



Merging structure and sequence

The challenges of designing software to help teams document and execute mission critical events

Andrew Frederick Cowie
Operational Dynamics
Sydney, Australia

andrew@operationaldynamics.com

Extended Abstract

Background

People are a big part of operating and maintaining any large IT platform. While senior management tends to focus on the cost of hardware and software, unfortunately little energy and even less resources are spent ensuring that the people who make the systems run are adequately supported. But, being the resilient and inventive types that they are, systems professionals work out their own tools to make their lives better.

We're all used to automating administrative tasks in our environments with scripts. Indeed, various ways of doing so have become a cornerstone of the systems administration community.¹ There isn't, however, much in the way of automation to help support the *people* side of maintaining systems.

Massive changes and upgrades are a significant part of the life-cycle of any large site:

- ◆ software upgrades;
- ◆ launches of new versions of websites;
- ◆ database administration such as a system wide reworking of table space configurations of a live production database, or an upgrade of the underlying database software itself;
- ◆ deploying security patches (once or twice, no big deal – multiple issues times thousands of machines become a real headache);
- ◆ hardware refresh, such as the in-situ replacement of the entire suite of networking gear (switches, routers, firewalls, everything) at the expiry of that equipments' lease. A surprisingly common situation – it can be a tad challenging to keep a production site running and connected to the Internet when you rip the switching fabric out from underneath it.

These types of event all share a number of characteristics. They are complex, involving numerous interdependent systems and people both internal and external to the team carrying out the procedure. They impact the services that organizations depend on critically for their day to day operations. As such the event can only be allowed to disrupt those services minimally, if at all. Because of all these factors massive change events require considerable advance planning. The need to “get it right the first time” and the fact that numerous people need to be co-ordinated in their interactions drives the necessity to clearly document the procedures for such events.

Strangely, despite the frequency with which major change activities take place across industries of every description, there is surprisingly little available to support the planning and documentation of such events. Even gurus in the industry use simple text documents to record their procedures.²

¹ Mark Roth identified the two trends in system configuration management tools as ones which either “generate” (and overwrite in the process) config files, and those which “converge” a configuration to an expected norm by editing the in-place files. See the introduction to [9] for an excellent discussion of the pros and cons of these approaches, and the ultimate imperative to be able to work with both.

² Conversation with Tom Limoncelli at LISA '03 and also [2]. A major point of his tutorial on Massive Changes and

Part of the nature of mission critical events is their uniqueness to the organization in question and so there has been limited scope to share information about them. Certainly the fact that most such events involve sensitive (from a security standpoint) internal infrastructure means that the details of the techniques used to plan them and any learnings from executing them are rarely published, resulting in a fair bit of “reinventing the wheel.”³

Software to help teams execute procedures

Of greater concern, however, is the lack of support for the execution of procedures. In particular, coordinating the actions of a group can be very challenging when their actions need to occur within tight time frames and need to take place in carefully specified sequences.

Drawing from a somewhat unusual background,⁴ the author has attempted to apply the situation estimation techniques used when planning military operations in uncertain environments to the equally chaotic world of systems administration. In particular, the lack of a tool to document and facilitate the execution of procedures has been evident. Quantifying the nature of a needed support tool, developing a suitable design, and reporting on our implementation of that design is the focus of this paper.

A number of challenges presented themselves:

- ◆ Finding a suitable presentation: organizing the information which makes up a procedure is difficult. Especially as an event grows in scope and number of people participating in it increases, a document describing that event can quickly become unwieldy. A hierarchical style can work, but engenders the next problem:
- ◆ Writing a procedure in a word processor is difficult as one has to constantly fight to keep formatting consistent and numbering in sequence. Working in HTML at first seems attractive (because it would allow both on online as well as printed procedures) but has the same problems of keeping the formatting consistent arise, and writing HTML manually is especially labourious.
- ◆ Change control: if a word processor such as Microsoft Word or OpenOffice Writer is used, the files themselves are binary, meaning traditional tools like CVS or subversion cannot be effectively used to track changes made to a procedure as it is developed and modified.

Further, during the execution of procedures, other problems develop:

- ◆ Preventing people from rushing ahead: a natural human trait is to lose the forest for the trees. Often staff would highlight their own tasks on a printed copy, and would then proceed to work through their steps without waiting for everyone else. This is in turn related to:
- ◆ Communication: it's difficult for people who aren't in the same place (say, some in the office, some in the data center, and some working remotely from home) to keep in touch with each other. Conference call bridges can be helpful, but still people need a shared sense of context about “where they are” in a sequence.

So we have a situation where we need to find a way to record the structure of events, but at the same time need a way to make that structure interactively available to the participants and to provide context against real time running.

