Part III: Concept computing
The Taxonomy Revolution, Part III: Concept Computing

A Taxonic whitepaper, by Taxonic B.V., Rijswijk, The Netherlands

http://taxonic.com

Copyright 2012 © by Jan Voskuil

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations (unless others hold the rights to these, as indicated), recitation, broadcasting, reproduction on microfilm or in any other way, and storage in data banks. Any commercial or non-commercial uses must be cleared by the rights holder. For questions regarding usage, please contact info@taxonic.com.

The use of general descriptive names, registered names, trademarks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.
Acknowledgements

The subject of applying semantic technologies in business processes is a vast and multi-disciplined area. I am thankful to many friends and colleagues for sharing their insights and generously offering input, encouragement and moral support in various forms. I want to thank Vincent van Dijk and Patries Kordelaar of Pharosius, Diederik Wentink, self-employed and founder of the Be Informed User Group, Guus de Mari and Marijn Jansen of Delft TopTech, prof. Hennie Daniëls of University of Tilburg, Bram Smeets of Orange11, Ronald Heller, Jeroen van Grondelle, Freek van Teeseling, Frank Buystendijk and Geert Rensen from Be Informed, Peter Mika of Yahoo!, Arleen Zank of Coronado Group, Erik Reeves of SumoBrain and Anne Hunt and Gam Dias of First Retail. Special thanks goes to Rocky Kahn and Phil Aucutt of TeamPatent for giving my train of thought a series of twists at the right moments.

Extra special thanks to Mills Davis (Project10X), Coos van der Togt (O&i) and Menno Gülpers (Be Informed) for, among other things, commenting on earlier versions of this paper.

Finally, I thank Jan Verbeek (Be Informed) and Larry Goldberg (KPI) for vision and inspiration. It is a privilege to be able to stand on their shoulders.
# Contents

1 Concept Computing ........................................................................................................ 5
   Concept Computing and Case Management .................................................................. 6
   Enterprise Decision Management ................................................................. 7
   Adaptive Case Management ..................................................................................... 9
   Concept Computing and the Knowledge Worker ..................................................... 11
   Concept computing and continuous change ......................................................... 13

   Bridging the gap is more difficult than it seems .................................................. 13

   How concept computing solves the problem ...................................................... 14
   Concept computing and BMP 3.0 ........................................................................... 15

2 Concept computing: how it works ............................................................................. 16
   Inference engines: facts instead of instructions .................................................... 16
   Propositional logic and taxonomies ................................................................. 17
   Classification taxonomies and context taxonomies ............................................ 19
   Examples with Be Informed BPP ......................................................................... 20

   A simple example: the knowledge model ........................................................... 20

   A simple example: the web application ............................................................... 22

   A slightly more complex example ....................................................................... 23

3 Executing business processes .................................................................................. 24
   Concept Computing meets Case Management ..................................................... 25
   Managing knowledge models in real-life ............................................................ 27
   Creating a business thesaurus .............................................................................. 29
   Benefits of concept computing ........................................................................... 31

4 Where to go from here ............................................................................................... 33
   Platforms for concept computing, or Against BRE’s .......................................... 34
   Skills for concept computing ............................................................................... 36
   Other semantic technologies ............................................................................... 37

5 Concluding remarks and take-aways ..................................................................... 38
   Endnotes .............................................................................................................. 40
   Bibliography ...................................................................................................... 32
Semantics plays an increasingly important role in business processes and IT-systems. This has profound consequences for organizations. Computer systems can now reason based on the semantic features of a specific case. While the concept of such self-reasoning systems has been proven already in the 1970’s, real world solutions now start hitting the market. Consequently, techniques developed in the field of information retrieval and library science are applied to business modeling. Information retrieval, search technology and business process management converge, resulting in revolutionary change. This series of whitepapers explores the backgrounds and the consequences of this change.

Part I addresses the basic concepts: knowledge models, taxonomies, ontologies. Part II explains applications of these concepts in the domain of information retrieval and search. Part III is about using knowledge models to support business processes.

Part III consists of three major sections. We start with a short overview of concept computing. In the next section, we discuss the underlying technology, to get a feel of what is going on under the hood. Finally, we elaborate on real-world benefits that this approach can yield for organizations.

The leading product in this area is Be Informed Business Process Platform. Working examples will be presented in this series based on this product. Other examples of products in this field are, for instance, Pega Systems, IBM WebSphere Operational Decision Management, Oracle Policy Automation and Isis Papyrus.

1 Concept Computing

Automating business processes based on knowledge models is the new trend. Knowledge models are understandable and transparent to business persons. By making these directly executable, the development cycle is cut short, and the resulting IT-systems support business processes in a more user-friendly way.

When we talk about a particular domain, we use concepts to structure the things in it. In Part II, we have seen how knowledge models capture these concepts, and, by doing so, enhance findability and foster knowledge management. Now, these very same models are used to execute business
processes. In a presentation at the Semantic Technologies and Business conference in San Francisco, 2012, semantic web visionary Mills Davis coined the term *concept computing* to refer to such systems.¹

Concept computing is using executable knowledge models to create model driven IT-systems. The model driven approach to system development cuts the development process short. The distance between business and operational system is shortened.

Concept computing works on concepts rather than on data as its primary fuel. Data remain as important as ever, but concepts and their interrelations are now used to make it explicit what they mean. Consequently, these concepts can be used to reason *about* the data. The result is a brand new approach to process automation.

**Concept Computing and Case Management**

Case management systems support administrative processing and provisioning of requests for a product or a service. They constitute a particularly important class of process automation systems, and concept computing is making its break-through in precisely this domain. In the realm of case management, concept computing uses knowledge models to reason about a case — an application for a permit, or a loan request, for example —, and determines at each step in the process which actions can be carried out to bring the case to its desired result, as quickly and as ergonomically as possible.

This contrasts with more traditional approach to case management. Within this approach the focus has always been on the flows within linear processes, rather than on the knowledge underlying these flows. Hence, flowcharts are used as the main modeling language. This fits within the Business Process Management paradigm or BPM for short. BPM can refer to managing business processes in the broadest sense of the word, but in this section, we use it in a more restricted sense, meaning the flow chart approach to process modeling.

In that more restricted sense, BPM creates process models in terms of instructions that describe activities and the order in which they are performed. This is also a form of model driven system development, since the models can be executed by a process or workflow engine. There are widely used standards for this, such as OMG’s Business Process Modeling Notation.² The semantics that these models can express is limited, however. A box stands for an activity, an arrow means “when finished, go to the next”. But business processes are not about which activity precedes which. They are
about executing complex policies, regulations and legislation, about composing products and services tailored to the customer’s needs, based on rules that often change, and about finding relevant information from disparate sources — in short, about things that are specific to the given business domain. While BPM leads to a form of model driven system development, the models are semantically not rich enough to be of true value.

