Elements of a business process management system: theory and practice

Duncan R. Shaw
Nottingham University Business School, The University of Nottingham, Nottingham, UK

Christopher P. Holland and Peter Kawalek
Manchester Business School, The University of Manchester, Manchester, UK, and

Bob Snowdon and Brian Warboys
Department of Computer Science, The University of Manchester, Manchester, UK

Abstract
Purpose – To construct, test and illustrate a sophisticated and theory-based architecture for analyzing business process management systems (BPMS) used for business process change.
Design/methodology/approach – Exploration of business process modeling-based BPMS via a meta-survey of academic and business literatures. Two main dimensions are used based upon semiotics and a block-based BPMS pyramid architecture. Each block is a core technology required for the functioning of the BPMS and include: the subject being modeled; the software formalism; the IT infrastructure; the modeling language and notation; and the underlying technical infrastructure.
Findings – Theoretically explains and empirically illustrates each core technology in the proposed architecture then does the same for the architecture, its arrangement as a whole and its interrelationships. Recognizes the lack of a theoretical basis for business process modeling constructs and the dangers that this generates. Explains why automatic BPMS require formal construct transmission from subject modeled to modeling hardware and software.
Research limitations/implications – The architecture’s core technologies span numerous disciplines so each set of literatures introduces the component concepts and their bases but is not exhaustive.
Originality/value – This paper proposes a considerably more sophisticated framework for BPMS analysis than is currently available; it is theoretically and not just empirically based; it uses a novel method of theoretical justification concerned with the transmission of modeled properties and characteristics between several technological media; and it illustrates the innovative analytical use of this architecture and the practical use of BPMS with three different case vignettes.

Keywords Business process re-engineering, Modelling, Electronic commerce

Paper type Research paper

Introduction
An organization’s current performance depends upon its business processes’ collective ability to achieve its fundamental objectives. However, an organization’s longer-term
performance depends upon its ability to satisfy exogenous change requirements such as product lifecycles, new competitors or other types of environmental change. We refer to managing the process of changing an organization’s business processes as flexibility and we define it as that characteristic which enables an organization to deal with the complexity (Anderson, 1999), or the emergent variety (Beer, 1979), of its environment. This paper is concerned with business process flexibility: the ability to change organizational capabilities repeatably, economically and in a timely way. Thus, we focus on a meta-process: the process of changing a process (Warboys et al., 1999, p. 26; Ellis and Keddara, 2000) and business process management systems (BPMS) the collection of technologies that allow humans to better manage this meta-process.

First we define our research problem and justify its alignment with BPMS and our research method. Then we review the different technologies that are required to build a BPMS, propose an architecture that describes their salient inter-relations and then illustrate the use of BPMS in three different case vignettes. Finally we discuss the contribution of our architecture and implications for further research.

Conceptual foundations
Our research is concerned with the use of information systems technologies to improve organizations’ abilities to better manage the process of changing their internal and external processes. Commonly the technologies that are used for this are called BPMS, which are significant extensions of workflow management (WFM) (van der Aalst et al., 2003a, p. 5) (also, see section 3.1). BPMS are able to support business process management because their technical systems are joined to the business processes of the organization’s wider socio-technical system (Mumford, 2000), which they help to manage. In effect they are part of the same system. A business process is a socio-technical system, executed by humans and machines, and a BPMS is purely a technical system.

In order to investigate this we require an analytical framework and the development of this framework, which we have called the BPMS pyramid architecture, is the research problem of this paper. Such an architecture needs to span both social and technical systems in order to manage human and machine-based business processes and be run on an information technology platform. In fact an investigation of the actual join between the social and technical systems is an implicit use of the architecture although this is not covered in detail here.

Research methodology
Our methodology is to explore the current state of BPMS and its foundation upon business process modeling (BPM) via a meta-literature survey. Our meta-literature survey is a survey of more than 20 literature surveys that is based upon the premise that any categorization, taxonomy or framework of BPM must in itself be based upon a meta-model of the models that it contains. Such articles cover more research than single model-only articles and so contain a wider range of modeling concepts and ideas.

