LAND AND WATER
www.csiro.au

# Review of The Invisible Modelling Environment (TIME)

Ross Searle and Dave Penton

CSIRO Land and Water Science Report 4/12
15 December 2012

CSIRO Land and Water Science Report series ISSN: 1834-6618

## Citation

Searle R. and Penton D. (2012) Review of the Invisible Modelling Environment. CSIRO, Australia.

## Copyright and disclaimer



## Important disclaimer

# Contents

# Figures



# Tables

# Acknowledgments

As this review is CSIRO focussed, the majority of input gained throughout the review was sourced from CSIRO staff, with additional inputs from a limited group of key external stakeholders. The main mechanism for gathering input to the review was a workshop held on the 16th of May 2012 at CSIRO, Canberra, bringing together the key CSIRO technical staff involved in TIME development as well as key eWater staff. Attendees at this workshop included Dave Penton, Jean-Michel Perraud, Tim Smith, Dominic Snowden, Andrew Freebairn, Ben Leighton, Robert Bridgart, Nick Murray, Matt Stenson, Dave Penton, Ramneek Singh and Ross Searle from CSIRO and Joel Rahman and Geoff Davis representing eWater.

The authors wish to specifically acknowledge the assistance of Geoff Davis (eWater) for his invaluable assistance in providing detailed information to the review.

# Executive summary

The Invisible Modelling Environment (TIME) is a model development framework, supporting model developers in the creation and testing of algorithms and in the development of standalone modelling applications (Rahman *et al.*, 2005). TIME provides a framework for spatial and temporal data analysis. TIME has been developed under the sponsorship of eWater CRC and its predecessor CRC for Catchment Hydrology. As well as providing a modelling framework, TIME supports model users with a range of data analysis and management, model processing and visualisation tools. TIME comprises a collection of .NET classes, libraries and visualisation components for developing models and applications.

With the completion of the eWater CRC and transition to eWater Limited in July 2012, it is opportune for CSIRO to strategically review and assess its position in relation to the future development of TIME going forward. The purpose of this review is to consider the current technical capacity of the code base, how short term transitional arrangements will be implemented and potential future directions that CSIRO may wish to pursue with TIME. This review is intentionally internally focussed on CSIRO. We acknowledge the broader TIME development community and the broad range of applications built upon TIME, however the focus of this review is CSIRO's use of and future involvement in the development of TIME.

CSIRO has invested heavily in the development of TIME and has provided much of the intellectual input as well as the majority of the coding effort. CSIRO is actively involved in the continued development of TIME and supports the development and maintenance systems.

At present, the TIME code base (as defined in section 3.1) contains approximately 360,000 lines of code. There are 2040 individual classes (a grouping of lines of code that perform a specific function), and 16968 methods (code that performs actions). The TIME code base currently has 153 registered users. Of these 57 are considered to be active. All these users are based in Australia. These users are distributed over 31 individual organisations (Table 2) across 5 States. There are 84 federal agency, 27 university, 22 state agency and 19 private sector registered users.

TIME is a core component of a large range of formal software products, as well as numerous research programs. Currently the TIME code base is incorporated into approximately 23 eWater supported applications. These products have currently been downloaded from the eWater Toolkit web site a total of 38,360 times. The "eWater Source" products are a major focus of this product range and are extensively used by government and private enterprise in addressing water quantity and quality policy questions. TIME is also integrated into a range of CSIRO specific products and projects such as the "Hydrologists Workbench" and the Australian Water Resources Assessment (AWRA) system, a continental water balance monitoring system that is being developed jointly by CSIRO and the Bureau of Meteorology.

A review of the strengths and weaknesses of TIME was conducted and this was used to guide suggestions for improvements to the existing code base. A variety of research and development activities to further enhance the functionality of TIME is also presented. Most if not all of these suggested activities will require significant investment and commitment. With this investment, the TIME environment would better support ongoing science in a broader suite of domains (e.g. soils, agriculture and environment).

It is recommended that CSIRO should remain as an active participant in the further development of the TIME code base. Given the significant investment and the broad range of internal and external dependencies on TIME, it would not be prudent to abandon the development and maintenance of the TIME code base. A formal roadmap for the development of TIME should be established. It is also recommended that formal governance arrangements should be put in place to guide future development and maintenance of the TIME code base.

# 1 Introduction and Purpose of Review

The Invisible Modelling Environment (TIME) is an environmental modelling software framework (Rahman *et al.*, 2003) developed through collaborative activities under the CRC for Catchment Hydrology and its successor eWater CRC. CSIRO has been heavily involved in the TIME modelling framework development since its inception in 2003. CSIRO has directly contributed much of the scientific knowledge and code development that has gone into the TIME code base.

With the completion of the eWater CRC and transition to eWater Limited in July 2012, it is opportune for CSIRO to strategically review and assess its position in relation to the future development of TIME going forward.

The purpose of this review is to consider the current technical capacity of the code base, how short term transitional arrangements will be implemented and potential future directions that CSIRO may wish to pursue with TIME. This review is intentionally internally focussed on CSIRO. We acknowledge the broader TIME development community and the broad range of applications built upon TIME, however the focus of this review is CSIRO's use of and future involvement in TIME.

TIME is core to numerous existing software applications and scientific projects within Australia. At the time of initial development, TIME enabled a major leap forward in model framework development and consequently, how modelling applications in the hydrology domain were developed. In the ensuing years, much progress in this domain has occurred around the world. Given the ubiquitous nature of the modelling issues TIME addresses, many other groups worldwide have also implemented solutions to various components of the modelling framework paradigm.

From CSIROs perspective, it is important to understand how TIME fits into the current broader modelling framework domain and assess the relevance of TIME. This review does not attempt to conduct a comprehensive analysis of all existing environmental modelling software frameworks, but rather to assess the functionality of TIME in regard to similar efforts in this domain.

# 2 History of TIME

With the increased utility of personal computing resources during the 1980s opportunities arose to undertake problem analysis through the development of purpose built computer models. The hydrology domain was a particular area of focus for such activity. The nature of problems addressed by these purpose built models was diverse and ranged across scales both temporally and spatially including discrete events, long term averages, daily and seasonal dynamics, point and spatial estimates and total catchment outputs(Rahman *et al.*, 2003). Each of these models tended to written from scratch and often developed unique solutions to common functional requirements. They were written in a vast range of programming languages, computer platforms and employed problem specific design approaches

Scientists within the Cooperative Research Centre for Catchment Hydrology and CSIRO identified this divergence of approaches and duplication of effort and set about exploring solutions to this issue. Domain centric software modelling framework development was at the time an active area of research and development across a number of disciplines.

In 2002 a comprehensive review of existing purpose built hydrologic models and frameworks was undertaken by the CRC for Catchment Hydrology. As well as this, a detailed user requirement survey was carried out within the Australian hydrologic modelling community(Marston *et al.*, 2002). As part of this process, several frameworks were evaluated and tested. None of the existing frameworks was deemed suitable to meet the requirements determined via the review process. With advances in software development technologies, an opportunity arose that made it feasible to develop a new framework that better suited the needs of environmental modellers within and beyond the CRCCH (Rahman *et al.*, 2003)

TIME was developed to support several key stages of model development. TIME was designed to support the development of new model components, using one of a number of languages, along with the testing of those model components in a generic test-bed, providing a high level of data handling, analysis and visualisation. TIME attempted to supports the integration of modules into applications with highly customised, visually rich user interfaces, utilising a number of re-useable components for data handling and visualisation (Rahman *et al.*, 2003).

Over the ensuing years TIME evolved to support the requirements of a broad range of applications. Both the CRC for Catchment Hydrology and eWater CRC had a focus on the development of software tools to support researchers and land managers in decision making. These products make up what is now known as the eWater Toolkit ( http://www.toolkit.net.au/ ). Functionality tended to be added to TIME as the need arose supporting an ever expanding user base of these tools. Today, TIME is a core component of 23 formal software products, as well as numerous research programs.

The vast majority of the TIME software development has been undertaken by CSIRO software developers within the Land and Water Division. As well as developing the core TIME modelling framework many of these developers have also been involved with the coding of various toolkit products. These developers and associated staff have also been responsible for training a broad range of users in TIME development methods and processes.

At the time of its inception, there was no other modelling framework readily available that included the suite of capabilities, functionality and design principles that were encompassed by TIME.

# 3 What is TIME

The Invisible Modelling Environment (TIME) is a .NET based model development framework, supporting model developers in the creation and testing of algorithms and in the development of standalone modelling applications (Rahman *et al.*, 2005). TIME provides a software framework for spatial and temporal data analysis.

TIME automates most of the repetitive tasks of model development, including the automatic generation of user interfaces, the management and analysis of time series data and the handling and visualisation of temporal and spatial data. The TIME software framework is designed to allow model developers to concentrate on the development and application of models, without having to invest heavily in developing code to handle the more routine tasks of data IO and visualisation.

Environmental modelling frameworks support scientific model development by providing model developers with domain specific software libraries which are used to aid model implementation (Lloyd *et al.*, 2011).

Frameworks for environmental modelling typically support rapid model development and integration by generic, re-useable components for data handling and visualisation. Environmental modelling frameworks range from high level abstract architectures with little or no domain dependence such as Common Component Architecture (Armstrong *et al.*, 1999) and OpenMI (Gregersen *et al.*, 2007) through to more invasive, domain specific frameworks such as Tarsier (Watson and Rahman, 2004). TIME attempts to strike a balance between providing a range of concrete methods and tools to ease model development while being implemented in framework not specifically tied to an individual scientific domain.

