Web Service Composition Languages: Old Wine in New Bottles?

Wil M.P. van der Aalst[*]
Department of Technology Management
Eindhoven University of Technology
P.O.Box 513, NL-5600 MB Eindhoven, The Netherlands
w.m.p.v.d.aalst@tm.tue.nl

Marlon Dumas   Arthur H.M. ter Hofstede
Centre for Information Technology Innovation
Queensland University of Technology
GPO Box 2434, Brisbane QLD 4001, Australia
{m.dumas, a.terhofstede}@qut.edu.au

Abstract

Recently, several languages for web service composition have emerged (e.g., BPEL4WS and WSCI). The goal of these languages is to glue web services together in a process-oriented way. For this purpose, these languages typically borrow concepts from workflow management systems and embed these concepts in the so-called "web services stack". Up to now, little or no effort has been dedicated to systematically evaluate the capabilities and limitations of these languages. BPEL4WS for example is said to combine the best of other standards for web service composition such as WSFL (IBM) and XLANG (Microsoft), and allows for a mixture of block structured and graph structured process models. However, aspects such as the expressiveness, adequacy, orthogonality, and formal characterization of BPEL4WS (e.g. reachability) have not yet been systematically investigated. Although BPEL4WS is not a bad proposal, it is remarkable how much attention it receives while more fundamental issues such as semantics, expressiveness, and adequacy do not get the attention they deserve. Therefore, we advocate the use of more rigorous approaches to critically evaluate the so-called standards for web services composition, and to learn from 25 years of experiences in the workflow/office automation domain.

1. Introduction

Web Services is an emerging paradigm for architecting and implementing business collaborations within and across organizational boundaries. In this paradigm, the functionality provided by business applications is encapsulated within web services: software components described at a semantical level, which can be invoked by application programs or by other services through a stack of Internet standards including HTTP, XML, SOAP, WSDL, and UDDI [8]. Once deployed, web services provided by various organizations can be inter-connected in order to implement business collaborations, leading to *composite web services*.

Business collaborations require long-running interactions driven by an explicit process model [1]. Accordingly, a current trend is to express the logic of a composite web service using a business process modeling language tailored for web services. Recently, many languages have emerged, including WSCI [5], BPML [6], BPEL4WS [9], XLANG [21], WSFL [16], and BPSS [17], with little effort spent on their evaluation with respect to a common benchmark. Such a comparative evaluation will contribute to establishing their overlap and complementarities, to delimit their capabilities and limitations, and to detect inconsistencies and ambiguities.

As a step in this direction, we have conducted an analysis of some of these languages [1, 2, 23], namely BPML (Business Process Modeling Language) and its "sibling" language WSCI (Web Services Choreography Interface) [2], and BPEL4WS (Business Process Execution Language for Web Services) and its parent languages XLANG and WSFL (Web Services Flow Language) [23]. The analysis is based on a framework composed of a set of *patterns*: abstracted forms of recurring situations found at various stages of software development. Specifically, the framework brings together a set of *workflow patterns* documented in [4, 3], and a set of *communication patterns* documented in [19].

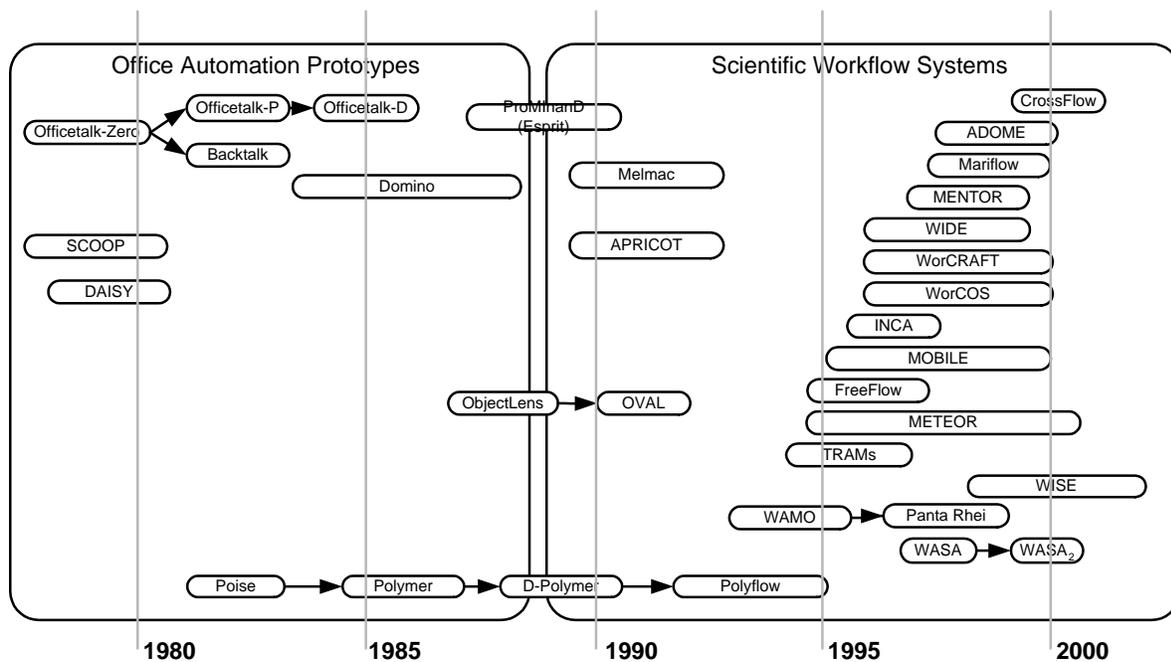[*]Also affiliated with the Queensland University of Technology.

