Service-Oriented Workflows: The DySCo Framework

Giacomo Piccinelli and Anthony Finkelstein
University College London, UK

Scott Lane Williams
Hewlett-Packard, USA

Workflow is the most popular choice among businesses for capturing and managing their operational knowledge. The authors propose an extension to traditional workflow that enables Web services to be composed into business solutions.

From a business perspective, Web services represent a new channel for the offer as well as the acquisition of business capabilities.[3] The full automation of the interaction process between providers and consumers is a peculiarity of the Web service channel. Beyond electronic data transfer, automation extends to all aspects of business interaction.[2] The negotiation of terms and conditions for service delivery and the management of service-level agreements are just some examples. Since their early definition, composition has been a central concept for Web services.[12]

In the Web service model, the provider of a new Web service WS drives the composition of internal and external capabilities in order to produce a new capability. Both internal and external capabilities are modelled as Web services that act as components for WS. Similarly, WS can be used as component for other Web services.

The fact that a capability is available internally or needs to be acquired externally reflects on the business as well as technical design of the service. The Web service model enforces the separation between a service component and the related service provider. Different providers can be selected for the same capability under different circumstances.

From a business as well as a technology perspective, Web services enable dynamic integration for service providers. The explicit management of the interaction processes associated to the delivery of a service is a fundamental aspect of Web services.

INTERACTION PROCESSES

The decupling of service logic and service provider is closely related to the most noticeable feature of the Web service model: process-oriented interfaces.

In traditional component models, function signatures often represent the only information on the interaction requirements of a component. While the invocation of a function can trigger complex interaction processes, interaction logic is not formally exposed. In the Web service model, the interaction processes associated to a Web service are explicitly formalized and exposed. WSCL[1] (Web Service Choreography Interface) and BPEL4WS[4] (Business Process Execution Language for Web Services) are examples of languages for the formalization of interaction processes associated to Web services.

An immediate benefit of process-oriented interfaces is that they open new possibilities for static as well as dynamic composition of services. As an example, the interaction protocols between individual Web services can be automatically negotiated and adapted.[11] Workflow provides an established framework for handling the interaction processes of individual Web services, as well as the orchestration of needs intrinsic in business solutions based on the composition of multiple Web services.

WORKFLOW

Workflow is a broad term that encompasses models, methodologies, and technologies related to the modelling and management of business processes. A business process captures the operational logic for the coordination of resources towards the achievement of a business objective. Workflow processes represent an embodiment of a business processes.[5]

Resources are modelled as independent units that have the capability to perform specific tasks. While resources may be capable of autonomous behaviour, the tasks specified in a workflow are performed only upon request. The coordination logic in the workflow model is based on an orchestration approach. A single logical entity is in charge of maintaining the state of the process, and to request the resources to perform tasks.

W ::= ε	*empty process*	
| $\mathbf{t_r(\sigma)}$	*task*	($\sigma : \mathcal{N} \to \mathcal{V}$ $r \in \mathcal{R}$ $t \in \mathcal{T}$)
| **W.W**	*sequence*	
| $\mathbf{W +_c W}$	*choice*	
| **W || W**	*concurrency*	
| **! W**	*loop*	

Table 1. Basic formalisation for a workflow process. N is a set of identifiers for data items. V is a set of values for data items. R is a set of resources. T is a set of task definitions.

Basic Formalisation

A basic formalisation for workflow processes is presented in Table 1. The atomic element in the specification of a workflow process (Tab. 1) is the task. A task is defined by a name *t* that indicates the activity required to a resource, as well as a name *r* that indicates the resource that has to perform the activity. The specification for a task includes indications on the information that is supplied to a resource, and on the information that the resource will produce as a result of the activity. Such information flow is modelled with a function σ that links parameter names (*N*) to their value (*V*). The flow logic for the process is expressed by the sequence, choice, concurrent, and loop operators.

The execution logic of a workflow process can be summarised by the rules in Table 2. The notation X $\xrightarrow{\alpha}$ Y indicates the progress of a process from the stage X to the stage Y. The stage of a process is defined by the information base of the process at a give time (Ω), and the description of the part of the process that remains to be executed (W). The progress of a process can also generate management data, as well as other types control information (α).

