Formalisms for IT Management Process Representation

Vitalian A. Danciu
Munich Network Management Team
Ludwig-Maximilians-University
Oettingenstr. 67
D-80538 Munich, Germany
danciu@mnm-team.org

Abstract— **With recent years' accelerated convergence to process oriented service management frameworks, IT management is adopting business methods. As organisations model and document their management processes, they apply suggestions and best practices found in document collections like the eTOM or the ITIL. Formal representation of the emerging management process definitions can be accomplished by means of process or workflow definition formalisms. However, many IT management processes may differ from other business processes in that they are executed by technical personnel. More important, an IT infrastructure as the target of management processes offers compelling opportunities for automation at an operational level.**

To help leverage these opportunities, formalisms for process representation need to express IT management process details at a technical level. This paper analyses formalisms designed for business process representation, assesses their suitability to express IT management process definitions, and categorises the examined formalisms according to IT management requirements.

I. Introduction

The importance of process oriented management increases steadily. A major driving force behind this increase is the focus on IT Service Management (ITSM). ITSM itself is strongly motivated by the requirements of cost control and customer orientation, and catalysed by the availability of ITSM process frameworks. A growing number of organisations define their IT management processes relying on frameworks like the IT Infrastructure Library or the Enhanced Telecom Operations Map. This ongoing transition to process-centric thinking and adoption of best practices introduces the need to manage the processes themselves, while offering opportunities for process automation at different levels of process detail.

For a long time, IT divisions have been largely exempted from the accountability imposed on other organisation parts (e.g. enterprise production divisions). In order to reduce costs, a certain level of cost and benefit controlling is being demanded from IT centres in most companies. This demand is being addressed by introducing a service view and adopting documented management processes based on best practices collections and process frameworks.

For the time being, many organisations still focus on modelling and documenting their processes. To collect on the promise given by process oriented management, process execution needs to be automated wherever possible. Depending on the process template (i.e. the collection of best practices provided for a process) and its adaptation to organisation specific needs, more or less parts of a process (i.e. subprocesses) are suitable for automation. In general, a coarse distinction between three kinds of subprocesses can be made:

a) Automated subprocesses: that can be executed without human interaction. For ITIL, candidates include parts of the service support processes, e.g. parts of incident or configuration management process specifying direct enactment of management operations on the infrastructure.

b) Manual subprocesses: that are executed by hand. Manual processes may of course be supported by work-flow systems; the decisive characteristic is the need of human decision making during the process. Any subprocess dependent on human interaction, e.g. a Change Advisory Board (CAB) meeting or provider-customer negotiations, falls into this category.

c) Hybrid subprocesses: that can be described as automated processes containing distinct parts that need human interaction.

This distinction becomes important when considering the overall management tasks performed within an organisation. Available process frameworks concern themselves with management from a service perspective in order to achieve a tighter business alignment. Though not thoroughly investigated, it seems plausible that other management disciplines (e.g. system management) could profit from a process oriented approach.

To make the transition from modelling and documentation of processes to their assisted/automated execution, the documented processes need to be expressed in a suitable machine readable formalism. Subprocesses of different kinds (with respect to the distinction above) can be provided automation support by different kinds of tools: manual subprocesses can be assisted by work-flow systems, automated subprocesses can be executed on – more or less specialised – Operations Support Systems (OSS) tools, while hybrid processes require a cooperation between such tools.

Existing languages for process formalisation originate in the domain of business processes. A cursory review of these formalisms suggests that they focus on the assistance of manual processes or highly encapsulated (i.e. business logic hidden behind a high-level API) automatable process parts.

The objective of this paper is to evaluate a selection of process formalisms regarding their suitability to encode operational IT management processes. The evaluation encompasses analysis of the expressiveness regarding the execution of actions/functions, the definition of data types and the support for a priori defined objects and roles as well as the support for data path modelling provided by each formalism.

The formalisms themselves are discussed in Section II, including their origin and interrelations. The evaluation criteria derived from general requirements posed by operational IT management are explained in Section III and applied to each language in Section IV. The analysis results are summarised in Section V along with conclusions regarding the use of the formalisms examined in the context of IT management processes.

II. Background and related work

Formalisms suited or intended for process representation have been devised from different perspectives. This paper discusses a selection of languages that differ in scope, degree of technology binding and, as Section IV will show, general applicability. This section will introduce these formalisms in short and show the relationships between them, as well as the relationships between the standardisation bodies maintaining them. In addition, examination of these and similar languages under different aspects are discussed.

