Tools and Processes to Support the Development of a National Platform for Urban Research: Lessons (Being) Learnt from the AURIN Project

Richard O. Sinnott, Christopher Bayliss, Luca Morandini, Martin Tomko
The Australian Urban Research Infrastructure Network (AURIN)
University of Melbourne, VIC, 3052 Australia

rsinnott@unimelb.edu.au

Abstract

The development of large-scale software systems remains a non-trivial endeavour. This is especially so when the software systems comprise services and resources coming from multiple distributed software groups, and where they are required to interoperate with heterogeneous, independent (and autonomous) distributed data providers. The use of software development and management tools to support this process is highly desirable. In this paper we focus on the software development and management systems that have been adopted within the national Australian Urban Research Infrastructure Network (AURIN - www.aurin.org.au) project. AURIN is tasked with developing a software platform to support research into the urban and built environment - a domain with many diverse software system and data needs. In particular, given that AURIN is tasked with integrating a large portfolio of sub-projects offering both software and data that needs to be integrated, deployed and managed by a core team at the University of Melbourne, we illustrate how tooling and support processes are used to manage the software development lifecycle and code/data integration from the distributed teams and data providers that are involved. Results from the project demonstrating the ongoing status are presented.

Keywords: Code Management, Collaborative Development Environment, Software Testing, Urban Research.

1 Introduction

Despite many years of experience, the development of complex software infrastructure and systems remains a challenge (Stojanovic 2005). This is especially so when the software systems are developed by distributed teams of software engineers from a multitude of organisations and where stakeholders beyond the software development teams are protagonists in the infrastructure efforts. For many major organisations such efforts are commonplace and systems and processes are well embedded into the way in which software is developed, managed and ultimately released as products with each iteration (software release) building upon the previous version. However in many circumstances, this building upon a common platform with extensive experiences and feedback from the end users/customers is simply not possible. This is often the case in the area of research-focused projects dependent upon IT. For many research projects, e.g. those sponsored by governments, the software development activities required to support the particular research endeavour are, what can best be described as a greenfield, i.e. with no pre-existing systems already running that require enhancing/tuning, but largely building from scratch. In such circumstances, the architecture of complex research-oriented systems benefits greatly from re-use of existing software components, typically these will come from a variety of sources and/or require implementation without any existing prototype in place. Even with this recycling of software systems however, the adaptation, integration and management of various subsystems to meet the needs of the research community is a far from trivial process – especially when the research needs evolve with the development of the infrastructure itself. That is, often researchers are unaware of the capabilities required of the underlying research platform until the platform exists and facilitates their research.

There have been ranges of software engineering approaches and methodologies that have been explored historically to manage such efforts (Boehm 1988, Filman 2004, Booch 1996). More recently the area of agile software development has gained widespread endorsement as the best way of developing complex systems and ensuring that they meet the end user research community needs through a rapid prototyping and release driven approach (Martin 2003). However for many agile software approaches, the assumption is often that the software developers themselves are physically co-located and directly interacting with each other for iterations of the code and infrastructure release. Often this is not the case especially in large-scale distributed software engineering activities. In such circumstances, tool support for managing the software engineering process of multiple software teams is essential. This is the focus of this paper. In particular the paper focuses on the tools and processes that have been adopted within the Australian Urban Research Infrastructure Network (AURIN) project (www.aurin.org.au).

The rest of the paper is structured as follows. Section 2 provides an introductory overview of the AURIN

project. Section 3 describes the software development framework that represents the AURIN systems architecture. Section 4 describes the tools and process that have been adopted to manage the process of developing the AURIN infrastructure and the lessons learnt in their usage. Section 5 describes the history of development of the AURIN platform and example use cases that have been supported of increasing complexity. Finally section 6 offers some conclusions and identifies future work plans for the AURIN project as a whole.

2 AURIN Overview

The Australian Urban Research Infrastructure Network (AURIN) project (www.aurin.org.au) has been funded through the Australian Government's Department of Industry, Innovation Science, Research and Tertiary Education (DIISRTE). The project is lead out of the University of Melbourne. The project formally commenced in July 2010 with the overarching remit for the '*establishment of facilities to enhance the understanding of urban resource use and management*'. AURIN is a large and complex project with government investment of $20m to run over the project lifetime. AURIN was initially expected to run to mid-2014, but has since been agreed with DIISRTE to run to mid-2015.

The AURIN project is tasked with providing urban and built environment researchers with a research environment offering seamless access to data and tools for interrogating a wide array of distributed data sets crossing government, industry/commercial and academic domains. The intention is to support multiple research activities that will enhance the understanding of key issues of Australia's past, current and future major urban settlements. This will allow better understanding of a range of phenomenon including (but not restricted to): the impact of population growth and changing demographic profiles of cities; the nature and context of urban environments in which diverse people live, e.g. the future challenges on transport networks, housing, employment, through to the health and well-being of individuals and societal groups, e.g. the elderly.

Prior to AURIN no such national urban research facility existed. Rather a wide range of largely independent silos of data and information existed with no possibility to support the interconnected and multifaceted research challenges associated with urban settlements. Similarly a range of bespoke tools and processes has often been used for the analysis of these data sets. Pockets of expertise in how to use these tools and data sets have been the norm. AURIN is tasked with development of a common data platform with associated analytical tools to provide a "*lab in a browser"* offering seamless access to distributed and heterogeneous data sets and associated tools.

It is essential to note that in developing an infrastructure to tackle such multi- and inter-disciplinary demands, it is paramount that the infrastructure is developed to be flexible, scalable and extensible. Thus there is no fixed (closed) set of data providers, data sets and tools that represent urban and built environment research. Rather, the AURIN infrastructure has to be developed to accommodate the flexible access to and use of data from a range of diverse organisations including new data providers and data sets almost *on the fly*.

To structure and scope the work on AURIN, the first year of the project (June 2010-June 2011) focused on community engagement and outreach on what the urban and built environment research community would like AURIN to do, be and ultimately deliver.