At the highest level of abstraction, TIME is represented as a layered system (Figure 1), with components in each layer interacting with the layers below it. Each layer contains a family of components and, in some cases, small frameworks supporting a specific aspect of model development.

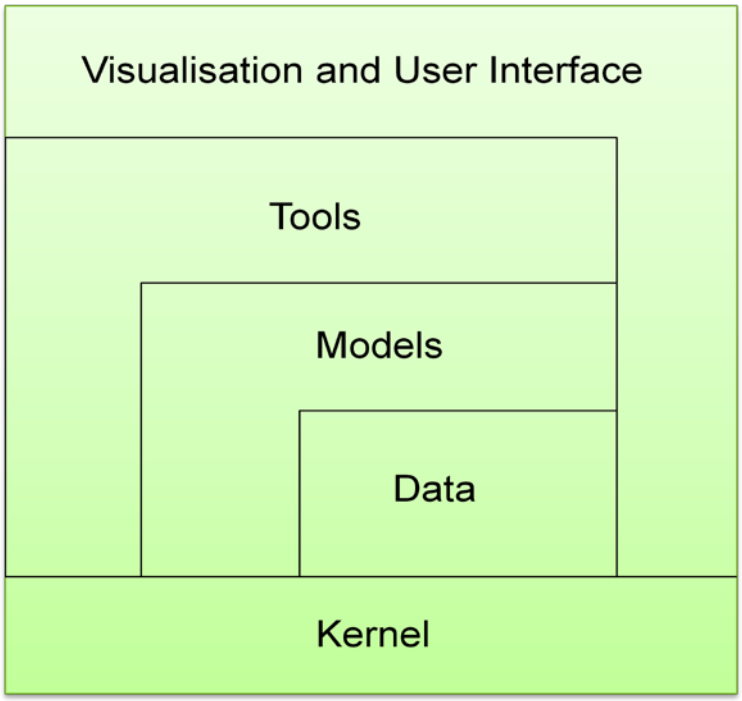


**Figure 1. Main Architectural Layers of TIME (Rahman et.al., 2003)**

When it was initially developed, TIME differed from most existing modelling frameworks in a number of ways, particularly in its use of metadata to describe and manage models as well as the flexibility given to model developers to 'pick and choose' the components of TIME relevant for a given project. Functionality that is embedded as an immutable 'core' layer in other frameworks is included in applications under TIME on an as-needed basis using optional, interchangeable components. This flexibility extends to components that manage data and models, recognising that one approach does not necessarily fit all applications (Rahman, et al., 2003).

TIME makes use of the metadata capabilities of .NET to automate several tasks, such as user interface generation, that are not possible with many modelling frameworks based on commercial development tools. By automating these common tasks, the model code does not become directly coupled to the implementation of those tasks, relieving model developers from code maintenance tasks stemming from framework evolution (Rahman, et al., 2003).

TIME attempts to find a middle ground, by using a commercial development platform (.NET) that is easily accessible to users. .NET allows the elementary integration of components written in different languages, including Visual Basic, Fortran and C++ (Meyer, 2001).

TIME supports the deployment of models as graphical applications, command line applications and active web pages.

## 3.1 Defining TIME in Terms of the Code Base

For the purpose of this review and to avoid confusion on the exact scope of discussion, it is useful to explicitly define what parts of the full eWater code base are considered to encapsulate the TIME code. Currently the TIME code base is managed as part of a much larger code base encompassing a much broader scope of activity around the whole suite of eWater Toolkit products. The larger code base contains products such as eWater Source and specific applications contributed by the various eWater partners. Currently there are approximately 28 products in the broader managed code base. Most if not all of these products have dependencies on the TIME code base.

The TIME code base has been developed in parallel with this range of eWater products, hence the exact boundary of exactly what should be included in the TIME code base can be blurred. While there is no definitive answer as to the exact lines of codes that make up the TIME code base we can make a reasonably precise attempt to define it in terms of C# projects contained in the code repository. The projects that essentially make up TIME have been agreed to by participants in the review as those that are listed in Table 1.

**Table 1. List of C# projects that make up the TIME code base**

| | |
|---|---|
| TIME | Tools.GRID |
| Amnesia | Tools.Visual |
| DataAnalysis | Visualisation |
| NetCDF | Winforms |
| ScenarioManagement | WebFormComponents |
| Testing.UnitTests | VisualTime |
| TIME.Tools | TIME.NetLP |

## 3.2 Functionality

As well as providing a modelling framework, TIME supports model users with a range of data analysis and management, model processing and visualisation tools. TIME comprises a collection of .NET classes, libraries and visualisation components for developing models and applications (Murray *et al.*, 2006). In computer programming, a class is a grouping of lines of code that perform a specific function within the program.

TIME has native support for roughly 70 time series, image and GIS file formats for manipulating data without needing to write conversion routines. TIME also has a range of visualisation controls that support display and interaction with layers, graphics, and parameter manipulation. The TIME libraries also contain mathematical and statistical routines.

The functional components of TIME can be broken down into the following broad groups of classes:-

- **Core** – This is the main functional component of the TIME code base. It is the “Kernel” part depicted in Figure 1. Core is essentially the model running, unit handling and modelling framework components.
- **Applications** – command line and Windows forms based GUI for access to some TIME functions.
- **Data Analysis** – temporal and spatial data analysis routines. Raster and vector analysis routines, terrain and hydrological analysis, data conversion routines, time-series analysis routines.
- **Data Types** – spatial and temporal data IO handling. TIME supports a broad range of spatial and temporal data types.

- **Scenario Management** – framework for representing different model realisations.
- **Science** – analysis routines for linear algebra, hydrology, mathematical solvers, probability, uncertainty and statistics.
- **Tools** – a diverse range of ancillary model tools and utilities.
- **Tests** – unit tests covering the TIME code base.
- **Visualisation** – functionality for viewing spatial and temporal data sets.
- **Webforms** – prototype methods for enabling TIME on the web.
- **WinForms** – suite of re-usable windows forms components to aid in typical modelling applications.

A more detailed listing of the specific functionality within TIME is given in Appendix 1 : TIME Functionality Table. For a detailed description of the complete functionality and use of TIME please refer to the TIME Reference Manual (Murray *et al.*, 2006).

A breakdown of the relative size of each of functional group in terms of coding effort (percentage of total lines of code) is given in Figure 2.

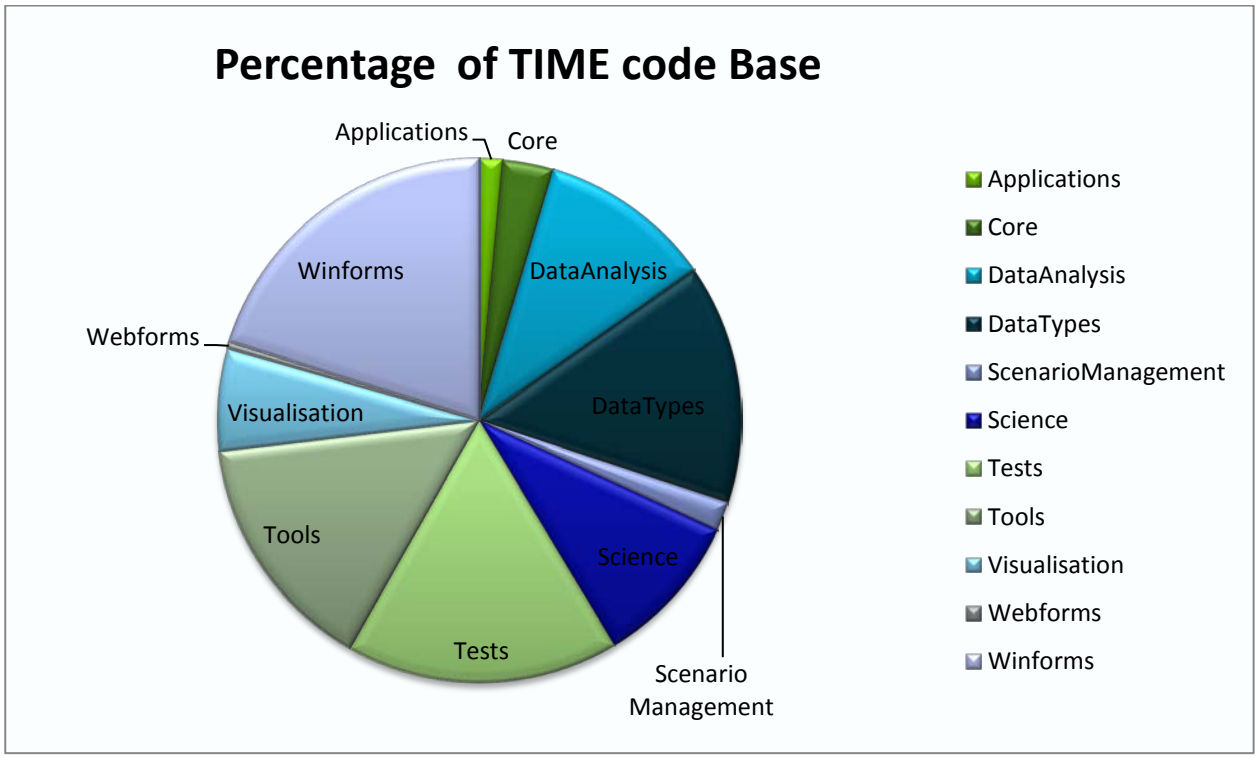


**Figure 2. Relative size of the various functional components (percentage of total lines of code) within the TIME code base.**

## 3.3 TIME Context

At present the TIME code base contains approximately 360,000 lines of code. There are 2040 individual classes, and 16968 public methods and properties.