Figure 1. Historic overview of early systems and research prototypes (Taken from [18]).

There have been other comparisons of some of the languages mentioned in this paper. These comparisons typically do not use a framework and provide an opinion rather than a structured analysis. An example is [20] where XPDL, BPML and BPEL4WS are compared by relating the concepts used in the three languages. Unfortunately, [20] leaves open many important questions.

In the remainder of the paper, we first review the history of workflow management and introduce the set of workflow patterns. Then, we discuss the topic of web service composition and compare BPEL4WS, BPML, XLANG, WSFL, and WSCI using the workflow patterns and some basic communication patterns.

2. Workflow management: A historical perspective

An interesting starting point from a scientific perspective is the early work on office information systems. In the seventies, Skip Ellis [10], Anatol Holt [13], and Michael Zisman [24] already worked on so-called office information systems, which were driven by explicit process models. It is interesting to see that the three pioneers in this area independently used Petri-net variants to model office procedures. During the seventies and eighties there was great optimism about the applicability of office information systems. Unfortunately, few applications succeeded. As a result of these experiences, both the application of this technology and re-

lated research almost stopped for a decade. Hardly any advances were made in the eighties. In the nineties, there was a renewed interest in these systems. The number of workflow management systems developed in the past decade and the many papers on workflow technology illustrate the revival of process-aware office information systems. Today workflow management systems are readily available. However, their application is still limited to specific industries such as banking and insurance. As indicated by Skip Ellis in [11] it is important to learn from these ups and downs. The failures in the eighties can be explained by both technical and conceptual problems. In the eighties, networks were slow or not present at all, there were no suitable graphical interfaces, and proper development software was missing. However, there were also more conceptual problems: there was no unified way of modeling processes and the systems were too rigid to be used by people in the workplace. Most of the technical problems have been resolved by now. However, the more conceptual problems remain. Good standards for business process modeling are still missing and even today's workflow management systems enforce unnecessary constraints on the process logic (e.g., processes are made more sequential that they need to be).

Figure 1 gives a historic overview of office automation and workflow prototypes [18]. Figure 2 provides a historic overview of commercial workflow management systems. These two figures show that: (i) workflow management is not something that started in the nineties but already
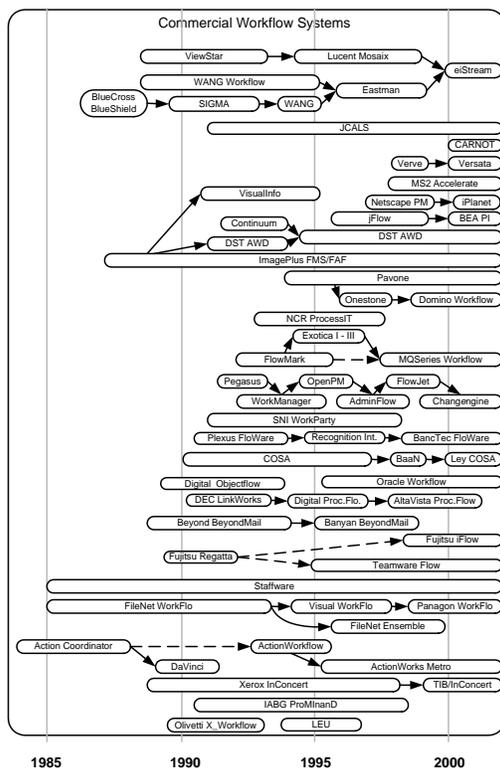
Commercial Workflow Systems

