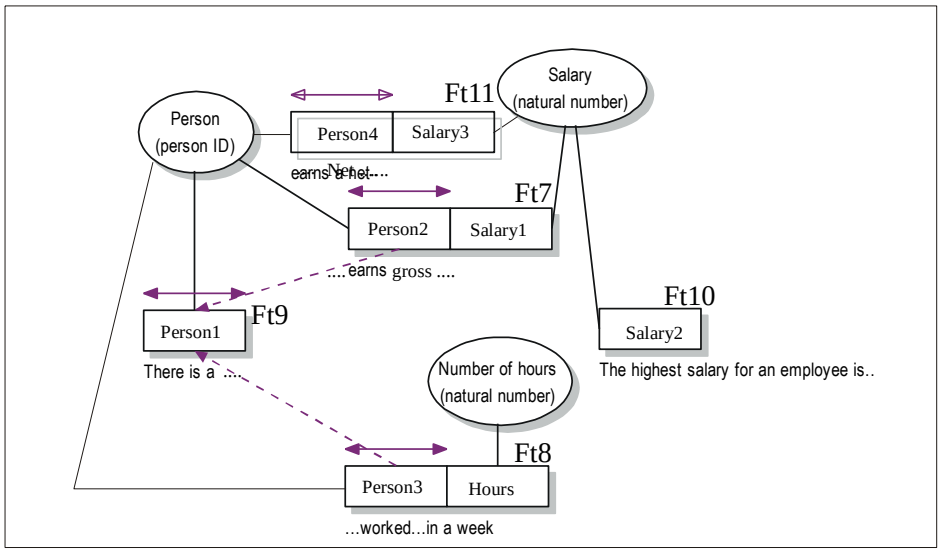

Fig. 5. Fact-based data model for the payroll example.

The content of the fact-based data model in figure 5 can be summarized as follows. There exists fact types that declare the existence of a person (Ft9), that declare that a person earns a gross salary (Ft7), that a person has worked a specific number of hours in a week (Ft8), that there is a highest (gross) salary for an employee (Ft10), and that a person earns a net salary (Ft11). An overview of the fact-based modeling constructs, that are used in figure 5 is given in [17].

We have now discussed all possible situations under which fact instances can be created. We will now synthesize the modalities under which a conceptual process type within an indefinite SoI and UoD can be transformed onto a specific conceptual process type that is defined within the borders of a known application UoD and SoI. The most important modality is the responsibility under which a process instance is executed. If this responsibility lies outside the application's SoI the process will always be modeled as an *enter* process type. We will now consider the modalities from the frameworks of Gorry and Scott-Morton [9] and Kim et al. [14] to characterize the proto-process type configurations and how they map onto the actual

processs type configurations when the processes are performed under the responsibility of active users within the SoI.

From table 1 we see that when a conceptual process is performed under the responsibility of (a) user group(s), within the SoI and the ingredient fact types are not known within the enterprise data model or the process type is unstructured this will always lead to a *strict-determination* process type in the the typology of this paper. If a conceptual process type is not explicit but has (an) ingredient fact type(s) that are contained in the enterprise data model then a process type will always be modeled as a *mixed-determination* process configuration. Finally a structured and explicit process type will always be modeled as a derivation process configuration.

Table. 1. Additional modalities for process configurations performed under responsibility of user groups within enterprise SoI.

Gorry-Scott Moton	**Ingredient fact type(s)**	**Derivation rule**	**Kim**	**Conceptual process types in this paper**
structured			explicit	Derivation
structured	In enterprise Data model	Outside SoI	implicit	mixed-determination
structured	Not in entpr Data model	Within SoI	implicit	strict-determination
structured	Not in entpr Data model	Outside SoI	implicit	strict-determination
semi-structured	In enterprise Data model		tacit	mixed-determination
semi-structured	Not in entpr Data model		tacit	strict-determination
unstructured			tacit	strict-determination