If we use some back of the envelope calculations we can get a very rough idea of the size of the financial investment in the development of the TIME code base. If we consider that the development of the TIME

code base occurred over an 8 year period and multiply this by some rough staff costings, we can estimate that at least $5 million has been invested in the development of TIME. These calculations do not take in to account ongoing management and maintenance and are based on some very loose assumptions but are intended to only give a ballpark figure. This figure presented is most likely at the lower end based on conservative cost assumptions used.

The TIME code base currently has 153 registered users. Of these 57 are considered to be active users (Davis and Penton, personal communication). This is a similar number reported by Rahman et.al. in 2005. All these users are based in Australia. These users are distributed over 31 individual organisations (Table 2) across 5 states. There are 84 federal agency, 27 university, 22 state agency and 19 private sector registered users.

**Table 2. Number of TIME users per organisation**

| Organisation | No. of Users | Organisation | No. of Users |
|---|---|---|---|
| CSIRO | 71 | Catchment Simulation Solutions | 1 |
| UC | 14 | United Water International Pty Ltd | 1 |
| BMT WBM | 10 | Department of Water SA | 1 |
| QDERM | 8 | DNR NSW | 1 |
| Melbourne Uni | 7 | Griffith University | 1 |
| SKM | 6 | EPA SA | 1 |
| Bureau of Meteorology | 5 | Flowmatters | 1 |
| eWater | 5 | G-M Water | 1 |
| MDBA | 3 | ANU | 1 |
| QEPA | 3 | SARDI (SA) | 1 |
| WA Water | 2 | Melbourne Water | 1 |
| Uni Newcastle | 1 | Monash Uni | 1 |
| Australian Rivers Institute, Griffith University | 1 | NSW Office Water | 1 |
| UQ | 1 | SA Department for Water | 1 |
| University of Adelaide | 1 | DPI | 1 |

**Table 3. Number of TIME users per Australian political jurisdiction**

| Jurisdiction | No. of Users |
|---|---|
| Federal | 113 |
| QLD | 14 |
| VIC | 11 |
| SA | 5 |
| ACT | 5 |
| NSW | 3 |
| WA | 2 |

TIME is a core component of a large range of formal software products, as well as numerous research programs. Currently the TIME code base is incorporated into approximately 23 eWater supported applications. These products have currently been downloaded from the eWater Toolkit web site a total of 38,360 times. The “eWater Source” range of products are a major focus of this product range and are extensively used by government and private enterprise in addressing water quantity and quality policy questions. TIME is also integrated into a range of CSIRO specific products and projects such as the Australian Water Resources Assessment (AWRA) system, a continental water balance monitoring system that is being developed jointly by CSIRO and the Bureau of Meteorology.

# 4 Intellectual Property Rights Considerations

The TIME code base since its inception has been developed in a collaborative environment under the eWater CRC and its predecessor the CRC for Catchment hydrology. As a result of these collaborative arrangements the intellectual property (IP) of the code base has always been held by the CRCs in a shared rights arrangement.

The CRC for Catchment Hydrology (CRCCH) held the IP in trust from the CRCCH partners. With the conclusion of the CRCH in 2005 all IP went into a trust, which is now administered as background IP on behalf of CRCCH partners by eWater Ltd. The IP transferred was formally captured in an IP register including the Catchment Modelling Toolkit and associated models as well as the source code for E2 and other models that resided on CSIRO's subversion system. Much of this IP exists as code, which resides within the TIME repository (developed as part of the CRCCH). With the establishment of eWater CRC in 2005, a separate IP register in the trust was created to hold eWater CRC IP created from 2005-2012. This is also administered on behalf of eWater CRC partners by eWater Ltd. Since 2005, the TIME codebase has continued to grow as a function of CSIRO/eWater activities. However, some (small) additional code has also been added to TIME, through development that occurred outside the agreed CSIRO/eWater activities.

An eWater CRC IP register has been developed and continuously been updated by the eWater CRC to date, CSIRO staff have been deeply involved in the formulation of the eWater CRC IP register. There are few impediments to the use of IP as at 30 June 2012 as per the eWater participants agreement (agreement terms 22 and 23). The eWater IP as of 30 June 2012 belongs to eWater partners including CSIRO and all commercial rights will remain with eWater. All eWater CRC partners have access to the IP in trust generated to June 2012. CSIRO remains free to utilise this IP as part of its ongoing/usual business activities.

Moving forward from 1/7/2012, eWater intends to operate as a not for profit company in a partnership model, whereby the partners are given access to the core model code (and IP). It is intended for CSIRO and eWater to move ahead in a special partnering arrangement, where CSIRO is the 'preferred' development partner for substantial model development. The form and content of this is being drafted in a "Letter of Agreement" between the two organisations. This agreement confirms CSIRO's ability to take the existing software and develop it further, both independently and with eWater, and at the same time exercise full usage of the software in its usual business practices. It is also understood that neither party wants to have independent model platforms developed, and the anticipated situation is for development to be linked and co-ordinated between them.

# 5 Systems Infrastructure

The current TIME coding environment utilises a suite of commercial and freeware software platforms to enable code development workflows. These systems form the basis of the workflows developed around TIME and the broader "eWater Source" coding. The use of this software suite enables quality control in a distributed development environment. Previous to July 2012 these systems are all run on CSIRO hardware.

Components of the development environment include :

**Visual Studio** http://www.microsoft.com/visualstudio/en-us

Microsoft Visual Studio is the Integrated Development Environment (IDE) utilised by coders contributing to the TIME code base. Its supports a number of different development languages including C#, VB.NET, F#, Python and C++.

**Subversion** http://subversion.apache.org/

Subversion is an open source version control system developed by the Apache Software Foundation. Subversion centrally manages the versioning of the code base developed in a distributed environment. Currently the source code repository manages 4.3 million lines of code. Both CSIRO internal developers and external developers contribute to this managed code base. The repository is backed up daily on a CSIRO server but in effect the source code is backed up on every user's local machine.

**Jira** http://www.atlassian.com/software/jira/overview

Jira is a web based issue tracking software used for all major software development projects in eWater/CSIRO collaborations. Used as a management tool to assign work tasks and track the progress of tasks. Jira currently hosts 50 individual projects, however not all are currently active.

**Confluence** http://www.atlassian.com/software/confluence/overview

Confluence is the wiki used by the developers. It allows easy communication of information amongst the developers.

**TeamCity** http://www.jetbrains.com/teamcity/

TeamCity is a continuous integration server and distributed build management tool. When a change is made to any code e.g. a model in RiverManager is modified, TeamCity recognises this and then sends a job off to compile the code, run unit tests and regression tests to make sure the change doesn't change anything unexpected and the code still builds. When this system detects problems the appropriate developers are automatically alerted.

**Crucible** –http://www.atlassian.com/software/crucible/overview

Crucible is a tool that assists developers in conducting code reviews.

Table 4 summarises the cost of each of the components of the current development infrastructure.

**Table 4. Systems infrastructure summary**

| Software | Provider | Cost | Location |
|---|---|---|---|
| **Visual Studio** | Microsoft | $481 per user (approx 20 users) | Individual PCs |
| **Subversion** | Apache Software Foundation | freeware | https://subversion.toolkit.net.au/svn |
| **Jira** | Atlassian | $4000 (51-100 users) | http://jira.toolkit.net.au |
| **Confluence** | Atlassian | $1600 (26-50 users) | http://confluence.toolkit.net.au |
| **TeamCity** | Jet Brains | $3000 | http://teamcity.toolkit.net.au |
| **Crucible** | Atlassian | $1200 (11-25 users) | |

## 5.1 Arrangements Post June 2012

With the establishment of eWater Limited after June 2012, a number of changes will occur to the setup and location of the various systems. The system setup after June 2012 is summarised in Table 5. eWater will take responsibility for a number of the existing systems, while CSIRO will have to continue supporting a number of the existing systems. A more detailed explanation of the new system setup can be found at http://confluence.toolkit.net.au/display/PRT/Handover+Process.

**Table 5. Planned systems infrastructure configuration post June 2012**

| Software | Post 2012 |
|---|---|
| **Visual Studio** | Same |
| **Subversion** | eWater will host via an external service provider. Alternative source version control software is currently be investigated by eWater staff. Depending on the outcomes of this, the Subversion repository may be moved to a different software platform. |
| **Jira** | eWater will extract their required projects and transfer the remaining projects to IM&T supported Jira at zero cost. |
| **Confluence** | eWater will host all eWater relevant projects, and the remainder will be transitioned to IM&T supported instance at zero cost. |
| **TeamCity** | eWater have purchased a new server license and 9 agent licenses. CSIRO have purchased a new server license and 12 agents. |
| **Crucible** | eWater will not extract anything from crucible. Transitioning to IM&T systems at zero cost. |

# 6 Governance Framework

Throughout the development of TIME there has been no formal governance structure in place to provide strategic guidance. At the strategic level, apart from adhering to the original conceptual framework and design principles, the ongoing development of TIME has tended to be guided by application needs and direct user feedback. Future strategic development needs have tended not to be addressed in a structured fashion.

A relatively strong governance of TIME has tended to occur at an operational level. A group of 3 or 4 senior developers provide oversight regarding major changes and additions to the code base. Once again there is no formal governance structure, but rather an informal arrangement, where these senior developers are consulted by others when significant alterations are being considered.