ViewStar — Lucent Mosaix — eiStream
WANG Workflow — Eastman
BlueCross BlueShield — SIGMA — WANG
JCALS
CARNOT
Verve — Versata
MS2 Accelerate
VisualInfo — Netscape PM — iPlanet
Continuum — jFlow — BEA PI
DST AWD — DST AWD
ImagePlus FMS/FAF
Pavone
Onestone — Domino Workflow
NCR ProcessIT
Exotica I - III
FlowMark — MQSeries Workflow
Pegasus — OpenPM — FlowJet
WorkManager — AdminFlow — Changengine
SNI WorkParty
Plexus FloWare — Recognition Int. — BancTec FloWare
COSA — BaaN — Ley COSA
Digital Objectflow — Oracle Workflow
DEC LinkWorks — Digital Proc.Flo. — AltaVista Proc.Flow.
Beyond BeyondMail — Banyan BeyondMail
Fujitsu Regatta — Fujitsu iFlow
Teamware Flow
Staffware
FileNet WorkFlo — Visual WorkFlo — Panagon WorkFlo
FileNet Ensemble
Action Coordinator — ActionWorkflow
DaVinci — ActionWorks Metro
Xerox InConcert — TIB/InConcert
IABG ProMInanD
Olivetti X_Workflow — LEU

1980   1985   1990   1995   2000

Figure 2. Historic overview of commercial workflow management systems (Taken from [18]).

in the seventies with the work of Ellis (*OfficeTalk*) and Zisman (*Scoop*); and (ii) the number of commercial systems has considerably grown in recent years. When considering web service composition languages it is important to take this into account and use experience and knowledge from the workflow domain, i.e., do not re-invent the wheel.

3. Workflow patterns

For a critical evaluation of web service composition languages, we use the set of workflow patterns described in [4, 3]. We have used these patterns to compare the functionality of 15 workflow management systems (COSA, Visual Workflow, Forté Conductor, Lotus Domino Workflow, Meteor, Mobile, MQSeries/Workflow, Staffware, Verve Workflow, I-Flow, InConcert, Changengine, SAP R/3 Workflow, Eastman, and FLOWer). The result of this evaluation reveals that (1) the expressive power of contemporary systems leaves much to be desired and (2) the systems support different patterns. Note that we do not use the term "expressiveness" in the traditional or formal sense. If one abstracts from capacity constraints, any workflow language is Turing complete. Therefore, it makes no sense to compare these languages using formal notions of expressiveness. Instead we use a more intuitive notion of expressiveness which takes the modeling effort into account. This more intuitive notion is often referred to as suitability. See [15] for a discussion on the distinction between formal expressiveness and suitability.

For a detailed description and discussion of the patterns we refer to [4, 3]. Just as illustration we describe Workflow Pattern # 16 (WP16: Deferred Choice).

WP16 Deferred Choice A point in a process where one among several alternative branches is chosen based on information which is not necessarily available when this point is reached. This differs from the normal exclusive choice, in that the choice is not made immediately when the point is reached, but instead several alternatives are offered, and the choice between them is delayed until the occurrence of some event. **Example:** When a contract is finalized, it has to be reviewed and signed either by the director or by the operations manager, whoever is available first. Both the director and the operations manager would be notified that the contract is to be reviewed: the first one who is available will proceed with the review.

Note that WP16 is different from the "normal choice" (WP 4: Exclusive Choice): The choice is not based on a decision or data but on a choice resolved by the environment.

Table 1 summarizes the results of the comparison of the workflow management systems in terms of the selected patterns. For each product-pattern combination, we checked whether it is possible to realize the workflow pattern with the tool. If a product directly supports the pattern through one of its constructs, it is rated +. If the pattern is not *directly* supported, it is rated +/-. Any solution that results in spaghetti diagrams or coding, is considered as giving no direct support and is rated -. These rating should be applied with care as indicated in [4].