A. An overview of process languages

In the following the formalisms treated in this paper are introduced and related standards and formalism–specific background work is discussed.

a) Business Process Execution Language: One of the languages currently en vogue is the Business Process Execution Language for Web Services (BPEL4WS, or in short: BPEL)[8]. An XML-based orchestration language, BPEL is native to the Web Services (WS) domain and, for practical purposes, constrained to it. It is used for specifying invocations of APIs declared in the Web Service Definition Language (WSDL)[11]. The actual API invocations are typically executed by means of the Simple Object Access Protocol (SOAP). BPEL is well known in the main-stream due to the popularity of web-based services, and its use can be expected to spread further.

b) XML Process Definition Language: A language less known in the main-stream is the XML Process Definition Language (XPDL, [7]) released by the Workflow Management Coalition (WfMC). XPDL is stated to be a textual equivalent of the Business Process Modeling Notation (BPMN, [20]), a graphical process modelling formalism released by BPMI.org/OMG (see Section II-B). BPMN provides a base for the development of graphical process modelling and visualisation facilities. Both languages make few assumptions as to process implementation technology. The XPDL specification does, however offer a BPEL4WS binding in addition to the generic facilities provided.

c) Business Process Modeling Language: Three years ago, BPMI.org released the Business Process Modeling Language (BPML)[9], a textual, XML-based language for process representation. While still supported by process modelling tools, BPML's territory is being taken over by BPEL.

d) Business Process Specification Schema: A cooperation between the United Nations Centre for Trade Facilitation and Electronic Business (UN/CEFACT) and the Organization for the Advancement of Structured Information Standards (OASIS) under the label of Electronic Business using eXtensible Markup Language (ebXML) develops the Business Process Specification Schema (BPSS)[1]. The BPSS includes an XML Schema definition intended to facilitate automation centred on business document exchange.

e) Unified Modeling Language: Although it is designed to be a general-purpose, (mostly) graphical language – with an accent on software engineering needs – the Unified Modeling Language (UML) can be employed to model (business) processes.

The obvious choice UML diagram kind for process representation is the *activity diagram*. An evaluation of activity diagrams for work-flow modelling can be found in [14], where control flow aspects of activity diagrams are examined.

Several approaches to UML-based business process representation have been presented, some of them relying on former UML versions' extension mechanisms to provide language elements adapted to business process requirements (e.g. [15]). The current version 2.0 of the specification acknowledges the need for process modelling by incorporating stereotypes supporting process representation, as well as extensions in the context of activity diagrams. Examples include a better parameter mechanism and input/output *ports* for associating activities and the data objects necessary for their execution.

UML 2.0 is specified in four documents[5], [4], [3], [2] that cover different aspects of the language. For the purposes of this paper, the Superstructure specification [5] is the most relevant. UML has a textual correspondent in the XML Metadata Interchange (XMI) [6], which is a supporting standard for the Meta Object Facility (MOF)[3].

f) Petri nets: A well–researched alternative for specifying processes are petri nets. A small, flexible mechanism with a strong formal foundation, petri nets have been employed successfully in work-flow engines.

By annotating petri net nodes, the requirements of IT management process formalisation may be met provided a machine readable form suitable for automation support is available. When clearly specified/standardised, such annotations would define a new formalism for process representation based on petri nets. Its definition would include a specification of syntax as well as a semantics description for (at least) a fixed set of

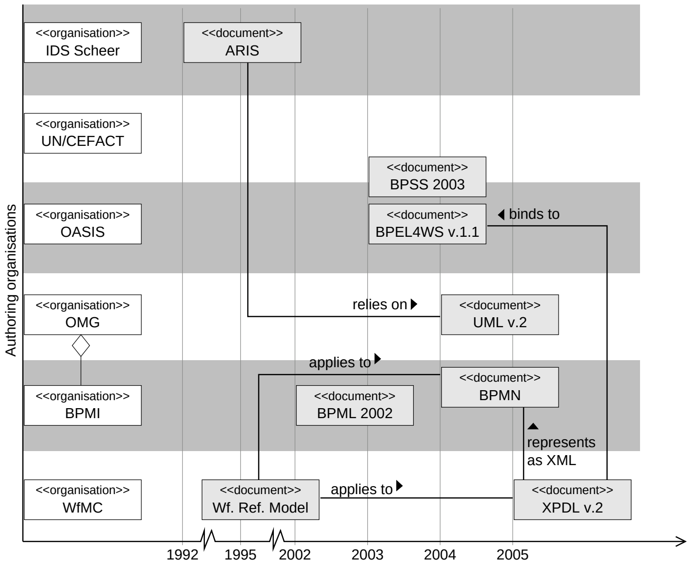

Fig. 1. Relationships between standardisation bodies and specifications

expressions to be used in annotations.

Analysis of such a language does make sense in the scope of this paper; a representative formalism based on petri nets, event-driven process chains, is described in the following and assessed in Section IV. Serving only as the underlying formalism, the petri net per se is exempted from analysis in this paper.

g) Event-driven Process Chains: Another established language for process representation is found with ARIS [16], specifically its Event-driven Process Chains (EPC). Petri nets by structure, EPCs are directed graphs containing activities, events as well as control flow elements and exploit the idea of event-driven process execution.

ARIS, really an architecture for process oriented management, includes a specification of a graphical representation of EPCs. A more concise description of language concepts and notation can be found in [10]. Some parts of ARIS, some of them unrelated to process modelling, rely on UML to represent technical information related to classes/objects.