Two major trends currently challenge the BPM-approach to model driven case management systems. These trends converge with the concept computing movement, lending it extra urgency. These trends are:

- **Enterprise decision management.** By separating business rules from process information, business logic becomes much more manageable.
- **Adaptive case management.** Knowledge work requires higher degrees of freedom than traditional BPM can offer.

Both trends pose new requirements for business process support systems. Since concept computing provides a platform that meets those requirements, it is at the center of a new round of innovation in IT. In the next subsections, we discuss these two trends in turn.

**Enterprise Decision Management**

The business rules community has been active since the 1990, seeking insight in the way one can optimally deal with business rules. This activity has lead to a steady stream of publications, large scale community events, the evolution of new tools — generally referred to as Business Rule Engines, and more recently as Decision Management Systems —, and numerous large-scale implementations of these across the industry and across the globe.

One of the most important publications in this area is no doubt *The Decision Model* by Barbara von Halle and Larry Goldberg, published in 2010. It offers a rigorous method of analyzing business rules in terms of a unified decision model. When done properly, this results in a clear separation of business rules from processes. Consequently, the process descriptions can be much simpler, and the business logic more transparent. Systems built using this approach are resilient to change, since each piece of business logic is implemented only once — instead of several times in different parts of a process model.

Business logic defines the way an organization behaves. As such, it underpins the organization’s integrity, innovative strength, intelligence, and its very identity. It has its own existence, independent of how it is executed, where it is executed, and how it is executed — manually or through some computer system. Therefore, business logic should be managed as a separate asset,
and not be buried in process descriptions or program code. Von Halle and Goldberg’s decision model is a way to achieve this. It is a structured representation of the business logic that is independent of process, data or technology. In much the same way an Entity Relationship Diagram (ERD) is a technology independent representation of data structure, a Decision Model captures the rules of a business. Another similarity between ERD models and decision models is that a rigorous algebra is available for normalizing the models to achieve optimal efficiency and simplicity.

Concretely, a decision model is a set of tables, each representing a so-called rule family. A simple example is given in Figure 1. Rule families can be thought of as decision tables subject to a set of rigorous principles governing the internal structure of each rule family, and the relations each rule family has with other rule families.

Von Halle and Goldberg’s approach intends to achieve three important goals:

- Simplicity. The models are simple to understand and easy to manage.
- Declarative descriptions. The rule models are declarative, that is, they can be applied in any sequence, using any technology. The models abstract away from processing requirements and flow details.
- Integrity. Each model is complete and consistent.

What makes this approach truly revolutionary is that it cuts the system development lifecycle short, giving business persons much more direct control. As tools become available that can execute the rule models directly, system development and change management are driven by the business: the rules are executable, hence, do not have to be translated to technical design artifacts that, in turn, are translated to programming code. One class of tools that can do this are Business Rule Engines. We talk about BRE’s
in the last chapter of this paper. Another class is platforms for concept computing.

The relation with concept computing becomes clear when we take a closer look at the rule family in Figure 1. The subjects in the in middle row are concepts and as such part of the business ontology. Concepts such as "Person Mortgage Situation" and "Person Employment History" — and their interrelations — need to be carefully defined and managed.

The rules expressed in Figure 1 can be captured in the same knowledge models that define the semantics of the core concepts. The result is a knowledge model. For instance, the rule family in Figure 1 could be represented as in Figure 2.

There is much to say about these modes of representation. At this point, the relevant observation is just that BPM is primarily about procedural aspects of organizational behavior, leaving business logic in the dark. Recent thinking about business logic challenges the validity of this approach. Carefully representing business logic in its own right takes away a lot of the complexity otherwise hidden in process models. Concept computing supplies both a modeling language to create such representations, and an execution environment to turn them into working applications.

**Adaptive Case Management**

The second trend challenging the BPM-approach to model driven case management systems is the adaptive case management movement. Adaptive case management intends to offer better support for knowledge work, which is essentially any type of work that requires human judgment at some stage. In Part I, we quoted Mastering the Unpredictable by Keith D. Swenson. The authors of the essays bundled in this important publication all share one basic observation: the way IT-systems support knowledge work must be improved by giving human judgment the prominent place it deserves. Peter Drucker formulates it as follows:

"The most important, and indeed the truly unique, contribution of management in the 20th Century was the fifty-fold increase in the productivity of the manual worker in manufacturing. The most important contribution management needs to make in the 21st Century is similarly to increase the productivity of knowledge work and the knowledge worker." ³

Unfortunately, BPM-based case management systems are most often unwittingly architected to deliver the opposite of what knowledge workers
require. The underlying reason for this is that the paradigm for BPM is the assembly line metaphor. This metaphor is misleading because knowledge work is fundamentally different from working on an assembly line. Drucker mentions six important features of knowledge work. BPM-based systems support none of them. BPM-based systems therefore lead to friction in processes where human judgment plays a role:

- **Problem solving.** Knowledge workers need support for problem solving tasks, but the system focusses exclusively on sequencing tasks.
- **Freedom.** Problem solving requires a certain amount of freedom, but the system shoehorns workers in a predefined process flow.
- **Collaboration.** Knowledge workers need to share their expertise in effective ways, but the system focusses exclusively on individual work items flowing through a process.
- **Quality.** In knowledge work, quality is at least as important as quantity, but the system focusses exclusively on quantitative aspects.
- **User experience.** Knowledge workers are assets that must be enticed to work for your organization, but they experience the system as unfriendly to users.
- **Constant innovation.** Knowledge worker’s tasks evolve continuously through innovation, but the system does not innovate because it is so difficult to modify it.

The problem of predefined execution paths is also a topic with decision management, as we have seen. In the previous subsection, we noted the importance of representing business rules declaratively, because a decision model intends to capture all and only business rules, therefore abstracting away from procedural aspects.

Both lines of reasoning converge because knowledge and process are fundamentally different concepts. If we go back to the rule family in Figure 1, it is obvious that in order to determine whether or not the likelihood of a person defaulting on a loan is high, three tasks need to be performed: assess the applicant’s employment history, assess his mortgage situation, and evaluate miscellaneous loans assessment. Since there are three tasks, they can be executed in six different sequences.

The combinatorial explosion of possible sequences of tasks is one of the reasons BPM deals badly with complexity in general and business logic in particular. The knowledge worker must be given the freedom to choose the order that is most logical in the particular context of a particular case. If information about the applicant’s mortgage situation is earlier available than other information, then assessing this aspect of the case takes place first, unless the knowledge worker sees fit to decide otherwise.
Proponents of BPM may of course retort that process modeling languages such as BPMN allow one to express that the order between certain tasks does not matter. But if order does not matter, there are apparently no process requirements. The pertinent tasks should therefore not be represented in a process model in the first place.

The underlying challenges that knowledge work poses for BPM are related to the differences between what Max J. Pucher, founder and president of ISIS Papyrus and prolific author on adaptive case management, calls designed and emergent interaction models. On the designed end of the spectrum, we find processes with a relatively low degree of complexity and a high degree of predictability. On the other end of the spectrum we find emergent processes, where people use social skills and intellectual proficiency to find creative solutions. BPM, workflow and the assembly line metaphor are appropriate on the designed end of this scale, whereas adaptive case management and knowledge work is oriented towards the emergent end. See figure 3.

