AUTOMATING WORKFLOWS IN MEDIA PRODUCTION
Building an Infrastructure for a Service Oriented Architecture with a Business Process Management System

Steven Van Assche, Dietrich Van der Weken, Bjorn Muylaert
VRT MediaLab, Gaston Crommenlaan 10 (bus 101), B-9050 Ghent, Belgium

Stein Desmet, Bruno Volckaert
IBCN-INTEC, Ghent University, Gaston Crommenlaan 8 (bus 201), B-9050 Ghent, Belgium

Keywords: Service oriented architecture, media production, open source, business process management system, user-generated content.

Abstract: This paper describes our experiences with building an infrastructure for automating workflows in media production based on service oriented architecture (SOA). An SOA deals with distributed software services that interact with each other. By adopting an SOA in cooperation with a Business Process Management System (BPMS) we aimed at increased efficiency and control, shorter setup times, and increased flexibility. We used open source or free products where possible, and the end result is a professional architecture suitable for small-scale to medium-scale media enterprises. Key concepts are the use of JMS as messaging layer for asynchronous, long-running service interactions (which are typical in a media production environment), the orchestration of services leading to processes with more business meaning, the graphical description of these business processes followed by the automatic generation of executable code (BPEL), support for human interactions in the processes and compliance with the WS-I Basic Profile 1.1. Our architecture is illustrated with a use case in which we automated a process that deals with the intake, review, transcoding and publishing of user-generated content.

1 INTRODUCTION

In every mid-scale to large-scale media production environment, production processes run across multiple heterogeneous systems that were originally not designed to work together. Therefore, system integration is needed in order to automate the required system interactions. The traditional system integration approach consists of putting in place a central monolithic "integration system" that accepts messages from the production systems, and that subsequently transforms and routes these message towards other production systems. These integration systems are internally very complex, difficult to scale horizontally, and based on proprietary technologies. Under the influence of Internet technologies and architectures, current integration systems and integration architectures are increasingly based on open standards and on distributed software components.

Service Oriented Architecture (SOA) is rapidly becoming the recommended system integration architecture as SOA-enabled software platforms are coming into maturity and more and more tooling is becoming available that supports SOA design and implementation methodologies. The vision of SOA can be considered threefold: 1) enable the reuse of functionality and data sources, 2) allow the orchestration of services based on a declarative description instead of programming, and 3) promote a horizontal market of service providers. In the scope of this paper, the envisioned goals of SOA are predominantly reuse of functionality on the one hand and services orchestration on the other. This will allow for higher degrees of automation in media production and will result in both setup and operational cost savings.

Van Assche S., Van der Weken D., Muylaert B., Desmet S. and Volckaert B. (2008).
AUTOMATING WORKFLOWS IN MEDIA PRODUCTION - Building an Infrastructure for a Service Oriented Architecture with a Business Process Management System.
In *Proceedings of the Third International Conference on Evaluation of Novel Approaches to Software Engineering*, pages 31-38
DOI: 10.5220/0001761400310038

Historically, the market for professional broadcast equipment has been rather vertical, with large vendors providing complete "silo" solutions. As in other markets, it is anticipated that the professional broadcast market will become more horizontal over time, and thus be more compatible with the SOA vision.

In order to assess the value of SOA for media production, some workflows in a selected use case were automated using a Business Process Management System (BPMS) that orchestrates distributed services and allows people to get involved in the automated processes through human tasks. The architecture was designed, technical and technological choices were considered, services were designed and implemented, the infrastructure was built, and the processes were designed and implemented. The primary goal of this work was to evaluate whether automating workflows in an SOA yields increased efficiency and control, shorter setup times and increased flexibility. Another goal was designing and building an SOA reference architecture and reference infrastructure for media enterprises. A third goal was the identification, design and implementation of required services for media. Because of the limited scope of the implemented use case, only very basic functionalities have been realised. More complex use cases are needed to drive the development of services with more added value.

Although the prime target audience for this work are professional media production enterprises such as broadcasters and production houses, the work is relevant to non-media enterprises also because of the generic architectural challenges and the use of generic technologies.