We use two main dimensions for our analysis of articles. The first one is a dimension made up of three semiotic sub-dimensions: syntactics, semantic and pragmatics and the second dimension is the set of core technologies in our BPMS pyramid architecture (Figure 1). We used articles that surveyed a number of business process model construct lists, then categorized how these constructs lists fitted the three semiotic dimensions
within each of the core technologies. Our use of semiotic theory, the theory of signs, has many precedents and reflects the ability of information systems to be taken as sign carriers and models to be taken as signs (Liu, 2000).

For example, we categorize Curtis et al.’s (1992) three constructs (agent, role and artifact) as syntactic and semantic modeling constructs. The four perspectives (functional, behavioral, organization and informational) are categorized as part of a model abstraction technology and we categorize the seven goals (see section 3.1) as pragmatic incidences of BPM technologies. We call the blocks in the BPMS pyramid core technologies and we refer to them by using capitalized first letters. Each core technology block only affects others in the vertical dimension, i.e. it rests upon the core technologies that are required for it to function as part of a BPMS. Blocks on the same level have no interdependencies and the architecture bifurcates at formal model constructs and software application signifying that there are no interdependencies between each “leg”.

Next we describe the BPMS pyramid architecture with some example technologies and then use case vignettes on SAP’s enterprise resource planning (ERP) configuration model, RosettaNet’s B2B interaction model and Intalio’s automatic order fulfillment system (for LexisNexis) to illustrate in practice how our architecture can be used to analyze a spectrum of business process management tasks from process standardization to business process automation. Finally, we discuss the contribution of our architecture and implications for further research.

Elements of a BPMS
Based upon an extensive review of the academic literature and business practice we have identified a number of interrelated elements that are common to all BPMS.
A block diagrammatic representation was chosen to display the BPMS pyramid architecture to signify that each core technology block is a prerequisite for the core technologies that it supports (Figure 1).

**Business process management system**

During business operations, and business process improvement projects, processes are composed into more complicated business processes and eventually into services (Kalakota and Robinson, 2003, p. 70). The composition process itself includes many design decisions that configure the resulting process composite by choosing which processes to use in the composition, where they are sourced as internal or external services, how they are configured and what process is used to change the composition. It is this variety of possible design choices and the complexity of how this variety changes, due to service offer changes and service-need changes, which the BPMS manages.

Howard Smith and Peter Fingar describe a BPMS as a modeling, integration, and execution environment for the design, manufacture and maintenance of business process and point out that “Just as relational database management systems supported the aggregation of business data and the creation of enterprise data models, a BPMS achieves the same for business processes” (Smith and Fingar, 2003, p. 15). Hollingsworth (2004, pp. 300-02) describes a BPMS as supporting a similar process design-execution-redesign cycle via an evolution of workflow management systems (WfMS) and their convergence with enterprise application integration and world wide web technologies. The addition of these later two technologies led to a greatly increased potential for BPMS and WfMS to support change in inter-organizational processes (Sayal et al., 2002; Basu and Kumar, 2002). We make no distinction in this paper between inter-organizational processes (between companies) and intra-organizational processes running between different parts of a single company because of the inherent recursiveness of social systems.

**Enactable business process model**

Curtis et al. (1992, p. 77) list five modeling goals: to facilitate human understanding and communication; to support process improvement; to support process management; to automate process guidance; and to automate execution support. We suggest that these goals plus our additional goals of to automate process execution and to automate process management, are the goals of using a BPMS. These goals, which form a progression from problem description to solution design and then action, would be impossible to achieve without a process model.

This is because an enactable model gives a BPMS a limited decision-making ability, the ability to generate change request signals to other sub-systems, or team “members,” and the ability to take account of endogenous or exogenous changes to itself, the business processes it manages or the environment. Together these abilities enable the BPMS to make automatic changes to business processes within a scope limited to the cover of its decision rules, the control privileges of its change request signals and its ability to recognize patterns from its sensors. Warboys et al. (1999, p. 38-44) divided models up into five characterizations, which overlap:

1. **Static models.** Where the model’s representation of the subject modeled is a “snap shot,” i.e. the model does not represent the subject’s dynamic behavior.
(2) **Dynamic models.** Where the model’s representation of the subject includes the subject’s dynamic behavior.

(3) **Passive models.** Where changes in the subject cannot influence the model after the model is created.

(4) **Active models.** Where the subject and the model influence each other as part of the same system.