Unlike the other languages presented in this paper, ARIS does not originate from a standardisation body, but from IDS Scheer, a company with close ties to academic research in the field of business processes and automation.

B. Interrelations of process formalisms

Despite their differences in scope and target audience, the process formalisms discussed do relate to each other. The specifications created by a standardisation body often reflect the interest of associated (industry) stakeholders. Thus, interrelationships between formalisms, as well as the relations between the standardisation bodies, may allow conclusions regarding the direction of future standardisation efforts.

The documents reviewed in this paper include work of the WfMC, OASIS, UN/CEFACT, OMG, BPMI and IDS Scheer. Figure 1 depicts the release of documents originating from these different organisations over time. The horizontal swimlanes hold documents released by the organisation shown on the left aligned along a time scale. The relations between organisations and documents are marked in the diagram and suggest a certain degree of convergence. On the level of organisations this has manifested in cooperations and mergers, while a higher alignment of the standardisation documents can be seen in recent releases. While only the merger between BPMI and OMG constitutes a strong binding between two of the standardisation organisations, several other cooperations exist (not shown in the figure) that are manifested in common web presences of the cooperation partners.

The relations between documents are also manifold. WfMC's Workflow Reference Model has provided a common understanding of work-flows and has been taken into account by the current versions of BPMN and XPDL, as well as by former versions. XPDL and BPMN are strongly related in that XPDL aims to provide a textual, machine readable format for BPMN's graphical representation of processes. BPMN, on the other hand, provides an explicit binding to BPEL4WS by defining a mapping of elements and constructs into BPEL4WS. The ARIS model family relies on UML (since former versions of the UML) in parts, though the event-driven process chains (EPCs) treated in this paper are based on petri net concepts.

The overall picture seems to suggest a convergence of organisations as well as of standardisation efforts.

C. Pattern-based analysis of process language

Surveys and analysis of process formalisms have been performed extensively before, albeit from a different view point. Notably, pattern based analysis of work-flow formalisms and products has been pursued for some time at Eindhoven University of Technology and has resulted in a pattern catalogue [18]. The catalogue is applied to different formalisms to compare their expressiveness from a work-flow perspective.

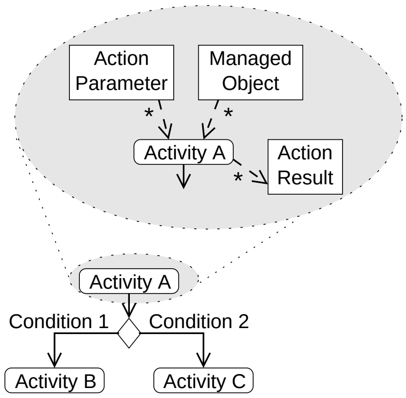

Fig. 2. Pattern example: Exclusive choice pattern.

The patterns employed in this approach are abstractions of recurring control flow in work-flows/processes. They concern themselves with the coordination of generic actions, excluding the technical details of actions from view. The pattern collection has been effectively employed for testing the support for a specific kind of process structure in formalisms and architectures/products.

IT management process modelling and implementation must take into consideration language aspects that support the mapping of management process specification onto technical IT management. Patterns can help by supplying the outer, structural framework for the information associated with an action. Figure 2 shows (in its lower part) the "Exclusive Choice" pattern as an UML activity diagram congruent to the one depicted in [19]. By design, it does not take into account the inner form of its elements (in this case, actions and conditions).

This paper, in contrast, revolves around requirements originating in technical IT management and therefore focuses on the details associated with a formalism's elements rather than on the overall structures allowed by a formalism. The upper part of Figure 2 indicates (organised around `Activity A`) some of the features sought-after in a formalism for technical IT management processes.

III. Requirements of IT Management processes

The process formalisms discussed in this paper were not designed with management processes in mind. They are intended for use in any business process scenario. Although management processes are a type of business processes themselves, they can be defined based on specific assumptions as to the execution environment and the involved personnel. At least in part, the processes introduced into IT management are tightly adherent to the infrastructure providing IT services. Manual subprocesses are executed by persons with knowledge about the process activities (e.g. activities pertaining to service support) that at the same time are (typically) knowledgeable about the function of the OSS tools employed to execute the activities. In consequence, the knowledge "distance" between the process activity (e.g. registering a new configuration item) and the software supporting it (database, application server etc) is relatively small. In contrast, knowledge about a business process (e.g. in automated manufacturing or content management) does not typically imply knowledge about the tools utilised to support it.

This difference can be exploited to achieve a higher degree of automation and integration through adaptation of OSS tools to execute process parts (semi-)automatically by directly interacting with the IT infrastructure. For this purpose, automated and hybrid subprocesses (as described in Section I) require access to object databases, coupling with monitoring tools as well as integration into facilities for enactment of administrative measures.

In concrete terms, the process automation facility must have access to definitions of entities (persons, components/systems, accounts etc), roles and domains; it must be coupled to the monitoring facilities available, in order to be able to execute process parts in response to events; and, it must be provided a means to influence the managed objects by executing management operations.