It was widely accepted that the heart of the AURIN project would be providing programmatic access to urban and built environment data sets in a manner that supported the researchers and their associated research processes. To overcome the data deluge and associated research processes adopted by many which can be classified as "Google-like", i.e. searching for relevant research data using search engines, which typically return masses of relevant and irrelevant data to the researchers, it was identified that targeted access to specific data for specific urban phenomenon was required. To this end the first year of detailed requirements and community engagement resulted in the identification of a key set of strategic urban and built environment research areas to be realized through targeted implementation stream (lenses). Each of these lenses has their own data sets, services and tools that need to be brought together. Ten aspirational lenses were identified including:

1. Population and demographic futures and benchmarked social indicators;
2. Economic activity and urban labour markets;
3. Urban health, well-being and quality of life;
4. Urban housing;
5. Urban transport;
6. Energy and water supply and consumption;
7. City logistics;
8. Urban vulnerability and risks;
9. Urban governance, policy and management;
10. Innovative urban design.

Each of these lenses has an associated expert panel that have (are) directly shaping the focus of the lens activities. Typically these panels identify core data sets and tools that are required to support the particular urban research of interest. Once identified, a typical scenario is that a range of lens-specific sponsored projects is funded through AURIN. These projects have their own software development activities. However a foundational principle of AURIN was that support for multi- and inter-disciplinary research would be possible. Thus rather than having ten separate lens subprojects, it was identified that the inter-connection and interoperability across these lenses was essential. To this end, a common unifying e-Infrastructure was needed. A core technical team at the University of Melbourne is tasked with implementing and coordinating this overarching e-Infrastructure.

At the time of writing, the current core e-Infrastructure has been undergoing development since September 2011 (with the full complement of staff in place since April 2012); the first three of these lens areas have started implementation and each of these has a multitude of lens-specific subprojects occurring. Lenses 4-6 are now at the formal contracting stage (with

subprojects to begin in early 2013) and the other lenses are currently in the early scoping stage or have yet to commence. It should be emphasized that each lens represents a significant urban research area in its own right. However a key challenge (and research opportunity) is that all of these areas are themselves inter-related. As one example, understanding the changing profile of population demographics in cities and the current and future urban landscape is essential for planning urban transport, housing, energy and water, and provisioning of healthcare.

A major challenge facing the core technical team and outsourced subprojects funded through AURIN is the number of concurrent software development activities. Thus it is expected that there will be 42 subprojects running contemporaneously in 2013. Dealing with this volume of projects is both a logistical challenge, e.g. dealing with legal aspects of negotiation with data providers for example, as well as a technical challenge. This latter point is the focus of this paper: how can a wide variety of (often domain-specific!) distributed software engineers and data providers work together to deliver a common urban and built environment research platform. The foundation for this effort is the AURIN technical architecture.

3 AURIN Technical Architecture

The AURIN e-Infrastructure has been designed around a client-server based service-oriented architecture model built upon a variety of flavours service implementations including Representational State Transfer (REST)-based services and Open Geospatial Consortium (OGC) web service flavours with other flavours of web-services in progress, e.g. statistical data-oriented web services (SDMX). The focus of the core architecture has been to establish a loosely coupled, flexible and extensible service-based architecture. In this model, individual functional components can be reused in different situations. The implementation details of each component are hidden as much as possible from the external applications and end users. The overall AURIN technical architecture is shown in Figure 1.

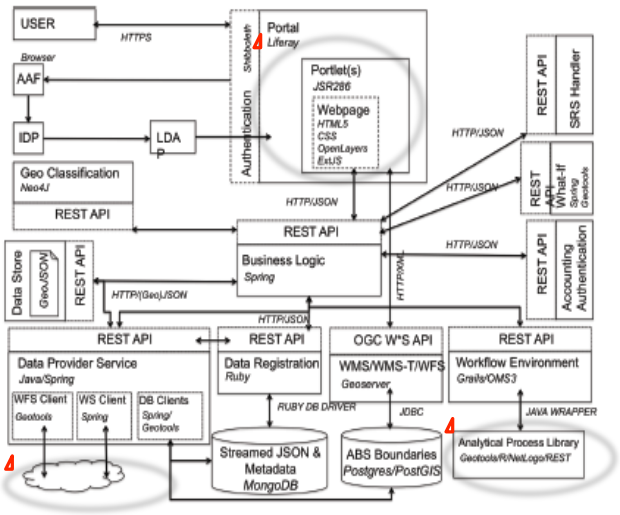

Figure 1: AURIN Implementation Architecture

The detailed description of the components and how they are used to support discovery, access and use of data including mapping, visualisation and a variety of data analytics are described in (Sinnott 2011). Of particular relevance here are the ways in which the core technical team and the external, i.e. the distributed non-core technical teams, coordinate and integrate their codes and data into the core AURIN e-Infrastructure. These possibilities are represented through the three red ovals depicted in Figure 1 and include the AURIN portal/user interface components; AURIN data provider components and the AURIN analytical process library.

3.1 AURIN Portal Interface

The AURIN portal provides a single interface to all of the data sets, services and tools offered through AURIN. The portal is (currently!) the only way in which users can discover data, access data, analyse data, visualise data. The user interface components exposed within the portal environment are deployed as JavaScript (JS) objects from a small set of JS libraries such as ExtJS, ProcessingJS and for interfacing with map data, OpenLayers. JavaScript Object Notation (JSON) objects and the geospatial flavor of JSON (GeoJSON – www.geojson.org) are used for data transmission between the business logic layer and the user interface, and a pattern of linked JS objects created from these data assures linked visualizations, e.g. brushing based on mouse-over events.

The portal interface represents a key coordination point for all of the software development activities of AURIN both for the internal core technical team and for the associated external dependencies, e.g. for user interfaces required for lens-specific software tools and/or data sets.

3.2 AURIN Data Provider Service

The Data Provider Service exposes a REST-based API that is queried by the internal AURIN components in order to access and query the distributed data services. The Data Provider Service uses real-time information on the data services (and their associated data models), their availability and potentially the load of the services themselves.

The Data Provider Service provides both externally facing REST-based and OGC-based service interfaces that are typically used to discover, query and where appropriate return data sets from a range of data providers typically using provider-mandated/preferred solutions. That is, it is often not possible for AURIN to mandate that a given data provider uses a particular technical solution. Rather the AURIN e-Infrastructure must work with whatever data provider technologies are proposed/used by the associated organizations (like the Australian Bureau of Statistics).