XML seemed an obvious choice as a medium to within which to document procedures – it is naturally hierarchical, excellent at placing that structure around textual data, and because it is text it is well suited to the line oriented version control tools mentioned above.

Upgrades was the importance of documenting the procedures behind such an event, but he indicated that there wasn't software available to could help people run through such processes.

³ Particularly telling was the observation in [3] that “we believe our work is unique in requiring all these tasks to happen on a large scale in a relatively short time”

⁴ The author was an infantry officer in the Canadian Army before returning to Unix and Linux systems administration. Bio available at <http://www.operationaldynamics.com/about/staff/andrew/>

Unfortunately, XML is difficult to work with in dynamic interactive systems. In that domain, a well established model is the web application, where by an HTML user interface is generated by an application server and then passed to the client as a web page. In order for this to work, however, OLTP systems typically depend on an unique identification number to be used as a primary key to identify what database records are to be updated. Users click on links which send events back to the server where changes can be propagated into the backing database. The information necessary to enable this interaction is drawn from a database query and embedded by the server into the links it creates as it generates the web page interface.

Elements in an XML document are not typically numbered in a uniquely identifiable manner⁵ and so cannot be translated into an interactive interface in quite the same manner. Instead, one only has a series of similar nodes at each hierarchical layer. Humans can view that list as being ordered, but there is nothing permanent which helps pin down which particular node a user event might be referring to⁶ which makes co-ordinating client and server difficult.

Certainly, order is important⁷ – In the case of massive changes and upgrades, the sequence which events are to occur and what activity can happen concurrently is the very essence of the plan being followed. So the technical problem reduces to finding ways to bridge between loose structure and strict sequence.

One obvious goal was to try and make information about a procedure be available on as many platforms as possible – in other words to use a web based interface. A common approach used by many websites to effect their presentation is to use XSLT to generate HTML from the source XML documents, and so at first this was thought a promising approach. However, the degree of interactivity desired turned out to not be possible with a standard web application architecture because the request/response nature of HTTP leads to problems pushing notifications of changed state out to clients. Also, the problem of being able to map from XML to user interface and back again remained.

We struck on what we believe to be a novel approach – doing the translation from XML to HTML on the client side in a dedicated application so that the client would implicitly have information about which node element was linked to each presented piece of the user interface. A few existing user applications take the approach of displaying HTML based on underlying XML data structures,⁸ but we have extended the idea so that the XML space can remain the basis for exchange between client and server.

On the server side, two sets of data are held. First is the XML form of the procedure itself, post-processed to assign numerical IDs to task elements. The second is a table which records the sequence of events related to the running of that procedure – when each task was completed, in what order, and by whom. Because the XML originally transmitted to the clients contained these sequence numbers, they form the basis of tying together all subsequent client/server interaction related to the execution of the event.

Having moved away from a web based client, we realized that we could have a web based presentation after all. The transformation from XML structure to user interface presentation was really two steps – one to generate the HTML used as the basis of the presentation to the user, and a second to add the programmatic hooks and callbacks to enable the interactive part of the GUI – and that both could be accomplished using XSLT. While we remained convinced that a thick client was

⁵ Interestingly, the XML specification [11] provides for each element to have an attribute of type ID for just this purpose, though few applications seem to use it.

⁶ XPath [12], while good at specifying node sets, does not facilitate uniquely identifying specific nodes, especially if the document being processed has changed between one run and the next. XPath is, however, a crucial part of XSLT.

⁷ A long running theoretical debate has been running in the configuration management community about whether installation and configuration actions need to be strictly ordered. See [6],[10]. Kaines's analysis of the difficulty of ensuring atomicity in automated systems administration actions applies equally to human procedures discussed here.

⁸ See [yelp] and [gnotime].

the way to go for people participating in the procedure, it became clear that by just running the second translation we could create a read only presentation of a procedure (as a web page) – suitable for printing and review. And, in the broadest traditions of XML, the underlying structure of the procedure remains accessible for both storage and for future development.

Having established an effective tool, the paper concludes with consideration of the potential of its application to more routine events. While targeted at massive events because of their complexity, the approach of capturing the actions staff are to take and giving them guide to follow when executing appears to also be of use for smaller scale activities. At first this was unexpected, but lesser events, though routine, are still work being done on mission critical systems. Mistakes are made. In the aftermath, promises are made “not to let it happen again”. However, most organizations lack a way to record the procedure that was used, and so lack a place to record the feedback. By using the a procedures system to encapsulate such smaller events, and developing a culture where people respect and use the procedures, costly mistakes can be reduced. Finally, such a knowledge store provides a good foundation for passing arcane knowledge on to new staff.⁹

This does not mean that procedures must be formal and complex in the ISO 9000 certification sense of the word – the documents produced by such audits tend to be bulky, verbose, and ultimately unused.¹⁰ No, the trick is to generate procedures which are flexible and easy to use.