To integrate such technical aspects in the process specification, a certain degree of support in the formalism employed for the process definition is required.

The requirements described in the remainder of this section provide a basis for comparable examinations of the process formalisms.

A. Actions

Due tasks in work-flow or process definitions are often called *activities*. On an operational, technical management level, we refer to due tasks as management *operations* or *actions*. The following requirements refer to actions in the latter sense, i.e. more like function calls against an API than human-executed procedures. Requirements to action specification are detailed in the following.

a) Unique name or identifier: To be able to map an action from a process specification to a feature of a process support tool, the process formalism employed must allow specification of unique action identifiers.

b) Input or formal parameters: Many examples of (graphically) modelled processes show only the due actions; the input data for the action is neglected or implied from the context. Machine execution of an action requires explicit specification of input data to an action.

c) Output or return values: The output of an action, be it a value, a document or a status code signifying success or failure, is often needed as input of another process part. As with action input, explicit support for modelling output data is required.

d) Control flow: Though not an intrinsic of actions themselves, language constructs for control flow determine how actions are executed. Common constructs include those for parallel execution (forking and joining execution threads), and conditional branching.

e) Error handling: Actions may fail during execution and/or cause further faults in the system executing them. The minimum requirement for error handling is error detection, which suffices to ensure that a process is executed correctly, or not at all. However, this is a quite spartan mode of error handling. Extended requirements appropriate for realistic process support include:

- Notification in case of error. This could mean the notification of an operator if the system is incapable of handling the error condition. Such notification is a criterion applied when more sophisticated error handling (as described below) is missing.
- Error handlers. As with most programming languages, some process formalisms include the concept of error handlers, that are invoked when the normal control flow

is interrupted by a fault. The characteristics of such error handlers are dependent on the way the actions themselves are specified. Hence, for the purposes of comparing process formalisms, the mere existence of a error handling concept remains the only criterion.
- Recovery from error handling. Upon successful error handling, it may make sense for the process to continue at the point of interruption, thus requiring a mechanism for storing and retrieving that position in the process specification. As with error handlers, the criterion in our scope is the mere existence of a recovery mechanism.

B. Events and messages

Management processes are often invoked (or resumed) as a result of events or messages originating from a change of state in a system or from operator input to the process. In addition, such events may transport information to the process. In the same way, messages can be sent from within a process to convey information to human or machine recipients. Some of the formalisms examined rely on message exchange for the greater part of their control flow or for their information exchange. The analysis criteria with regard to messaging support are described in the following.

a) Ability to expect messages: To be able to react to messages, facilities to specify expectance of a message need to be included in a formalism. Optionally, the type of the message may be included in the specification, as detailed further on in this section.

b) Ability to send messages: Apart from notifications in case of errors, processes may need to send messages about their progress, successfully performed important action etc. The specification for the sending of a message may include more or less detail, such as message type and optionally enclosed information (see below).

c) Information transport in messages: As with low-level mechanisms relying on asynchronous messages (e.g. SNMP traps), it does make sense to allow piggy-backing of process internal information in messages.

d) Typing and naming of messages: To allow automated reaction to messages, the format of messages must be known or accessible to the tool supporting the process parts dealing with a certain kind of message. Obviously, the format must also be advertised to the recipient. Well-known means of achieving this is unique naming of message types associated with a definition of their content.

C. Conditional expressions

Requirements for conditional expressions do not originate solely from IT management process needs. In more technical management processes, however, a powerful conditional expression mechanism can be leveraged to make control flow decisions based on system state or data, thus promoting automation endeavours. For this reason, the formalisms in this paper are examined regarding the expressiveness of their conditional expressions, as well as regarding the language elements where these may be included.

a) Values: Values can be constants that are part of the process definition, they can be held in attributes of the process or of the runtime system (e.g. process support tools) or they can originate from actions (as return values) or arithmetic expressions. Requirements regarding values are found in Section III-E.

b) Operators: Conditional expressions require operations that can be relational ($<$, $\leq$, $=$, $\geq$, $>$) to compare values, as well as logical operators to create complex conditions, including *and*, *not*, *or* etc.

c) Arithmetic expressions: Common cases of conditional expressions include values obtained by the evaluation of arithmetic expressions in the process definition. Hence, the formalisms are examined for the support of basic arithmetic operators (add, subtract etc.) and their applicability in different contexts.

D. Reference to managed objects

IT management relies more and more on system and service models based on object oriented modelling frameworks. Examples of such frameworks include DMTF's Common Information Model (CIM, [13]), Telemanagement Forum's Shared Information/Data Model (SID, [17]) or models based directly on the UML.

The modelled infrastructure and the provided services obviously have high relevance to the management processes defined. Therefore, a formalism used for IT management process definition should provide a way to reference these a priori defined object collections. At the very least, a concept of objects in the formalism could be employed to reference tailored "copies" of the (mostly object oriented) management information in the models.