Note that a pattern is only supported directly if there is a feature provided by the (graphical) interface of the tool (i.e., not in some scripting language) which supports the construct without resorting to any of the solutions mentioned in the implementation part of the pattern. For example, WP6 (Multi-choice) can be realized using a network of AND/XOR-splits. However, this does not mean that any workflow system supporting WP2 (Parallel Split) and WP4 (Exclusive Choice) directly supports WP6.

From the comparison it appears that no tool supports all the selected patterns. In fact, many of these tools only support a relatively small subset of the more advanced patterns (i.e., patterns 6 to 20). Specifically the limited support for the discriminator, the state-based patterns (only COSA), the synchronization of multiple instances (only FLOWer), and the cancellation of activities, is worth noting.

pattern	product														
	Staffware	COSA	InConcert	Eastman	FLOWer	Domino	Meteor	Mobile	MQSeries	Forté	Verve	Vis. WF	Changeng.	I-Flow	SAP/R3
1 (seq)	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
2 (par-spl)	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
3 (synch)	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
4 (ex-ch)	+	+	+/-	+	+	+	+	+	+	+	+	+	+	+	+
5 (simple-m)	+	+	+/-	+	+	+	+	+	+	+	+	+	+	+	+
6 (m-choice)	-	+	+/-	+/-	-	+	+	+	+	+	+	+	+	+	+
7 (sync-m)	-	+/-	+	+	-	+	-	-	+	-	-	-	-	-	-
8 (multi-m)	-	-	-	+	+/-	+/-	+	-	-	+	+	-	-	-	-
9 (disc)	-	-	-	+	+/-	-	+/-	+	-	+	+	-	+	-	+
10 (arb-c)	+	+	-	+	-	+	+	-	-	+	+	+/-	+	+	-
11 (impl-t)	+	-	+	+	-	+	-	-	+	-	-	-	-	-	-
12 (mi-no-s)	-	+/-	-	+	+	+/-	+	-	-	+	+	+	-	+	-
13 (mi-dt)	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
14 (mi-rt)	-	-	-	-	+	-	-	-	-	-	-	-	-	-	+/-
15 (mi-no)	-	-	-	-	+	-	-	-	-	-	-	-	-	-	-
16 (def-c)	-	+	-	-	+/-	-	-	-	-	-	-	-	-	-	-
17 (int-par)	-	+	-	-	+/-	-	-	+	-	-	-	-	-	-	-
18 (milest)	-	+	-	-	+/-	-	-	-	-	-	-	-	-	-	-
19 (can-a)	+	+	-	-	+/-	-	-	-	-	-	-	-	-	-	+
20 (can-c)	-	-	-	-	+/-	+	-	-	-	+	+	-	+	-	+

Table 1. Main results of evaluation of workflow products using the workflow patterns [4, 3].

4. Web service composition

After putting workflow management into an historical perspective and introducing workflow patterns as a means to evaluate languages/systems, we focus on web service composition and investigate how we can apply results from the workflow domain to the web services domain.

The goal of web services is to exploit XML technology and the Internet to integrate applications than can be published, located, and invoked over the Web. A typical example of a web services application is the Galileo system that connects more that 42,000 travel agency locations to 37 car rental companies, 47,000 hotels, and 350 tour operators. To truly integrate business processes across enterprise boundaries it is not sufficient to merely support simple interaction using standard messages and protocols. Business interactions require long-running interactions that are driven by an explicit process model. This raises the need for web service composition languages such as BPEL4WS, WSFL, XLANG, WSCI, and BPML.[1] Before discussing BPEL4WS and the likes, we focus on the typical technology they build on, i.e., SOAP, WSDL, and UDDI.

SOAP (Simple Object Access Protocol) is a protocol for exchange of information in a decentralized, distributed environment using typed message exchange and remote in-vocation. WSDL (Web Services Description Language) is an XML format for describing network services based on a standard messaging layer like SOAP. A WSDL document defines services as collections of network endpoints, or ports. In WSDL, the abstract definition of endpoints and messages is separated from their concrete network deployment or data format bindings. This allows for the reuse of abstract definitions: messages, which are abstract descriptions of the data being exchanged, and port types which are abstract collections of operations. The concrete protocol and data format specifications for a particular port type constitute a reusable binding. A port is defined by associating a network address with a reusable binding, and a collection of ports defines a service. UDDI (Universal Description Discovery and Integration) is the definition of a set of services supporting the description and discovery of: (1) businesses, organizations, and other web services providers, (2) the web services they make available, and (3) the technical interfaces which may be used to access those services. Simply put: UDDI can be used to build "yellow pages" for web services. At this point in time, there seems to be consensus on the use of SOAP, UDDI, and WSDL. Therefore, we assume these standards to be in place in the remainder.