2 USE CASE DESCRIPTION

The use case deals with the intake, review, transcoding and publishing of user-generated content, and was defined together with the *Vlaamse Radio- en Televisieomroep* (VRT, the public broadcaster of the Flemish part of Belgium). The term "User-generated Content" refers to different kinds of contributions of non-professional end users to an online medium. Popular examples of websites based on user-generated content include YouTube, MySpace and Flickr. User-generated content is used at VRT in multiple contexts, f.e. http://www.16plus.be and http://www.toyinima.be. VRT's current automation framework is an ad-hoc solution and is very brittle. As user-generated content becomes increasingly popular, and multiple VRT websites need similar functionality, it is expected that a more professional automation infrastructure will be required.

VRT was specifically interested in shorter setup times, increased operational efficiency, a higher level of control, and increased flexibility. Extra constraints put forward at the beginning of the work in this use case were the use of open source or free products, and the targeted developer role of so called "empowered developers" who have both good technical and business knowledge.

The automation infrastructure we have built allows end users to upload their own video material intended for publication on a user-generated content website. An uploaded video item will be prepared automatically for publication, except that in some cases human intervention will be necessary to successfully finalize the publication. With respect to these human interactions, we can identify two roles participating in our process. The first role is the process administrator who is responsible for approving the format of the uploaded item in case the format was not accepted or refused automatically by the system. The process administrator thus deals with exceptions on a technical level. The second role, the reviewer, is responsible for checking whether the uploaded item is suitable for publication w.r.t. the item's content. This can be useful in those cases where the targeted website is aimed at a specific audience (e.g. children), and content moderation is required.

Figure 1 shows a functional overview of the demonstrator that has been built. The end user uploads video content through the Upload Servlet that stores the uploaded file on a local drive and triggers the business process. The business process subsequently calls the FileInfo and FileHandling services in order to retrieve technical information for the video file and to move the file between the different storage locations in the setup, respectively. It calls an e-mail notification service at the end of the publishing process. The business process also coordinates human interaction for exception handling and content approval. The specific business rules for transcoding are isolated in a sub-process TranscodeHandling. A second sub-process deals with the technical particularities of the invocation of the Vodtec transcoder. A façade component is required to translate the SOAP/HTTP call into XML-RPC/HTTP to the transcoder. The FileHandling and FileInfo services both call the RemoteFile component that can access local and remote files and that can copy and move those files.

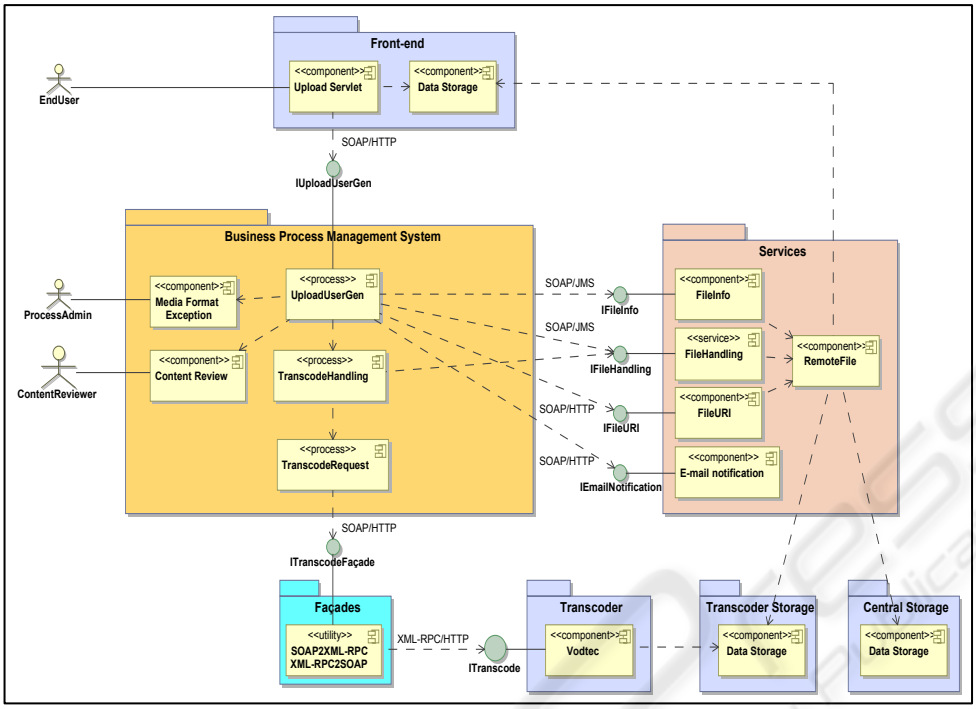

Figure 1: Functional overview of the demonstrator setup.