Management objects may relate to several of the expression classes mentioned in this section. For instance, they may be referenced in actions, either as parameters or as targets of an action; Their attributes may be part of conditional expressions; and they may be referenced when creating messages or events. Therefore, the formalisms examined in this paper are checked for the following characteristics:

- Existence of the concept of process-external objects.
- References to objects in action input or output.
- References to objects in conditional expressions.
- Direct references to externally defined objects.

E. Data types, variables and values

Business work-flows have a weaker relation to typed data than is common in IT-centric environments. Instead, data is encapsulated in forms or "documents" and often transported in the form of character strings. Independently of its encoding, IT management processes interacting with IT infrastructure could take advantage of clear definitions of data types.

Beside facilitating the design of process-supporting applications, a set of data types is indispensable if automation of process parts is desired. A basic set of data types includes general purpose types for the representation of numeric and textual data as well as special types such as time and date representation. To summarise, process formalisms are examined as to the support for representation of:

- Integer and floating point numbers
- Character strings
- Date/Time expressions
- Boolean values

IV. Analysis of process formalisms

In this section, the scheme described in the previous section (IV-A) is applied to the formalisms mentioned in Section II-A. A description of the language accompanies the analysis in an attempt to capture the intent of the formalism's authors with respect to its mode of use.

A. Assessment scheme for process formalisms

In essence, the analysis is performed by means of a list of criteria that are applied to every formalism analysed. The list is a summary of Sections III-A through III-E capturing the most important formalism features to check for. The leftmost three columns of Table I show the examination criteria at a glance.The columns captioned with a formalism name contain the grade of compliance of that formalism to a criterion. A "$\surd$" denotes the criterion to be satisfactorily fulfilled. When set in parenthesis "($\surd$)" it denotes partial fulfilment, while a "$\times$" marks a failed criterion.

Examination of a formalism is not about searching for formalism elements with certain names but identifying features of a language that can perform a certain desired function. Therefore, in the following discussion of the examined formalisms, alternatives to the direct specification of language elements (i.e. opportunities for constructing missing functions from elements present in the formalism) have been taken into account.

B. BPEL4WS

As its name suggests, BPEL is intended as a formalism for *executable* business processes. In consequence, it addresses those requirements aiming at enabling automation. IT management specific bindings, however, are less prominently developed.

a) Actions: BPEL defines actions as invocations of remote API functions, typically in a business partner's domain. The target domain to be invoked is identified by a `partnerLink` and a `portType` expression. The action to be invoked is identified by its symbolic name. It can receive literal formal parameters and be assigned a variable name to be used as a container for the return value. References to externally defined objects (e.g. MOs) are not supported.

Control flow features include parallelisation of actions and conditional branching by means of `switch` statement. Fault signalling is supported by an exception mechanism and facility for error handling is provided through `compensationHandlers`. The latter consist of alternative actions to be executed when the primary action fails.

b) Messaging: BPEL specifies event handlers capable of invoking an operation or instantiating a process in reaction to receiving an event. Conversely, event handlers can be invoked (remotely) to transmit an event. Format guidelines for information transport in events are not provided.

c) Support for objects: Externally defined objects are not easily made available to a BPEL process. Similarly, actions are associated with scalar variables for parameters and result.

d) Conditional expressions: Conditions are specified for special language elements, such as `switch` constructs or execution thread joining. They can be temporal conditions or expressions formulated in XPath. Relational and arithmetic operators are supported.

e) Data types: BPEL relies on common XML data types for user data providing integer, string and date/time types.

C. XPDL/BPMN

This specification combo consisting of a graphical notation and its mapping to XML target process authors from the business domain. For instance, the BPMN defines graphical elements designed to be easily comprehensible by non-technical personnel, e.g. by use of icons representing messages or stereotype actions. Nevertheless, many of the requirements formulated for operational IT management processes are addressed by their specifications.

Because of the equivalence in expression of these languages BPMN is described representatively for both formalisms (with the exception of formalism features where BPMN and XPDL differ). Its specification includes normative text regarding the graphical representation of language elements and the behaviour of compliant modelling tools.

a) Actions: BPMN specifies atomic or compound actions ("activities") that are associated with zero or more `InputSets` providing data to the action as well as zero or more `OutputSets` that represent the action result. Input and output are defined as `Artifacts` that can be of a `DataObject` type and encapsulate documents or parameters made available to the actions.

Control flow can be modelled by means of *gateways* that specify flow forks and joins as well as conditional and event-based branching. BPML specifies different control flow variants, including the `Exception Flow`, that is triggered by an event (i.e. error notification) and constitutes an alternative control flow path that can be merged into the `Normal Flow` by means of the general joining mechanisms provided by the formalism.

b) Messaging: The atomic actions ("tasks") are typed and are optionally associated with incoming and outgoing messages. Special task types denote acceptance and transmission of messages as alternatives to action execution. Events can contain messages that are named and may contain a set of `Properties` consisting of named strings.