(5) **Enactable models.** These are models that are modeled in a medium that allows them to execute and thus become active.

Only an enactable business process model gives a BPMS the ability to automatically manage business processes because a single system that is part model and part business process—that-is-modeled is able to:

1. signal a change in the business process (the subject) to the controlling machine via a change in the model and in reverse; and
2. cause change in the business process via a change in the model made by the machine.

Active models can only do (1) but a BPMS needs the ability not just to make processes changes but also to react to process changes cause by itself or other agents. For example, a car assembly simulation running on a laptop is an active model and update’s itself to reflect an increasingly more complete car progressing along the line.

For the BPMS to be automatic the model has to be enacted by a machine, which will use a software application to do so, and for a process model to be enactable by a machine it has to use formal model constructs (Figure 1).

**Formal model constructs**

Warboys et al. (1999) wrote, “Models, either physical or graphical, provide a way of mapping and preserving a clear relationship between model and real world subject.” They then list four things that are necessary for a model to exist: the part of the reality that is the subject modeled; the model itself; the relationship between the model and the subject modeled; and an observer, user or creator of the model. A model is a planned abstraction of reality represented in a form that is usable by a human. If the model is an active model then a machine must enact it. Without the model there would be no connection between machine and business process.

A model construct is taken from Curtis et al. to mean those things that “…collectively form the basis of a process model …” In this paper we develop this further:

An enactable model is a composition of model constructs that is derived from the properties of the physical, hardware or software modelling medium that together naturally display characteristics that exactly replicate those of the subject abstraction.

Model constructs are like prefabricated construction elements and examples include Curtis et al.’s agents, roles and artifacts. They are independent of the means, technique and process of modeling, and there is no theoretical basis for any assembly of business process model constructs that we have seen. Several theory-based constructs have been used to make contributions from areas such as classification theory, speech-act theory
and semiotics (Wand and Weber, 2002, p. 365). Wand and Weber themselves use ontological theory to propose a research agenda framework for modeling grammars, modeling methods, modeling scripts and modeling uses but made the point that this theory like the others is a basis for the representation or communication of the model rather than a theoretical basis for model constructs that describe an abstraction of the subject. The BPM literature pointing to supporting theories for modeling introduces semiotics (Falkenberg et al., 1998; Stamper, 1987; Liu, 2000), Shannon’s (1948) communications theory, classification theory (Parsons, 1996, p. 1438) and ontology theory (Wand and Weber, 2002; Green and Rosemann, 2000).

The lack of a theoretical basis for BPM constructs is quite clear when what the constituents of a theory are compared with the model constructs and frameworks (i.e. constructs) of constructs that we have seen. Theories consist of “what” – the variables, constructs and concepts that describe the subject of interest; “how” – the ways that they relate to each other; and “why” – the reasons for existence of the “what” and their relationships of “how” (Whetten, 1989). Theories do not consist of “lists of variables or constructs” because such lists are arbitrary. Also they lack proofs of completeness and explanations of the relationships between the listed items (Sutton and Staw, 1995). The model constructs in the BPM literature that we have seen “merely list properties” (Lindland et al., 1994, p. 42), thus there is no way to judge the relative or absolute quality of different lists of model constructs.

Some of the lists of model constructs describe all three semiotic dimensions of the subject-model-modeler relationship, like Becker et al. (2000, p. 32), and some seem unaware of the independence of the subject-model (i.e. semantic) relationship and the model-modeler (i.e. pragmatic) relationship. Others like Giaglis (2001, p. 211) seek to develop architecture of dependencies like our BPMS architecture. Bart-Jan Hommes identifies 20 different techniques and over 350 different process modeling tools that use different modeling constructs (http://is.twi.tudelft.nl/~hommes/tools.html, accessed April 7, 2004). In fact, there is a similar over abundance and lack of differentiation in the related but different (Smith and Fingar, 2003, p. 3) area of workflow modeling that has led to the development of a workflow language with the name “Yet Another Workflow Language.” Even model quality models do not have a theoretical basis for their modeling constructs (Hommes and van Reijswoud, 2000). We have not been able to find such a basis but we can list the potential dangers of its absence: no proof of abstraction scope and depth, no absolute value judgments between competing models and no economically efficient limit to the number of model constructs used.