At present the Data Provider Service supports access to and use of a range of remote service solutions including: OGC WFS services, REST-based and SOAP-based Web Services, and data sources directly accessible through JDBC. These are currently implemented, using a combination of Hibernate, Spring and Geotools Java-based libraries. The support for non-SQL databases (MongoDB in particular) is also supported and used for processing of real-time Twitter-based information.

3.3 AURIN Analytical Process Library

Many of the needs of AURIN researchers are driven by access to and usage of analytical tools and routines. The AURIN core e-Infrastructure offers a range of *basic* analytical routines such as linear regression, however for many researchers access to richer analytical routines is essential. These can cover algorithms that allow performing geospatially-oriented weighted measures or a variety of cluster analysis. Often domain experts using statistical packages such as R, SAS or STATA to realize such routines and algorithms combining advanced statistical knowledge with urban and build environment experiences. Incorporation of such expertise (routines and how best to utilize them) into the AURIN e-Infrastructure is essential to the overall success of the AURIN platform for collaboration.

4 Distributed Code Development Tools and Processes

To support the AURIN project and its evolving set of needs and requirements both regarding the core technical e-Infrastructure and the multitude of external subprojects that are occurring, software development tool support is essential. To this end the AURIN project has adopted processes and a range of tools that are shaping the overall software engineering efforts including: distributed code versioning tools; coordination, bug tracking and feedback tools; software documentation tools; integrated testing tools, and deployment and management tools.

4.1 AURIN Agile Process

The AURIN project has adopted an agile methodology for its software development plan. Agile software development is an iterative method of determining requirements based on rapid prototyping efforts as the key way to elaborate software requirements and as a model to move towards systems that meet customer's needs (in this case the AURIN research community). An agile methodology is particularly suitable for AURIN since the requirements are extremely complex with a multitude of end users with varying expectations. Put another way, there is no single documented specification of what the AURIN e-Infrastructure should do or be, rather these requirement specifications are growing with the prototype versions of the platform. As such other development models such as sequential design ala Spiral or Waterfall models (Boehm 1988) are not suitable, since they are typically not able to cope with constant changes in requirements from end users and the associated AURIN service/data providers. Indeed, despite the year spent by AURIN on enumerating the needs of the community on the AURIN platform, the level of abstraction identified was not at an implementation level. Instead, the AURIN agile process is one based largely upon real-time reactive design, build, test and deploy.

The core AURIN team themselves are physically co-located at the University of Melbourne in a single office space. This co-location has been specifically and deliberately established to support this project and the team-based coordination efforts. That is, there is no single team member that has the complete infrastructure responsibility (at an implementation level) nor the skill sets to deliver all of the AURIN needs. Rather, it is the pooling of efforts and resources across the team that is needed.

The actual core AURIN technical team comprises a range of targeted roles including: Portal / User Interface e-Enabler; Security e-Enabler; Data/Metadata e-Enabler; Data architect; Platform infrastructure support; Statistical Geospatial e-Enabler; Geospatial e-Enabler; Workflow e-Enabler and a Middleware/business logic e-Enabler.

The full complement of staff on the core technical team has been in place since April 2012. A senior project manager responsible for the information infrastructure design supervises the day-to-day activities of the core AURIN technical team. The agile methodology that has been adopted utilizes SCRUM-based (Schwaber 2009) systems development, where the senior project manager represents the *ScrumMaster* tasked with the day-to-day efforts of the team. A key goal of a SCRUM-based methodology is organized around the SCRUM-based concept of *sprints,* which involve rapid prototyping to complete the next iteration of the AURIN e-Infrastructure and/or its components. The SCRUM product owner is the AURIN Technical Architect and weekly meetings are organized where results of the latest sprint are discussed and/or demonstrated.

The physical co-location of the core AURIN e-Infrastructure staff allows for immediate feedback and discussions on software development issues that arise. Even with this however and the associated tools that are adopted (see below), there is a need to ensure that the efforts of the team are properly coordinated and synchronized. To support this process, the AURIN e-Infrastructure team runs a daily stand-up session where their daily plans and development issues are identified and discussed. This daily process is augmented with a white-board tracking of efforts and issues as shown in Figure 2 where the horizontal rows represent the individual team members and the vertical columns the specific activities that have been identified for completion as part of the current sprint. The left hand column represents the starting point of a given activity through to the right hand column, which indicates when a particular activity has been successfully completed. As well as providing an overview of the activities of the individual team members, this approach allows to see where bottlenecks and delays are arising for individuals and across the team, with the added (and inadvertent!) advantage of visibly incentivizing team members.

Figure 2: White-board based Work and Activity Scheduling

This model of software development has major advantages for rapid prototyping but does obviously not cater for remote software engineers. To this end, the project has organized a series of *CodeSprints* where the distributed software teams working on lens specific projects come together with the core technical team and, through close coordination of the AURIN senior project manager, work on joint development and integration activities. Thus far three *CodeSprints* have taken place with a fourth planned in November 2012.

The AURIN e-Infrastructure requires far more interaction between the technical teams than physical face-to-face meetings every quarter however. To this end, the AURIN project has adopted a portfolio of distributed software development and management systems.

4.2 AURIN Code Versioning Tools

For many distributed software development projects, network-accessible code versioning systems have been widely recognized as an essential component for the implementation of distributed systems. A range of code management systems currently exist including for example: Code Versioning System (CVS) (http://sourceforge.net/apps/trac/sourceforge/wiki/CVS), Mercurial (http://mercurial.selenic.com), Subversion (http://en.wikipedia.org/wiki/Apache_Subversion) and Git (https://github.org). The AURIN project has adopted Git for its code management and versioning system.

Git is the fastest growing source and revision management system, originally developed for the management of code commits by the open source development community contributing to the development of the Linux kernel. Git provides the ability to develop code in a collaborative manner without the need for a single centralized repository (but allowing the use of one). As such, it is particularly appealing for use in AURIN, where elements of the code may have to be forked and shared with large numbers of external providers for extensive periods of time, before being tested, evaluated and subsequently committed (rolled in) to the core platform.