The execution of a task (step) relies on a function (w) to extract form Ω the information required by the resource. In the same way, a function (v) feeds into Ω new information generated by the resource. Only one resource is involved in each task (r∈R). A function φ defines the management information produced by the execution of the task. Management information depends on the task, as well as on the resource and data involved in the execution of the task itself.

$$\textbf{(step)}\quad \Omega{::}t_r(\sigma) \xrightarrow{\varphi(t,\,r,\,\sigma')} \Omega{<}\sigma'{::}\varepsilon \qquad \text{where } \sigma' = \rho\,(t, r, \Omega{>}\sigma)$$

$$\textbf{(loop)}\quad \Omega{::}!W \xrightarrow{\tau} \Omega{::}W.(!W)$$

$$\textbf{(seq1)}\quad \frac{\Omega{::}W1 \xrightarrow{\alpha} \Omega'{::}W1'}{\Omega{::}W1.\,W2 \xrightarrow{\alpha} \Omega'{::}W1'.\,W2}$$

$$\textbf{(seq2)}\quad \Omega{::}\varepsilon.\,W2 \xrightarrow{\tau} \Omega{::}W2$$

$$\textbf{(choice)}\quad \Omega{::}W1 +_c W2 \xrightarrow{\tau} \Omega{::}WX \quad \textit{where } X = \text{eval}(c) \in \{1,2\}$$

$$\textbf{(comp1)}\quad \frac{\Omega{::}W1 \xrightarrow{\alpha} \Omega'{::}W1'}{\Omega{::}W1 \,||\, W2 \xrightarrow{\alpha} \Omega'{::}W1' \,||\, W2}$$

$$\textbf{(comp2)}\quad \frac{\Omega{::}W2 \xrightarrow{\alpha} \Omega'{::}W2'}{\Omega{::}W1 \,||\, W2 \xrightarrow{\alpha} \Omega'{::}W1 \,||\, W2'}$$

$$\textbf{(comp3)}\quad \frac{\Omega{::}W1 \xrightarrow{\alpha} \Omega'{::}W1' \qquad \Omega{::}W2 \xrightarrow{\beta} \Omega''{::}W2'}{\Omega{::}W1 \,||\, W2 \xrightarrow{\mu(\alpha,\beta)} \Omega' \triangledown \Omega''{::}W1' \,||\, W2'}$$

Table 2. These rules specify the progress logic for a workflow process. The symbol τ indicates that no management information is produced. The function μ combines the management information generated by the progress of the two branches of the process. The function∇ aligns different versions of an information base.

The execution of a loop is handled by sequential repetitions of the process in the body of the loop (loop). A sequence (seq1/2) implies the complete execution of the first process before any progress can happen for the second process. Conditional branching (choice) implies the evaluation of a condition, and the subsequent execution of the selected process. Concurrent processes (comp 1/2/3) progress independently, unless otherwise constrained by their internal logic.

A comprehensive definition of the theoretical framework underpinning workflow can be derived from the work of authors such as C.A.R. Hoare and R. Milner.[7, 8]

Workflow Management

Workflow management systems (WfMSs) provide the link between theory and practice for workflow.[6] The WfMC (Workflow Management Coalition - www.wfmc.org) defines a WfMS as a system enabling definition, creation, and execution of workflow processes. A WfMS includes software components to store and interpret process definitions, create and execute process instances, and control their interaction with process participants and applications. Additionally, a WfMS offers administrative and audit functions on the system overall and for individual process instances. A detailed description of the standard reference architecture for WfMSs can be found in [5].

The list of commercial products for workflow management is almost endless. From IBM to SAP, Microsoft, BEA, HP and Oracle, the offer of virtually every major software manufacturer includes complete solutions or at least technology components for workflow management. Products differentiate in anything from performance to graphical interfaces, richness in the set of adapters and connectors for legacy applications, and domain-specific libraries of process definitions. As a reference, the Workflow and Reengineering International Association (www.waria.org), publishes a comprehensive directory of workflow-related products.

SERVICE-ORIENTED WORKFLOW

Traditional workflow adopts a very basic model for the resources involved in a business process. A resource is an entity that can execute a task. Data can be provided to and generated by the resource during the execution of the task. Task execution is beyond the scope of the process, and of the workflow management system.