Finally, machine enactability requires that the modeling of the business process must be done by formally creating a model using formal model constructs. Non-formality causes errors during model enaction in software that take the form of uncontrolled divergences between the model and the subject modeled that are not due to purposeful abstraction and ambiguity (van der Aalst et al., 2003a, p. 6). Formal models are built using two “tools”: they have to be written in a formal modeling notation using an ontology-based modeling grammar. The two prerequisites are described in the following sections (Figure 1).

**Formal modeling notation**

The modeling notation is the set of signs and sign combinations used to represent the model constructs, which in turn represent an abstraction of the subject modeled.
In English these are the letters of the Roman alphabet although they could also be any diagrammatic, or audible, symbol and use any media from paper to a PC. For model constructs to be formal they need to be represented in a formal modelling notation. Such a notation has a constant interpretation of the notation’s symbols. Non-formal notation generates errors due to exceptions, inconsistencies and omissions in the notation’s syntax and semantics (Stamper, 1987; Liu, 2000, p. 29). A constant interpretation of the notation’s symbols simply means that once defined the interpretation of a symbol does not change except via another process, which is also defined. Business process modeling notation (BPMN) is an example of a formal modeling notation (www.BPMI.org for draft, accessed April 7, 2004). One of many other examples is the Petri Net, which also possesses a formal grammar and is a formal mathematical and graphical language (Giaglis, 2001, p. 216; van der Aalst et al., 2003a, p. 7).

Ontology-based modeling grammar
The modeling grammar is the set of rules that govern the meaning and structures of the modeling constructs that are used in the model (Rosemann and Green, 2002). In English this is the meaning of a word that is spoken or written. Wand and Weber (2002, p. 364) called it the “conceptual-modeling grammar” and described it as a set of constructs and a set of rules describing how to combine them. The modeling grammar contains the rules governing the use of a semantic component which is the meaning of the concepts embodied by the modeling constructs. It also has a syntactic component, which describes how the concepts relate to each other.

Like the modeling notation the modeling grammar has to be formal for a machine to be able to enact the modeling constructs that it describes. Therefore, the semantic and syntactic components of the modeling grammar also have to be formal. Automation would not be possible if the grammar contained what Wand and Weber (2002, p. 365) called: construct overload (homonyms), construct redundancy (synonyms), construct excess (a meaningless word) and construct deficit (a wordless meaning). Any of these four cases would generate errors due to exceptions, inconsistencies, duplications and omissions in the grammar’s set of concepts and structural relationships. Wand and Weber advocate that a modeling grammar should be ontological since an ontology can be used to formally describe real subjects. An ontology-based modeling grammar is a set of coherent and systematic definitions and rules that describe the concepts in the grammar and how they relate to each other in terms of their meaning and structure. Lindland et al.’s (1994, p. 44) short description of the modeling language L brings together the concepts of syntax, semantics, alphabet, modeling constructs, notation and grammar. Business process modeling language (BPML) is an example of one business process language that uses a formal modeling grammar to describe the chosen model abstraction (www.BPML.org).

Model abstraction
The modeling grammar describes an abstraction of reality (the subject) called the modeling abstraction. This is a subset of the total set of characteristics and properties of the subject and is constructed out of model constructs. The abstraction is made when the model is created using the model constructs and the scope or focus of the abstraction is chosen by the modeler depending upon what characteristics of the subject are to be modeled and in what level of detail. There is a distinction between
abstraction depth, which is the degree of detail, and abstraction area which is the scope or range of subject characteristics and properties modeled (Giaglis et al., 2002, p. 7; Dewhurst et al., 2002, p. 421).

**Business process managed (subject modeled)**
The subjects modeled are the actual business processes managed by the BPMS.

**Software application**
A software application is a prerequisite for a BPMS model because for the model to be enactable it has to be modeled in a medium that allows the subject and the model influence each other as part of the same system. For automated enactment only computer software is able to do this, e.g.: Intalio’s N³ Designer (Smith and Fingar, 2003, p. 10). The level of decision automation varies between the uses of the BPMS and is different to, for example, BPM tools that do not support all of Curtis et al.’s (1992, p. 77) goals of process modeling. BPMS have access to the elements of BPM tools such as: a library of best practice business processes; manual graphical manipulation of configurations; automated change of application configurations; links to communications systems; and possibly, integrated project management and change control (Holland and Kelly, 2002). BPMS can be designed to automatically choose between process reconfiguration options as well as to facilitate manual choice. The blocks below software application signify that the software application that runs the business process model in a BPMS requires both a technical architecture to enact the model on and a software language that encodes the model in a form that is executable on a technical architecture.

