Open Source Software for Workflow Management: The Case of YAWL

Michael Adams, Arthur H.M. ter Hofstede and Marcello La Rosa
Queensland University of Technology, Brisbane, Australia

"This column is distinguished from previous Impact columns in that it concerns the development tightrope between research and commercial take-up and the role of the LGPL in an open source workflow toolkit produced in a University environment. Many ubiquitous systems have followed this route, (Apache, BSD Unix, ...), and the lessons this Service Oriented Architecture produces cast yet more light on how software diffuses out to impact us all."

Michiel van Genuchten and Les Hatton

Workflow management systems support the design, execution and analysis of business processes. A workflow management system needs to guarantee that work is conducted at the right time, by the right person or software application, through the execution of a workflow process model.

Traditionally, there has been a lack of broad support for a workflow modeling standard. Standardization efforts proposed by the Workflow Management Coalition in the late nineties suffered from limited support for routing constructs. In fact, as later demonstrated by the Workflow Patterns Initiative (www.workflowpatterns.com), a much wider range of constructs is required when modeling realistic workflows in practice.

YAWL (Yet Another Workflow Language) is a workflow language that was developed to show that comprehensive support for the workflow patterns is achievable. Soon after its inception in 2002, a prototype system was built to demonstrate that it was possible to have a system support such a complex language. From that initial prototype, YAWL has grown into a fully-fledged, open source workflow management system and support environment (www.yawlfoundation.org).

DEVELOPMENT

Managing the development of the YAWL environment is driven by the desire to service both academic (research and teaching) and industrial user communities. Development has primarily taken place at the Queensland University of Technology, Australia. Working at a university allows the focus and inherent freedom to undertake research, which has contributed greatly to taking the environment forward. Conversely, a university’s resources are generally limited when it comes to software development and finding sufficient funding for development has been an ongoing challenge. A careful balance needs to be struck between the aim to advance research insights in the field and the objective to see uptake of the environment in practice. Interestingly, these have not proven to be mutually exclusive, as research inspiration can be drawn from practical applications and a strong conceptual and formal foundation makes use in practice a more attractive proposition.

Many open source software products have grown from university-based projects; prominent examples include Apache Web Server and BSD Unix descendants. Some of the driving factors behind the decision to open source YAWL included the desire to make research impact (in a field that was already crowded with languages and tools) and the opportunity for collaboration and feedback from the community. For example, collaborative efforts with industry led to the development of support for persistence, which enables the restoration of a workflow instance with minimal data loss after a system failure. From a research perspective, this was not within our sphere of interest, but a satisfactory solution to this problem was imperative for industrial uptake. Conversely, other industry-supported initiatives were aligned with our research interests, such as the development of a sophisticated solution for workflow instances to interact with users.

Like Firefox and OpenOffice, the YAWL environment is released under the LGPL (the GNU Lesser General Public License) with the spirit to encourage developers to contribute modifications and enhancements, while not restricting its use in larger proprietary works. An entity called The YAWL Foundation acts as custodian of all intellectual property (IP) related to the YAWL environment, and all contributors are asked to sign a deed of assignment. This serves to indemnify the Foundation from any copyright or IP infringement issues, while providing the right to distribute the software on the contributors' behalf. We believe these arrangements have worked well to encourage the general developer community to donate back extensions and enhancements, but also to assure industry partners that any work not involving changes to the core environment will remain their proprietary property.

Testing of workflow management software is complex, due to stringent requirements on user interfaces, integration with other systems, and the inherent distributed and concurrent nature of workflow processes. While no entirely satisfactory solution to the challenge of testing has been found, besides unit and system-based testing it has proven invaluable to have a community of users beta test new versions and report any problems they encounter.

ARCHITECTURE

The YAWL environment conforms to a Service Oriented Architecture (SOA), which helps to alleviate the contrasting needs of researchers, educators and practitioners by providing a framework of core components to which new services may be added. It comprises an extensible, intercommunicating set of RESTful Web services, some of which interface with end users, while others interface with other services and applications.