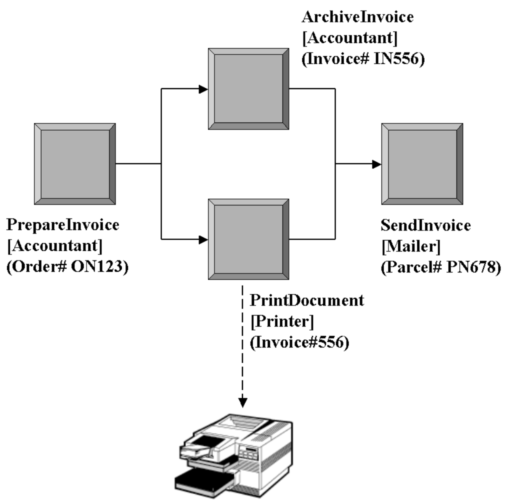


Figure 1. In traditional workflow, the process of invoicing a customer includes all the necessary steps to the generation and delivery of the invoice.

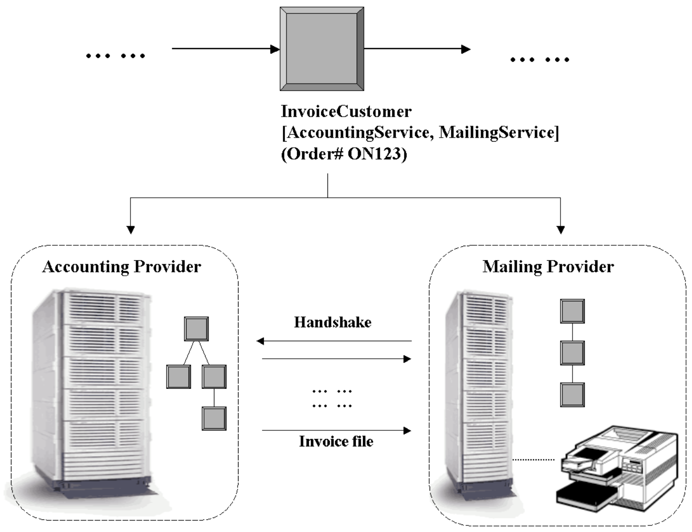


Figure 2. In a workflow that leverages Web services, invoicing of a customer simply requires identifying and composing the relevant services.

As an example of traditional workflow, Figure 1 captures part of an invoicing process. The process first requires an Accountant to prepare the invoice. As soon as the invoice is completed, the accountant archives the related information. At the same time, the invoice is printed on the appropriate forms. When the required data are archived and the document printed, the invoice is posted to the customer. In the example, the Mailer can be a person that collects the document from a known printer, and performs the basic activities required for posting a letter.

Partially due to the resource model adopted, traditional workflow enforces *concentration* and *atomisation* of process logic. Atomisation refers to the limited knowledge by the resource about the related process logic. In the invoicing example, the Accountant has no explicit knowledge that archiving follows invariably invoice preparation. Concentration refers to the fact that the process contains the coordination logic related to all the resources. Coordination takes the form of strict orchestration, whereby a central entity assigns tasks to the appropriate resources at the appropriate stage in the execution of a process. While not incompatible with Web services, traditional workflow leverages only a fraction of the potential deriving from service-oriented engineering of resources.

Figure 2 captures part of an invoicing process that is equivalent to the one in Figure 1 but that is based on Web services. Accounting and mailing (which now also includes printing) capabilities are modelled as Web services. Each Web service exposes the interaction process associated to the delivery of the service itself. The preparation and mailing of an invoice now derive from the cooperation of the providers for the accounting and mailing services. In the invoicing process, a single step is present indicating the appropriate services and the objective of their cooperation.

The overall operational knowledge that needs to be specified has not changed. Archiving still follows preparation, and posting will not be done before the archiving is complete. Still, modelling resources as Web services offers to the designer of a business process a more modular and realistic view of the business resources involved in the process.