We can conclude from the analysis in this section that the two frameworks that we have used from the literature (Gorry and Scott-Morton [9] and Kim et al. [14]) are not sufficient for determining the exact process configuration in case the UoD and SoI are finite. It turns out that the responsibility of the user who 'performs' the process instances of the conceptual process type in combination with the precise knowledge on the 'status' of the ingredient fact types, in terms of whether they are contained in the enterprise data model, determine the resulting process configuration as defined in this article. The main problem in terms of the applicability of the Gorry and Scott-Morton and the Kim et al. frameworks lies in the notion of 'falsifying' the claim that something does not exist. The framework that we have introduced in this paper, however, only needs an answer to the question whether ingredient facts can be found on example documents within the UoD of the enterprise subject area, or whether there exist active users within the given SoI, that have access to a given derivation rule.

4 A Procedure for Deriving the Process Base of an Enterprise Subject Area

The interaction between the UoD (what fact types are relevant for the enterprise subject area) and the SoI (what active users are contained in the enterprise subject area) if not properly managed can be a risk resulting in project delays and project cost overruns in the development life cycle of business information systems. This phenomenan is known as 'scope-creep' [18] and is characterized by a human tendency to widen the *SoI* over and over again, thereby extending the application's UoD with new examples who in turn lead to an extension of the SoI and so on. To overcome these problems concepts like *Rapid Application Development (RAD)* [19] and *timeboxing* [20] emerged in the project management of IT development. These approaches have had a big impact on the project lead times and they enforce information and business analysts and user management to clearly demarcate the enterprise subject area (UoD and SoI) in the analysis stage of the project. The embodying of these demarcation requirements within the *process-oriented perspective* enforces business analysts to decide what informational activity belongs to the *environment* of the 'system' and what informational activity has to be considered part of the system that is subject of the analysis. It should be noted that the *enter* process types **never** have a process type argument, because instances of such a conceptual process type do not have to be instantiated within the *SoI* under consideration. In figure 6 the procedure for the determination of process type signature for given *UoD* and a known *SoI* is summarized.

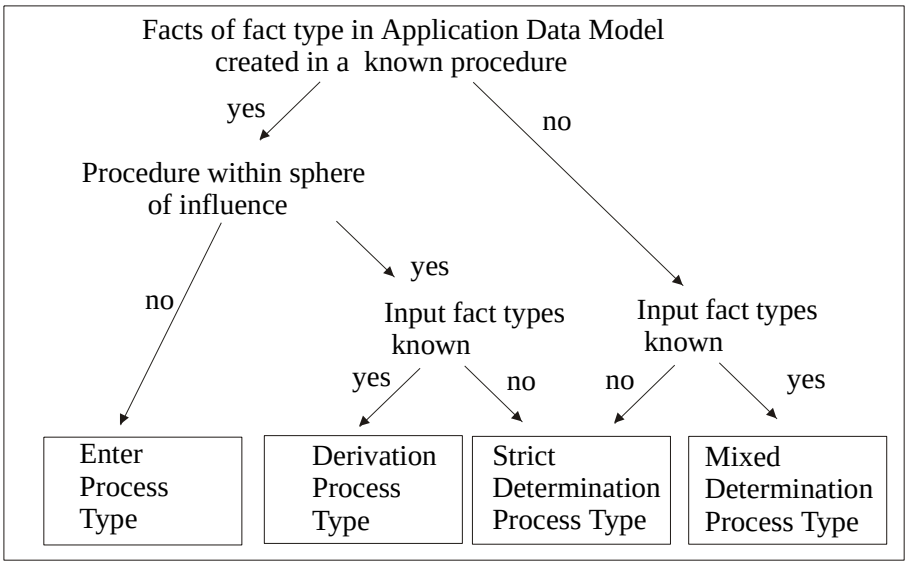

Fig. 6. Procedure for the determination of process type signature for given UoD and known SoI

We can now easily derive an application process base for a given UoD and SoI by applying the decision tree from figure 6. We note that different user groups might use different conceptual process types to create instances of a given fact type. After we have determined all relevant conceptual process types within the sphere of influence we in principle have atomic process type that subsequently can be grouped within a user group to form compound process types.