Domain	*Element*	*Criterion*	*BPEL4WS*	*BPSS*	*XPDL*	*UML*	*EPC*
Actions and control flow	name/ID	existence of	√	(√)	√	√	×
	formal parameters	named	(√)	×	√	√	(√)
		typed	×	×	√	√	×
	return value	existence of	√	(√)	√	√	(√)
		complex value	(√)	(√)	√	√	(√)
	reference to MOs	support for	×	×	×	√	(√)
		as target	×	×	×	√	(√)
	parallel execution	support for	√	√	√	√	√
	conditional branching	support for	√	√	√	√	√
	error notification	support for	√	√	(√)	√	×
	error handler	existence of	√	×	√	√	×
	recovery from error	support for reentry	×	×	√	×	×
Conditional expressions	operators	relational	√	×	√	√	(√)
		logical	√	×	√	√	(√)
		arithmetic	√	×	√	√	×
Events and messaging	message	expect	√	×	√	√	×
		send	√	×	√	√	×
	typing	name/ID	×	×	(√)	√	×
		format definition	×	×	×	(√)	×
	information transport	support for	√	×	√	√	×
Types	integer numbers	in process definition	√	×	√	(√)	×
	floating point numbers	in process definition	×	×	√	(√)	×
	character strings	in process definition	√	×	√	(√)	×
	boolean type	in process definition	×	×	√	(√)	×
	date/time expressions	in process definition	√	×	√	(√)	×
Objects	object concept	existence of	×	×	√	√	(√)
	reference to objects	in action input	×	×	(√)	√	(√)
		in action output	×	×	(√)	√	(√)
		in conditional expressions	×	×	(√)	√	×
	externally def. objects	direct ref. to	×	×	√	(√)	×

TABLE I

ASSESSMENT CRITERIA AND EVALUATION RESULTS

c) Support for objects: Objects can be referred to via the `Participant` class that encapsulates an entity or role expression identified by a name. A mapping to externally defined objects is not supported explicitely in the BPMN specification, though a facility to reference such objects is present in XPDL.

d) Conditional expressions: Conditions can make use of relational and logical operators. They can be associated with flow control elements (gateways).

e) Data types: XPDL defines basic data types including all data types required as per Section III, and additional data types for representation of externally defined entities as well as *participants* of the process.

Complex data types include records, arrays, unions enumerations and lists.

D. ebXML/BPSS

The Business Process Specification Schema is a quite high-level business process description language. It has a focus on business document exchange and can be described as a document flow language more than a work-flow language.

a) Actions and control flow: There are several definitions of activities/actions in BPSS, however they have quite different semantics compared to the concept of actions as described in Section III-A. For instance, a `BusinessAction` is a named coarse description of an interaction with a business partner. Its instances are associated with `DocumentEnvelopes`.The BPSS describes a state–machine–like concept (capturing business state) that can be employed for modelling business interactions.

b) Messaging: Processes can be described as message driven but not as event driven in the IT management sense: messages always originate with business roles (i.e. machine or human actors from different domains) and constitute an exchange of business documents. While event mechanisms employed in IT management (e.g. low-level traps, notifications from management tools) are similar in structure, the messaging concept of BPSS has more in common with documents exchanged by email. A BPSS `BusinessDocument` is a named entity wrapped in one or more `DocumentEnvelopes`, which contains state information to determine whether the document included is intended as a request or a response. In the latter case, a simple boolean flags a positive/negative response. Additionally, an envelope containing a document may transport one or more attachments related to the document.

c) Support for objects, roles and domains: The two roles of *requester* and *responder* are the only roles defined in BPSS. There is no apparent mechanism to refine or complement these roles. Domain expressions are not supported and while objects may be described inside documents, no reference can be made to predefined objects as described in Section III-D. Document format or structure is not given in the specification.

d) Conditional expressions: Collaboration between two roles is governed by pre- and post-conditions. Conditions can be also be imposed on the association of a `BusinessDocument` to a `DocumentEnvelope` in order to determine if the envelope is suitable for the document it wraps. However, there is no formal specification of condition features or syntax.

e) Data types: Data types for use in process modelling are not defined. Internally, the language relies on the common XML data types to represent information.

E. UML

The UML has been used for process definition before even though it does not provide specialised means for that purpose. It does, however, provide generic means to express most of the elements noted in Section III. In this case, generic means are at the same time a blessing and a curse: while conveying the ability to express all required elements, they make necessary the introduction of conventions regarding accepted ways of expressing elements. This analysis focuses on Activity Diagrams. Process representation options available by use of other UML features and the use of extension mechanisms have not been taken into account.

a) Actions: The requirements formulated for management actions are addressed by a group of activity diagram classes. `Activities` model the execution of primitive functions as well as invocation behaviour, transmission of signals and the accessing of object attributes. The requirements regarding functional parameters of actions and return values are satisfied by the `ParameterSet/Parameter` classes that allow the association of input/output objects to actions. Parallel execution of actions as well as conditional branching is provided by means of the well–known `DecisionNode` (diamond) and `ForkNode` (parallelisation/synchronisation bar) elements. Error handling is modelled explicitely by use of the `ExceptionHandler` class.