THE DySCo WORKFLOW

The way in which Web services have developed and evolved is such that different degrees of integration have been achieved with different types of pre-existing technology. In the case of workflow, integration came almost immediately for the communication layer. Individual functions of traditional business objects were transformed into basic Web services. Each Web service would cover the equivalent of a traditional task; hence very limited changes were required to models and technology of existing WfMSs. DySCo takes the integration between workflow and Web services one step further

DySCo (Dynamic Service Composer) is the result of a two-year project involving University College London (UK), University of St. Petersburg (Russia), University of Ferrara (Italy), University of Hamburg (Germany), and Hewlett-Packard (UK and USA). The objective of DySCo was the development of a conceptual and technology framework supporting the dynamic composition of Web services. The Web services considered in DySCo are business-level services, similar to those represented in Figure 2.

The main aspect of composition targeted by DySCo is the dynamic reconfiguration of the interaction processes between providers of service components. Extending the example in Figure 2, two new providers could replace the mailing provider: one covering the printing and one covering the actual mailing. The business logic of composition does not change: first printing and then mailing. The coordination logic for the providers does instead change. The provider of the accounting service needs to coordinate with two distinct providers.

In addition to a service-oriented extension to the workflow model, DySCo supports automatic re-factoring of the coordination logic for providers of service components.[9]

Composition Model

The composition model proposed in DySCo retains the overall structure of traditional workflow (Table 1/2), with exception of the task. In DySCo the focus shifts from activity to interaction. Resources become roles, and the task becomes an interaction step. In an interaction step, sets of roles interact in order to achieve a given objective.

In traditional workflow, the task description contains sufficient information for the resource to understand precisely the activity to perform. In an interactive step, the step description contains sufficient information for the roles involved to understand precisely the way in which they have to interact with each other. As in traditional workflow, information from the overall state of the process may be provided as input for the execution of an interactive step. An interactive step may also produce output information that contributes to the state of the overall process.

Given the set R of role names and the set S of names of interactive steps, the entry for the task in the process grammar of Table 1 becomes:

$$s_r(\sigma) \quad \textit{interactive step} \;\; (\, \sigma : \mathsf{N} \rightarrow \mathsf{V} \quad r \subseteq \mathsf{R} \quad s \in \mathsf{S}\,)$$

The entry (step) for the process evolution rule in of Table 2 is replaced by:

$$\textbf{(int-step)} \;\; \Omega{::}s_r(\sigma) \xrightarrow{\varphi\,(s,\,v,\,\sigma')} \Omega \vee \sigma'{::}\varepsilon$$

$$\text{where } \sigma' = \rho\,(s, r, \Omega \mathsf{w} \sigma) \quad r \subseteq \mathsf{R} \quad v \subseteq r$$

The involvement of multiple roles in an interactive step affects both the output function ρ and the tracking function φ for the step. The output function takes into consideration the contributions from all the roles involved in the step. The tracking function can give different perspectives on the execution of the interactive step depending on the subset v of roles defined by an observer. The granularity as well as the complexity of the interaction can vary for different steps. Still, interactive steps maintain the atomicity property typical of tasks in traditional workflow.

The use of roles enforces a level of indirection between capabilities and capability providers. In particular, the interaction between the multiple roles in an interaction step is independent from the providers that will cover each role. Similarly to resource names in traditional workflow, role names are used consistently across different steps in one process. The equivalent of a task in traditional workflow can be obtained with an interactive step that specifies only one role. Similar conditions can also be recreated if one provider covers all the roles in an interactive step.

Coordination Model

The coordination model adopted in DySCo operates at role level, and is entirely based on peer-to-peer interaction. The approach is to derive from a DySCo process D a set of traditional workflow processes $\{P_j\}$, such that the result of the union of all the P_j is equivalent to D. Equivalence can be defined formally using the classic notion of bisimulation[8], which defines two processes equivalent if they exhibit identical observable behaviour.