Definition 11. A process base for a given UoD, user group and SoI contains all *conceptual process types* and *enter process types* that exist within the *SoI* for the fact types in the information grammar of that UoD and user group.

In figure 7 we have given the complete process base for the salary example in a graphical format. We note that for each fact type from the models in the data-perspective at least one process configuration must be contained in the application's process base. To determine to what process type a process instance belongs, that creates an instance of a fact type (that can be created in 2 or more process types), we need an enterprise impulse base, that specifies under what conditions a *specific* process type will be instantiated to create an instance of such a fact type.

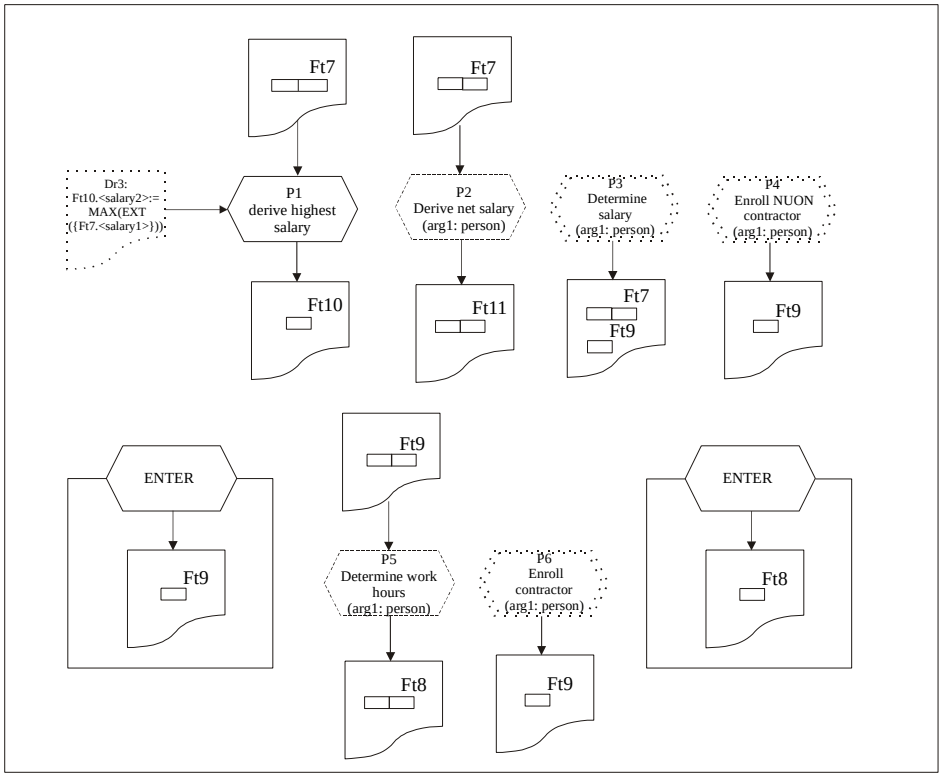

Fig. 7.'As-is' application process base for salary example.

5 The Meta-Process Model

In this section we will give the meta model for the process-oriented perspective. The meta model for the process-oriented perspective or *meta process model* (see figure 1) is a specific application model for the data perspective that is based upon the UoD of a process analyst. The meta process model determines the possible contents of any process base. So a process base of any application UoD and SoI must be an instance of the meta process model. In figure 8 we have given the meta process model expressed as a UML class diagram [21].