**Concept Computing and the Knowledge Worker**

The reason that concept computing works so well for supporting knowledge work is that it marries a declarative representation of knowledge to a goal oriented approach.

Declarativeness means that the knowledge representations abstract away from procedural aspects. The system uses its computational power to derive these procedural aspects autonomously.

Goal-orientatedness means that the system focusses on the intended goal and uses knowledge representations and case data to infer how to get there as fast and conveniently as possible.

Let us consider an example to see what this means. If the desired goal of a case is to obtain a loan for the applicant, the system will start its reasoning from there. In order to obtain that loan, a finite set of preconditions must be fulfilled. One of these preconditions is (for example) that likelihood of defaulting is low. To establish this, three more basic aspects of the applicant need to be established first — such as given in the example rule family in Figures 1 and 2. For each of these, a separate task with a separate set of electronic forms will be presented to the case worker. The case worker is free to decide which one of these tasks he executes first.

These two properties — goal-orientatedness and declarative modeling — set concept computing clearly apart from the BPM-approach to model driven system development. It is these properties that allow case management
systems to assist knowledge workers in choosing a direction rather than taking over the steering wheel. The new paradigm is the navigation device instead of the assembly line. A navigation device is goal-oriented, in that it starts reasoning from the destination you want to reach. It uses a declarative representation of the roads, and computes the procedural aspects autonomously. If you take a detour to avoid a problem or to buy a cup of coffee and sandwich, it will graciously compute an alternative route to the destination instead of treating the event as an exception. Human judgment is in the lead, not the machine.

Like a navigation device, a case management system based on concept computing helps case workers to navigate a complex maze of possible routes to finalizing each case at hand. Let us go back to Drucker’s requirements for supporting knowledge work: a focus on problem solving, a certain amount of freedom, support for collaboration, a focus on quality, a great user experience and constant innovation.

If a particular route is blocked due to unusual circumstances, the case management system graciously computes an alternative route, thus offering room for judgment and freedom to the knowledge worker.

To support problem solving tasks, a case management system should make necessary information findable. Enhancing findability is one of the core functions of knowledge modeling, as we have seen in Part II. Links to relevant information are easily incorporated in knowledge models. Therefore, the case management system can provide helpful information at each step on the way: links to the business thesaurus, links to relevant policies and legislation, links to diverse sources of knowledge, links to similar cases.

Concept computing also fosters collaboration. Finding other case workers who have experience with exactly the problem at hand is a problem of findability, and knowledge modeling is a technique well suited to solve it.
All this leads to a much better user experience for knowledge workers. And last but not least, since business rules drive the process, quality aspects can be monitored effectively.

But what about constant innovation? We discuss this in the next subsection.

**Concept computing and continuous change**

There are two ways one can interpret adaptivity in adaptive case management, and both ways are equally appropriate and equally important: the system is able to adapt to changing needs of its users while managing a case, and the system is able to change the way it works in response to changes in the environment, such as new legislation. In the previous subsection, our focus was on what concept computing means to end users. We now shift our attention to designing, building and maintaining systems based on concept computing.

**Bridging the gap is more difficult than it seems**

Concept computing narrows the gap between system development and business behavior. One language is used across the board — the language of the business. This claim needs careful backup, because it has been made so often, and so far, it has never convincingly lived up to its promise. Already in 1972, when the newly standardized Common Business Oriented Language, or COBOL for short, hit the market to replace Assembler, pundits and thought leaders boldly proclaimed the end of programming as a discipline. Business persons would be able to write the necessary programs themselves, because COBOL resembles English (and, so the argument went, business people normally speak English). Instead of writing up specifications in English and having these translated into Assembler programs, business people would, from now on, write the specifications directly in COBOL!

And indeed, it soon appeared that even the simplest business person was able to understand and write COBOL statements like ADD ITEM-PRICE TO TOTAL. The problem is that to answer to real-world business

![Figure 4. In 1972, COBOL promised to close the gap between business and IT.](image)
needs, millions and millions of such statements are necessary. You need highly trained IT-personnel to deal with that kind of complexity. And so we were back at square one, because it is well-known that highly trained IT-personnel has problems understanding even the simplest business problem. Notwithstanding its resemblance to English, COBOL did not help closing the gap between business and IT.

A similar story can be told for BPM. Business people understand and can draw simple process flow diagrams. However, long before enough detail is added so as to make the process flow diagrams executable by a process engine, a level of complexity is reached where specialized IT-skills are necessary. BPM has equally failed to live up to its promise to bridge the gap.

Why should it be different with concept computing? The answer is: semantics. Concept computing is based on directly executing knowledge models. These models describe the semantics of the language used in the business.

**How concept computing solves the problem**

What makes it so hard to bridge the gap between business and IT is the problem of dealing with complexity. You need structuring principles to keep complexity manageable. In the business world, these principles are laid down in policies, work instructions and governance structures. The structuring principles used in IT-system development are of a completely different nature. They are applicable, independently of the business domain where the IT-system is applied. Hence, people understanding complexity in one domain need a translating service when talking to people trained in dealing with complexity in the other domain.

Concept computing is about directly capturing the policies, work instructions and governance structures of the business in executable models. With concept computing, the focus is, therefore, exclusively on dealing with business complexity. No translation needed.

In Part II, we discussed the MeSH-thesaurus, which is a knowledge model that describes the meaning of medical terms. This thesaurus is written and maintained by a team of specialists. This indicates that creating and maintaining knowledge models requires special skills, but these are not IT-skills. Rather, these skills are oriented towards analyzing, organizing and structuring business vocabulary and business logic, so that business persons can use these more effectively. The knowledge models that are the result of this analysis are therefore expressly meant to be understandable to business persons. It is logical that lay people and medical professionals alike can understand and use the MeSH-thesaurus effectively. It is transparent to them.
The central premise of concept computing is that such knowledge models can now be made directly executable. The behavior of a system based on concept computing can be changed, therefore, by making modifications to the knowledge model — by changing rules or relations between concepts.

This leads to better responsiveness to change for two reasons:

- The chain from business need to a working solution is cut short. No more hand-over of specs and endless translation steps.
- The models are closer to the business; hence, the business has more direct control.

An additional source for resilience to change derives from the nature of knowledge models themselves.

Since the concepts in a knowledge model are linked to each other, impact of changes can be determined by simply navigating the links. A sound knowledge model contains explicit links to policies, legislation and business requirements explaining why rules are in effect. When the law changes, there is a clear and systematic method of determining the impact of that change — namely, by following the links from the units of legal text to functions in the knowledge models.

Conversely, following a similar systematic approach, proposed changes in a system’s behavior can be checked against legal requirements. This adds considerably to the ability to adapt the knowledge model to a changing environment — and hence, indirectly, the behavior of the case management system supporting the business processes.