The day to day governance of changes to the TIME code base are facilitated by the systems architecture as described in section 5. The version control system has defined levels of access control that can be assigned to all developers. By default all new users are only given read permission to the code base. There has been a strong ongoing commitment to providing an appropriate level of training to all new users so they are able to quickly learn the concepts and conventions used by the TIME development team. As users become more skilled in the coding of TIME they can be granted read/write permission if deemed appropriate by the senior developers. Developers with read/write permission to the repository are termed “trusted” developers. These trusted developers are free to make additions and changes to the code base as they deem appropriate.

The code base is covered by a comprehensive set of unit test that are run whenever code changes are committed to the repository. These tests automatically flag when problems arise and developers are notified of these problems. This enables quick resolution of potential issues.

There is also a process of code review that is undertaken within the development team. The code review process allows an opportunity for developers within the team to receive feedback on the quality of their work from within the team. It is also a good way of sharing information and knowledge within the team. Over the development period of TIME, the application of this code review process has been variable.

As a consequence of this training, and “trusted” developer paradigm, the coding development tends to be managed by exception. This approach to development leads to a flexible and more agile development process, where it is more efficient to deal with exceptions as they arise rather than forcing a rigid development procedure on developers.

It is recognised by the developers that these governance arrangements are typically most successful when applied to relatively small teams working across a relatively small number of individual projects. Application of these same governance approaches to larger development environments can be more challenging and typically requires concerted effort to make them effective.

# 7 Critical Dependencies

There have been no critical dependencies identified for TIME, however they are some strong dependencies that may be worth reflecting on.

The large majority of the TIME code base is written in the C# programming language. Whilst C# is an industry standard language at the present, programming languages and software development concepts are rarely static. The Windows environment is about to undergo some significant alterations and the impacts of these on programming paradigms is as yet unclear. Early public announcements around the release of the latest version of the Windows OS, suggested a reduced level of support around C#.Net into the future, however this scenario now seems increasingly unlikely.

C# is a proprietary programming language of Microsoft. Thus, in theory the future of the C# programming language is mostly controlled by this one private company. That said, there is an open source alternative to C# called Mono. Whilst TIME is not currently fully compatible with Mono, it would not be a major effort to make TIME fully Mono compatible. This may result in some loss of functionality relying on third party components within TIME. Nevertheless, some recent significant TIME-related developments, like the AWRA-L data assimilation, have been run on Mono and Linux, without significant loss of functionality.

Most of the software development team responsible for the coding of TIME are part of the Environmental Information Systems (EIS) group, within the CSIRO Division of Land & Water. Thus there is a strong ownership and knowledge of TIME amongst a relatively confined group. In large organisations such as CSIRO reorganisations of work groups over time is the norm. If the EIS groups focus or function was to be diminished, one strong driver for the development of TIME would be reduced. Adverse impacts of dependency are somewhat mitigated by the presence of strong partner organisations also involved to varying degrees in the development of TIME. eWater Limited in particular has just recently acquired significant intellectual capacity and capability in relation to the TIME code base.

# 8 Code Base Quality and Health

The TIME code base has evolved over a period of approximately 10 years. During this time, over 50 developers from a range of organisations in numerous geographic locations working on a large number of disparate projects have contributed to the TIME code base. Such a broad collaborative effort will always put pressure on the optimal management of a code base. To deal with this, the TIME development team have put in place a variety of systems and procedures to ensure the best possible management of the code base given the inherent limitations of the environment in which the team work.

There are a range of automated code assessment tools which are run at regular intervals across the code base to analyse the code base health. Key to testing the day to day health of the code base is the TeamCity continuous build environment. As code changes are committed to the code base, test builds are fired within the TeamCity environment to ensure these changes have not had a deleterious impact. If problems are found, appropriate people are automatically notified and fixes are implemented.

As well as the continuous build environment there are analyses that are run across the code base to test the health and quality of the code base. Table 6 gives an example of the type of output that can be generated from the code metrics analysis to assist in pinpointing areas of the code base that require work to optimise. To fully utilise the value of these analyses one must drill down into the lower levels of the code base. The summary table presented is merely an example and does not reflect the understanding to be gained by fully exploring and interpreting this analysis. Whilst one can not necessarily ascribe a definitive health rating using these code metrics, they provide developers with powerful analysis to target code improvement.

**Table 6. Example of the Code Metrics analysis of the TIME code base (green – good, yellow – acceptable, red - poor)**

| Assembly | Maintainability Index | Cyclomatic Complexity | Class Coupling | Depth Of Inheritance | Lines Of Code |
|---|---|---|---|---|---|
| TIME.dll | 80 | 95 | 24 | 7 | 810 |
| TIME.DataAnalysis.dll | 70 | 42 | 35 | 7 | 86 |
| TIME.Tools.dll | 84 | 27 | 17 | 4 | 235 |
| TIME.Tools.Visual.dll | 73 | 66 | 31 | 10 | 265 |
| TIME.Visualisation.dll | 80 | 28 | 19 | 7 | 83 |
| TIME.Winforms.dll | 79 | 25 | 23 | 9 | 123 |

The Microsoft Developer Network documentation defines the metrics as below (Microsoft, 2012) :

**Maintainability Index** – Calculates an index value between 0 and 100 that represents the relative ease of maintaining the code. A high value means better maintainability.

**Cyclomatic Complexity** – Measures the structural complexity of the code. It is created by calculating the number of different code paths in the flow of the program. A program that has complex control flow will require more tests to achieve good code coverage and will be less maintainable. Less can be better.

**Class Coupling** – Measures the coupling to unique classes through parameters, local variables, etc. Good software design dictates that types and methods should have high cohesion and low coupling. High coupling indicates a design that is difficult to reuse and maintain because of its many interdependencies on other types. Less is better.

Depth of Inheritance – Indicates the number of class definitions that extend to the root of the class hierarchy. The deeper the hierarchy the more difficult it might be to understand where particular methods and fields are defined or/and redefined. High is not good.

**Lines of Code** – Indicates the approximate number of lines in the code. A very high count might indicate that a type or method is trying to do too much work and should be split up. It might also indicate that the type or method might be hard to maintain. Lower can be better.

As well as assessing the quality of the code it is important to quantify the level of automated test coverage over the code base. Table 7 presents a summary of the test coverage over the TIME code base. There appears to be relatively good test coverage over most of the code base with room for improvement in some of the visualisation classes. By nature these style of classes van be difficult to create useful automated testing for, due to their interactive nature. It should also be noted that quality of the tests is an important consideration. A hundred percent coverage of poor quality tests is not useful.

**Table 7. Test coverage as percentages over the TIME code base**

| Assembly | Class, % | Method, % | Statements, % |
|---|---|---|---|
| **TIME** | 76.1% (535/ 703) | 56.1% (3847/ 6858) | 52.3% (34316/ 65572) |
| **TIME.DataAnalysis** | 30.2% (42/ 139) | 27.8% (247/ 890) | 31.8% (4749/ 14946) |
| **TIME.Tools** | 63.4% (161/ 254) | 53.3% (1222/ 2293) | 51.6% (8239/ 15960) |
| **TIME.Tools.Visual** | 0% (0/ 80) | 0% (0/ 1061) | 0% (0/ 13361) |
| **TIME.Visualisation** | 28.3% (36/ 127) | 14.5% (207/ 1429) | 14.5% (1410/ 9755) |
| **TIME.Winforms** | 4.1% (12/ 290) | 3% (105/ 3459) | 3.1% (962/ 31279) |

Just as important as reviewing statistical measures of code health, it is also wise to apply a common sense review of the code base. During the developer workshop held in Canberra, a brief analysis of the structure of the code base was conducted to determine the scope of the TIME code base being considered in this review (Section 3.1). It was considered that there would be considerable benefit to be gained from a small scale tidy up of the structure of the code base. Despite the best of efforts, some classes have been located in illogical locations in terms of the overall solution structure. There are also obsolete and superfluous classes that could be removed from the existing code base. This was not considered to be a task that should be too onerous and with all of the organisational changes taking place it was thought that this would be a worthwhile undertaking.

Another common measure often considered when reviewing code health is the level of documentation associated with a code base. TIME has a User Reference Manual (Murray *et al.*, 2006) and a set of TIME Training Workshop Notes. Both of these documents are useful when starting out developing code in the TIME environment. There is also a document currently in development focusing on how to write a plug-in. There is also a reasonably comprehensive set of example code available for coding beginners. All of this existing documentation tends to be targeted at entry level coders.

What TIME is missing is a broader overview document explaining the concepts and functionality of TIME for less computer coding literate potential users. At present it is very difficult for a new comer to TIME to actually understand the full extent of what TIME is capable of being used for unless they delve deep in to the internals of the code base.

The actual TIME code base also tends to be poorly documented – although this can be variable. It is accepted that modern programming languages are meant to be self describing to a certain extent, however there needs to be more information in the code base about the code provenance and more descriptive explanations of functionality of classes. At present, there is a steep learning curve for developers once they

move past entry level and at present this tends to be supported by knowledge exchange between coders, not documentation contained within the code base.

# 9 SWOT Analysis of TIME

At the review workshop (May, 2012), an analysis of the strengths, weaknesses, opportunities and threats to the TIME code base was undertaken. The methodology of a formal SWOT analysis was only loosely applied but the SWOT categories provide a robust framework for structuring discussions. Presented below, in no particular order is a summary of the analysis.