GitHub (www.github.com) is a commercial code repository based on Git. AURIN opted to host its code in a private GitHub repository to reduce the load on the internal system administrators and leverage access to the wealth of functions provided. It also allows the checking in of code from external parties without having to expose the internal infrastructure at AURIN. At present AURIN has paid for ten Github instances that are used extensively across the project for the core e-Infrastructure development and the associated subprojects.

One of the most important features of Git for AURIN is its code branching and merging model. Instead of cloning software into a separate directory, as is the case with many code versioning systems, Git allows switching between branches in a single working directory. Thus instead of only having branches for major development Git allows routine creation, merging and destroying of multiple, *ad hoc* branches. Indeed each feature or bug can have its own branch, merged in only when it is resolved. This model allows the AURIN software developers to experiment quickly and encourages a rapid development cycle, where they can work in parallel without always having immediate dependencies on each other.

4.3 AURIN Coordination, Bug Tracking and Feedback Tools

AURIN uses a project management Web application called Redmine (http://www.redmine.org/) for the assignment and tracking of development tasks. This environment provides an important capability for tracking bugs, support and feature requests. These so-called "issues" can be assigned to particular developers, versions of the system, and internal deadlines, the progress followed by other collaborators and managers, and statistics about the progress can be collected. It also provides an environment where collaborative documentation is built and maintained in a wiki. It is important to make this environment available as much as possible for externally contracted teams who are interacting with AURIN during the outsourced phases of development – to this end, the first steps following the inception of a new subproject are the activation of the GitHub and Redmine accounts.

4.4 AURIN Software Documentation Tools

To improve collaboration and support the automated documentation of AURIN software subsystems based upon ReST-ful APIs, the AURIN project has adopted Swagger (http://swagger.wordnik.com). Swagger is a both source code-level a tool for documenting ReST-ful APIs, and a web-based user interface to browse and test thee APIs by sending API requests; indeed, Swagger allows software engineers to test an API without having to write a single line of code. As such Swagger provides a key communications tool that supports collaborative development of APIs. To achieve this, a first prototype API is developed with Swagger - typically including representative data. This allows the API's users to play with the API, and the API's developers to gather users' feedback and modify the API prototype accordingly. Once the API prototype matures (is accepted), the full API can be implemented. This is especially useful when dealing with distributed software teams.

Swagger is available for multiple languages and frameworks. Within AURIN it is used with the Node.js and Spring frameworks. Whilst it works in language-specific ways, the same annotation-driven mechanism is used throughout. Swagger works by defining request parameters, routing and descriptions as JSON objects defined within the code itself, alongside the function definition. In the case of Node.js this is highlighted below (italics representing the Swagger annotations).

```
exports.putGraph = {
        "spec": {
          "description" : "Adds a graph to database",
          "path" :
        "/graph/datasets/{datasetid}/graphs/{graphid}",
          "notes" : "",
          "summary" : "Inserts a graph in BPnet or
JSONGraph format",
          "method": "PUT",
          "params":
          [param.path("datasetid", "ID of dataset", "string"),
                param.path("graphid",
```

```
                "ID of graph", "string"),
        param.query("format",
                "Format of graph to be inserted (bpnet|
                                        jsongraph)", "string")],
        "responseClass" : "Response",
        "errorResponses" : [],
        "nickname" : "putGraph"
      },
      "action": function (request, response) {
                var dsid= request.params.datasetid;
                var graphid= request.params.graphid;
                var format= request.query.format;
```

Figure 3: Swagger Annotations for Bi-Partite / JSON Graphs

The Swagger web-based user interface that allows automated testing of such APIs is shown in Figure 4 with the request URL, the body and response for testing a Graph-oriented API.

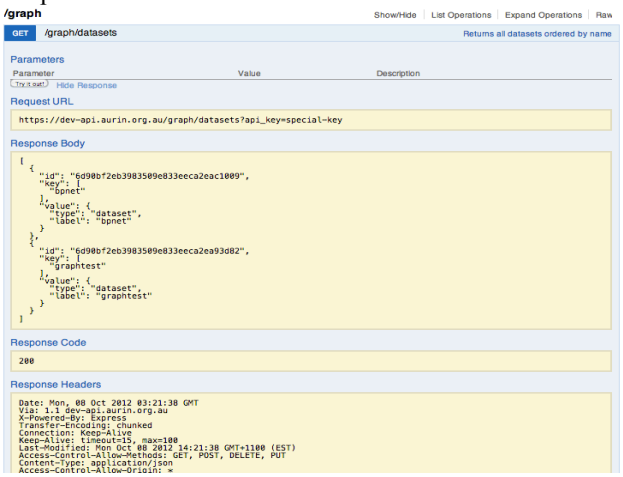

Figure 4: AURIN Swagger-based ReST Testing

It is noted that Swagger is still evolving and does not yet represent a completely mature ReST-based testing and documentation framework. Thus not every feature is used consistent across languages, e.g. full functional support of JSON Schema is not yet supported.

4.5 AURIN Integrated Testing Tools

To support the development and build environment and associated activities in software testing and integration, the AURIN project has adopted the Jenkins software environment (http://jenkins-ci.org/). Through Jenkins it is possible to automatically highlight the status of overall builds comprised of many independent software systems including those from AURIN lens projects and open source systems upon which these are often built as shown in Figure 5.

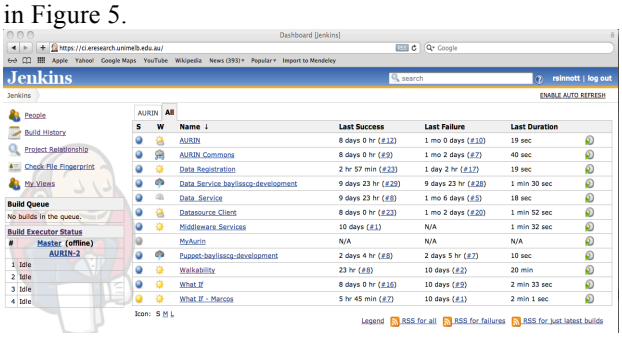

Figure 5: AURIN Jenkins-based Continuous Integration

Specifically, the AURIN project has adopted Jenkins to automate many of the typically manual processes associated with continuous software testing and integration. Advanced capabilities to support code coverage and usage are supported in Jenkins. This allows for streamlining of codes that are developed or contributed to the AURIN environment. With Jenkins, every time a new revision of the code is committed it automatically downloads, builds and tests the code in a clean environment. This ensures that any problems introduced, e.g. due to eccentricities in an individual developer's personal working environment are identified before the new code is circulated.