Knowledge workers find themselves in a constantly changing environment. Constant innovation of the way they do their job is necessary, as Peter Drucker observed. Concept computing now makes this possible.

**Concept computing and BMP 3.0**

Does concept computing spell the end of BPM? Or does it add to BPM, creating a new synthesis? It all depends on how one interprets the term. When construed to mean, in general, the art of structuring business processes, concept computing can be viewed as a valuable addition to the BPM-toolset. Analogously to the Web 2.0 – Web 3.0 dichotomy, one could say that BPM 2.0 is BPM plus collaborative features, and that BPM 3.0 adds rich, executable semantic models. BPM 3.0 is BPM plus concept computing.
2 Concept computing: how it works

So far, we have seen that concept computing offers a brand new approach to system development. It converges with enterprise decision management and adaptive case management, offering a compelling proposition for IT-innovation in many organizations. To understand why concept computing is so different, some insight is necessary in how it works. This chapter briefly explains the basic principles, and discusses a simple example.

In discussing the use of taxonomies in semantic search in Part II, we touched on the subject of query expansion. If you are searching for literature on birds, the search engine can expand your query with narrower terms for bird, such as duck and seagull. This ensures you don’t miss out on articles on these narrower subjects. This is, in fact, a form of using a taxonomy in an inference: a duck is a bird, therefore, if bird is a relevant topic, then duck is a relevant topic. Inference engines can go much further than this.

Inference engines: facts instead of instructions

To see how inference engines work, we go back to the early days of computer science. In the 1970’s, there was an influential movement that focused on the difference between procedural programming languages and declarative languages such as Prolog. Although Prolog is almost a dead language now, like Latin and Sanskrit it offers insight in current problems.

Using a procedural programming language, you tell the computer to follow a procedure and execute activities in a predefined order. In a declarative language, on the other hand, you just define facts and rules — where rules can be thought of as complex facts. The inference engine, or inferencer for short, will use these rules and facts, apply them to a given problem in the best order possible, and arrive at the best solution. It does not need instructions: it uses rules and facts only.

Consider the example of classical Prolog in Figure 6. In this example — where “:-” means “if..., then...”—, we have fed the inferencer with three declarative statements: (1) if something is a philosopher, it is human; (2) if

```prolog
philosopher(X) :- human(X);
human(X) :- mortal(X);
philosopher(socrates);
?- mortal(socrates);
=yes
```

Figure 6. An example of Prolog.
something is human, then it is mortal; and (3) Socrates is a philosopher. We then ask the question whether Socrates is mortal (Prolog-style), and the inferencer concludes that this can indeed be inferred from the set of known facts. In other words: yes.

What made this sensational at the time is that you must find a way to make the inferencer efficient. The easiest solution to the puzzle would be to iterate over all facts at each step of the inference, just to see which are relevant at that particular step. Adding facts to the knowledge base, however, would soon cause a combinatorial explosion, bogging down the computer. Only when smart algorithms were developed, such as the famous Rete algorithm, the inference engine became a viable tool. In fact, there is a close similarity here with navigation programs. Navigation software also relies on smart algorithms. Without these, it would take geological periods of time analyzing the (declarative) representation of roads to find the optimal route to the user’s destination.

The Prolog inferencer is goal-oriented. Using backward chaining, the Prolog inferencer will find the fastest route to the goal. In the above example, the inferencer starts with the hypothesis that “mortal(socrates)”. This hypothesis is proven if “human(socrates)” is true. Given the logic of implication, it follows that that proposition is true if Socrates is a philosopher, which is a known fact in the example’s knowledge base.

Goal-orientedness is again a parallel with a navigation system: once it has a destination, the navigation system will reason backwards to find out the fastest route to get there.

Thus, Prolog stands to procedural programming languages as concept computing stands to classical process flow chart modeling. This is not just a manner of metaphorical speech: it points at an abstract but real property of certain approaches to problem solving. The principles behind Prolog have left the cookbook of cool algorithms and now, after all those years, form the basis of a new round of innovation in business process automation.

**Propositional logic and taxonomies**

The Prolog inference engine — like all inference engines — uses propositional logic as its basis. A proposition is anything that can be true or false. In fact, from a semantic point of view, the meaning of a proposition is either one of two values, T or F. In terms of these truth values, we can proceed to define logical operators that combine elementary propositions into complex propositions. For instance, the operator AND.
Q" is only true in case both P and Q are true. Similarly, we can define an implication relation: “if P then Q”. Such an implication is only false if P is true and Q is not. The rule “if it rains, then the streets are wet” is only disproved in case it rains and the streets remain dry. In all other cases, the implication is true: As long as it doesn’t rain, the rule doesn’t care about the streets being wet or not. We can represent these properties of propositions and operators in truth-tables, such as in Figure 7.

From a logical point of view, all an inference engine has to do is to interpret facts and rules in terms of such tables, and use these in determining truth and falseness of all propositions involved in an inference.

The Prolog inference engine adds something extra to this: it analyses atomic propositions in terms of terms and predicates. Thus, instead of propositions like P and Q, we have propositions like “philospher(socrates).” Predicate logic is much more powerful than plain propositional logic. Such extra power also adds more complexity, and for the purposes of concept computing, simple propositional logic will do just fine.

To reason with taxonomies, we have to take taxa as propositions, hence, as things that can be true or false. This stretches the imagination to some extent, because a taxon normally refers to a class, not to a truth value. It is intuitively strange that the taxon “bird” should be true or false per se. If, on the other hand, you interpret this taxon as a proposition meaning “the animal I am trying to classify is a bird,” then truth and falseness become much more plausible. In a similar vein, the relation from a subclass to its parent class can be defined as implication: duck implies bird.

It is important to keep in mind that interpreting taxa as propositions leaves something implicit. An implication like “if duck then bird” is true if ducks are birds, but we need to trust that we apply the two taxa to the same animal — which is essentially a leap of faith. In predicate logic, Prolog-style, we could make this assumption explicit by writing “duck(X) :- bird(X)” as opposed to “duck(X) :- bird(Y)”.

But we won’t. The higher goal we want to achieve is to keep the modeling language as simple as possible, adding just enough complexity and semantics.
to be able to express basic truths that are relevant in a business context. We steer away from complexity and go for practical value.

When reasoning with taxonomies, you often want something — an animal, the desired car, the requested permit to stay — to be classified. The underlying assumption is that in order to qualify as falling under the top node in the taxonomy, you have to qualify as one of the taxonomy leaves.

Thus, in Figure 8, the goal would be to establish if a given animal is a bird. The arrows are interpreted as implication: if duck, then bird. The inference engine backward chains starting from bird and tries to establish that the animal is a bufflehead, a mallard, or a gull.