## 9.1 Strengths

- The vast majority of the TIME code base is written in Microsoft C#. The .NET programming languages are considered to be industry standard languages amongst the IT community. Thus there is good access to state of the art cutting edge functionality and a large pool of developers in existence as well as a wide range of tools available.
- There is an open source alternative to Microsoft C#. Mono is an open source implementation of Microsoft's .NET Framework based on the ECMA standards for C# and the Common Language Runtime.
- C# has reasonable speed performance compared to other coding languages, even compared to natively compiled languages.
- C# utilises the “Just in Time” compilation paradigm which in theory gives it platform portability although currently there are limited realisations of this.
- It is relatively easy to obtain a minimal level of competence in programming within the TIME environment.
- The TIME code base architecture was developed with a strong focus on making it very simple to extend its functionality through the addition of modular components termed “Plug-ins”. Plug-ins provide a powerful and flexible way to utilise the functionality of TIME. This capacity makes TIME a highly extensible modeling and programming environment.
- Through the use of system reflection, TIME implements some powerful internal semantic features, which add to the flexibility and robustness of TIME as a modeling framework. The use of semantics is a highly desirable feature of modern modeling frameworks.
- TIME supports a broad range of both spatial and temporal data types from unique model specific formats through to industry standard formats. Once again the extensibility of TIME makes it relatively easy to add new data types as required. Currently TIME natively supports the input and output approximately 45 data formats.
- The current TIME coding environment and systems (Section 5) provide a robust development environment.
- TIME has 57 active developers. This makes for a relatively strong coding community with a degree of forward momentum. At present, this development community is centered on a relatively small domain area, and exists solely within Australia.
- The development of TIME has been driven by pragmatic requirements. Often the development of TIME has been resourced by particular short term project funding, thus there has been a strong emphasis on code focused on delivery of domain specific outcomes. This has lead to a range of tools being developed driven directly by user requirements.
- The adoption of the TIME framework has been supported and underpinned by a strong emphasis on delivering appropriate levels of training to potential users.
- New capability added often. With an active development community and an ever increasing user community there is always new functionality and tools being added to TIME.
- The basic modeling framework concepts encapsulated by TIME are relatively straight forward, thus are easy to pick up and apply.

- Some parts of the TIME code base have been thoroughly tested. Being incorporated into over 30 products that have been downloaded by 38,000 people and with over 50 active developers, the TIME code base has been comprehensively applied and tested in real world applications. There is also extensive unit test coverage of the code base itself to help identify problems as they arise.

## 9.2 Weaknesses

- Discoverability of the functionality of TIME is difficult. Unless you are a code developer it is very difficult to determine the exact capabilities of the TIME modeling framework. Even if you are code developer, it can still be difficult to ascertain the total of TIME functionality.
- Related to this is a lack of a readily useable front end to TIME. Currently there is no easy to use front end for non programmers to use to access the full range of TIME functionality. Over the years a number of products (Visual TIME, TIME Shell) have been developed to allow generic access to TIME functionality but these products were not widely utilised and have slipped from use.
- It is difficult to pin point a stable release of TIME. Due to the dynamic nature of the development environment change is continual and with no defined release cycle it can be very difficult to know exactly which version of TIME you are using.
- TIME is not very efficient at dealing with large datasets. Attempts have been made to introduce methods for efficiently handling large data sets but these are not always optimal and may not follow the most modern design principals and paradigms.
- TIME is largely not thread-safe. The basic framework architecture of TIME was developed before threading approaches became common. Hence there is only limited support to ensure thread safe execution.
- As with threading. TIME does not support parallel and or distributed processing very well. Once again, these computing approaches were not common when TIME was conceived and thus the basic framework does not have intrinsic support for these computing approaches. There is no simple “one size fits all” solution to parallelisation and it is unlikely that any sophisticated modeling framework can be made ubiquitously parallel processing capable.
- There has been limited or no real uptake of TIME by the broader scientific community. One of the original aspirations of TIME was to provide a modeling framework that was very flexible and extensible without the practitioner requiring high level programming skills. However, training attracted mostly software developers and subsequently most users of TIME typically have a strong coding background.
- C# does not tend to be the preferred programming language of most scientists. Languages with interactive capabilities such as Python and R are emerging as the preferred languages of the general scientific community. While an attempt was made to provide an interactive console in VisualTIME, and R and Python have been previously used in close conjunction with TIME, a lot of potential is unrealized.
- There is no fully implemented scripting language interface to TIME. A scripting interface call “Boo“ was partially implemented but has never been utilised to any great extent. Scripting is common in many scientific work flows.
- The TIME documentation is not complete. Although there are good training notes and a user guide there is no comprehensive piece of documentation that describes all of the functionality and concepts of TIME.
- Unit test coverage is not complete.
- TIME still contains calls to obsolete C# objects and data structures. While not a necessarily negative situation the more modern alternatives are considered to be better coding practice.
- Some parts of the TIME code base have Windows operating system dependencies. This reduces the deployment opportunities, thus reducing the potential user community.

- Provenance of the code base is poor: There is often very little description of the source and derivation of many of the algorithms. Components are not self-describing.
- There is limited support for self describing data sets such as NetCDF and no support for data transport architecture and protocols such as OPeNDAP (Cornillon *et al.*, 2003). Since the inception of TIME, the scientific and IM&T communities have driven the development of a range of efficient, self describing data types and data access methods. TIME has not necessarily kept up with these advances.
- There is no perceived long term "roadmap" or development plan for TIME. One internal developer has remarked during the review that he is specifically excluding the use of TIME from products he is involved in developing, that would be eminently amenable to the use of TIME, as he had no confidence in the long term support for the code base. This is not a desirable situation as it leads to inefficiencies, duplication and fragmentation of effort and resources.
- TIME is perceived to be by some practitioners, as domain specific i.e. hydrology/river systems
- The development of TIME has largely been sponsored by on major client group, that being eWater and its predecessors. Thus some opportunities to incorporate additions of functionality from Source Urban and other areas of closely related code development have been missed.
- The code base is susceptible to becoming "brittle". Due to the plug-in architecture it is not difficult for hidden interdependencies between Plug-ins and core components to be introduced.
- There are very few publications about the TIME modeling framework in the literature. This means that the use of TIME has been restricted to users within Australia where interpersonal networks have prevailed to promote awareness.
- There is no external presence for TIME such as a web site. If you are not a partner in the existing eWater organisation you have no access to the code base.
- The current development and management of the TIME code base is heavily focused on the Environmental Information Systems (EIS) group within CSIRO Land and Water. Such a defined locus of development for TIME heightens potential negative consequences of organisational change.
- Licensing and IP "encumbrance" may be limiting the potential uptake of TIME. Anecdotal evidence was provided by review participants of instances where TIME had been passed over for use in scientific projects, as the current licensing, IP and commercial model is too restrictive.

## 9.3 Opportunities

- Open-Sourcing of the TIME code base would potentially expand the user base. Expanding the user base may also broaden the developer base available to work on TIME. This would have the benefit of reducing the reliance of TIME on the small developer base. Open sourcing would give access to a much broader range of scientific disciplines. Broader adoption across a range of disciplines could lead to increased and improved functionality. The code base is not currently amenable to broadly available open sourcing as is, and would need some additional resourcing to realise this opportunity. Open sourcing would also create an overhead in the management and communication within the broader open source development community, but the potential benefits could be significant. Changing TIME to an open source license without the overheads of making broadly accessible could also realise some benefits in terms of ease of collaboration.
- Tidy up and promote the use of Visual TIME and TIME Shell as a means to accessing the functionality of the TIME code base. Linking with scripting languages or incorporating TIME within the CSIRO's "WorkBench" software may provide may provide alternative means to the same end.
- Enhance and build on the existing documentation and make this more publicly available to broaden the knowledge and understanding of the functionality in TIME within the scientific community.

- TIME has a relatively large existing user base across a diverse group of organisations within the Australian hydrology domain, a number of who have significant influence on the future directions of hydrologic modeling within Australia.
- The TIME development team has considerable knowledge and skills in scientific computing, data management best practice and development environments and systems. There are numerous groups external to the EIS group within CSIRO that could benefit significantly from these skills and knowledge.
- There are a number of groups within CSIRO that specialize in dealing with large data sets. TIME developers could leverage off these groups to enhance large data handling within TIME.
- There is a large base of scientific modelers utilising the Linux computing platform. With little effort TIME can be made to run under Linux using the Mono compiler. This would give access to TIME functionality to a much broader scientific user base.
- Promote the continued and expanded integration of TIME into scientific workflows used within CSIRO and partner organisations
- A number of the scientific computing languages that are currently popular amongst the scientific community, such as R and Python, are highly extensible and come with extensive packages of functionality. Improving the interoperability/availability of TIME features through these languages could be a very valuable path to adoption.
- There is a large scientific modeling community out there totally unaware of the capabilities of TIME. Through increased levels of scientific publications describing the uses to which TIME is being applied, a much broader potential user community can be informed of the existence of TIME.
- There is very little brand recognition of the TIME name. The main user group who affiliate with the TIME name is the relatively small group of coders. Most users of TIME functionality are exposed to it through applications built with TIME. Thus there is not broad awareness of a software product called TIME. This presents an opportunity to consider re-branding TIME with a more descriptive name that may improve identifiability and discoverability.