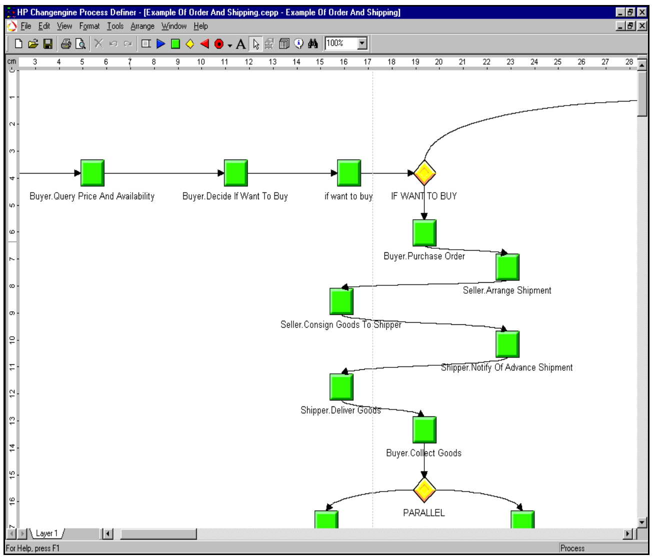
Given a DySCo process D and the set R of all the roles involved in D, let us consider P(R) the set of all the possible subsets of R. Let us also consider C(P(R)) the set of all the subsets of P(R), such that $c \in$ C(P(R)) implies that the union of all the sets in c is equal to R. Given a set $c = \{c_j\} \in$ C(P(R)) the coordination model defines a set $W = \{W_j\}$ of traditional workflow processes such that there is a one-to-one relation between the elements of c and W. Each process W_j contains the orchestration logic for the Web services of the provider covering the roles in the corresponding c_j, as well as the interaction and synchronisation logic with other providers. The interaction logic in W_j refers to the exchange of service-level information between providers. Synchronisation logic refers to the signalling between providers required in order to align the global flow of execution.

The coordination model preserves the level of indirection introduced by the composition model between a capability (captured by a role) and the Web services that implement the capability.

Implementation Framework

The implementation framework developed for DySCo includes development tools (Figure 3), as well as execution infrastructure to support the complete lifecycle of a composite Web service.

The composition logic for a Web service can be specified using a graphical environment based on an extension of the design system iGrafx™. The information is encoded using XML, and can be easily exported to other modelling tools. The graphical notation for processes derives from the reference standard specified by the WfMC.


HP Changengine Process Definer - [Example Of Order And Shipping.cepp - Example Of Order And Shipping]
File Edit View Format Tools Arrange Window Help
100%
Buyer.Query Price And Availability
Buyer.Decide If Want To Buy
if want to buy
IF WANT TO BUY
Buyer.Purchase Order
Seller.Arrange Shipment
Seller.Consign Goods To Shipper
Shipper.Notify Of Advance Shipment
Shipper.Deliver Goods
Buyer.Collect Goods
PARALLEL
Layer 1
For Help, press F1
Process

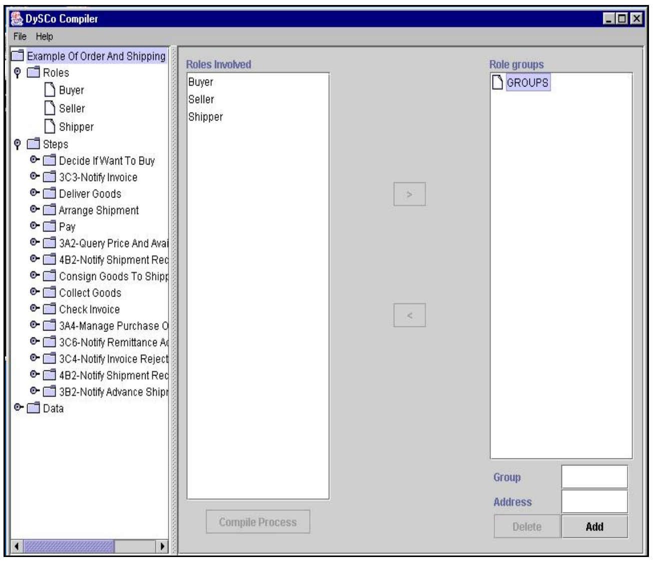


Figure 3. Two components of the DySCo implementation framework. From the top down, the first image captures the design environment for DySCo processes. The second image captures the tool for projection generation. The view on the projection generator shows the creation of role groups.