**Classification taxonomies and context taxonomies**

In a business context, classification problems are most often about bringing together two independent taxonomies. Take, for instance, the process of handling a request for a visa. There is a classification of different types of visa, each applicable to a specific type of situation that the applicant is in. In such a case, we call the visa classification the classification taxonomy, and the taxonomy describing the possible situations of the applicant, the context taxonomy. The trick is that you can derive the correct visa by evaluating the applicant’s situation using one or more contextual features. Another useful term for a context taxonomy is feature taxonomy.

The basic idea behind this is well-known and has a long history. We have come across it in Part I when we discussed single-access keys in biological classification. A field guide to birds in Scotland provides a kind of decision process to classify birds, based on what the bird watcher may observe about a bird: its environment, its behavior, its color, and so on. Conceptually, classifying a bird and determining the correct type of visa represent exactly the same kind of decision process.

There are strict relations between classes in a context or feature taxonomy on the one hand and the related classification taxonomy on the other. In the next subsection we illustrate this with examples.
Examples with Be Informed BPP

Be Informed business process platform (BPP) supplies a modeling language and an execution environment that turns these models into working computer applications. In this subsection, we use this tool to illustrate how concept computing works, based on a simple example from classification in high-school biology. Such examples have the benefit that they are clear, easy to understand, and that they are accompanied by clear classification criteria available in the Wikipedia.

A simple example: the knowledge model

Our example is the classification of four kinds of insect: butterflies, moths, thread-horns and soldier flies. Butterflies and moths are Lepidoptera, thread-horns and soldier flies are flies, also called Diptera. In Figure 9, this taxonomy is expressed using the Be Informed modeling language. Next, we define three feature taxonomies:

- Number of wings. Flies have two wings, butterflies and moths have four.
- Antennae structure. Butterflies and moths are distinguished by antennae structure. With butterflies, these are filamentous, with moths, they look like feathers and have a comb-like structure.
- Antennae length. Within the group of flies two groups are distinguished based on length of the insect’s antenna.

This leads to three different taxonomies of classification criteria, as in Figure 10. We have given the concepts in these taxonomies a different color to emphasize the conceptually different nature of a feature taxonomy. Note that the relation between each specific feature and its superclass is the same as in Figure 9. This makes sense: filamentous antennae structures form a subset of

![Figure 9. A simple classification taxonomy in Be Informed.](image-url)
We now put the classification taxonomy and the feature taxonomies together. We link the appropriate features to the classification taxa using a relation called “requires”. This relation indicates that the feature is a defining characteristic of the class defined in the classification taxonomy. Classifying as a moth “requires” an antennae structure of the comb-like type. This is shown in Figure 11.

This knowledge model is a completely declarative representation of how our four different kinds of insects relate to each other and which features uniquely identify them. Nothing is said about how you would apply these criteria, or in which order, during the classification process. An effective inference engine can derive that autonomously, based on the concepts in this representation.

Figure 10. Three simple feature taxonomies.

antennae structures.

Figure 11. A simple ontology connecting a classification taxonomy and three feature taxonomies.
A simple example: the web application

Let us now use this model to create an interactive web application that works like an electronic single-access key. You could run it on your smart phone while spending your vacation as a hobby entomologist in the field.

In Be Informed BPP, this is called a Knowledge Instrument. All we have to do to create one is to specify the goal concept — in this case, Insects. In doing so we tell the inference engine that our goal is to classify something as an insect. The inferencer will then backward chain through the semantic network. When it has reached a point where it cannot make further inferences, it will interactively ask the user for input about the current concept.

Also, when navigating the ontology, the inferencer will first follow “requires” relations, then “subset of” relations — at least, by default, because this can be configured. Having created the Knowledge Instrument, we can now run it and see what happens.

So we start the web application. The inferencer starts navigating the ontology, starting from “Insect.” Its first stop is when it reaches the concept “Number of wings.” Since it can make no further inferences about this concept being applicable or not, it asks the user in an interactive dialogue, as shown in Figure 12.

The answer determines which of the two branches of the classification taxonomy is relevant. If our insect has four wings, the inferencer will continue to ask only about antennae structure, not about length: length is only relevant for flies. The full deduction is shown by default at the end of each interactive session, as shown in Figure 13.

Under the hood, each concept is interpreted as a proposition that can be true or false. The relations between the concept, such as the “requires” and the “subset of” relations create complex propositions. The inferencer can deduce truth values based on the truth tables defining the relations. The “requires” relation has the same definition as logical conjunction, while the “subset of” relation equals material implication. See the truth
tables in Figure 7 above

To this basic level of semantics, an additional level is added: the label on the relation not only tells the inferencer how it can infer truth values, but, in addition, also how to traverse the semantic network. This leads to a rich and intuitive modeling language and a powerful execution environment to put these models into action.

A slightly more complex example

A running example of a more complex classification task is Volkswagen car-configurator, which we discussed in Part I and II. In this task, three taxonomies are used — feature taxonomies describing different car features. We discussed the complexities for searching and filtering that arise when we add a business rule saying that car color depends on model. Concretely, if the model is Passat, then the color black is not available. It is instructive to see how concept computing may deal with such a business rule. The model in Figure 14 illustrates how we could model the task using Be Informed BPP.

In this example, the relation labeled “is option when (or FALSE)” means that the color black is only an option when the car model is Golf. Otherwise, it is not an option and evaluates to false automatically. The inference engine uses this knowledge in its decision process. When during the interactive dialogue the user makes it known that he desires a Passat, the only color options displayed are silver and blue. When the user chooses for a Golf, black is also listed as an option for color, alongside silver and blue.
Notice that the business rule implicitly adds a constraint to the order in which values for three features can be determined: before you can determine the car color, it is necessary to know first which model is chosen. The inference engine will automatically detect this and act accordingly.

This shows how adding a simple declarative business rule creates procedural complexity. What makes concept computing so powerful is that it can deal with this automatically.

3 Executing business processes

So far, we have considered knowledge models to support decision processes, such as classifying an insect or choosing a car configuration. The same knowledge modeling techniques can also be used to support the execution of business processes in general. This gives rise to fascinating results. In this section, we take a closer look at how the Be Informed modeling language achieves this and what the benefits are. In doing so, we draw on a publication by Jeroen van Grondelle and Menno Gülpers, product architects at Be Informed.⁶
Concept Computing meets Case Management

The basic idea is to enrich the modeling language for decision processes with concepts specific to other types of business processes. Besides an inference engine, specialized engines are supplied that detect the presence of these concepts and generate specific behavior based on these. Examples of such predefined concepts are Activity, Time Limit, Business Object, Document and so on.

A link labeled “creates” that connects an Activity to a Document means that that specific activity generates a document. This triggers an engine specialized in generating documents. The Document Generation Engine can easily be configured with behavioral properties one would expect of such an engine, so that the correct text and property values are inserted in the document, and the correct styling is used.

Two important concept types in the modeling language are Activity and Case. Several activities need to be performed on a case to reach its end point. An activity has preconditions and postconditions. Consider the example in Figure 15. The stereotypes between angled brackets, such as <<Activity>>, indicate the type of the concept, hence, the kind of behavior it implies.