Web service composition languages build directly on top of WSDL. A process in BPEL4WS both provides and/or uses services described in WSDL. Note that a WSDL service is composed of ports that provide operations. Each

---

[1]These languages are also known as web service flow languages, web service orchestration languages, and web-enabled workflow languages.

operation either sends a message (one-way), receives and sends a message (request-response), sends and receives a message (solicit-response), or receives a message (notification). WSDL services and the corresponding operations are glued together to provide composed services. To glue such services together a process model is needed to specify the order in which the operations are executed. A web service composition language provides the means to specify such a process model. An important difference between WSDL and a language like BPEL4WS has to do with state management. WSDL is in essence stateless because the language is not aware of states in-between operations. The only state notion supported is the state in-between sending and receiving a message in a request-response or solicit-response operation. Any technology supporting a web service composition language will have to record states for processes that are more complex than a simple request-response. Only by recording the state it is possible to determine what should/can be done, thus enabling long-lived business transactions.

The BPEL4WS specification builds on IBM's WSFL (Web Services Flow Language) and Microsoft's XLANG (Web Services for Business Process Design). XLANG is a block-structured language with basic control flow structures such as sequence, switch (for conditional routing), while (for looping), all (for parallel routing), and pick (for race conditions based on timing or external triggers). In contrast, WSFL is graph-oriented, and relies mainly on the concept of *control links*. Graphs defined using control-links can be nested but need to be acyclic. Iteration is only supported through exit conditions, i.e., an activity/subprocess is iterated until its exit condition is met. The control flow part of WSFL is almost identical to the workflow language used by IBM's MQSeries Workflow. This may be surprising given the fact that this workflow language is very different from most others. For example, the semantics of control links in MQSeries Workflow is defined in terms of the so-called "dead-path elimination" principle. The idea of dead-path elimination is that both positive and negative values can be propagated through control links, determining whether the activities in a path should be executed or not. Using this idea, it is possible to capture not only the basic "merging" patterns, namely Synchronization (WP3) and Simple Merge (WP5), but also the more advanced Synchronizing Merge (WP7), which is not supported by many mainstream workflow products. Although dead-path elimination is a nice feature, it is quite exotic and not supported by most systems.

The correspondence between WSFL and MQSeries Workflow can easily be explained by the fact that both languages are defined by the same people. Similar comments can be made for XLANG and Microsoft's BizTalk Orchestrator. XLANG is based on the current middleware solution of Microsoft and therefore hardly qualifies as a "standard".

Moreover, the merger of these two languages (WSFL and XLANG) which are based on different paradigms, leads to a language (BPEL4WS) with clearly overlapping constructs (i.e. lacking orthogonality). For example, the simple patterns (WP1–WP5) can be specified using either structured activities (switch, sequence, etc.), or control links, or even combinations of both as discussed in [23]. Hence, process designers using BPEL4WS require guidance regarding when to use which style (XLANG-style or WSFL-style).

BPEL4WS is not the only standard in the area of web service composition that has been proposed in recent years. Sun, BEA, SAP, and Intalio have introduced another candidate for web services composition: WSCI (Web Service Choreography Interface), which has been taken as one of its inputs by the recently created W3C Web Services Choreography Group (www.w3.org/2002/ws/chor). Intalio has also initiated the Business Process Management Initiative (BPMI.org) which developed the BPML (Business Process Markup Language). OASIS and UN/CEFACT support ebXML (Electronic Business using eXtensible Markup Language). Part of ebXML is BPSS (Business Process Schema Specification), yet another standard having a similar scope as BPEL4WS, WSFL XLANG, WSCI, and BPML, but with subtle differences that deserve further investigation. The abundance of overlapping standards for web services composition is overwhelming. In fact, the collection of competing web services standards without clear added value has been termed the *Web Services Acronym Hell* (WSAH) [1].