**Technical infrastructure**
The operating system and hardware that the Software Application runs on, e.g. Windows and Intel on a PC.

**Software language**
A software application is encoded in a software language, for example, a model modeled in the notation BPMN can be formally encoded within the language BPML for execution on the software application that is part of Intalio’s BPMS N³ Designer. BPML is an XML-based language in that it has an XML syntax. Another example is the modeling language PML executing on ProcessWeb (Warboys et al., 1999, p. 210).

The formal relationship between BMPN and BPML preserves the unambiguous mapping between the characteristics of the initial subject abstraction and the set of Model Constructs enacting on the software. There can be no one-to-one mapping between the characteristics of the initial model and those of the enacting model unless the software notation and the software grammar are formal because mapping ambiguities generate one-to-many mappings. One example of converting from a modeling language medium to a software language medium is van der Aalst et al.’s (2003b, p. 604) modeling of an organization in UML then their conversion of the UML model into XML.

**Software notation and software grammar**
Software is written in a language so it is also has a notation and a grammar. Software notations and software grammars have to map on to their respective modeling
notations and modeling grammars without error so that the model constructs can physically enact in a way that is characteristic of the aspects of the model that they are designed to represent. This is why the notation BPMN is designed to have a formal mapping to the modeling language BPML, which in turn is linked to a software language by writing it in XML. The syntactic rules of XML are combined with construct semantics from a software formalism to create the software grammar. The software grammar in effect models the modeling constructs that were initially used to model the subject’s properties and characteristics, i.e. it is a model of a model – a transmission of properties and characteristics.

**Software formalism**
The software formalism is the set of rules and concepts that are used as building blocks for the construction of the software notation and grammar. Formality assures the mathematical proof of their properties. When they are composed into conceptual building blocks they must capable of representing the characteristics of the model constructs that in turn represent the true fundamental aspects of business processes. BPML accomplishes because its formalism is \(\pi\)-calculus (www.BPMI.org, accessed June 22, 2004) (Milner, 1991), which contains rules and conceptual building blocks suited to first order business process modeling.

**Proposed architecture**
The elementary core technologies that make up our BMPS pyramid architecture are shown in Figure 2 with relevant examples. Our architecture is similar to the standards “stack” of the WFM coalition’s reference model (Hollingsworth, 2004, p. 310) in that it is a parsimonious combination of the technologies required for a functional BPMS. Jablonski also proposes an architecture that is based upon a process model called

---

**Figure 2.** The BPMS pyramid architecture
MOBILE, which is divided into a build-time architecture and a run-time architecture (Jablonski, 1994). All three are designed to be BPMS architectures so some similarity is to be expected but the major difference between our pyramid architecture and the two others is that ours is derived by tracing two “subject property transmission links.” These are the subject-model link and the model-information system link. The subject modeled is the base of the left “leg” and the information system is the base of the right “leg”. We conceptualize the modeling of the properties and characteristics of the subject business process by the modeling medium (via compositions of its own inherent physical properties and characteristics) as a transmission of these physical properties and characteristics. Property information is transmitted up the left “leg” and down the right “leg”. As such it bears some relation to Shannon’s (1948) communications theory and semiotics (Liu, 2000), which are also concerned with information transmission.

To illustrate the value of the architecture in practice we next use it to compare three different examples of model-driven business process management that display varying uses of BPMS: an electronic commerce standard, a brand of ERP software and an automatic BPMS (Tables I and II). Each core technology in our architecture is given a column and each of the three examples is given a row. At the intersection of each column and row we describe the occurrence, if any, of the core technology within the example.

Discussion
An examination of the three case vignettes in the tables shows how the model construct formality and the scope of the model abstraction affect the utility of the model for process management. The RosettaNet, SAP and LexisNexis rows in Table I illustrate the contrast in functionality that modeling formality can enable. In the top row the shallowness of RosettaNet’s model abstraction does not allow model constructs to be proscribed in enough detail so the standard is not actually standardized at a level below main messaging. Thus, optional fields and customization occurs and then automated interaction with new partners becomes impossible due to ambiguity. Interestingly, Sayal et al. (2002) have described how HP’s BPMS, the HP process manager, can be applied to managing RosettaNet’s B2B interaction standards.