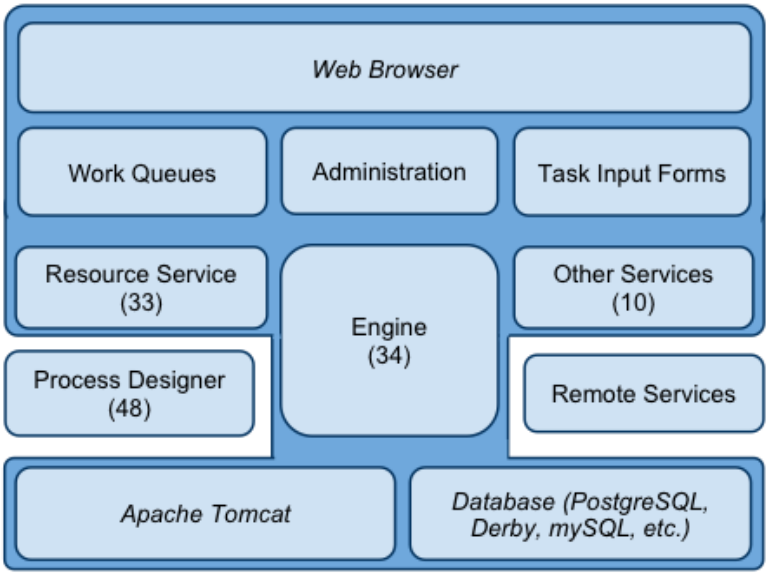

Figure 1: YAWL environment architecture – KLOCs of core components shown in brackets.

At its core is the Workflow Engine (Figure 1), responsible for the creation, routing and synchronization of tasks as defined by a process model. The Engine delegates responsibility for task execution to a service chosen from an extensible pool of available services, each one designed to handle tasks in a particular way. For example, the core Resource Service allocates (human) resources to tasks, and displays the data associated with these tasks in a worklist. Other services may invoke external Web services or applications, perform data transformations, dynamically modify the process according to business rules, and so on.

The standard user interface is provided as a series of Web pages, some of which deal with work queue presentation and manipulation, some of which cater for administrative tasks (launching new process instances, task delegation and escalation, user management etc.), and some of which provide input forms for capturing user data.

The environment runs in a servlet container (by default, Apache Tomcat) and depends on an external database management system for process persistence and logging. YAWL is written primarily in Java (with some Java Server Pages, Java Server Faces and Javascript). XML is used for data definition and transport, and XPath and XQuery for data expressions and transformations.

The environment is OS independent, and auto-installers are provided for Windows, OSX and Linux, as well as a CD image complete with operating system. Its components may also be installed individually, allowing the leveraging of different or extant servlet containers and database systems (a wide variety are supported).

Functional flexibility has always been a design priority. In fact, because of its high modularity, every part of the environment, including all user interfaces, may be modified and/or substituted by end users/developers.

SIZE AND VOLUME

Being open source software, there are some difficulties in determining the exact YAWL user demographic (there are no sales figures, for example). However, the uptake of the YAWL environment may be determined using a variety of other metrics. As at the end of September 2010, there had been over 105,000 downloads of the YAWL environment (Figure 2) from its primary host site, SourceForge (www.sourceforge.net/projects/yawl). For each of the last two years there were almost 20,000 unique visitors to the host site, from over 100 countries. Lines of code (LOC) in the core environment have grown from the 20,000 of the first release (2004), through 50,000 of version 0.7 (2005) and 66,000 of version 0.8 (2007), to 125,000 of the current version, an indication of its increasing functionality and complexity. In addition, we try to use existing technology wherever possible; around 50 third-party open source libraries are currently used.

Compared to previous impact articles, YAWL's shipping volume, measured by the number of annual downloads, exceeds that of the FMS and MR Scanner software, is less than the ECU car software and much less than RealPlayer. YAWL's code size is relatively small; the larger code sizes of the other systems are to some extent a reflection of their access to far greater numbers of engineers and developers, although our ability to include the open source community through the efficient reuse of third party libraries allows us to realise more functionality than may be indicated through code size comparisons alone.

A major difference with many of the other columns is that YAWL provides a platform on top of which others may write applications. In fact, unlike the software from Honeywell for airplanes (Jan/Feb 2011), the software for medical equipment from Philips (July/Aug 2010) and the software for cars from Bosch (Jan/Feb 2010), our users include software engineers in addition to those who use it out-of-the-box. Instead of building a focused application for a specific use case, we build a platform that should allow many engineers and organizations to do many different things. Defining a clear roadmap using the open source approach is a challenge. There is some similarity with the multimedia platform for mobile phones from Real (March/April 2010). It is interesting to note that they also apply the open source approach to part of their products. Open source may be well suited for platforms designed to be extended by other software engineers.

In terms of user support, we offer a number of different mechanisms, including paid consultancies, forums, emails and seminars. The YAWL forum has over 340 registered members.