b) Messaging: Activity diagrams allow the modelling of transmitting and expecting typed signals (events) using specialised actions (`SendSignalAction`, `AcceptEventAction`. Similarly, event payload relaying can be expressed by means of `SendObjectAction`. A means for format definition is not provided explicitely; the type of the objects transmitted can be used to map to the payload format.

c) Support for objects: The UML's support for object definitions in the context of actions and events is quite good. References to external object definitions may be facilitated by the fact that such definitions (e.g. deposited in a CIMOM) are mostly based on UML[1]

d) Conditional expressions: The preferred way to express conditions in the UML is the Object Constraint Language (OCL). OCL statements can be associated with e.g. `DecisionNodes` in activity diagrams to implement conditional branching. Support for relational, logical as well as arithmetic operations is included.

e) Data types: Support for complex data types (classes) is inherently good, while primitive types as those listed in III-E can be expressed as attributes of classes/objects. A precise primitive type definition (e.g. including value ranges of number types), however, is not provided.

F. Event-driven Process Chains

EPCs describe process partitions (chains) incorporating actions, conditions, objects and logical connectors for control flow. The language elements are intended for high-level, human-readable representation of processes. To achieve a consistent machine-readable form, a set of conventions regarding the inner structure of the elements addressed would need to be created.

It should be noted that the event-driven process chain formalism is embedded within the ARIS family of modelling techniques that relies among others on entity-relationship modelling for data and UML for class/object modelling. In consequence, some aspects of process modelling are addressed by other formalisms than EPCs. An extension of EPCs termed *enhanced event-driven process chains* allows reference to entities relevant to the process (e.g. organisational entities).

a) Actions: Actions in EPCs are textual descriptions of tasks and can be associated with objects, thus allowing the modelling of action input and output. However, support for a formal representation is not provided. Parallelisation and joining of process threads as well as conditional branching is supported by means of logical operators (OR, XOR, AND).

b) Messaging vs. EPC events: Events in EPCs serve to invoke a chain (i.e. instantiate a process) as well as to determine a change of process state (e.g. an altered attribute value) or a certain point in time. Although EPCs rely heavily on events, the event concept used is not intended for transmitting messages within or outside processes. EPC events have a declarative nature and their semantics overlap with those of conditions. In other words, an event can consist of a statement (e.g. "data available") that triggers a process partition if deemed to be true. Readers familiar with petri nets will recognise the origin of the EPC event/condition concept.

c) Support for objects: EPC actions can be associated with objects and organisational units. However, a formal representation of these entities is not supported directly.

d) Conditional expressions: Conditions and events are described using the same graphical notation. They are not differentiated between explicitely.

e) Data types: Data types adhering to the requirements described in this paper are not defined.

V. CONCLUSIONS

This paper has presented requirements for the representation of technical IT management processes as well as an assessment

[1]Note that although CIM uses a somewhat different meta-model, it is compatible with UML when only access to single object definition is regarded.

of process formalism based on those requirements. Although the languages analysed pursue similar goals, the findings show striking differences in their suitability for the representation of technical process details.

Good candidates include the UML, XPDL/BPMN and, to some extent, BPEL4WS. The latter is constrained by its focus on Web Services, but that constraint is increasingly offset by the popularity of web based frameworks and by the proliferation of BPEL itself.

XPDL and BPMN were designed with automation in mind. In consequence, they fulfil the automation requirements shared by common business and technical processes. The growing support for these formalism in business process modelling tools further motivates them as a choice for IT management process modelling.

The UML exhibits very good characteristics with regard to the requirements formulated. Since it is an established formalism, modelling tool support is also abundant.

BPSS is suitable for modelling only the most high-level, business-like IT management processes. It does not fulfil the requirements described in Section III. It is clear that it is intended to deliver some support for business document exchange. It makes few assumptions regarding its target business domain and therefore it cannot provide support for detailed process representation.

ARIS's event-driven process chains likewise focus on flow control and include summary support for operational IT management needs.

To summarise, all languages provide suitable control flow features but generally lack expressiveness regarding the formalisation of actions. UML seems the best choice for the operational processes targeted in this paper (automatic and hybrid subprocesses). However, the modelling of business processes outside the scope of IT management may force the selection of a formalism more accessible to business people than the comparatively formal UML. In such cases, when a common language is needed for use with both core business processes and IT management processes, BPMN/XPDL may be the language of choice.

A. Further work

The work having been presented here covers a narrow aspect of IT management process formalisation.

The criteria used have been treated equally, even though the formalism features examined may have different degrees of importance to management processes. A differentiation with respect to the impact of formalism features and a complementation of the analysis using weights should yield more accurate verdicts on the usability of a given formalism for management process representation.