3 ARCHITECTURAL AND TECHNOLOGICAL DECISIONS

3.1 Message Format

The preferred technical interface for the services is SOAP because it is supported by most, if not all infrastructure components in a typical SOA environment, such as application servers and BPMS. An alternative to SOAP is XML-RPC, but XML-RPC is not really standardized and therefore support is rather limited in typical (back-office) application servers and in BPM systems. It also suffers from some serious functional and technical drawbacks which makes it ill-suited for an enterprise environment. Because of our choice of SOAP, the XML-RPC interface of the transcoder in the setup needs to be wrapped by a façade component that translates SOAP to XML-RPC for the request and XML-RPC to SOAP for the reply. Interoperability tests in our heterogeneous environment quickly revealed that the level of web services technology support in general and SOAP support in particular is limited to what WS-I Basic Profile 1.1 specifies: SOAP 1.1, WSDL 1.1, UDDI 2.0, XML 1.0, XML Schema and HTTP 1.1. The selected encoding of the SOAP messages is set to document/literal/wrapped encoding as it is the industry standard.

3.2 Message Transport

Traditionally, web services are invoked using HTTP as transport layer. HTTP restricts service interactions to synchronous short-lived interactions, and is not suited for the asynchronous long-lived service interactions as encountered in media production. Therefore, a JMS messaging layer is put in place for asynchronous message transport for the long-lived service interactions.

Besides providing reliable messaging, Message Oriented Middleware (MOM) like JMS brings other interesting capabilities such as message queuing. This allows a service listening to a message queue to have messages delivered in-order (f.e. important when updating metadata of a media item) and to control the number of concurrent service instances that are being executed (f.e. for scheduling resource-intensive operations like transcoding). MOM is a very mature concept and high performing solutions are available on the market able to cope with very high message throughputs.

The message content of the JMS messages is still SOAP. Currently, state-of-the-art web services development tooling and infrastructure systems like Enterprise Service Buses (ESB) and BPMS support the combination of SOAP with JMS, at least in the Java world.

3.3 Service Definitions

In practical development of web services, two divergent practices have emerged (Akram, 2006): the "code-first" approach (also known as "bottom up") and the "contract-first" approach (also know as "top down" or "WSDL first"). The code-first approach involves auto-generation of the WSDL file from service implementation classes using tools that leverage reflection and introspection. Alternatively, the contract-first approach involves writing the original WSDL and XML Schema, and generating service implementation classes from the WSDL file.

The code-first approach is often appealing to the developers because of its simplicity, but the contract-first approach is preferred in an enterprise environment. Platform and language interoperability problems are prevented, because both the client and server are working from a common set of interoperable XML Schema types. Defining a common platform-independent type system also facilitates separation of roles, whereby client side developers can work in isolation from server side developers. The contract-first approach is the most suitable for developing robust, interoperable services. However, in practice not all development environments support the contract-first approach in a sufficient manner.

The WSDL specification and the WS-I Basic Profile recommend the separation of WSDL files into distinct modular components in order to improve re-usability and manageability. These modular components include: 1) XML Schema files, ideally one per XML type; 2) an "abstract" WSDL file for the message and portType definitions; 3) a "concrete" WSDL file with bindings and endpoint addresses. It is also recommended to have separate namespaces for each of the three (types of) files. The namespaces are versioned using version numbers or using the revision date.

3.4 Enterprise Service Bus

An Enterprise Service Bus (ESB) is an SOA infrastructure system that provides an integration backbone. It acts as a shared messaging layer for connecting applications and other services throughout an enterprise computing infrastructure (Desmet, 2007). An ESB provides enhanced services such as message routing, message transformation, use of technology adapters, and service orchestration.