4.6 AURIN Common Build Environment Tools

To provide a common software build development, the AURIN core technical team has adopted the Maven (http://maven.apache.org) software build and management system. Maven is an open source tool that supports the building and management of Java-based projects.

Maven and its project object model (POM) utilizes a set of plugins that are shared by all AURIN projects. This provides a uniform AURIN build environment for all AURIN software developers that addresses many common challenges facing distributed software systems including support for tackling software dependencies, configuration challenges and unit tests. The project also includes core components in Maven. This provides an easy way for Maven clients to update their installations so that they can take advantage of any changes that been made to Maven itself. This latter feature allows support for installation of new or updated plugins from third parties or Maven itself.

4.7 AURIN Deployment and Management Tools

A key part of the AURIN development and management environment is to provide integrated deployment and configuration of phased implementations of systems. At present two versions of the e-Infrastructure are supported: *production* (accessible at https://porta.aurin.org.au) and an on-going development version of the e-infrastructure which is used for prototyping purposes and maturing the software systems to production level. This *development* version is available at http://portal-dev.aurin.org.au. It is planned that a further staging environment will also be rolled out in due course to help n the transition from development to production versions. The production version is deployed within the Australia Access Federation whilst the development version is available through the Australian test federation.

To support this process the AURIN project has adopted the Chef configuration and management software (http://www.opscode.com/chef). Chef provides a coherent management approach for the specification and delivery of deployment of e-Infrastructure components through *recipes* and *cookbooks*. These allow specification and bundling of the underlying software systems e.g. the OS versions required, the prototyped versions of software components and their dependencies and indeed the database resources and how they should be deployed onto particular virtual machines. A key advantage of Chef is

that it allows association of software bundles onto resources (VMs) with roles assigned for future access, usage and monitoring of these resources. Firewalls are also used to sandbox systems to ensure that systems developed in development and separated from production. In AURIN a single dedicated software engineer is responsible for the deployment and configuration of systems.

5 AURIN Phased Implementations

The proof of the AURIN development methodology, as with any other software engineering project can best be assessed by the successful software systems that are developed and ultimately used by the end users. The project has had two major releases of the e-Infrastructure with a third planned for mid-October 2012.

5.1 AURIN e-Infrastructure Mark-I

The first AURIN platform was largely a proof of concept system to demonstrate the viability of the approach that was to be taken. This system was described in (Sinnott 2011) and completed in a 3-month rapid prototyping effort by a small subset of developers - since at this time the full complement of the AURIN technical team was not yet in place. This Mark-I prototype version of the e-Infrastructure allowed access to a small subset of the data providers with basic visualization and analytical tools. One of the primary data resources in the Mark-I version of the AURIN e-Infrastructure was from Landgate in Western Australia. Work was also undertaken in this release using streamed social media from Twitter. The user interface and visualization/analytical capabilities for this Mark-I version of the e-Infrastructure are shown in Figure 6 where data from Landgate is being analyzed and visualized.

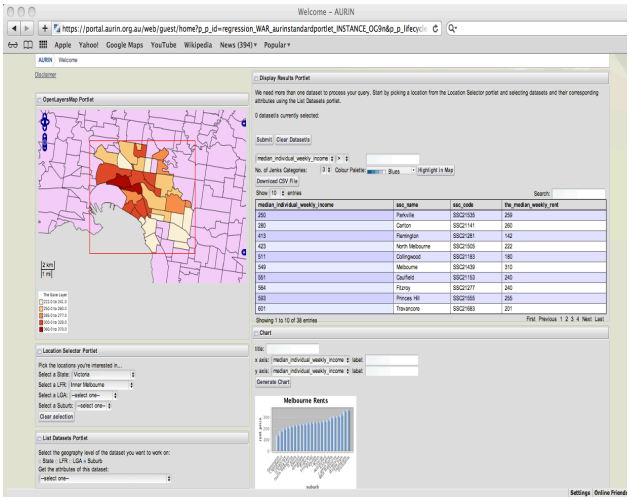

Figure 6: AURIN e-Infrastructure Mark-I (circa July 2011)

Despite its somewhat limited functionality, this initial implementation was extremely informative and influential to AURIN as a whole from a variety of perspectives. Firstly it allowed establishing a grounding and understanding of the data sets that would be dealt with in AURIN, and the geospatial information systems through which many of these data sets would be delivered. Secondly and importantly this version of the e-Infrastructure helped to bring an implementation-oriented focus to the numerous technical and management boards involved in overseeing the project. To that point, a huge effort had been expended on discussions, documentation and planning on what the system might be. This provided a basis for customer engagement far beyond the more abstract discussion that had hitherto been taking place. Thirdly, this version of the system allowed assessment of the integration of the AURIN portal (which at the time was based upon the LifeRay portal framework) with the Australian Access Federation (www.aaf.edu.au) - the federated authentication system adopted by the project. In a similar, vein this version of the e-Infrastructure enabled exploration of a suitable workflow environments and their prototyping – this was based upon the Object Modeling System version 3 (OMSv3). Experiences in developing and using OMSv3 in the AURIN context and results of applying OMSv3 on Cloud based infrastructures described in (Javadi 2012).