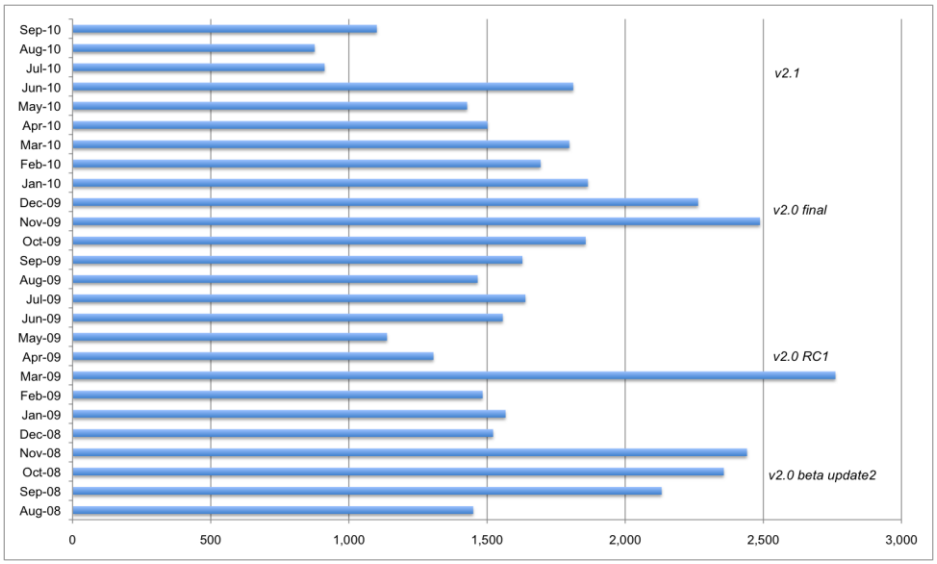

Figure 2: Downloads of the YAWL environment for the period Aug 2008 – Sep 2010, showing major release dates.

YAWL has been used as a teaching tool in over 20 universities worldwide. Industry uptake is diverse; organizations that are using or have indicated an intention to use YAWL operate in domains such as armed services in the USA, utilities in the UK, insurance in Spain, healthcare in Germany and CRM in India.

A recent survey on a sample of over 200 users indicates that YAWL is used mainly for process automation (33%) and simulation (27%), followed by documentation/requirement analysis (22%) and process improvement (18%). In industry, YAWL is predominantly used by solution architects (25%) and software engineers (20%), followed by developers (13%) and managers (11%).

The YAWL environment is a relatively young software product. 53% of respondents have just started to use or evaluate it, 32% have been using the system for less than six months, while only 12% have been using it for up to 5 years.

The schedule of new YAWL releases is to some extent dictated by the community. However, we consider regular formal releases to be very important, not least as an indicator of continuous activity; there have been 27 full release versions to date.

WHAT THE FUTURE MAY BRING

Which path YAWL takes into the future depends on how well we can engender industry and community involvement. We are committed to expanding YAWL's user base through increasing the usability, stability and functionality of the environment. However, our primary constraint is a lack of engineering resources. Consequently, we operate under a constant threat to continuity. Being open source means that, unlike commercial software products, increasing shipping volume does not correspond to an ability to hire more engineers.

One solution is to attract more paid consultancies, where an organization will pay for the cost of engineering resources to add a certain feature they specifically desire, or

to provide one-on-one support during an implementation phase. These arrangements provide the potential to expand our engineering base, at least for the period of the consultancy.

At the same time, we seek to encourage the ‘developer community’ to make contributions to the project. We have moved to a more interactive style, for example through our second generation Web site which gives prominent place to user forums and requests for involvement. However, while there has been increased activity, we have found that most represent support requests, rather than developer contributions. Another possible direction may be to move the core environment to a support organization, encapsulating support, maintenance and marketing activities, so that our engineering resources can be applied to opening up new functional avenues.

We don’t see YAWL as a serious competitor to the large, closed source workflow systems such as those offered by SAP, Oracle and IBM. However, their high cost of ownership makes YAWL an attractive alternative for small to medium enterprises. Also, the richness of YAWL’s interfaces allows it to be comfortably attached to, or embedded in, other systems (for example, the SAP-developed universal worklist connector for YAWL).

Industry partners are forging new paths into areas such as perioperative care and administrative process re-engineering, demonstrating innovative applications of the environment that can be used to ‘advertise’ its capabilities. While it is impossible to see too far over the horizon, we believe actively pursuing these strategies will ensure YAWL’s continued growth.