An ESB is a strongly recommended component in an SOA with so called legacy systems that do not have a SOAP interface. In reality, hardly any IT environment is free of legacy systems (w.r.t. SOAP support), and thus ESB is positioned as an essential infrastructure component in an SOA. In our use case, most services were designed and implemented from scratch and thus have a SOAP/HTTP or SOAP/JMS interface. One exception is the transcoder service that has an XML-RPC interface. Because adapting SOAP to XML-RPC is not straightforward at all (even in an ESB), our current infrastructure does not have an ESB, and the adaptation is performed in a custom façade component.

3.5 Business Process Management

The intake and publishing process in the use case is a long-running business process that involves human interaction. Automating this process is done through first modelling it in Business Process Modelling Notation (BPMN) and then converting it to an executable business process in Business Process Execution Language (BPEL) that is run by a Business Process Management System (BPMS).

The Business Process Modelling Notation (BPMN) is a standardized graphical notation for drawing business processes in a workflow. The primary goal of the BPMN effort was to provide a notation that is readily understandable by all business users, from the business analysts that create the initial drafts of the processes, to the technical developers responsible for implementing the technology that will perform those processes, and finally, to the business people who will manage and monitor those processes. Thus, BPMN creates a standardized bridge for the gap between the business process design and process implementation.

WS-BPEL 2.0, commonly known as BPEL 2.0, is a business process modelling language that is executable. WS-BPEL defines a model and a grammar for describing the behaviour of a business process based on interactions between the process and its partners. The interaction with each partner occurs through Web Service interfaces described using WSDL 1.1. The WS-BPEL process defines how multiple service interactions with these partners are coordinated (orchestrated) to achieve a business goal, as well as the state and the logic necessary for this coordination. WS-BPEL also introduces systematic mechanisms for dealing with business exceptions and processing faults.

The service endpoints (whether HTTP or JMS binding) are currently statically configured in the BPMS. However, binding dynamically to endpoints

via a Universal Description, Discovery, and Integration (UDDI) registry is probably a recommended practice in a large SOA environment.

3.6 Monitoring

A typical SOA results in a highly distributed environment. It is vitally important to monitor and log the service interactions in order to verify the correct behaviour of the services and debug the business processes. Separate tools are available for monitoring and logging SOAP/HTTP messages and SOAP/JMS messages.

Monitoring is also recommended on other levels, such as machines, software systems, and processes. Combining all this monitoring information into a coherent dashboard view relating processes with messages, systems and machines is the ultimate goal, but no products are available on the market to provide such a view.

3.7 Storage Location Abstraction

Media content is stored in multiple storage locations in the use case: on the intake server, the central storage server, the transcoder, and the publish server. The storage locations can be addressed on the servers via their respective addresses, ports and protocols and credentials (user/password combinations). As such, a file residing on the intake server might have a Uniform Resource Identifier (URI) like ftp://intake.geisha.vrt.be/video/file1.mpg, while a transcoded version of that file on the publish server has http://publish.geisha.vrt.be/cat.mp4 as URI.

In order to prevent the process designer of needing to know the concrete addresses of the storage locations and to allow easy changes to the system setup without changes to the processes, we introduced the “geisha” scheme that binds a logical location to a physical one. The same video residing on the intake server would then be identified by the URI geisha://IncomingFiles/file0001.mpg. The services that deal with the media files are configured to translate a logical location into a physical one.

4 INFRASTRUCTURE SETUP

The infrastructure for the demonstrator was predominantly setup with open source or free software systems. The services and adapters run on MS .NET or Tomcat/Axis2. We chose Intalio Community Edition as the business process management system and ActiveMQ as messaging middleware. HTTP web services monitoring is done with Amberpoint Express and generic infrastructure monitoring with OpenNMS.

4.1 Services

A basic service in media production environments is the file movement service. The FileHandling service has operations for copying a file from its source location to a destination location (using the File Exchange Protocol whenever possible), verifying the existence of a file, and deleting a file. The input parameters to the operations are Uniform Resource Identifiers that can refer to a physical location or a logical location through the geisha:// scheme. The FileHandling service can translate logical locations into physical locations while performing the requested task. Another service, the FileURI service, provides a mapping service returning the physical location for a given logical location. This service is necessary when file locations need to be externalized. The FileInfo service is a media-specific service returning technical information about a media file, such as duration, audio and video codecs, etc. This information is used by the business processes to determine what tasks need to be performed on the media. The services were implemented as Microsoft ASP.NET 2.0 web services. As they were developed first, a code-first approach was taken and WSDLs and XSDs were derived from the web service code. The choice for Microsoft ASP.NET 2.0 goes against the requirement for the use of open source or free products, but it was dictated by the service developer’s competencies. Actually, the presence of Microsoft ASP.NET 2.0 web services in the setup proved very useful in determining the technological standards required to achieve interoperability in heterogeneous environments. The FileInfo and FileHandling services operate on large files and some of their operations take up to one minute to complete, and that is why a JMS binding was preferred. The open source Enterprise Service Bus ServiceMix was tried for bridging the HTTP web services towards the JMS messaging backbone, but this was too unwieldy and a specific JMS client for MS .NET (called NMS) was used instead.