The majority of the software systems that formed the basis for the Mark-I version of the AURIN e-Infrastructure were largely discarded when worked commenced on the Mark-II version. There were several reasons for this, but the most important one was that the initial work was primarily to understand and articulate the problem through implementation, and to demonstrate that the overall vision was realistic and achievable. In doing this, the user interface had specifically developed/crafted user interface components (portlets) targeted to individual data sets and tools. It was rapidly recognized that this model would not scale given the volume of data to be made accessible. Similarly data and metadata considerations both in terms of access, usage storage and provenance were identified as crucial, but not supported in Mark-I. Instead it was recognized that the whole AURIN e-Infrastructure itself had to be data-driven (Sinnott 2012). To this end, a major focus was focused on the definition and realization of the architecture shown in Figure 1.

5.2 AURIN e-Infrastructure Mark-II

Driven by user demand, the Mark-II e-Infrastructure offered a variety of ways to geospatially drill into urban and built environment data and hence target the data sets of associated with particular regions (Figure 7).

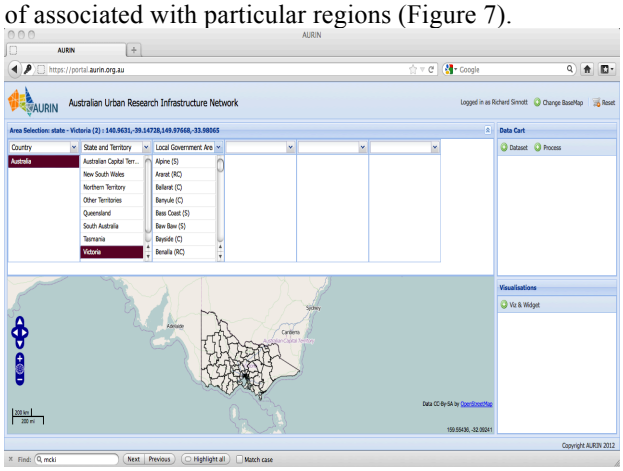

Figure 7: AURIN Mark-II Geospatial Filtering (May 2012)

This filtering of data is essential in the urban domain given the proliferation of data and information available from multiple organisations. This data selection can be

done graphically (using zoom features of the data visual interface given as a map of Australia) or through use of query interfaces that allow direct specification of the region of interest, e.g. Australia/Victoria/Melbourne as shown in Figure 7. A Google-like search interface was also offered to select particular regions or data of interest, e.g. search for data sets associated with "*employment*".

To improve the overall performance of the user experience in accessing and using data, the geometrical boundaries of spatial regions, e.g. Census Districts, Statistical Local Areas, Local Government Authorities, are stored in topologically correct representations at multiple resolution levels. In particular whilst the detailed boundaries are always used for analytical purposes, generalized boundaries are used for client-side display. This radically increases the overall responsiveness of the user interface and hence user experience. Details of how this has been achieved are described in (Tomko 2012).

The Mark-II version of the AURIN e-Infrastructure incorporated a range of services and tools developed through the core technical team and through the associated externally funded subprojects. Some of the core subproject capabilities included in AURIN e-Infrastructure Mark-II release was the data registration service developed by the Centre for Spatial Data Information and Land Administration (CSDILA). This service provides an automated mechanism to harvest metadata from OGC-compliant web feature services. The service also allows for manual additions and refinements of metadata from data providers.

Several lens specific data-oriented subprojects were incorporated into the Mark-II release including some of the data sets from the Population Health Information Development Unit (PHIDU – www.publichealth.gov.au) at the University of Adelaide. PHIDU has a rich source of health and other aggregated data sets from across Australia. Voting and a range of associated classification data were included in this release from the University of Queensland eResearch Group – drawing on work previously undertaken by the ARC funded Research Network in Spatially Integrated Social Science (www.siss.edu.au/). Data sets from the Centre of Full Employment and Equity (CofFEE – http://e1.newcastle.edu.au/coffee/) were also included in this release. Some of the data providers and the associated data sets along with their associated variables are shown in Figure 8.

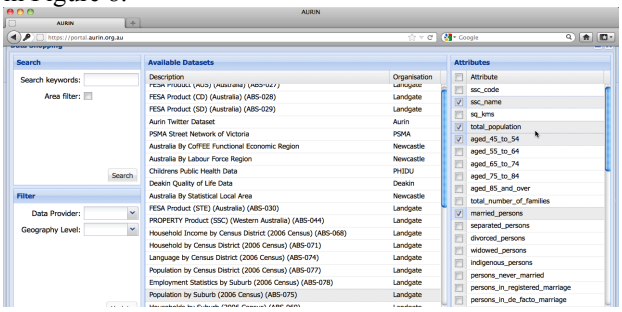

Figure 8: Mark-II Data Providers and Variables from a Provider

With this *shopping* interface, as with the Mark-I AURIN e-Infrastructure, data could be accessed from a range of federated (distributed) data providers and brought into the AURIN research space. Following the data shopping, i.e. once data had been returned to the AURIN environment a range of charting (Figure 9), mapping (Figure 10) and basic analytical capabilities (Figure 11) were offered.

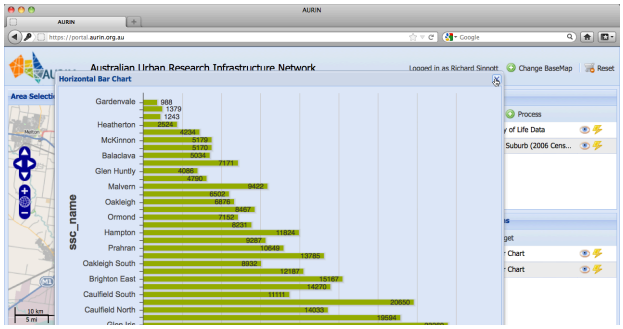

Figure 9: AURIN Mark-II Charting (May 2012) showing the total population of Statistical Local Areas in the Local Government Authority Glen Eira (from Landgate and based on the 2006 Census)

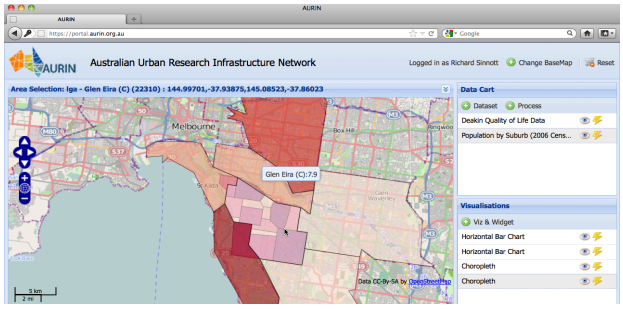

Figure 10: AURIN Mark-II Visualisation (May 2012) showing a choropleth overlaying data from the Australian Unity Quality of Life survey with specific focus on population safety, i.e. how safe do you feel in your suburb and population density for the Local Government Authority of Glen Eira