The goal of a Housing Benefit Case is to publish a decision. In order to do so, eligibility has to be decided first, as the “precondition guard” on the link between the activity and the document indicates. This is done in the Assess activity, which requires that the Request details are available. It can thus be inferred that the activity called “Accept request” is the first step in the

Figure 15. A model describing the Housing Benefit process.
process. The modeling environment allows you to specify the actual properties of the business object Applicant and the case object Request details. Based on this, forms are automatically generated so that the user is prompted to enter the necessary data when performing the Accept request activity.

The Eligibility for housing concept takes you to the part of the ontology where that decision is modeled. Its structure will look similar to the insect classification and car configuration models discussed above, although the concepts will of course be different.

The modeling environment will automatically generate a web application to provide an execution environment for the models. When opening a specific case, the default lay-out will present a main panel for navigating the case information and the dossier. Alongside the main panel, additional smaller panels are shown for search and for activities. The overall lay-out is shown in Figure 16.

Let us zoom in on the activities panel. It indicates which activities are available for the case being viewed at that point in time. The user can click a task in this list to activate the task. The user will then be prompted for the
Figure 17 illustrates the activities panel that represents the model in Figure 15. Apparently the Accept activity has already been completed for the case at hand, as the check mark indicates. The Assess activity is ready for execution. Clicking the link will execute the activity, which, as can be seen from the above model, involves taking a decision about eligibility. A modal panel on the screen will guide the user through the decision process, using the same kind of interaction seen in the insect classification example discussed previously.

The activity Publish is not yet available, as the pause-sign indicates: first, the Assess activity has to be completed.

Thus, the model shown above in Figure 15 is directly executable in the form of a web application. It does not function as input for a team of programmers. No translation necessary.

**Managing knowledge models in real-life**

A growing number of organizations use concept computing for knowledge and rule management purposes. In practice, the ensuing knowledge models can grow fairly large. At IND, the Dutch immigration office, the complete ontology comprises more than 40 thousand concepts and 70 thousand rules.

To keep ontologies of that size manageable, modularization is necessary. To be able to live up to its promise of closing the gap between business and IT, it is essential that the platform supports modularization in a way that is understandable to business persons. Therefore, we discuss two aspects of this by way of example.

At the finest level of granularity, the Be Informed modeling environment provides support for linking concepts across canvasses. These canvasses are called tiles and comprise the basis for the modularity of knowledge models. Concretely, you could open the above knowledge model of the Volkswagen car configurator and move each of the three taxonomies to a separate tile. The local model then looks as in Figure 18.

The little arrow in the top left corner of each of the orange concepts indicates that they are reference concepts. You can double-click on each to navigate to the tile on which the original resides. The modeling environment offers tooling
for searching and finding individual concepts, finding all the places where a concept is used, validating constraints on models, and so on.

At the coarsest level of granularity, the ontology is split up in self-contained functional modules that together cover all business functions an organization performs. Each module contains a set of models underlying a separate unit of functionality. The description of the business functions in these terms is called the Target Operating Model or TOM for short. An example of a real-life TOM for the patent domain is given in Figure 19.

The TOM is model of the organization and its business functions. It details, at the highest level of abstraction, how more elementary functions are coherently grouped so as to constitute a working whole. When the TOM and its underlying models are exported, the result is a technology-independent specification of an organization’s functional architecture.

**Figure 19. A Target Operating Model.**
Figure 21. Overview of roles as presented in the business thesaurus.

Within the TOM, several different types of module are distinguished. This helps understanding the generic characteristics of a module. For instance, the fact that Natural Persons in Figure 19 is a Registration reveals that Natural Persons is about persisting data and the relations between these data in a database. From the name of the module one can legitimately infer that these data are about natural persons. The module contains taxonomies and other knowledge models describing all there is to know about persons, within the patent domain.

Creating a business thesaurus

There is more to knowledge models than just being able to execute them. The models can also be viewed in the form of a so-called Be Informed knowledge base — essentially, a navigable web application that displays the concepts in the underlying knowledge model. Each concept is displayed on a separate page. The page also displays hyperlinks representing the relations which that concept has to other concepts. Such a knowledge base essentially constitutes a business thesaurus. The thesaurus can be enriched with extra information, yielding a powerful instrument for knowledge management. In Part II, we have discussed the notion of a thesaurus as a pillar of effective knowledge management. Leveraging the Be Informed business thesaurus is more than a handy side-effect of an implementation. The thesaurus can be seen as the basis for functionality, rather than the other way around.

Let us briefly illustrate how this works. Figure 20 shows an example of a Be Informed knowledge base.
model. It displays an activity called Assess. Users in the role Assessor are authorized to execute this activity, as the “performed by” relation indicates. The orange box with the checkbox designates that there is apparently a precondition that must be fulfilled before the activity becomes available to users. The blue concepts indicate that there are two system tasks available for performing the Assess activity, each with electronic forms associated to them.

This model is directly executable. At the same time, however, the model can be viewed through a navigation instrument. The Be Informed modeling environment makes a number of access paths available by default.

For instance, the thesaurus user can start the navigation by opening an overview of user roles, as illustrated in figure 21. One can move one level deeper by clicking a role in the list. The resulting view displays the activities that a user with that role — in this case, the role called Assessor — is authorized to perform, as in Figure 22. Each activity in the list is a hyperlink to a detailed description of that activity. The screenshot in Figure 23 displays the description page of the activity Assess.

As can be seen, the information contained in this thesaurus is rudimentary. However, since the thesaurus is a view on the underlying, executable models, it can be truly said that the documentation “is” the application. The importance of this can hardly be underestimated. When the underlying permissions change, both the behavior of the system and the information displayed in the thesaurus, change in unison. If we retract the permission for Assessors to execute the activity Assess, this will have immediate effect for

**Figure 22. The description of the Assessor role.**

**Figure 23. The activity Assess as displayed in the business thesaurus.**
the end user of the system and for the thesaurus.

It is possible to enrich the content of the business thesaurus with extra information. The screenshot in Figure 24 shows the same description page of the activity Assess as the one in Figure 23, but now enriched with descriptive information, a link to the relevant passage in the work instructions, a link to a relevant piece of legislation, and links to requirement descriptions. Some information may be hidden for certain user roles. For instance, requirements documentation is not of interest for business end users.

A useful business thesaurus arises only through thorough design, never by accident. A separate whitepaper discusses the major design choices and their consequences in detail. The Be Informed business thesaurus has such great potential to bring knowledge management and information management together.

**Benefits of concept computing**

The goal-oriented, declarative approach to business process modeling has a profound impact on business and IT. We have discussed the benefits of concept computing at a more conceptual level in the first chapter of this paper. In their paper quoted earlier, Jeroen van Grondelle and Menno Gülpers analyse these benefits in more detail. Let’s see what they come up with, using the terminology developed in the previous chapters.