Video transcoding is done through a transcoder service that is able to transcode a media file from one encoding to another. It was provided to us by a third party. Its XML-RPC interface needed to be adapted to a SOAP interface to be compatible with our infrastructure. We tried to achieve this with

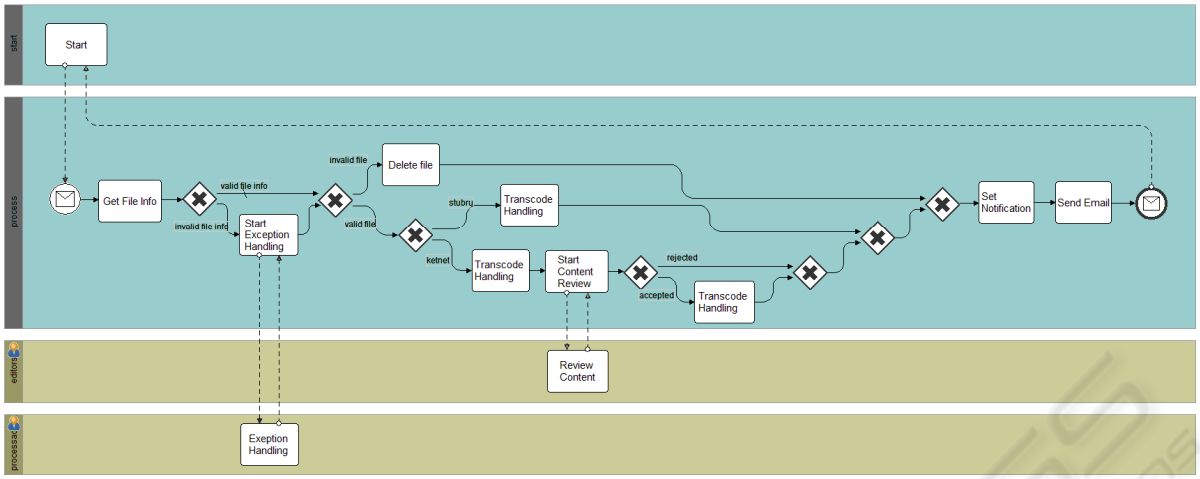

Figure 2: Intake and publishing business process.

ServiceMix, but finally a custom façade component implemented with Apache Axis2 and the Apache XML-RPC library was a far less complex solution. Axis2 is an open source web service stack developed at the Apache Software Foundation, and is available in both Java and C implementations. It not only supports SOAP style web services, but it also has integrated support for REST style web services.

ServiceMix is an open source Enterprise Service Bus product, also provided by Apache. It is based on the Java Business Integration (JBI) specification, developed by Sun under the Java Community Process. Two different types of JBI components are distinguished: Service Engines (that provide services to other components, and may consume other services as well) and Binding Components (that provide connectivity to external systems or services through a particular transport protocol, such as HTTP, JMS or JDBC).

4.2 Processes

Business process management addresses how organizations can identify, model, develop, deploy, and manage their business processes, including processes that involve IT systems and human interaction. Intalio offers a range of solutions for business process management and control. The first is the Open Source Edition of the BMPN Designer, BPEL Server and Workflow Engine. The BPEL Server is part of the Apache Foundation under the ODE project. The BPMN designer is part of the Eclipse Foundation under the STP project. The Workflow Engine is fully open sourced and available under the Eclipse Public License.

The second platform option and the one we have chosen is the Community Edition. All three components are available for download together and have all the necessary components to run as a platform. The Community Edition runs on the Apache Geronimo Application Server and the MySQL database.