## 9.4 Threats

- The intellectual property rights for TIME are held in trust by eWater Limited on behalf of its partner organisations. This arrangement as it currently exists is satisfies CSIRO's requirements going forward, but given that CSIRO is not ultimately in control of the fate of eWater there is potential that these IP arrangements may change in the future.
- The development of the TIME code base has largely been driven by eWater Limited and its predecessor CRC's. eWater and its partners currently play a major role in providing the necessary support for the development of TIME and setting the directions for the development of TIME. Until recently, eWater provided financial support to the majority of developers working on TIME development. The creation of eWater Limited is a new paradigm for delivering research and development services to the hydrology community and as such there are inherent uncertainties as to the long term outcome of this approach. Any diminution of eWater support will provide challenges to the future development of TIME.
- The large majority of the TIME code base is written in the C# programming language. Whilst C# is an industry standard language at the present, programming languages are continually evolving to meets new demands. The Windows operating system is about to undergo some significant changes and the impacts of these on support for the C# programming language are unclear. A fall in support for C# amongst the programming community would potentially create issues for TIME although this would be a slow process and provide plenty of opportunity to adapt.
- Organisational change within CSIRO is a threat to the future support of TIME. The core developer team currently resides in a relatively small team (EIS) within CSIRO. This team has developed under the prevailing CSIRO research priorities and strategic goals. As with all large organisations, these priorities will change over time as will organisational structures. Being a relatively small group

within CSIRO, IES is susceptible to organisational change. At present there is a critical mass of knowledge in the EIS to sustain TIME development, but in would not take a large reduction in developer numbers to fall below critical levels.

- Until now, the development of TIME has largely been sponsored through large National science funding initiatives formed around desired political outcomes. As the political landscape evolves these funding initiatives may change focus into science domains not normally associated with the development of TIME.
- Related to this, is the issue of strategic direction. If, in to the future, the development of TIME was to be solely funded through ad-hoc project requirements, there is the risk that TIME will suffer from a lack of strategic direction and regress in to an esoteric problem specific solution with little or no relevance in the broader modeling framework domain.
- With the current changes in organisational arrangements that are occurring, there is potential that disparate objectives of the various TIME development partners may emerge over time. Without strong commitment from the development partners to support a unified code base, it would be relatively easy to envision a divergence of the code base over time. This would be a serious deleterious outcome.
- Internationally, the area of modeling framework development is an area of very active research and development. There are also many freeware software products and code bases that provide functionality comparable to various components of TIME. The utility of these comparable softwares is rapidly increasing, often supported by large and active user bases. If the development of TIME was to stagnate it would quickly lose relevance and possible forfeit the niche it presently fills.

# 10 Alternatives to TIME

TIME was initially conceived and developed to fill an identified functionality gap in the modelling domain. One of the guiding design principles of the TIME modelling framework was that uses should be able to do as much of the data manipulation and analysis within the TIME framework as possible without having to resort to expensive third party commercial solutions. At this time there were very few, if any, freely available tools that had the range of functionality required to deliver optimal modelling solutions to the scientific community. Thus, the original developers of TIME had to code a lot of functionality into TIME from scratch.

In the ensuing years the open source community has been exceptionally active in addressing and delivering solutions to the same types of problems that TIME provides solutions to. To this point, there appears to be no one, freely available software framework that covers the complete TIME functionality gamut. There are many commercial and free-ware alternatives to specific parts of the functionality provided by TIME, now in existence. Below is a very brief summary of just a few freely available alternatives to sections of the TIME functionality. Most if not all of these have very active development communities and are highly extensible and multi platform compatible. These alternatives provide functionality across a broad spectrum of scientific domains.

## Spatial Analysis

**GRASS** is free Geographic Information System (GIS) software used for geospatial data management and analysis, image processing, graphics/maps production, spatial modelling, and visualization. (http://grass.fbk.eu/).

**GDAL** (Geospatial Data Abstraction Library) is a library for reading and writing raster geospatial data formats, and is released under the permissive X/MIT style free software license by the Open Source Geospatial Foundation It is already used by TIME to handle some spatial data formats. As a library, it presents a single abstract data model to the calling application for all supported formats. It may also be built with a variety of useful command-line utilities for data translation and processing. (http://www.gdal.org/).

**Quantum GIS** (QGIS) is a powerful and user friendly Open Source Geographic Information System (GIS) that runs on Linux, Unix, Mac OSX, Windows and Android. QGIS supports vector, raster, and database formats. (http://www.qgis.org/).

**DotSpatial** is a geographic information system library written for .NET 4. It allows developers to incorporate spatial data, analysis and mapping functionality into their. The free open source data viewer and GIS package **MapWindow** is built on DotSpatial. (http://www.mapwindow.org/ )

**SAGA** is the System for Automated Geoscientific Analyses. SAGA is a Geographic Information System (GIS) software designed for an easy and effective implementation of spatial algorithms. SAGA offers a comprehensive, growing set of geo-scientific methods. (http://www.saga-gis.org/en/index.html)

## Temporal\Vector Analysis

**R** is a language and environment for statistical computing and graphics. R provides a wide variety of statistical (linear and nonlinear modelling, classical statistical tests, time-series analysis, classification, clustering etc) and graphical techniques, and is highly extensible. R is becoming widely used within the general scientific community. (http://www.r-project.org/)

## Model Orchestration\Frameworks

The **OpenMI** (Open Modelling Interface), (Gregersen *et al.*, 2007) standard defines an interface that allows time-dependent models to exchange data at run-time. When the standard is implemented, existing models can be run simultaneously and share information at each timestep making model integration feasible at the operational level. OpenMI is merely a specification and relies on model developers doing their own implementation.

(http://www.openmi.org/)

The **OMS** (Amerman *et al.*, 2002) is a framework consisting of: a library of science, control, and database modules; a means to assemble the selected modules into a modelling package customized to the problem, data constraints, and scale of application; an automatic generation of a friendly user interface; and creation of a compiled, ready-to-run, version of the package. The framework is supported by utility modules such as data dictionary, data retrieval, GIS, graphical visualization, and statistical analysis.

**FABM** the Framework for Aquatic Biogeochemical Models (Trolle *et al.*, 2012) is a Fortran 90 programming framework for biogeochemical models. FABM can interface with various hydrodynamic models, and includes a repository of existing models of biogeochemical processes (http://sourceforge.net/projects/fabm).

Whilst none of the available alternatives covers the breadth of functionality embodied in TIME, they are currently possess vibrant development communities and are under active development. They are all implement effective solutions to their particular domains. They all have a significant web presence which supports discovery and subsequent development of these products.

It is interesting to note the niche that TIME continues to fill. There are very few, if any competing alternatives to the fundamental scope of TIME, i.e. implementing a generalised modelling framework, whilst providing a large range of concrete functionalities and tools.

It is clearly evident from this very brief review that there are many noteworthy alternatives to TIME but none in a comprehensive, integrated environment. Given the extensible nature of most of these alternatives it be wise practice in the future development of TIME to assess the specific capabilities of the alternatives and where possible integrate with, and build on these alternatives.

# 11 Potential Directions

This section aims to suggest some possible areas of work into the future to improve on the existing TIME code base. A number of the directions suggested have already been explored by various developers and may have been partially implemented or built for a specific one off purpose. There are many specific programming related improvements that could be made to TIME, but this section does not intend to go in to great detail regarding these. It attempts to offer guidance on general directions which may enhance the utility of TIME. Most if not all of these suggested activities will require significant investment and commitment. Suggestions are made in no particular order.

## 11.1 Building on Existing Functionality

- **Make TIME Open Source.** The problem set that TIME addresses is by no means unique to the existing user community. The utility of TIME may be a significant benefit to a broad range of scientific disciplines. Open sourcing TIME would be an efficient means of opening these benefits to a much broader user group. It may also have the benefit of increasing the resources available to further improve the code base through contributions from third parties. The code could be made open source and access to the code base could remain similar to what it is now i.e. relatively restricted. This type of approach would open up a range of collaboration opportunities. The other approach could be to make the code base more broadly publically available. If the code base was to be made more broadly available it is suggested that future development is not open slather, but rather a reasonably well managed process in which contributions would be assessed and implemented in a structured manner. This approach has an associated overhead for the custodian but the potential benefits may justify this cost. The code base in its current state is not highly suited to immediate open access and would require significant commitment to get it to a standard suitable for the open sourcing model.
- **Make TIME Mono compatible.** There is a large base of scientific modelers utilising the Linux computing platform. With little effort the TIME code base can be made to run under Linux using the Mono compiler. This would open up TIME functionality to a much broader scientific user base. This would also remove the dependence on the Microsoft Windows operating system.
- **Build a simple GUI for TIME.** A graphical user interface called VisualTIME already exists. It was designed to be a very generic and flexible GUI that provided access to a broad range of the TIME code base functionality. To achieve this flexibility, the forms presented to users tend not to be that easy to use or understand. A simple purpose built GUI that exposed some of the core functionality and tools within TIME based around the spatial and temporal processing tools would potentially open up TIME to a much broader range of non programming literate users. This GUI could build on the existing VisualTIME GUI or be a new purpose built GUI.
- **Build on model framework capabilities.** A prototype application giving the ability to visually link models together and run the result has previously been developed. The tool provides a rapid and simple way to develop new modelling functionality and is accessible to users who are not programmers (Rahman *et al.*, 2005). It no longer seems to be in the code base. This tool should be revisited and further developed to re-implement this functionality. This is a valuable application of one of the core capabilities of TIME.
- **Improve discoverability.** One of the key blockages to increased uptake of the TIME code base is the difficult nature of trying to actually ascertain exactly what TIME can do. This applies to both coders and general users. No specific approach to this is recommended but it is mainly a communication issue that could be addressed through documentation or simple discovery tools.