There has been quite a bit of work leveraging XML in the Systems Administration space. In particular, attempts to encapsulate knowledge and automate administrative tasks are growing.¹¹ We hope to continue this trend. This goal of this paper is thus to demonstrate our work thus far, to frame a debate about ways to construct such procedures, and to explore ways in which we can encourage interchange of knowledge amongst those in our profession faced with similar challenges.

Status

This project was initially developed internally to support our work in the field of managing mission critical events. It evolved into this considerable body of analysis which we are pleased to contribute to the informatics world.

The software resulting from this work is **xseq**. An initial revision focused on the specific problem of operations procedures has been in internal use for several months. A rework intended to provide the foundation to address the general problem of merging sequence and structure has largely been completed, and has been released under an open source licence to a limited set of contributors to ensure basic suitability across a range of target platforms and to get the usability issues right. Public launch of this code as free software will occur at Usenix '05.

⁹ For a detailed treatment of these issues, see “Surviving Change”, [13].

¹⁰ It seems common wisdom that procedures derived from bureaucratic processes are usually worthless, but assessing the effectiveness of ISO 9000 type activities is outside the scope of this paper. As anecdotal evidence, however, the author cites a 6 page approvals process for adding a printer [7] as an example of procedures for simple tasks gone horribly wrong.

¹¹ Many tools have emerged which use XML for structure, including configuration file generation [8], and automation of daily tasks [5]. And, while variations on make have been underpinning sys admin tools for years, the advent of ant brings an XML based build tool to our space [ant].

Outline

[Note: comments are only made concerning issues not discussed in the extended abstract above]

Problem overview

Theoretical framework

Relevant practices from other industries

Note that quite a number of unusual sources are listed as references. One of the weaknesses of many fields is to focus inwards and not see that practices employed in other industries and professions can apply to them as well. In particular, the practices employed in NASA space launch operations [4] and higher level military strategic planning as applied to business situations [1] bear significantly on the practice of IT operations and systems administration.

Field Experience

A significant factor not discussed above is the importance of rehearsals. Only by going over a procedure with all the involved staff and actually practicing the event in the sequence which events will occur can a level of proficiency be developed which will reliably ensure success.

Conclusions

Technical architecture

Assessment of conventional architecture options

Strengths of XML and XSLT

Difficulties of working with XML in an interactive setting

Evaluation of user interface options

Decisions & Resulting architecture

Technical Implementation Options

Choices: target operating systems, development platform, databases and test environments

Consideration of portability versus increased capabilities available if a specific platform is chosen. Evaluation of different database storage methodologies (considerable work went into trying to find an off the shelf solution which would neatly bridge the structure/sequence gap).

Experiences during implementation

Although this paper's contribution is intended to be commentary on the problems that arise when using structured data in sequential scenarios, naturally, as we implemented this software we ran into challenges integrating various other open source technologies, and here these experiences will be described. Notable were **java-gnome**, **Berkeley DB**, and the GCC Java compiler, **gcj** and it's ability to make executable linux binaries. We're pleased that we were able to make small contributions to those projects as well (bug fixes are always welcomed, but contributing knowledge of how to use them, and how to use them together seems to have been appreciated upstream).

Trials and Results

Experience with our use of `xseq` in real world scenarios.

Future directions

Broader applications: using procedures as a way to capture lessons learned.

Remarkably, one of the hardest things to do is to effectively capture the learnings from an event. To make a mistake once is understandable. Not making it again, that's tough. This applies not just to “forgetting” something, but also to organizational evolution over time – it is not always the same people conducting a massive change event the next time around.

Broader applications: procedures applied to routine systems administration activities

As a means of minimizing human error, reducing burden on senior staff, and as a foundation for knowledge transfer to juniors.

Similar problem spaces

While there is much meeting assistance technology out there (videoconferencing, shared white boards, distributed instant messaging platforms), there is still much to be desired about the conduct of teleconferences by small organizations. It is envisioned that this software could help the people on a teleconference call stay on track (sequence) while working through agenda (structured information). Interest in tools to help with this has come from both the non-profit sector and from a major global open source project.

And, surprisingly, in developing the ideas behind this work, the author received a request to work out a front end that would be suitable for working through kitchen recipes – after all, they are structured information which is worked through sequentially. The answer to who amongst the people listed below actually asked for this feature are expected to become a mystery on par with Fermat's Last Theorem.