RosettaNet on its own is purely a model of a set of interaction instructions whereas the SAP row shows how its internal process model abstracts down to the level of a single process instance (Soffer et al., 2003, p. 675) and uses the R/3 to manipulate a library of standard business processes. This conforms to Curtis et al.’s (1992, p. 77) goals of facilitating human understanding and communication; supporting process improvement; supporting process management; automating process guidance; and to some extent automating execution support. RosettaNet only supports the first goal fully and supports goals two and three partially.

The LexisNexis’ order fulfillment system, however, is a fully automatic BPMS that supports all five of Curtis et al.’s goals together with our suggested goals of automating process execution and automating process management. Its ability to support automated reconfiguration and re-composition of processes at machine speeds and costs is possible because the business processes that fulfill orders are formally modeled using BPMN in sufficient depth to remove ambiguity. The variety of possible requests placed upon the system is constrained by its web-based intranet fulfillment tool but
<table>
<thead>
<tr>
<th>BPMS</th>
<th>Enactable business process model</th>
<th>Formal model constructs</th>
<th>Formal modeling notation</th>
<th>Ontology-based modeling grammar</th>
<th>Model abstraction</th>
<th>Business process managed (subject modeled)</th>
</tr>
</thead>
</table>

Table I. BPMS core technology analysis. Part 1
<table>
<thead>
<tr>
<th>BPMS</th>
<th>Enactable business process model</th>
<th>Formal model constructs</th>
<th>Formal modeling notation</th>
<th>Ontology-based modeling grammar</th>
<th>Model abstraction</th>
<th>Business process managed (subject modeled)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERP software workflow engine and configuration: SAP</td>
<td>Some WFM. Support of manual reconfiguration of business processes using GUI and internal process models (van der Aalst et al., 2003a, p. 3)</td>
<td>Only for workflows. Manually controlled active model library within SAP's R/3 enterprise applications</td>
<td>Not formal: high level framework for a library of standard business processes called the R/3 reference model that are reconfigurable using the R/3 business engineer (<a href="http://www">www</a>. help.sap.com accessed January 26, 2004)</td>
<td>Not formal but standardized</td>
<td>No</td>
<td>Abstraction area covers only processes in the ERP system library. Abstraction depth down to a single instance of a process within a given configuration (Soffer et al., 2003, p. 675)</td>
</tr>
<tr>
<td>Automatic BPMS: LexisNexis' order fulfillment system</td>
<td>Automated order fulfillment process configuration for small legal firms' information needs using Intalio's N3 BPMS</td>
<td>Formal modelling constructs defined in BPMN using Intalio's N3 Designer</td>
<td>Yes, BPMN</td>
<td>Yes</td>
<td>BPMN allows any abstraction area and abstraction depth because its modeling constructs can be used recursively to any depth</td>
<td>Automatically fulfills high variety high volume, low margin orders using an internet distribution channel</td>
</tr>
<tr>
<td>Electronic commerce: RosettaNet</td>
<td>Software application</td>
<td>Technical infrastructure</td>
<td>Software language</td>
<td>Software notation and software grammar</td>
<td>Software formalism</td>
<td></td>
</tr>
<tr>
<td>---------------------------------</td>
<td>----------------------</td>
<td>--------------------------</td>
<td>------------------</td>
<td>----------------------------------------</td>
<td>-------------------</td>
<td></td>
</tr>
<tr>
<td>ERP software workflow engine and configuration: SAP</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>Automatic BPMS: LexisNexis’ order fulfillment system</td>
<td>SAP</td>
<td>Any ERP hardware</td>
<td>Advanced business application programming (ABAP)</td>
<td>ABAP</td>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

Table II. BPMS core technology analysis: Part 2

Business process management system
even if the variety of possible requests needed to be increased the model’s variety
could also be increased using BPMN to fully model the increased variety. The model
abstraction and business process managed columns of Table I illustrate how the
abstraction area and depth of the model that “drives” the BPMS varies depending upon
which processes are modeled and why. Greater abstraction area leads to greater
solution scope and greater abstraction depth leads to greater solution automation.