The criteria presented in Section III address important aspects of IT management processes with respect to automation, but do not constitute a comprehensive set. The selection of further criteria not yet addressed and the refinement of those used in this paper would help provide a more complete picture of the formalisms in question. While analysis of formalisms is a necessary first step, a requirements set aiming at amending suitable formalisms would be a future step on the way.

Current process frameworks are invariably designed from a service management vantage point. However, formalisation of work processes in other management disciplines, e.g. management of systems or networks, could also profit from a process-oriented approach. Most probably, the requirements given in this paper will apply to the needs of these disciplines. However, modelling processes for yet more technical management disciplines may pose additional challenges to the selection of suitable formalisms.

Finally, all processes – whether describing core business procedures or IT management – are meant to constitute a representation of goals. To ensure compliance with the original management intent, processes can be analysed and validated against these goals. Direct validation of process definitions is one option, perhaps with the aid of assertions included in the process definition. Another option is the analysis of operational policies derived from process definitions, as suggested in [12]. With this approach, reasoning techniques developed for management policies could be used for the analysis.

ACKNOWLEDGMENT

The author wishes to thank the members of the Munich Network Management (MNM) Team for helpful discussions and valuable comments on previous versions of this paper. The MNM Team directed by Prof. Dr. Heinz-Gerd Hegering is a group of researchers of the University of Munich, the Munich University of Technology, the Federal Armed Forces University Munich and the Leibniz Supercomputing Center of the Bavarian Academy of Sciences. Its web–server is located at http://www.mnm-team.org.

REFERENCES

[1] Business Process Specification Schema. Technical report, UN/CEFACT and OASIS, 2001.
[2] UML 2.0 Diagram Interchange Specification. Object Management Group Adopted Specification ptc/03-09-01, Object Management Group, September 2003.
[3] UML 2.0 Infrastructure Specification. Object Management Group Adopted Specification 03-09-15, Object Management Group, September 2003.
[4] UML 2.0 OCL Specification. Object Management Group Adopted Specification ptc/03-10-14, Object Management Group, October 2003.
[5] UML 2.0 Superstructure Specification. Object Management Group Adopted Specification ptc/03-08-02, Object Management Group, August 2003.
[6] MOF 2.0 / XMI Mapping Specification, v. 2.1. Object Management Group Formal Specification formal/05-09-01, Object Management Group, September 2005.
[7] Process Definition Interface – XML Process Definition Language. Workflow Management Coalition Workflow Standard WFMC-TC-1025, Workflow Management Coalition, October 2005. Version 2.0.
[8] Tony Andrews, Francisco Curbera, Hitesh Dholakia, Yaron Goland, Johannes Klein, Frank Leymann, Kevin Liu, Dieter Roller, Doug Smith, Satish Thatte, Ivana Trickovic, and Sanjiva Weerawarana. Business Process Execution Language for Web Services Version 1.1. Technical report, OASIS, May 2003.

[9] Assaf Arkin. Business process modeling language. Draft, BPMI, November 2002.
[10] J. Becker, M. Kugeler, and M. Rosemann, editors. *Process management – A guide for the design of business processes*. Springer-Verlag, 2003.
[11] Erik Christensen, Francisco Curbera, Greg Meredith, and Sanjiva Weerawarana. Web Services Description Language (WSDL) 1.1. W3C Recommendation, W3C, March 2001.
[12] V. Danciu and B. Kempter. From Processes to Policies – Concepts for Large Scale Policy Generation. In *Managing Next Generation Convergence Networks and Services: Proceedings of the 2004 IEEE/IFIP Network Operations and Management Symposium (NOMS)*, volume 2004, Seoul, Korea, April 2004. IEEE/IFIP.
[13] Distributed Management Task Force (DMTF). Common Information Model (CIM) Version 2.9. Specification, June 2005.
[14] M. Dumas and A. ter Hofstede. UML activity diagrams as a workflow specification language. In *Proceedings of the International Conference on the Unified Modeling Language (UML)*, Toronto, Canada, 2001. Springer-Verlag.
[15] Hans-Erik Eriksson and Magnus Penker. *Business Modeling with UML – Business Patterns at Work*. John Wiley & Sons, ISBN 0-471-29551-5, 2000.
[16] A.-W. Scheer. *ARIS – Business Process Modeling*. Springer, Berlin, 1999.
[17] TeleManagementForum. Shared Information/Data (SID) Model Concepts, Principles, and Domains. Technical Report GB 922 Member Evaluation Version 3.1, TeleManagement Forum, July 2003.
[18] W. M. P. van der Aalst, A. H. M. ter Hofstede, B. Kiepuszewski, and A. P. Barons. Workflow Patterns. 2002.
[19] Stephen A. White. Process Modeling Notations and Workflow Patterns. White paper.
[20] Stephen A. White. Business Process Modeling Notation (BPMN) Version 1.0. Technical report, Business Process Management Initiative, May 2004.

Printed in: Information Technology Management from a Business Perspective, p. 45-54, IEEE, 1st IEEE/IFIP International Workshop on Business Driven IT Management, Vancouver, Canada, April, 2006