Outside the web services domain there have been other initiatives to standardize the specification of executable business processes. Most notable is the initiative of the Workflow Management Coalition (WfMC). Since 1993, the WfMC has been active to standardize both a workflow process definition language and the interfaces between various workflow components. In August 2002 the WfMC released XPDL (XML Process Definition Language, Version 1.0 Beta) to support the exchange of workflow specifications between different workflow products. According to Jon Pyke, WfMC Chair and CTO Staffware, XPDL is consistent with BPEL4WS, but goes far beyond the standards for web service composition. Clearly, many people working on standards for web service composition have not benefit enough from the experiences in the workflow domain. However, it is also clear that the standards of the WfMC have not been adopted by the workflow vendors. Some of the systems can export to XPDL, but none of them can import XPDL from another system and still produce meaningful results. One of the reasons is that after working on workflow standards for more than a decade, there is still no consensus on the workflow constructs that need to be supported and their semantics. It is remarkable how many different interpretations of a join construct exist in contempo-

rary workflow languages: "Wait for all (AND-join)", "Wait for first and reset (XOR-join)", "Wait for first and ignore remaining ones", "Wait for all to come", etc.

5. Web service composition languages

A comparison of BPEL4WS, XLANG, WSFL, BPML and WSCI is given in Table 5. The ratings for BPEL4WS, XLANG, WSFL, BPML and WSCI in the table are taken from [2, 23]. As indicated before, a + in a cell of the table refers to direct support (i.e. there is a construct in the language which directly supports the pattern). A - indicates that there is no direct support. This does not mean though that it is not possible to realize the pattern through some work-around solution. In fact, any of the patterns can be realized using a standard programming language but this is meaningless.[2] Sometimes there is a feature that only partially supports a pattern, e.g. a construct that directly supports the pattern but imposes some restrictions on the structure of the process.

In [2, 23] we show constructs for each of the patterns mentioned. For example, BPML realizes the Deferred Choice (WP16) through the choice construct. The semantics of choice, i.e. awaiting for the arrival of an event and depending on the event selecting a pre-specified route, captures the key idea of this pattern, namely a choice is not made immediately when a certain point (i.e. the choice activity) is reached, but delayed until receipt of some kind of external trigger. BPEL4WS offers a construct similar to the choice in BPML. In BPEL4WS this construct is named the pick. In this paper, we do not show explicit solutions for the patterns and focus on the "big picture" shown in Table 5.

The following observations can be made from the table:

- As the first 5 patterns correspond to basic routing constructs, they are directly supported by all languages.

- BPEL4WS as a language integrating the features of the block structured language XLANG and the directed graphs of WSFL, indeed supports the union of patterns supported by XLANG and WSFL.

- BPEL4WS, in contrast to BPML, does offer direct support for the Multi Choice and Synchronizing Merge. This is a consequence of the "dead-path elimination" principle inherited from WSFL.

- BPEL4WS does not support the Multi-Merge pattern, while BPML directly supports it with some restric-

---

[2]Languages such as BPEL4WS, XLANG, WSFL, and BPML are Turing complete. They can be used to emulate a Turing machine, and thus can theoretically do any calculation. However, this observation is not relevant in the context at hand: Any programming language is Turing-complete, but this does not imply suitability for web services composition. Hence, we consider "direct support" rather than Turing-completeness.

tions. This is due to the fact that BPML, unlike BPEL4WS, supports invocation of sub-processes.

- Unlike many mainstream workflow languages, all the compared languages support the Deferred Choice.

- BPEL4WS, through the concept of serializable scopes, is the only one of the above languages to support the Interleaved Parallel Routing pattern, although with some restrictions.

- None of the compared languages supports *arbitrary* cycles.

When comparing BPEL4WS, XLANG, WSFL, BPML and WSCI to contemporary workflow systems [4] on the basis of the patterns discussed in this paper, they are remarkably strong. Note that only few workflow management systems support Cancel Activity, Cancel Case, Implicit Termination, and Deferred Choice. In addition, workflow management systems typically do not directly support message sending.

The trade-off between block-structured languages and graph-based languages is only partly reflected by Table 5. XLANG, BPML, and WSCI are block-structured languages. WSFL is graph-based. BPEL4WS is a hybrid language in the sense that it combines features from both the block-structured language XLANG and the graph-based language WSFL. Nearly all workflow languages are graph-based and emphasize the need of end-users to understand and communicate process models. Therefore, it is remarkable that of the five languages evaluated in Table 5, only WSFL is graph based. Moreover, in [16] no attention is paid to the graphical representation of WSFL. All the five languages are textual (XML-based) without any graphical representation. This seems to indicate that communication of the models is not considered as a requirement. In this context, we refer to the BPMI initiative toward a Business Process Modeling Notation (BPMN). BPMN is intended as a graphical language that can be mapped onto languages such as BPML and BPEL4WS [22]. Although not reflected by Table 5, the expressiveness of block-structured languages is limited to "well-structured" processes [15] where there is a one-to-one correspondence between splits and joins. In the case of BPML, this forces designers to rely on signals and related constructs (namely raise and synch) which appear to be workarounds to emulate a graph-based language.