The most important advantages of the Intalio platform are wide standards support and clear and well established user interfaces. The modeler is very usable, and supports BPMN elements well enough to be used for everyday modelling. Furthermore, the designer embeds a form editor based on XForms to be used with the Workflow Engine. Forms can be created graphically and are integrated with process models to support user tasks. We believe that Intalio|Designer has better than average usability. Besides this, the zero-code principle and one-click deployment model are very much appreciated. Finally, the administrative console coming with Intalio|Server offers sufficient control over active and running processes, gives a basic overview of all the process instances details and allows for a superficial monitoring functionality.

Next, we give a brief description of the different processes designed with Intalio|Designer. We have designed one main process, the UploadUserGen process, that utilizes two sub-processes, the TranscodeHandling sub-process and the TranscodeRequest sub-process. The former sub-process deals with the technical aspects of transcode requests (transcode or not, generate thumbnails or not) and the latter is responsible for sending transcode requests to the Vodtec façade and receiving notifications from it.

When a video file is uploaded through the Upload Servlet, a SOAP request is sent to the endpoint corresponding to the UploadUserGen process such that a new instance of this process starts. The UploadUserGen process is rather

business-oriented, which means that we tried to hide the more technical aspects from this process. A high-level overview of the main process is displayed in Figure 2. The horizontal bars, called pools, represent the different roles in the process.

In the first step of the UploadUserGen process the validity of the uploaded item is checked by invoking the FileInfo service. It can happen that the format of an uploaded media item is not recognized by the FileInfo service, although it is a valid media item. In those cases we assign a task to a process administrator who is responsible for handling such exceptions. The task will be listed in the process administrator's task list which becomes available after logging into the workflow platform of Intalio. When the process administrator rejects the format of the uploaded item, the item is deleted from the central file storage and a notification email sent to the end user finalizes the process. In case the process administrator approves the format of the item, the process continues.

The next decision in the process is related to the publishing destination. In case the item was uploaded for publication to the "stubru"-website, the item will be processed immediately by the TranscodeHandling sub-process. In this sub-process the necessary steps are covered to bring the uploaded item in the correct format for publishing.

The other possibility is that the item is targeted at the "ketnet"-website. In that case an additional content approval step is needed. The TranscodeHandling sub-process is invoked to generate 10 thumbnails for the reviewer as preview to the complete item. When the reviewer rejects the uploaded item, a notification email with the comment of the reviewer is sent to the end user. In case the reviewer accepts the uploaded item, the process continues and the item will be transcoded, if necessary. Therefore, again the TranscodeHandling sub-process is invoked in which the required steps are taken to bring the uploaded item in the correct publish format. To conclude, also in this case a notification email with the publish location and comment from the reviewer is sent to the end user.

4.3 Asynchronous Transport

ActiveMQ is an open source message broker from Apache, which implements the Java Message Service 1.1 specification. It is perhaps the most prominent open source JMS implementation available. Messaging functionality is not limited to JMS only, but is also extended to other protocols such as XMPP or REST.

Using ActiveMQ from any Java environment is fairly easy, and clients are available in other languages, such as Ruby, C++ or C#/.NET. Especially the C#/.NET clients are of importance to the demonstrator, as several of the services implemented in .NET needed JMS support. However, it turned out that support for C#/.NET is only very basic and is not yet fully stable.

Message persistence can be configured in various ways, depending on the level of performance and security needed. ActiveMQ 4.2 does not support message priorities, despite the fact that these are part of the JMS specification. Message priorities are expected to be introduced in future versions.

ActiveMQ has a Web console and a Java Management Extensions (JMX) interface allowing remote monitoring.

4.4 Monitoring

In the current infrastructure two types of monitoring are used: Amberpoint Express for the web services and OpenNMS for machines and software systems.

Amberpoint Express is a free HTTP web services monitoring tool for Tomcat/Axis, MS ASP.NET and IBM WebSphere Application Server. When installed on the server, it automatically configures the management system for each deployed web service without requiring code changes. In practice, Amberpoint Express intercepts the SOAP messages on the HTTP level and logs the service requests and corresponding replies. Through its browser-based interface, performance can be monitored, errors can be diagnosed, and web services can be tested. Unfortunately, Amberpoint Express only supports HTTP transport and not JMS; however the full Amberpoint SOA Management System does.