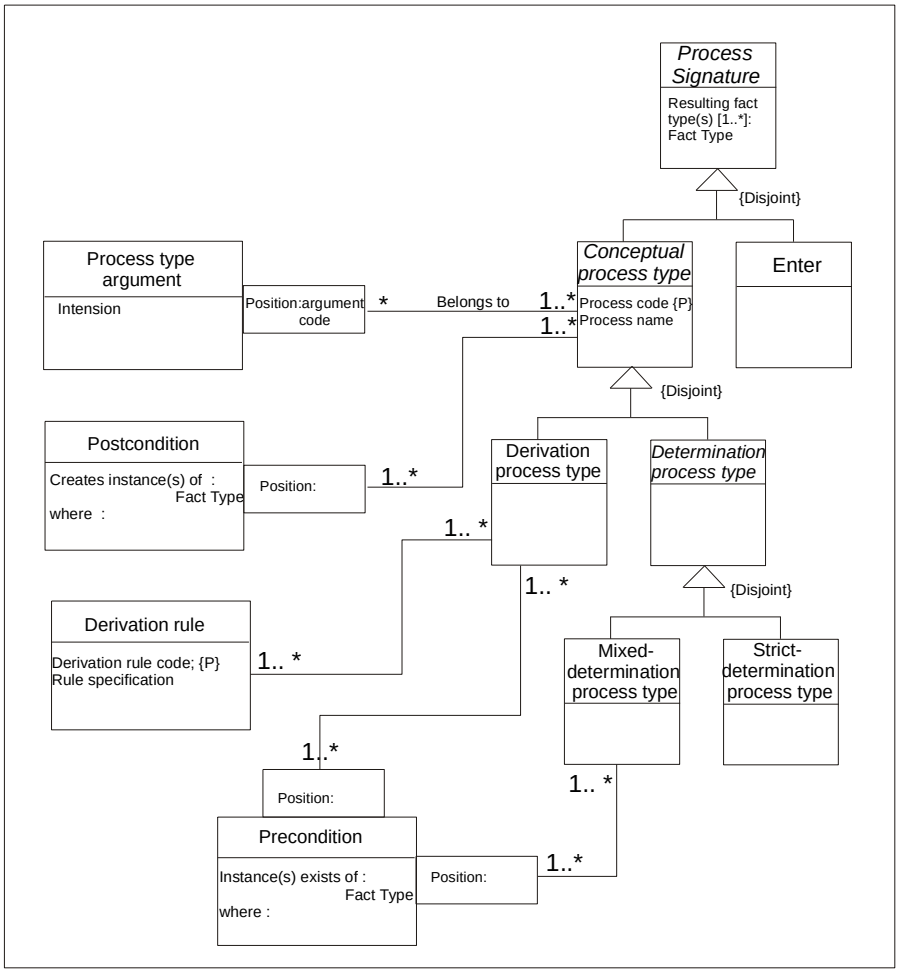

Fig. 8. Meta process model expressed as UML class diagram.

The root of the object-class hierarchy is the abstract object class *process signature*. The root class process signature has two subclasses namely the classes *conceptual process type* and the *enter* configuration. Every instance of the conceptual process type class has a post-condition. The object class conceptual process type, furthermore, has two subclasses: *derivation process type* and *determination process types.* The latter subclass is an abstract sub-class which has two leaf-classes; *strict-determination* process type and *mixed-determination* process type. The meta process model is linked to the meta model for the fact-based approach (for an example see [4]) via the (implicit) object class *fact type*.

6 Conclusion

The definition of three different conceptual process types in combination with the process border-concept of Sphere of Influence (SoI) has resulted in the existence of 3 conceptual process configurations (plus the *enter* configuration) for a given enterprise subject area with a known *UoD* and a known *SoI*. We have shown in this paper that the process configurations are not only determined by the level of 'structuredness' and 'tacitness' in a general sense, but in many instances they are determined by the borders in the data- and process perspectives, respectively. The ability to model conceptual knowledge processes that have a 'tacit' nature and the extent in which the 'codifiable' properties of these tacit knowledge processes can be modeled makes the constructs in the meta process model in this paper applicable in the field of Knowledge Management. The modeling constructs in the framework for the process base in this paper turn out to be applicable in every enterprise subject area, whereas the earlier frameworks of Gorry and Scott-Morton and Kim et al. are hard to apply in real-life situations because they do not have 'finite' border constructs for the enterprise subject areas at hand.