6. Discussion

To conclude, we return to the title of the paper "Web Service Composition Languages: Old Wine in New Bottles?". To answer this question we compared workflow management systems and web service composition languages using a set of patterns. The comparison reveals that web ser-

pattern	product/standard				
	BPEL	XLANG	WSFL	BPML	WSCI
Sequence (WP1)	+	+	+	+	+
Parallel Split (WP2)	+	+	+	+	+
Synchronization (WP3)	+	+	+	+	+
Exclusive Choice (WP4)	+	+	+	+	+
Simple Merge (WP5)	+	+	+	+	+
Multi Choice (WP6)	+	–	+	–	–
Synchronizing Merge (WP7)	+	–	+	–	–
Multi-Merge (WP8)	–	–	–	+/–	+/–
Discriminator (WP9)	–	–	–	–	–
Arbitrary Cycles (WP10)	–	–	–	–	–
Implicit Termination (WP11)	+	–	+	+	+
MI without Synchronization (WP12)	+	+	+	+	+
MI with a Priori Design Time Knowledge (WP13)	+	+	+	+	+
MI with a Priori Runtime Knowledge (WP14)	–	–	–	–	–
MI without a Priori Runtime Knowledge (WP15)	–	–	–	–	–
Deferred Choice (WP16)	+	+	–	+	+
Interleaved Parallel Routing (WP17)	+/–	–	–	–	–
Milestone (WP18)	–	–	–	–	–
Cancel Activity (WP19)	+	+	+	+	+
Cancel Case (WP20)	+	+	+	+	+
Request/Reply	+	+	+	–	–
One-Way	+	+	+	–	–
Synchronous Polling	+	+	+	–	–
Message Passing	+	+	+	–	–
Publish/Subscribe	–	–	–	–	–
Broadcast	–	–	–	–	–

Table 2. Comparison of BPEL4WS, XLANG, WSFL, BPML and WSCI using both workflow and communication patterns.

vice composition languages adopt most of the functionality present in workflow systems. Therefore, the statement "Old Wine in New Bottles" is justified. At the same time, it is remarkable that web service composition languages are more expressive than the traditional workflow products. This indicates that people developing these languages may have learned from experiences in the workflow domain. Moreover, web service composition languages also provide more explicit support for the basic communication patterns. In fact, these languages can be termed "communication-oriented process definition languages", since most of the atomic activities that they support are for sending or receiving messages. The communication patterns used in our analysis are directly borrowed from a previous proposal in the area of Enterprise Application Integration [19]. An analysis based on a more refined set of communication patterns which explicitly take into account aspects such as process creation, process correlation, retries, etc. is a possible direction for future work. The patterns documented in [12] may provide a starting point.

We hope that this paper will encourage researchers and developers to look into the history of workflow management and use frameworks such as the workflow patterns to analyse and compare competing languages. Also, as suggested by [14], researchers should mobilize in order to provide formal semantics and characterizations of emerging languages, using well-established process modeling formalisms such as communicating finite state automata, Petri nets, and process algebras. For example, it would be interesting to have formal proofs that languages such as BPEL4WS map to safe and non-deadlocking Petri nets, to have decision procedures for determining whether there are unreachable activities in a BPEL4WS process definition, as well as completeness theorems regarding emerging languages or subsets thereof. An early example of an effort aimed at providing a formal foundation for Web service composition (in the broad sense) is [7]. We believe that more work along this direction is needed before the field attains the level of maturity required for moving into durable standardization efforts.

References

[1] W.M.P. van der Aalst. Don't go with the flow: Web services composition standards exposed. *IEEE Intelligent Systems*, 18(1):72–76, 2003.

[2] W.M.P. van der Aalst, M. Dumas, A.H.M. ter Hofstede, and P. Wohed. Pattern-Based Analysis of BPML (and WSCI). QUT Technical report, FIT-TR-2002-05, Queensland University of Technology, Brisbane, Australia, 2002.