Credits

This work has been the result of both field experience and extensive consultation with others in the industry. In particular, the author would like to thank:

- ◆ Stephen White, Systems Administrator, University of Adelaide, Australia.
- ◆ Arjen Lentz, Technical Writer, MySQL AB.
- ◆ Martin Schwenke, Linux Kernel Hacker, IBM OzLabs, Canberra, Australia.
- ◆ Collin Boyce, Network Manager, Morehouse University, Georgia, USA.
- ◆ Angus Lees, Sydney Linux Users' Group, Australia
- ◆ Luke Kaines, Automation Guru, Reductive LLC, Tennessee, USA.
- ◆ Andrea Barisani, Systems Administrator, University of Trieste, Italy.
- ◆ Giles “Moof” Radford, Bergantells Usuaris de Linux a Mallorca i Afegitons, Mallorca, Spain.

References

Software

Process flow

[ultimus] A large scale corporate process flow design and management system. They have an impressive customer

reference list, but unfortunately feature very tight integration with the Microsoft Office platform [which makes them less ideal for Unix/Linux systems administration, even assuming an IT department could afford their software in the first place]. <http://www.ultimus.com/>

Document management

[simon] system configuration files are generated from an underlying database. An data store used for recording meta data about underlying XML documents and using XSLT to derive output files is described in [8].

[conglomerate] An XML editor specializing in text based documents such as DocBook with a novel presentation and user interface. <http://www.conglomerate.org/>

Systems configuration automation

[cfengine] The canonical tool for automated management of system configuration files. <http://www.iu.hio.no/cfengine/>

[isconf] A configuration management framework. <http://www.isconf.org/> and [10].

[psgconf] This software was introduced in a paper entitled "Preventing Wheel Reinvention", which is amusing since the author demonstrated a new tool to do configuration management. <http://www.feep.net/psgconf/>. See [9].

Innovative uses of XML

[ant] A cross-platform build tool in widespread use in the Java world. Although inspired by make, it uses build descriptions written in XML. <http://ant.apache.org/>

[yelp] The help system in the GNOME desktop project. Uses XSLT to render source XML files to HTML and displays them in a lightweight browser. <http://www.gnome.org/>

[gnotime] formerly gtr, this time tracker software generates journals of running projects and reports both from underlying XML data. Remarkably, rather than just displaying HTML, the links in the generated reports are actually live parts of the application, calling back to popup menus and allowing manipulation of the data. <http://gtr.sourceforge.net/>

Publications

- [1] Gordon R. Sullivan and Michael V. Harper, *Hope is not a Method*, Broadway Books, New York, 1996.
- [2] Tom Limoncelli, Tom Reingold, Ravi Narayan, and Ralph Loura, "Creating a Network for Lucent Bell Labs Research South", *Proceedings of the 11th Systems Administration Conference (LISA '97)*, San Diego, CA, October 1997.
- [3] Lloyd Cha, Chris Motta, Syed Babar, Mukul Agarwal, Jack Ma Waseem Shaikh and Istvan Marko, "What to Do When The Lease Expires: A Moving Experience", *Proceedings of the 12th Systems Administration Conference (LISA '98)*, USENIX, Boston, MA, 1998. p167-174
- [4] Gene Kranz, *Failure Is Not an Option*, Berkley, New York, 2000.
- [5] Thomas Stepleton, "Work-Augmented Laziness with the Los Task Request System", *Proceedings of LISA '02: Sixteenth Systems Administration Conference*, USENIX, Berkeley, CA, 2002. p1-12
- [6] Steve Traugott and Lance Brown, "Why Order Matters: Turing Equivalence in Automated Systems Administration", *Proceedings of LISA '02: Sixteenth Systems Administration Conference*, USENIX, Berkeley, CA, 2002. p99-120
- [7] "Centralized Application Systems Administration Support Procedures for the Defense Civilian Personnel Data System (DCPDS)", Department of Defense Civilian Personnel Management Service, 2002. <http://www.cpmc.osd.mil/vmo/documents/CentSysAdmProceduresOct2502.pdf>
- [8] Jon Finke, "Generating Configuration Files: The Directors Cut", *Seventeenth Large Installation Systems Administration Conference*, USENIX/SAGE, San Diego, CA, 2003. p195-204
- [9] Mark D. Roth, "Preventing Wheel Reinvention: The psgconf System Configuration Framework", *Seventeenth Large Installation Systems Administration Conference*, USENIX/SAGE, San Diego, CA, 2003. p205-211
- [10] Luke Kanies, "ISconf: Theory, Practice, and Beyond", *Seventeenth Large Installation Systems Administration Conference*, USENIX/SAGE, San Diego, CA, 2003. p115-123
- [11] The XML Specification <http://www.w3.org/TR/2004/REC-xml-20040204/>

- [12] The XML Path Language <http://www.w3.org/TR/xpath>
- [13] Andrew Cowie, "Surviving Change: Building redundancy into the one system that never has backups: the human system", *SAGE-AU '04 Annual Conference*, SAGE-AU, Brisbane, Queensland, 2004.