References

1. Bollen, P. *A Conceptual Modeling Language for the Adequate Design of Business Processes*. in *BPMDS '07* 2007. Trontheim, Norway.
2. Bollen, P., *Conceptual process configurations in enterprise knowledge management systems*, in *Applied computing 2006*. 2006, ACM: Dijon, France.
3. Halpin, T. and T. Morgan, *Information Modeling and Relational Databases; from conceptual analysis to logical design* 2nd ed. 2008, San-Francisco, California: Morgan-Kaufman.
4. Nijssen, G. and T. Halpin, *Conceptual schema and relational database design: A fact based approach*. 1989, Englewood Cliffs: Prentice-Hall.
5. Olle, T.W., et al., *Information Systems Methodologies- A Framework for Understanding*. 1988: North-Holland, Amsterdam.
6. Davenport, T., *Process innovation: reengineering work through information technology*. 1993, Cambridge: Harvard Business Press.
7. Hammer, M. and J. Champy, *Reengineering the corporation: a manifesto for business revolution*. 1993, Harper Collins: New York.

8. Nickols, F., *The difficult process of identifying processes.* Knowledge and process management, 1998. **5**: p. 14-19.
9. Gorry, G. and M. Scott Morton, *A framework for management information systems.* Sloan Management Review, 1971(fall).
10. Simon, H., *The new science of management decision making*. 1960: Harper & Brothers publishers.
11. Polanyi, M., *The tacit dimension*. 1966, London: Routledge & Kegan Paul ltd. .
12. Hertog, F.d. and E. Huizenga, *The Knowledge Enterprise: implementation of intelligent business strategies*. 2000: Imperial College Press.
13. McBriar, I., et al., *Risk, gap and strength: key concepts in knowledge management.* . Knowledge-Based Systems 2003. **16**: p. 29-36.
14. Kim, T.-G., S.-H. Yu, and J.-W. Lee, *Knowledge strategy planning: methodology and case.* . Expert Systems with Applications, 2003. **24**(3): p. 295 - 307.
15. Nonaka, I., *A dynamic theory of organizational knowledge creation.* . Organization Science, 1994. **5**(1): p. 14-37.
16. Anthony, R., *Planning and control systems: A framework for analysis*. 1965: Harvard University Press
17. Halpin, T., *Information Modeling and Relational Databases; from conceptual analysis to logical design*. 2001, San Francisco, California: Morgan Kaufmann.
18. Verner, J., S. Overmyer, and K. McCain, *In the 25 years since The Mythical Man-Month what have we learned about project management ?.* . Information and Software Technology 1999. **41**: p. 1021-1026.
19. Beynon-Davies, P. and M. Williams, *The diffusion of information systems development methods.* . The journal of strategic information systems, 2003. **12**: p. 25-46.
20. Jalote, P., et al., *Timeboxing: a process model for iterative software development.* Journal of Systems and Software, 2004. **70**: p. 117-227.
21. Booch, G., J. Rumbauch, and I. Jacobson, *Unified Modeling Language User Guide*. 2nd ed. The Addison-Wesley Object Technology Series). 2005: Addison-Wesley Professional. 496.

Appendix: Fact-Based Modeling concepts

Fact-Based Modeling (FBM) is a methodology for modeling information systems on the conceptual level. It is named after its main constituents: objects that play roles in relationships. The 'role-based' FBM notation makes it easy to define static constraints on the data structure and it enables the modeler to populate FBM schemas with example sentence instances for constraint validation purposes. In FBM (and other fact oriented approaches) the fact construct is used for encoding all semantic connections between entities. Figure 9 summarizes the symbols in the FBM modeling language that we will use in this paper.