- **Expand on the existing documentation.** Whilst the existing documentation is a useful resource, it does not fully cover the totality of TIME. A more comprehensive suite of documentation would make it easier for users to be more productive more quickly. The current documentation is a static suite of resources whereas TIME is a dynamic code base that evolves over time. A wiki based documentation system may provide more flexibility to update the TIME documentation as needed and simplify the process for developers to update information as required.
- **Build on existing model orchestration links.** The existing Source code base has classes for implementing the OpenMI (Gregersen *et al.*, 2007) standard to allow different models to communicate during simulations. Investigations to determine if it is possible to implement this functionality generically within the TIME modelling framework should be undertaken. This would broaden the utility of models developed within TIME by allowing them to work in concert with existing models and reducing the need for a one platform modelling solution.
- **Build on web interface capabilities.** Some work has already been done to develop simple web UI capabilities in TIME. As well as this, CSIRO has also incorporated TIME into a number of web service products for specific projects. With growing requirements for web based delivery of services it may be appropriate to explore the option for making TIME more amenable to this mode of delivery. The development of a generic web API for TIME would facilitate flexible delivery of TIME capabilities. Modelling components within TIME could be implemented as web services.

## 11.2 New Functionality & Research

- **Implement a scripting interface**. TIME currently has purpose built simple scripting interface that uses the Boo programming language (Oliveira, 2005), however this has never been utilised to any great extent. Scripting is common in many scientific work flows and the Python scripting language tends to be preferred by a significant portion of the scientific community. The implementation of a scripting interface using a language such as Python would add significant utility to TIME.
- **Implement parallel multi-threaded processing**. Investigate the opportunities to introduce parallel processing capabilities within the framework and build on the work Perraud *et al*. (2009) in this domain. With modern CPU speeds starting to plateau due to physical limitations, performance gains in modern computer hardware are being achieved through multi-core processing. Not all aspects of the framework are amenable to parallel execution, but certain data processing tasks, in particular raster data processing, could benefit significantly from this capacity. This is not necessarily a simple task but a well considered implementation could have significant benefit. Related to this, TIME should be made intrinsically thread safe where possible, to make robust when deployed in modern multi-core architectures.
- **Investigate distributed processing technologies**. Similar to the previous point, many computational performance gains are being made through distributed processing technologies. Previous investigative work has been done on incorporating these approaches into TIME but with this technology developing so rapidly it would be appropriate to undertake a review of the existing approaches to this, with a view to implementing some core distributed processing architecture into TIME.
- **Full Integration of GDAL into TIME**. The Geospatial Data Abstraction Library (GDAL) (Walter *et al.*, 2002) is an open source library for efficiently reading and writing raster and vector geospatial data formats. It supports over 120 raster and over 10 vector geospatial data formats. It also has built in projection support and a range of tools for processing geospatial data. Some integration of GDAL into TIME has already been implemented for dealing with large rasters, but a more comprehensive and native implementation should be considered.
- **Large data processing capability**. The core data classes within TIME were developed in the days before the processing of large data sets was common. It would be appropriate to review modern approaches and methods with the view to optimising the handling of large datasets within TIME.

- **Distributed data capability.** Given the types of analyses that TIME is typically used for, there is often a requirement to access data from disparate geographical locations and custodians. This makes the incorporation of distributed data access capabilities into the core TIME framework a very attractive proposition. This is currently a very active area of development worldwide and technologies such as OPeNDAP (Cornillon *et al.*, 2003) are being deployed extensively. OPeNDAP is a software framework for enabling distributed data access. Implementing OPeNDAP in TIME should be investigated.
- **GPU Processing.** Another commonly used means of improving computational performance in modern computing hardware is to utilise the General Processing Units within the computer to perform calculation intensive operations. It would be useful to investigate the potential of this approach in the TIME framework.
- **Modernise GUI components.** Much of the graphical user interface components of TIME are based on Microsoft WinForms technologies. This is generally considered to be an outdated technology and TIME would benefit from moving to one of the more modern approaches to GUI implementation approaches such as Windows Presentation Foundation. WPF applications can be deployed as standalone desktop programs, or hosted as an embedded object in a website.
- **Dynamically link to R.** The R data analysis environment has a broad range of powerful data analysis and statistical methods. It is rapidly being adopted by the broader scientific community as a tool of choice. Future data analysis requirements within TIME should leverage the capabilities already present in the R data analysis environment.

# 12 Recommendations

Recommendations are listed in a suggested order of importance

- CSIRO should use the shared code base resources and infrastructure currently being implemented by eWater. Fragmentation of the TIME code base needs to be avoided at all costs and contributing via the shared resources will facilitate this. At any point, if the need arose due to external influences, it is a simple task for CSIRO to obtain a copy of the existing code base.
- A formal roadmap for the development of TIME in collaboration with partners should be established. This would generate confidence amongst the user community in the long term viability of utilising TIME
- A formal process to establish the development priorities for TIME should be undertaken. There are many opportunities for further improvements and enhancements. Most of these will require significant investment and commitment. Finite resourcing constraints dictate a need for a considered approach.
- Where possible and appropriate, future development of TIME should integrate and or interoperate with the range of alternative or complimentary freeware solutions in existence.
- Broader applications of TIME outside of the hydrology domain should be explored and encouraged.
- CSIRO should remain as an active participant in the further development of the TIME code base. Given the significant investment and the broad range of internal and external dependencies for delivery related to TIME, it would not be prudent to abandon the development and maintenance of the TIME code base.
- The existence of the TIME framework should be promoted internally within CSIRO. They are many existing and future research programmes that could benefit from the application of TIME functionality.
- Where appropriate, CSIRO should continue to contribute to the development of TIME under the proposed shared IP arrangement with eWater holding IP in trust for the partner organisations. A considered decision as to the suitability of these IP arrangements should be made at the beginning of any new major projects. In some circumstances CSIRO could consider not contributing to the shared IP pool if CSIRO partnerships external to eWater had this requirement. This however, should be avoided if possible.
- The TIME code base should be “tidied up” to enhance its flexibility and to remove its dependence on other sections of the broader “eWater Source” code base, such that TIME becomes a generic domain agnostic framework.
- Formal governance arrangements should be put in place to guide future development and maintenance of the TIME code base. It is proposed that governance mechanisms be set up at three levels to provide strategic direction and support for TIME : -
    - Business Governance – a group to sponsor and support the ongoing maintenance, development and applications of TIME. This group should consist of management level members from CSIRO, eWater and partners at a minimum.
    - Technical Governance – a group to provide technical direction to the future development of TIME. This group should consist of technical members from CSIRO, eWater and partners at a minimum.
    - Operational Governance – arrangements similar to those that currently exist (Section 6) to provide for the day to day oversight of the development and maintenance of TIME.
- CSIRO should consider if it supports open sourcing the TIME code base. This would come at a cost and require ongoing commitment, but this has to be weighed up in terms of potential benefits in terms of visibility, impact and international adoption. If this is determined to be a viable development path, CSIRO would need to enter into discussions with eWater on this subject.

# 13 References


AMERMAN, R., B. W., L. AHUJA, O. DAVID, J. WERNER, J. CARLSON, R. KNIGHTON AND L. G., 2002. THE OBJECT MODELING SYSTEM (OMS): AN ADVANCED OBJECT-ORIENTED, MODULAR MODELING COMPUTER TECHNOLOGY FOR AGRICULTURAL PRODUCTION SYSTEMS. FORT COLLINS, COLORADO.

ARMSTRONG, R., D. GANNON, A. GEIST, K. KEAHEY, S. KOHN, L. MCINNES, S. PARKER AND B. SMOLINSKI, 1999. TOWARD A COMMON COMPONENT ARCHITECTURE FOR HIGH-PERFORMANCE SCIENTIFIC COMPUTING. IN: HIGH PERFORMANCE DISTRIBUTED COMPUTING, 1999. PROCEEDINGS. THE EIGHTH INTERNATIONAL SYMPOSIUM ON. PP: 115-124.

CORNILLON, P., J. GALLAGHER AND T. SGOUROS, 2003. OPENDAP: ACCESSING DATA IN A DISTRIBUTED, HETEROGENEOUS ENVIRONMENT. DATA SCIENCE JOURNAL, 2: 164-174.

FENTON, N., S.L. PFLEEGER AND R.L. GLASS, 1994. SCIENCE AND SUBSTANCE : A CHALLENGE TO SOFTWARE ENGINEERS. IN: IEEE SOFTWARE. INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS.

GREGERSEN, J.B., P.J.A. GIJSBERS AND S.J.P. WESTEN, 2007. OPENMI: OPEN MODELLING INTERFACE. JOURNAL OF HYDROINFORMATICS, 9(3): 175-191. AVAILABLE FROM HTTP://CSIRO.SUMMON.SERIALSSOLUTIONS.COM/LINK/0/ELVHCXMWWYWPEMUB6TQ7LJLBGNGY0QVP7_UGG3AQRBIAESMV926IDCFURIHOHRRQIWJ0K81NJXSTJM0MKKZMLVOMDZONUOXNZJONJRNBNYMKFPMUC6MUCXMLG2QGTEOZTUXMMK4ZSRBINE4XSTVJMRCWMBZIMEWZYKSWMGDPKDALC44ZX3OMNQBNZDZF. DOI 10.2166/HYDRO.2007.023.

LLOYD, W., O. DAVID, J.C. ASCOUGH, K.W. ROJAS, J.R. CARLSON, G.H. LEAVESLEY, P. KRAUSE, T.R. GREEN AND L.R. AHUJA, 2011. ENVIRONMENTAL MODELING FRAMEWORK INVASIVENESS: ANALYSIS AND IMPLICATIONS. ENVIRONMENTAL MODELLING AND SOFTWARE, 26(10): 1240-1250. AVAILABLE FROM HTTP://CSIRO.SUMMON.SERIALSSOLUTIONS.COM/LINK/0/ELVHCXMWQYWZH5UHIENPRBQH4KA_LVIEBD9W0O6XCFCZB6SI3K2UIDTNNCTZQXD6SYBUMUHSNF3J1BQZYYPDNMOU5FRGWJJJTUK0NKPMBP0RZMLKLMSAMGHKAJAYMGBSFQQMMIALMGJVPFGKGSALPKVYJQEAMVJTWTQ85HYWAW1R2XF3YDXVBWDH6TON. DOI 10.1016/J.ENVSOFT.2011.03.011.