For end users, the resulting IT-system offers more flexibility:
• **Knowledge workers in control.** A concept computing application infers which actions are available at any given point, as the example in figure 16 shows, instead of forcing a fixed linear progression through the steps. A knowledge worker is free to determine the best order in which activities are done. When an activity is not available yet, the system offers insight in the reason for this: wrong authorization, a time constraint, lack of information, and so forth. The ability to quickly resolve such questions is a great help in stimulating uptake.

• **Ad hoc interventions.** Explicitly modeled preconditions make it easier to repair a process when an exception occurs. For instance, a precondition may state that a certain document be available. Manually inserting such a document in the system may be all that is needed to let the process resume. In systems that only recognize sequences of actions, such remedial actions are not possible.

• **Flexible switching.** Most business processes are partly automated, and partly manual. Goal-oriented systems allow for transparent switching between the two. When straight through processing (STP) runs into an exception, the specific case can be treated manually. After “fixing” the current activity so that its postconditions are fulfilled, STP can simply resume, as in Figure 25. There is no need to list all possible exceptions up front and include these in a design, as in a procedural approach.

• **Feedback on process quality.** Like traditional systems, concept computing systems can report on different types of quantitative and time related KPI’s, such as mean time to process a request or number of requests per time unit. In addition, goal-oriented systems can report on which preconditions had which impact. The same information used to generate the panel displaying available actions can be used to report on what prevented automated processing. This kind of feedback is valuable input for business process improvement and optimization.

For the IT-department, compliance officers and process consultants, concept computing is also a leap forward. Major benefits include the following.

• **Executable specifications.** Declarative models, especially when enriched with links to policies and legislation, constitute an excellent vehicle for specifying IT-systems. These specifications are directly
executable. No more handover and sign-off processes needed. The process from business need to IT-solution is cut short dramatically.

- **Business thesaurus for transparency.** The business thesaurus describes the business ontology and greatly enhances knowledge management. Tracing between requirements and functionality is made transparent. The structure of the business support systems and the underlying rules are open for inspection. This supports business agility and decreases the cost of change.

- **Business ownership.** Goal-oriented, declarative business models are highly modular. The business ontology consists of progressively more fine-grained units separated by clearly defined criteria. The units are highly independent of each other. This fosters business ownership of the various units. One business unit owns the decision logic for deciding eligibility, another, the way applicants are registered, and so on.

- **Focus on high-quality concept definitions.** The traditional, procedural approach focuses on complexities in process flow. The proper definition of the concepts and rules therefore often receives less attention than it deserves. Declarative modeling allows one to focus on what matters: the language and behavior of the business. A high-quality thesaurus is just as important for describing business processes as for supporting search.

- **Design-time quality checks.** Declarative models with rich semantics offer new possibilities for consistency checks at design time. Algorithms based on graph traversal can be used to spot inconsistencies. Circular, conflicting or mutually exclusive preconditions can be detected at design time instead of waiting for error messages and bugs at runtime.

### 4 Where to go from here

Now that we have seen how concept computing can be applied to build adaptive case management systems and what the benefits are that it brings, we try to look into the near future. What can and must an organization do to optimally profit from the rise of concept computing? The following topics are important in this context:

- What kind of platform to use?
- What kind of skills to develop?
- What about other semantic technologies?

We discuss these topics in turn.
Platforms for concept computing, or Against BRE’s

Currently, there is only one platform on the market that can truly be said to support concept computing in business processes: Be Informed BPP. It is still a unique product, though not so very new anymore: at the time of this writing, Be Informed 4 goes to market. The 3-series has been with us for many years now and is used in many large-scale implementations.

Sooner or later, other technology providers will join in and the competition will start. It is interesting to think about which type of product may evolve in a platform for concept computing. An obvious answer would seem Business Rule Engines or BRE’s for short, such as Oracle Policy Automation or IBM WebSphere Operational Decision Management.

There are two major problems with BRE’s that must be tackled before a move towards concept computing becomes possible for them. Since many organizations currently tendering for case management solutions ask for a BRE as part of the desired solution — it is certainly a hot topic nowadays — it is worthwhile to dig a bit deeper.

The first problem is that in BRE’s, rules are treated as first-class denizens but concepts are not. Yet, as we saw earlier, a rule is best understood as a relation between concepts: relations between concepts must be established so that their meaning becomes explicit.

BRE’s, on the other hand, take data structures as their starting point. Take for instance WebSphere Operational Decision Management (WODM). In an IBM whitepaper on the subject, it is carefully explained that a software object model is the starting point for rule definition:

> Software developers typically express their understanding of the information required to run business systems in terms of an object model. [...] The object model defines how a software program manipulates data, and WODM allows all the terms in the object model to be translated in a vocabulary that policy managers can use when writing rules.⁹

This is illustrated with a two-column table that maps object model terms such as “CustomerClass” to “The customer’s priority level.” Consistent with an IT-centric view of the world, IT-artifacts are taken here as the point of departure. The business vocabulary is just a by-product. Concept computing is based on precisely the opposite approach: the business vocabulary is the fuel that drives the system. Retro-fitting WODM — or other BRE’s — to make this possible would amount to a fundamental change.
The second problem is the separation between rules and processes that most BRE’s impose, often supported by the slogan “Separate the know from the flow.” This fits in an architectural pattern where a Process Engine is placed alongside a Rules Engine. The one runs on process models, the other one rule sets. This is generally a bad idea.

As I have pointed out in more detail elsewhere, this easily leads to an inordinate amount of cross-references between process fragments and rules. One rule can be relevant in many process fragments. With tens of thousands of rules, maintaining the cross-references becomes a gigantic task. In practice, this leads to a brittle architecture that is not conducive to change. The underlying problem is that processes and rules are intimately related. Putting them in different components is the wrong approach to separate the two.

Perhaps the two oldest and most important IT-architecture principles are high coherence (use components to group together what belongs together) and low coupling (minimize the references from one component to another). Separating processes and rules in two distinct components violates both.

In addition, there is a governance issue, as Max Pucher points out in his article quoted previously. There is only one business process owner who tries to achieve a positive outcome for the customer and he or she defines both the process and the related rules.

Like a BRE, concept computing separates the know from the flow, but emphatically not by putting them in separate components. Rather, it separates them by abstraction. We define the concepts and rules describing objects, attributes, actions, events, and so on. As we have seen, they are all part of the same semantic network: the business ontology. The process is derived from that ontology. This is precisely the reason that the business ontology — or the thesaurus view on it — is so much more resilient to change. Again, retro-fitting a BRE to meet this criterion amounts to a major transformation.

In conclusion, there does not appear to be a simple, straightforward evolutionary path from the BRE to concept computing. It is easier to add a rule engine to a platform for managing ontologies than the other way around. Therefore, it is more likely that future technology providers of concept computing will come from the semantic technologies space, which we discussed in Part II. It is no coincidence that Be Informed itself started out once as a provider of semantic search solutions.
Skills for concept computing

With concept computing, the business sits in the driver seat, obtaining much more direct control. More control carries new responsibilities with it. To harvest all the benefits of concept computing, an organization must not only choose a platform, but also develop the necessary skills.