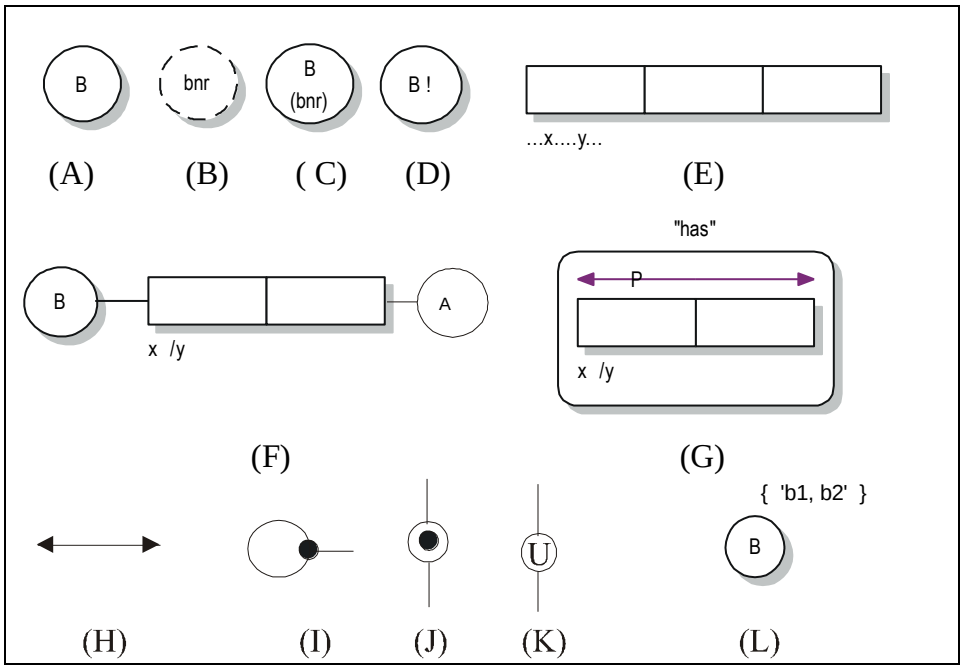

Fig. 9.: Main symbols in Fact-Based Modeling (FBM).

Atomic entities (figure 9A) or data values (figure 9B) are modeled in ORM as simple (hyphenated) circles. Instances of an entity type furthermore can exist independently (e.g. they are not enforced to participate in any relationship), which is shown by adding an exclamation point after the entity type's name (figure 9D). Simple reference schemes in ORM are abbreviated by putting the value type or label type in parenthesis beneath the name of the entity type (figure 9C). Semantic connections between entities are depicted as combinations of boxes (figure 9E) and are called facts or fact types in ORM. Each box represents a role and must be connected to either an entity type, a value type or a nested object type (see figure 9F). A fact type can consist of one or more roles. The number of roles in a fact type is called the fact type arity. The semantics of the fact type are put in the fact predicate (this is the text string …x…y… in figure 9E). A nested object type (see figure 9G) is a non-atomic entity type that is connected to a fact type that specifies what the constituting entity types and/or values types are for the nested object type.

Figures 9H through 9L illustrate the diagramming conventions for a number of static population constraint(s) (types) in ORM. A double-arrowed line (figure 9H) that covers one or more 'boxes' of a fact type is the symbol for an internal uniqueness constraint. The symbol in figure 9K stands for an external uniqueness constraint. A(n) uniqueness constraint restricts the number of identical instances of a role combination 'under' the uniqueness constraint to one. A mandatory role constraint (figure 9I) can be added to a role. It specifies that each possible instance of such an object type must play that designated role at all times. A disjunctive mandatory role constraint (figure 9J) is defined on two or more roles and specifies that each possible instance of the

object type connected to these roles must at least play one of these roles at any time. A subset constraint in figure 9K is sometimes depicted as an arrow: ----> between roles or role-combinations. It enforces that the population of the 'source' role at all times must be a subset of the population of the 'target' role. An in-depth treatment of Fact-Based Modeling can be found in [17].