In short these three examples use process models for different goals: RosettaNet is
set of malleable guidelines for B2B messaging and the processes that generate the
messages; SAP’s ERP suite uses the R/3 reference model for automated
document-focused decisions and for supporting manual configuration decisions; and
Intallio’s N3 automatically makes reconfiguration and process choice decisions.

Table II shows the startling contrast in software technologies required for
automated versus semi-automated and manual process management. Formality needs
to be preserved from the “ground up” in the entire bottom row except for the technical
infrastructure column. This is because the physical properties of the digital hardware
are loosely coupled (Weick, 1976) to the model characteristics that are modeled by
digital model constructs. This raises another BPMS attribute, the transmission of
model characteristics from enacting model to the digital processes that enact it.

At BPM ’03 van der Aalst et al. (2003a, p. 7) called for a formalization of the
relationship between business process management languages and their software
formalism. We believe that our BPMS pyramid architecture has begun to develop this
past the modeling language technology and through to a definition of a whole BPMS.
We have sought to explain what core technologies are required for a BPMS and how
they interrelate with the three basic infrastructures: the base reality of the business
process managed, the hardware infrastructure of the BPMS and the formalism of the
software that the BPMS application is written in. Our architecture describes varying
levels of automation from manual decision support to fully automatic process
reconfiguration “on the fly.”

**Conclusion**

We have defined a core technology as a medium for modeling and the set of core
technologies that are the building blocks for our BPMS architecture pyramid. We have
used the criticality of formal transmission and processing of modeling characteristics
between core technologies to suggest a theoretical basis for explaining why these core
technologies are required and why they interrelate so. Most of the literature does not
cover all the different core technology levels in the pyramid although Giaglis (2001,
p. 211) describes a much simpler hierarchical decomposition of “modeling elements”
each supported by the next in the hierarchy; the WFM coalition’s reference model has a
standards “stack” (Hollingsworth, 2004, p. 310); and van der Aalst et al. (2003a, p. 2)
describe four concentric IS layers. None of the literature describes and explains the full
set of levels and core technologies or their arrangement into a BPMS together with a
theoretical explanation based upon the transmission and processing of modeling
characteristics between core technologies.

Further research areas that present themselves include: the relationships between
modeled characteristics and modeling media properties; the practical benefits of BPMS
that require reduced manual supervision or can supervise and reconfigure other BPMS;
using BPMS in decision science and business intelligence systems; and using BPMS to
manage the main direct (e.g. customer relationship management, supply chain management) and indirect (e.g. human capital management) organizational and inter-organizational process classes in a value chain.

References
BPMJ
13,1

106


Further reading


About the authors
Duncan R. Shaw is Lecturer in Information Systems at Nottingham University Business School. After a career in management consultancy, he worked as a Manager for Motorola and then started a research career. His research and consultancy interests include the information systems and strategic business aspects of business-to-business interaction; network cultivation, orchestration and coordination; and the management of complex business environments. Duncan R. Shaw is the corresponding author and can be contacted at: duncan.shaw@nottingham.ac.uk

Christopher P. Holland is Professor of Information Systems at Manchester Business School. His research interests include information technology strategy, legacy information systems, electronic commerce, ERP systems, systems integration, inter-organizational systems and EDI, banking and implementation. He has consulted with a wide range of organizations in manufacturing, banking, information technology and retailing.

Peter Kawalek is Senior Lecturer in Information Systems at Manchester Business School. He has undertaken research and consultancy engagements across a number of sectors including insurance, telecommunications, manufacturing, rail and software development. Over a number of years he has developed a specialist interest in the public sector, working internationally with government agencies involved in information systems projects.

Bob Snowdon is Research Fellow at the Department of Computer Science in The University of Manchester. He worked for ICL and then the Fujitsu group and his interests centre around process modelling and the application of Beer’s Viable Systems Model.

Brian Warboyes is Professor of Software Engineering at The University of Manchester. His interests include the problems of very large systems design with specific emphasis on the development of techniques which enable the dynamic evolution of such systems.

To purchase reprints of this article please e-mail: reprints@emeraldinsight.com
Or visit our web site for further details: www.emeraldinsight.com/reprints