Coos van der Togt, founder and CEO of O&i and renowned practitioner of business rule management, points out that modeling legislation directly in executable rules cuts the development process short by many orders of magnitude, but also that in his experience, the skills involved in this constitute a delicate mixture of proficiency in interpreting law and policies, understanding execution practice, and modeling skills. These latter skills include, among other things, insight in and experience with the modeling principles discussed in Part I.

The complete set of knowledge models underpinning a business support system constitutes an ontology. A business thesaurus is essentially a view on this ontology, and the skills involved in setting up and managing either of them are largely identical. They differ significantly, however, from the skills involved in programming or requirements analysis.

In section 1 above, we discussed the specific skills involved in knowledge modeling are closer to the business, and how the resulting models are insightful to business persons. We illustrated this with the MeSH thesaurus, which is a thesaurus used in the medical field. The MeSH-team consists of specialists with a background in medical science, library science and knowledge management. They are not technicians.

When we think of business and IT in terms of demand and supply, the quality of the business ontology is the responsibility of the demand side. The supply side is responsible for technical aspects, such as the configuration of adapters for integrating with external systems, system performance, and database maintenance.

The art of ontology construction is an integral part of knowledge management and it includes all the skills needed for managing large scale business ontologies. Organizations that use Be Informed often have dedicated teams specialized in managing the ontology, changing it on a weekly basis to account for new rules and policies.

An important part of managing an ontology is managing the business rules comprised in it. Organizations that use BRE’s have specialized teams that translate legislation and policies into rule sets. While using a BRE does not
yield the full benefits of concept computing, the skills necessary for authoring rules largely overlap with ontology-management skills.

In the 1990, much work has been done on knowledge engineering, which is essentially a discipline of system engineering based on knowledge modeling. It has led to a number of important publications on knowledge modeling. Many of the practical insights in how to approach the representation of knowledge and rules are directly applicable to concept computing. Although knowledge engineering as a discipline has occupied a somewhat marginal place over the past decade, it is poised to make a comeback as concept computing continues to gain market share.

In conclusion, an organization that wants to make use of concept computing needs to develop its knowledge management skills.

**Other semantic technologies**

It is useful at this point to pay attention to what will happen when more and more organizations make a move towards concept computing. At that point in time, concept computing will have growing impact on the way organizations interact across their borders with each other.

The semantic models it uses can be published using W3C semantic web standards such as OWL, RDF and RDF(S). When published in such formats, parts of an organization’s knowledge are made available to partners. This is called Knowledge as a Service, or KaaS for short.

Models can then be shared and reused in different tasks in different areas. This will go a long way in realizing Tim Berners-Lee vision of the Semantic Web:

> “I have a dream for the Web [in which computers] become capable of analyzing all the data on the Web – the content, links, and transactions between people and computers. A ‘Semantic Web’, which should make this possible, has yet to emerge, but when it does, the day-to-day mechanisms of trade, bureaucracy and our daily lives will be handled by machines talking to machines. The ‘intelligent agents’ people have touted for ages will finally materialize.”

Behavior and language are intimately interrelated. Sharing data implies sharing a common language, and sharing a language means sharing a behavioral repertoire. Knowledge models created to support business process execution are therefore at the very core of the semantic web.
The pervasive availability of knowledge models published in formats that observe semantic web standards will revolutionize the way we do business. The same models can be used for many different purposes, both in the area of process execution and semantic search.

In Part II, we briefly discussed an example of this in the patent domain. During its processing, a patent application will be assigned metadata indicating the technical field it belongs to. This enhances its findability, but may also influence the process flow it goes through, such as determining the specific organizational unit that will process the application. In this example, one taxonomy is used for search and for process execution.

More generally, concept computing applications will integrate other semantic technologies, such as search. This is logical, since all these technologies are fueled by the same kind of knowledge models.

5 Concluding remarks and take-aways

We have now turned full circle. We started in Part I with the observation that search technology and process execution converge, and continued to discuss some foundational aspects of knowledge modeling. In Part II, we discussed how knowledge models are used in the area of findability, knowledge management and search technology. In this part, we focused on concept computing — business process automation based on executable knowledge models.

Concept computing yields great benefits for organizations when compared to the classical approach to model driven process support systems, which is based on flow chart models. The models in concept computing are declarative, that is to say, they abstract away from process flow as such. They are more expressive and therefore closer the reality of the business.

Besides concept computing, two major trends challenge the flow chart approach. Enterprise decision management demands that business logic be treated as an asset and must not be hidden in process models. Adaptive case management demands better support for knowledge workers and systems that are resilient to change. Concept computing supports both trends, which lends it extra urgency.

To see how concept computing really works, we developed a simple application for hobby entomologists to classify insects, and demonstrated how the same model driven approach can be used to execute business processes
in general. The models can be viewed as a kind of business thesaurus, a pillar of knowledge management.

Based on how concept computing works, its major benefits were explained. These can be summarized as a better user experience and more resilience to change.

In chapter 4, we concluded that current platforms for concept computing derive from semantic technology, and that this is probably also the case for future platforms. Organizations planning to make use of such platforms should develop their knowledge management skills to yield the full benefits of concept computing. As more and more organizations will commit to concept computing, knowledge models will be published, exchanged and shared. Tim Berners-Lee’s dream of the Semantic Web will become a reality.

The taxonomy revolution is taking something humans have been good at for millennia — creating classification systems and lay these down in knowledge models — to a new level. It merges these techniques with computer technology, yielding a new synthesis. The result is a brand new species of process support systems and a new way of creating networks of data sets linked at the semantic level. This will profoundly change the way we do business, the way we organize our environment and perhaps even the way we live our lives. The first signs of this are rapidly becoming apparent.
1 See Davis (2012).


3 See Drucker (1999).

4 See Max J. Pucher, “The Elements of Adaptive Case Management.” Published as Chapter 5 of Swenson (2010).


6 The image is taken from Van Grondelle and Gülpers (2011)

7 The following discussion and illustrations are quoted from Van Dijk, Voskuil en Wentink (2012).

8 See Van Dijk, Voskuil en Wentink (2012).

9 Stineman (2012).

10 See Voskuil (2011) for more elaborate discussion.

11 See for instance the seminal publication on the CommonKADS-methodology (Schreiber et. al. 2000).

12 See Berners-Lee and Fischetti (1999), Chapter 12.
Bibliography


About Taxonic

Taxonic is thought-leader in the application of semantic technologies in business processes. Focussing on dynamic case management, semantic search and linked data, we help organizations to become a master of adaptivity by leveraging the benefits of these innovations.

About the author

As CEO of Taxonic, Jan Voskuil is responsible for answering clients’ business challenges with innovative solutions. After obtaining a PhD in linguistics, Jan worked for a suite of dotcom startups in the field of artificial intelligence and disruptive technology. Before founding Taxonic, Jan worked as a business consultant for Logica.