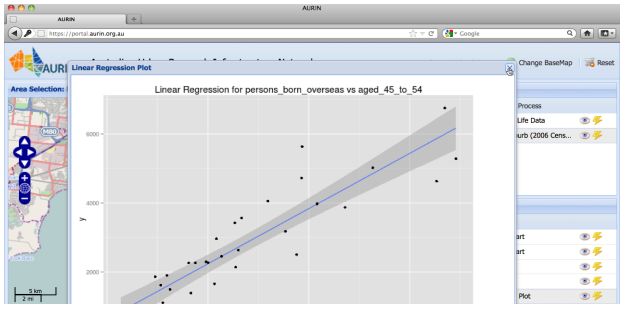

Figure 11: AURIN Mark-II Analytics (May 2012) showing the correlation (linear regression) between people born overseas and the age group 45-54 for the Local Government Authority of Glen Eira

5.3 AURIN e-Infrastructure Mark-III

The Mark-III AURIN e-Infrastructure is currently still under development (with the next formal release scheduled for mid-October 2012). This next release has been increasingly extended based upon the agile methodology that has been adopted. Included in the next release is a major increase in the number of data providers and data sets that are now provisioned. This includes a vastly extended set of data from PHIDU (with 156 separate data sets now incorporated); a variety of health survey data from VicHealth; data and services

from the Public Sector Mapping Agency (PSMA – www.psma.com.au) including access to the Geocode National Address File (GNAF) which allows to convert a valid Australian address into geospatial coordinates (latitude and longitude). A typical scenario illustrating these advanced data sets is show in Figure 12.

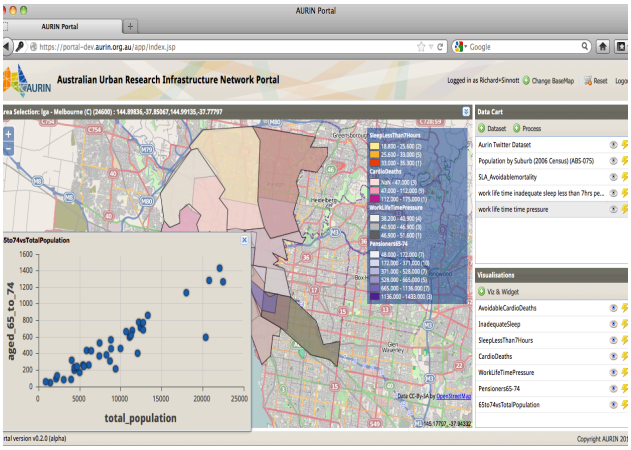

Figure 12: AURIN Mark-III and Enhanced User Interface (October 2012) showing data related to avoidable cardiovascular mortalities (PHIDU), those who sleep <7 hours and have increased work time pressure (VicHealth 2011 survey), population statistics (total population and those 65-74 years of age Landgate) for Melbourne

The user interface to the Mark-III version of the AURIN e-Infrastructure is also evolving. For example, advanced brushing techniques now allow data from map-based regions to be highlighted (and vice versa) as shown in Figure 13.

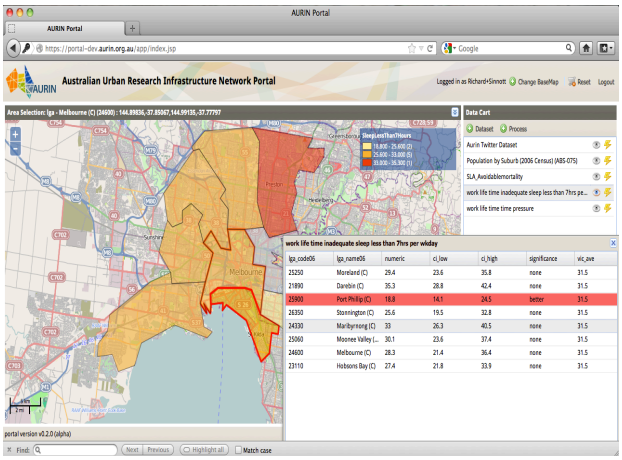

Figure 13: AURIN Mark-III and use of Brushing Techniques (October 2012) showing responses to survey questions on sleeping < 7 Hours (VicHealth 2011 survey) for Melbourne. PortPhilip is selected on the map and the associated data is highlighted.

Many of the AURIN subprojects for the first three lenses are now deep into their development activities and using the core collaboration and software management tools identified in section III. These include a range of advanced analytical capabilities from the CoFFEE group at the University of Newcastle; advanced walkability tools from the McCaughey Centre; health-based demonstrators with VicHealth and Western Health (in Perth) amongst many others.

A major enhancement in Mark-III of the e-Infrastructure is the enhanced utilization of workflow tools (based upon OMSv3). These workflow tools now allow definition of rich workflows coupling data and analytical services reflecting and capturing scientific processes. These workflows have been initially focused upon the walkability tools and use of PSMA data for geocoding addresses, and associated simulation and analytical services (implemented as part of the walkability services).

Many of the lessons learnt within the AURIN core technical team in development of the architecture identified in Figure 1, are now transferring to the collaborating partners. However this is still a non-trivial problem. The systems that are being developed are by their very nature, complex software engineering tasks for many urban research-oriented groups. To address this, the project has attempted to provide a core team technical buddy to the remote software engineering efforts. Given the number of projects expected to be running concurrently in 2013, this model will be stress tested with single technical staff members having to work across a multitude of domain-specific lens projects. Furthermore, to ensure that the expertise of any given core team technical developer is shared, the Project Manager/*ScrumMaster* has deliberately paired team members to work on each others individual software activities. This provides redundancy to the team and a shared understanding of the overall development activities.