[3] W.M.P. van der Aalst et al. Workflow Patterns Home Page. http://www.tm.tue.nl/it/research/patterns/.

[4] W.M.P. van der Aalst, A.H.M. ter Hofstede, B. Kiepuszewski, and A.P. Barros. Workflow Patterns. *Distributed and Parallel Databases*, 14(1):5–51, 2003.

[5] A. Arkin, S. Askary, S. Fordin, W. Jekel et al. Web Service Choreography Interface (WSCI) 1.0. Standards proposal by BEA Systems, Intalio, SAP, and Sun Microsystems, 2002.

[6] A. Arkin et al. Business Process Modeling Language (BPML), Version 1.0, 2002.

[7] L. Cardelli and R. Davies Service Combinators for Web Computing *IEEE Transactions on Software Engineering* 25(3):309–316, 1999.

[8] F. Curbera, M. Duftler, R. Khalaf, W. Nagy, N. Mukhi, and S. Weerawarana. Unraveling the Web Services Web: An Introduction to SOAP, WSDL, and UDDI. *IEEE Internet Computing*, 6(2):86–93, March 2002.

[9] F. Curbera, Y. Goland, J. Klein, F. Leymann, D. Roller, S. Thatte, and S. Weerawarana. Business Process Execution Language for Web Services, Version 1.0. Standards propsal by BEA Systems, International Business Machines Corporation, and Microsoft Corporation, 2002.

[10] C.A. Ellis. Information Control Nets: A Mathematical Model of Office Information Flow. In *Proceedings of the Conference on Simulation, Measurement and Modeling of Computer Systems*, pages 225–240, Boulder, Colorado, 1979. ACM Press.

[11] C.A. Ellis and G. Nutt. Workflow: The Process Spectrum. In A. Sheth, editor, *Proceedings of the NSF Workshop on Workflow and Process Automation in Information Systems*, pages 140–145, Athens, GA, USA, May 1996.

[12] G. Hohpe (editor). Enterprise Integration Patterns http://www.enterpriseintegrationpatterns.com, 2002.

[13] A.W. Holt. Coordination Technology and Petri Nets. In G. Rozenberg, editor, *Advances in Petri Nets 1985*, volume 222 of *Lecture Notes in Computer Science*, pages 278–296. Springer-Verlag, Berlin, Germany, 1985.

[14] R. Hull, M. Benedikt, V. Christophides, and J. Su. E-services: A look behind the curtain (Invited Paper). In *Proceedings of the International Symposium on Principles of Database Systems (PODS)*, San Diego CA, USA, June 2003. ACM Press.

[15] B. Kiepuszewski. *Expressiveness and Suitability of Languages for Control Flow Modelling in Workflows*. PhD thesis, Queensland University of Technology, Brisbane, Australia, 2003. Available via http://www.tm.tue.nl/it/research/patterns.

[16] F. Leymann. Web Services Flow Language, Version 1.0, 2001.

[17] P. Malu, J.J. Dubray, A. Lonjon et al. ebXML Business Process Specification Schema (BPSS), Version 1.05, 2002.

[18] M. zur Mühlen. *Workflow-Based Process Controlling: Foundation, Design and Application of Workflow-Based Process Information Systems*. Logos, Berlin, 2003.

[19] W.A. Ruh, F.X. Maginnis, and W.J. Brown. *Enterprise Application Integration: A Wiley Tech Brief*. John Wiley and Sons, New York NY, USA, 2001.

[20] R. Shapiro. A Comparison of XPDL, BPML and BPEL4WS (version 1.4). http://xml.coverpages.org/Shapiro-XPDL.pdf, 2002.

[21] S. Thatte. XLANG Web Services for Business Process Design, 2001.

[22] S.A. White et al. Business Process Modeling Notation (BPML), Working draft, Version 0.9, 2002.

[23] P. Wohed, W.M.P. van der Aalst, M. Dumas, and A.H.M. ter Hofstede. Analysis of Web Services Composition Languages: The Case of BPEL4WS. In *Proceedings of the 22nd International Conference on Conceptual Modeling (ER)*, Chicago IL, USA, October 2003. Springer Verlag.

[24] M.D. Zisman. *Representation, Specification and Automation of Office Procedures*. PhD thesis, University of Pennsylvania, Warton School of Business, 1977.