OpenNMS is an open source Java based Network Management System. It provides three different capabilities that are used in conjunction to offer network management. The first capability is that it is able to determine the availability of the various low-level services (HTTP, FTP, etc.) in the network, through the use of a polling mechanism. A second capability is the gathering and reporting of performance data on both the network and its services. OpenNMS does so by using the Simple Network Management Protocol (SNMP) and Java Management Extensions (JMX). Finally, its third capability is Event Management, which allows receiving and responding to events, such as a network outage.

5 CONCLUSIONS

The goal of the work described in this paper was the automation of several relevant workflows with a focus on achieving shorter setup times, and increased efficiency, control and flexibility. This had to be realized with open source or free products in a service oriented architecture, focusing on "empowered developers" with good technical and business knowledge.

First the architecture was designed and the infrastructure was built using several software systems: BPMS, messaging middleware, application servers, and monitoring tools. Table 1 shows the estimated effort that has gone into realizing the whole of the use case. A lot of time was put in product evaluation and selection, more specifically concerning open source products. Quite some time went into designing and implementing the services, and more or less the same time went into designing and implementing the processes. This was mainly due to the middle-out approach that was taken during development: the services and processes were developed together in an iterative way. As shown in the table, the reuse of existing services (the Vodtec transcoder) should not be underestimated, in particular when extensive configuration or technical wrapping is required. But in the end, automating workflows in media production with an SOA-approach proved to be perfectly feasible.

Table 1: Estimated effort for realizing the use case.

Topic	Effort (man-months)
Architecture	1
Product evaluation & selection	3
Demonstrator basic infrastructure	1
Service design & implementation	2
Transcoder setup & wrapping	1
Process design & implementation	2

The total time needed to implement the demonstrator was significantly longer than the initial setup time for VRT's current user-generated content automation framework, which is mainly due to the long product evaluation phase. The benefits with respect to setup times for new processes are really only achieved once the services provide at least some basic business-relevant functionality. Until that moment, a specific non-SOA based implementation is mostly faster to setup than an SOA-based implementation. But once all the services are available, implementing and deploying a new process takes only a very short time.

Concerning the request for increased efficiency and control, with the right monitoring and logging tools in place, it is possible to have a good insight into the behaviour of the technical infrastructure in general and the automated processes in particular. This comprehensive set of information allows developers to thoroughly debug and optimize the system interactions, and allows support people to track down the causes of problems very fast.

ACKNOWLEDGEMENTS

The research described in this paper was carried out in the framework of the IBBT-project GEISHA, which stands for Grid Enabled Infrastructure for Service Oriented High Definition Media Applications (https://projects.ibbt.be/geisha/). We would like to thank Vodtec (www.vodtec.com) for providing their transcoder free of charge.

REFERENCES

Web Service Description Language (WSDL) 1.1: http://www.w3.org/TR/wsdl

Simple Object Access Protocol (SOAP) 1.1: http://www.w3.org/TR/2000/NOTE-SOAP-20000508/

Universal Description, Discovery and Integration (UDDI): http://www.uddi.org/

XML Remote Procedure Call (XML-RPC): http://www.xmlrpc.com/

Java Message Service (JMS): http://java.sun.com/products/jms/

Web Services Interoperability (WS-I) Basic Profile 1.1: http://www.ws-i.org/Profiles/BasicProfile-1.1.html

Web Services Business Process Execution Language (WS-BPEL) 2.0: http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html

Business Process Modeling Notation specification (BPMN): http://www.bpmn.org/Documents/

ActiveMQ: http://activemq.apache.org/

Intalio, BMPS Community Edition: http://bpms.intalio.com

Amberpoint Express: http://www.amberpoint.com/solutions/express.shtml

OpenNMS: http://www.opennms.org/

Akram, A., Meredith, D., Allan, R., 2006. Best Practices in Web Service Style, Data Binding and Validation for use in Data-Centric Scientific Applications. In *Proc. UK e-Science All Hands Conference 2006.*

Desmet, S., Volckaert, B., Van Assche, S., Van der Weken, D., Dhoedt, D., De Turck, F., 2007. Throughput evaluation of different enterprise service bus approaches. In *Proceedings of the 2007 international conference on software engineering research en practice.*