In moving from the Mark-II to the Mark-III e-Infrastructure release, the access to and use of the AURIN e-Infrastructure has remained largely consistent, i.e. access is through the Australian Access Feature and users have to shop for data once a particular geospatial region has been selected. This consistency is an important feature to maintain to ensure that returning end users can benefit from improvements in the e-Infrastructure capabilities and not have to learn new user interfaces or techniques to access and use the system more generally.

6 CONCLUSIONS

One of the major challenges of distributed systems development that is often overlooked is not the distributed systems and hardware/software resources themselves, but the distributed teams that are often involved in these development activities. Tools to optimize the way in which these teams can coordinate their activities are essential yet are surprisingly not well recognized and adopted. This is in particular true as the types of *human* distributed collaborations vary, as much as software distributed collaborations. From teams where the members are (spatially) distributed, but belong to the same project and their resources, skills, and tools are matched, through teams that are contracted to deliver a specific type of software component based on well defined acceptance criteria, to merely *federated* contributors, i.e. software development resources that are leveraged because of their availability without the possibility to influence their direction or adherence to common project management and coding standards. This

last case represents many voluntary contributors from the open source development community, to large data providers that provide data resources to the general public, where AURIN cannot mandate the APIs and protocols that are used.

The AURIN work is far from complete – as noted the project runs to mid-2015. However the foundations for distributed collaborations and the processes that have been adopted from the project outset are now bearing fruit. Without these software development and coordination foundations, major risks would arise that could threaten the success of the project as a whole. The requirement to adopt key tools by the internal and external groups has meant that the overall software integration, management and coordination effort has been greatly simplified. Thus it is directly possible to check when a delivered piece of software meets the required integration testing for inclusion into the e-Infrastructure. As noted, it should be emphasized that these tools do not remove the overall challenges in developing and delivering distributed systems involving distributed teams, rather they are a mechanism to help to manage these challenges.

The AURIN project is running contemporaneously with many major e-Infrastructure investment activities that are currently taking place across Australia. Most notably are the $50m Research Data Storage Infrastructure (RDSI – www.rdsi.uq.edu.au), which has a specific focus on supporting storage of nationally significant research data sets, and the $47m National eResearch Collaboration Tools and Resources (NeCTAR – www.nectar.org.au) project, which has a specific focus on eResearch tools, collaborative research environments and Cloud infrastructures. The AURIN project has been engaging directly with these projects and related projects, e.g. the Australian Access Federation, in delivery of much of its underpinning infrastructure. For example, the AURIN portal and many of the associated services have been made available on virtual machines made available through NeCTAR. However given the ramping up of these projects, early issues with these projects has already arisen. To mitigate these risks and avoid the total reliance on VMs from NeCTAR or storage from RDSI, the AURIN project has purchased its own hardware systems, which are now used to augment the offerings of NeCTAR and RDSI.

The AURIN e-Infrastructure is very much a supporting activity. That is, the work in the e-Infrastructure development is not targeted at delivering novel IT solutions per se nor exploring research challenges in e-Infrastructure development, but on supporting the urban and built environment research community in *their* research needs. The AURIN research community is extremely diverse (with over 500 registered individuals and organisations) crossing a multitude of research disciplines. Whilst their feedback (positive and/or negative) will ultimately shape the AURIN e-Infrastructure, it is hoped that the underlying agile software engineering and tools described here will persist throughout the AURIN project lifetime and allow rapid evolution of systems in a tool supported manner.

6.1 Acknowledgments

The authors would like to thank the AURIN groups and committees that are directly shaping these efforts. The AURIN project is funded through the Australian Education Investment Fund SuperScience initiative. We gratefully acknowledge their support. In addition to the co-authors, the AURIN team comprises Ivo Widjaja (Portal/User Interface); Gerson Galang (Data e-Enabler); Jos Koetsier (Data/Metadata e-Enabler); William Voorsluys (Workflow e-Enabler); Damien Mannix (Infrastructure Support); Philip Greenwood (Statistical Geospatial Developer); Marcos Nino-Ruiz (Geospatial e-Enabler) and Sulman Sarwar (Middleware/Business Logic).

7 References

Stojanović, Z., Dahanayake, A., *Service-oriented software system engineering: challenges and practices*, Idea Group Publishing, 2005.

Boehm, B.W., *A spiral model of software development and enhancement*, Computer, vol. 21, Issue 5, May 1988.

Filman, R., Elrad, T., Clarke, S., *Aspect-oriented software development,* Addison-Wesley Professional, 2004.

Booch, G., *Object-oriented Development*, IEEE Transactions on Software Engineering, vol. 12, Issue: 2, Feb. 1986.

Martin, R.C., *Agile Software Development: Principles, Patterns, and Practices*, Prentice Hall 2003.

Sinnott, R.O., Galang, G., Tomko, M., Stimson, R., *Towards an e-Infrastructure for Urban Research Across Australia*, IEEE e-Science Conference, Stockholm, Sweden, December 2011.

Schwaber, K., *Agile Project Management with Scrum*, Microsoft Publishing, 2009.

Javadi, B., Tomko, M., Sinnott, R.O., *Decentralized Orchestration of Data-centric Workflows Using the Object Modeling System*, 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid 2012), Ottawa, Canada, May 2012.

Sinnott, R.O., Bayliss, C., Galang, G., Greenwood, P., Koetsier, G., Mannix, D., Morandini, L., Nino-Ruiz, M., Pettit, C., Tomko, M., Sarwar, M., Stimson, R., Voorsluys, W., Widjaja, I., *A Data-driven Urban Research Environment for Australia*, IEEE e-Science Conference, Chicago USA, October 2012.

Tomko, M., Sinnott, R.O., Bayliss, C., Galang, G., Greenwood, P., Koetsier, G., Mannix, D., Morandini, L., Nino-Ruiz, M., Pettit, C., Sarwar, M., Stimson, R., Voorsluys, W., Widjaja, I., *The Design of a Flexible Web-based Analytical Platform for Urban Research – Systems Paper*, ACM International Conference on Advances in Geographic Information Systems (ACM SIGSPATIAL GIS 2012), Redondo Beach, USA, November 2012.