The design environment covers processes as well as steps. In addition to the flow operators in the basic workflow model, specific design patterns allow the use of higher-level constructs. Multiple branching and conditional loops are examples of such constructs. The use of sub processes is also possible. Concerning interactive steps, the interaction logic for the roles is based on an abstract execution environment, defined by DySCo. The abstract execution environment includes a virtual repositories shared by sets of the roles, as well as direct role-to-role interaction. The specification of the steps is also based on workflow.

Libraries can be created for processes as well as steps. As an example of library for interactive steps, a library including all the currently specified PIPs (Partner Interaction Processes) specified in the RosettaNet standard is available in DySCo.[10]

The projection generation environment (Figure 3) supports the automatic generation of the coordination logic for different configurations of role groups. The tool can automatically identify the roles involved in the specification of composite Web service. It is then possible to specify a number of group names, and the association between roles and groups. Once all the roles have been assigned to at least one group, the tool generates for each group the workflow processes describing the related coordination logic. The activity can be repeated for different role groups. As specified in the DySCo coordination model, the projection generator adds synchronisation operations to the service logic required to the roles in a group. The formal framework on which DySCo is based makes possible formal proofs of equivalence between a set of projection and the process from which they derive.

The execution infrastructure for DySCo has been developed on top of HP Process Manager™ (HPPM). HPPM is compliant with the WfMC standards; hence, DySCo can be ported to any other WfMC-compliant system. The infrastructure includes facilities for the deployment, execution, and management of composite Web services. The infrastructure also supports the publication and retrieval of Web-service offers through UDDI registries.

VALIDATION

Standardisation efforts such as WSCI[1] and BPEL4WS[4] are indicators that Web services are moving towards workflow. The results achieved in DySCo demonstrate that the space for integration between workflow and Web services can go beyond basic communication level.

Specific validation of the concepts proposed in DySCo comes from standardisation organisations such as RosettaNet (www.rosettanet.org) and ebXML (www.ebxml.org). A case study developed in cooperation with RosettaNet demonstrates the advantages of explicit modelling of interaction processes for service providers.[10] A simplified version of the role-based specification of the interaction processes is at the bases of the CPP/A (Collaboration Protocol Profile/ Agreement) proposed by ebXML.

CONCLUSIONS

As Web services become the reference model for business resources, workflow can provide a powerful framework for composing individual Web services into complete solutions. Web services are moving towards workflow. Workflow should also evolve to leverage Web services.

References

1. A. Arkin et al., “Web Service Choreography Interface,” *SUN Online Documentation*, 2002.
2. E. Cerami “Web Services Essentials” O’Rielly and Associates, 2002.
3. M. Clark et al., “Web Services Business Strategies and Architectures,” Expert Press, 2002.
4. F. Curbera et al., “Business Process Execution Language for Web Services,” *IBM Online Documentation*, 2002.
5. L. Fisher (Ed.), “Workflow Handbook” *Workflow Management Coalition and Future Strategy Inc.*, 2002.
6. D. Georgakopoulos, M.F. Hornick, and A.P. Sheth, “An Overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure,” *Distributed and Parallel Databases,* Kluwer Academic, Vol. 3 No. 2, 1995, pp. 119-153.
7. C.A.R. Hoare, “Communicating Sequential Processes,” *Communication of the ACM*, Vol. 21 No. 8, 1978.
8. R. Milner, “Communication and Concurrency,” Prentice-Hall, 1989.
9. O. Nierstrasz, and T.D. Meijler, “Requirements for a Composition Language,” *Object-Based Models and Languages for Concurrent Systems*, LNCS 924, Springer Verlag, 1995.
10. G. Piccinelli, A. Finkelstein, and E. Stammers, “Automated Engineering of e-Business Processes: the RosettaNet Case Study,” *Proc. Int’l Conf. Systemic, Cybernetics, and Informatics* (SCI02), Orlando, Florida, 2002.
11. G. Piccinelli, W. Emmerich, C. Zirpins, and K. Schütt, “Web Service Interfaces for Inter-Organisational Business Processes: an Infrastructure for Automated Reconciliation,” *Proc. Int’l Conf. Enterprise Distributed Object Computing* (EDOC 02), IEEE, 2002, pp. 285-292.
12. W3C “Web Service Activity”, *W3C Online Documentation,* www.w3.org/2002/ws