MARSTON, F., R. ARGENT, R. VERTESSY, S. CUDDY AND J. RAHMAN, 2002. THE STATUS OF CATCHMENT MODELLING IN AUSTRALIA. IN: REPORT (COOPERATIVE RESEARCH CENTRE FOR CATCHMENT HYDROLOGY (AUSTRALIA)) ; 2002/04. CRC FOR CATCHMENT HYDROLOGY, CLAYTON, VIC.: PP: VI, 39 P.

MEYER, B., 2001. .NET IS COMING [MICROSOFT WEB SERVICES PLATFORM]. COMPUTER, 34(8): 92-97. DOI 10.1109/2.940017.

MICROSOFT, 2012. CODE METRICS VALUES. IN: MSDN. MICROSOFT.

MURRAY, N., J.M. PERRAUD, J.M. RAHMAN, S.P. SEATON, H. HOTHAM, F.G.R. WATSON, R. BRIDGART AND G. DAVIS, 2006. TIME - REFERENCE MANUAL.

OLIVEIRA, R.B.D., 2005. THE BOO PROGRAMMING LANGUAGE.

PERRAUD, J.-M.V., J; STENSON, M; BRIDGART, R, 2009. MULTI-THREADING AND PERFORMANCE TUNING A HYDROLOGIC MODEL: A CASE STUDY 18TH WORLD IMACS CONGRESS AND MODSIM09 INTERNATIONAL CONGRESS ON MODELLING AND SIMULATION. MODELLING AND SIMULATION SOCIETY OF AUSTRALIA AND NEW ZEALAND AND INTERNATIONAL ASSOCIATION FOR MATHEMATICS AND COMPUTERS IN SIMULATION, CAIRNS, PP. 1059 - 1065.

RAHMAN, J.M., J.M. PERRAUD, S.P. SEATON, H. HOTHAM, N. MURRAY, B. LEIGHTON , A. FREEBAIRN, G. DAVIS AND R. BRIDGART, 2005. EVOLUTION OF TIME. IN: A. ZERGER AND R. M. ARGENT, (EDS.) MODELLING AND SIMULATION SOCIETY OF AUSTRALIA AND NEW ZEALAND.

RAHMAN, J.M., S.P. SEATON, J.M. PERRAUD, H. HOTHAM, D.I. VERRELLI AND J.R. COLEMAN, 2003. IT'S TIME FOR A NEW ENVIRONMENTAL MODELLING FRAMEWORK. MODSIM 2003 INTERNATIONAL CONGRESS ON MODELLING AND SIMULATION(4): 1727-1732.

TROLLE, D., D. HAMILTON, M. HIPSEY, K. BOLDING, J. BRUGGEMAN, W. MOOIJ, J. JANSE, A. NIELSEN, E. JEPPESEN, J. ELLIOTT, V. MAKLER-PICK, T. PETZOLDT, K. RINKE, M. FLINDT, G. ARHONDITSIS, G. GAL, R. BJERRING, K. TOMINAGA, J.T. HOEN, A. DOWNING, D. MARQUES, C. FRAGOSO, M. SØNDERGAARD AND P. HANSON, 2012. A COMMUNITY-BASED FRAMEWORK FOR AQUATIC ECOSYSTEM MODELS. HYDROBIOLOGIA, 683(1): 25-34. AVAILABLE FROM HTTP://DX.DOI.ORG/10.1007/S10750-011-0957-0. DOI 10.1007/S10750-011-0957-0.

WALTER, G., F. WARMERDAM AND P. FARRIS-MANNING, 2002. AN OPEN SOURCE TOOL FOR GEOSPATIAL IMAGE EXPLOITATION. IN: GEOSCIENCE AND REMOTE SENSING SYMPOSIUM, 2002. IGARSS '02. 2002 IEEE INTERNATIONAL. PP: 3522-3524 VOL.3526.

WATSON, F.G.R. AND J.M. RAHMAN, 2004. TARSIER: A PRACTICAL SOFTWARE FRAMEWORK FOR MODEL DEVELOPMENT, TESTING AND DEPLOYMENT. ENVIRONMENTAL MODELLING & SOFTWARE, 19(3): 245-260. AVAILABLE FROM HTTP://WWW.SCIENCEDIRECT.COM/SCIENCE/ARTICLE/PII/S136481520300152X. DOI 10.1016/S1364-8152(03)00152-X.

# 14 Appendix 1: TIME Functionality Table

| | |
|---|---|
| Applications\TIMEShell | The command line interface for TIME |
| Applications\ToolkitLook | Sample app demonstrating common visual components of TIME and GUI guidelines |
| Applications\VisualTIME | General purpose for analysing data and running models |
| | |
| Auxiliary\Addins | Visual Studio plugins built by the TIME team |
| Auxiliary\Utils | Various utilities for managing the TIME codebase eg documentation and Latex tools |
| | |
| Core\Metadata | Contains metadata definitions (tags) |
| Core\Uncertainty | Methods for representing uncertainty in data |
| Core\Units | Definitions representing units of data (eg mm per day, kg, m3, etc) |
| | |
| DataAnalysis\CalibrationTool | Plugin for Visual TIME allowing optimisation of parameters of temporal models |
| DataAnalysis\CellOrder | Generates a processing order of a DEM based on elevation |
| DataAnalysis\DataConverters | Components for converting between common datatypes eg raster to polygons |
| DataAnalysis\RasterCropper | Interactive tool for cropping rasters |
| DataAnalysis\RasterTemplate | Examples of using the template (zonal) functions of rasters |
| DataAnalysis\RuleEngine | Tool for manipulating data based on a set of objective rules |
| DataAnalysis\ShortestPath | Finds the shortest path between a source node in a network and every other node. Uses Dijkstra's algorithm. |
| DataAnalysis\SpatialDataTool | Unfinished. Intended to be a VT plugin bringing together many spatial functions into a common interface (like a miniARCView) |
| DataAnalysis\StatisticsTool | Visual time plugin providing numerous statistics routines |
| DataAnalysis\Terrain mrVBF | An old and incomplete Multi-res valley bottom flatness implementation. There is a more up to date MrVBF in a class called MrVBF.cs inside Terrain/TerrainAnalysis. |

| | |
|---|---|
| DataAnalysis\Terrain\Terrae | Finite Element Modelling routines |
| DataAnalysis\Terrain\TerrainAnalysis | , Collection of general purpose Terrain routines |
| DataAnalysis\TimeSeriesTransform | Removed - do not use |
| DataAnalysis\UncertaintyAnalysis | Tools for defining data uncertainty. Alpha. |
| DataAnalysis\VectorOperations | A powerful set of tools written by Paul Peterson. Used for vector data manipulation vector polygon etc. eg clip, buffer, etc. |
| DataAnalysis\ZonalOperations | |
| | |
| DataTypes\IO | Routines for loading and saving data |
| DataTypes\Metadata | Metadata tags specific to datatypes & io types |
| DataTypes\NodeLinkNetwork | Implementation of Node-link networks |
| DataTypes\Polygons | Implementation of Polygons |
| DataTypes\RasterImplementation | Internal implementation of Rasters |
| DataTypes\TimeSeriesImplementation | Internal implementation of Time Series |
| | |
| NetLP | Implementations of Network Linear Program Solvers |
| | |
| Science\Algebra | Linear algebra (vectors etc) |
| Science\Economics | |
| Science\Hydrology | Where most fundamental hydrological methods should be located. Currently a suite of baseflow filters. |
| Science\Mathematics | Currently contains solvers for differential equations and root finders. |
| Science\Probability | Contains/Should contain fomulas related to probability laws. |
| Science\Probability\Uncertainty | This is where the implementation of classes handling data uncertainty should be located. |
| Science\Probability\Random number | generatros Tools for generating RNS according to different distributions |
| Science\Statistics | Contains/Should contain fomulas related to statistics (i.e. stuff that tries to get a probabilistic description from data). |
| Science\Utils | Currently, set of wheighting classes useful in ecomonics for multi-attribute decision systems. |
| | |

| | |
|---|---|
| Testing\Models | TIME models designed for testing purposes. |
| Testing\Prototyping | |
| Testing\UnitTests | |
| | |
| Tools\CanvasTool | Prototype interactive linking of models. Used by the System Model Tool. |
| Tools\DataTools | Set of general purpose Data manipulation and evaluation tools supporting optimisation tools, dynamic visualisation etc. |
| Tools\DIME | Unfinished but in development. Distributed processing framework for TIME. |
| Tools\ModelExecution | Tools for coordinating simulation runs |
| Tools\Optimisation | Tools for optimising model parameters |
| Tools\RasterGenerators | Generating Test Rasters |
| Tools\Reflection | Supporting metadata interrogation |
| Tools\Time | Calendar routines |
| Tools\Utils | Set of common array manipulation functions. |
| Tools\VisualTools | Generation of automatic interfaces for models |
| | |
| Visualisation\Colors | classes for using colour in TIME. Ie. ColorSchemes |
| Visualisation\Decorators | The bits and pieces used to create viewControls. Ie. Axis, AxisLabels, Chart Title. |
| Visualisation\Layers | Most of the common layers for visually representing data in a viewControl. |
| Visualisation\LegendItems | Helper classes for drawing legends in viewControl |
| Visualisation\Symbols | Small graphics for drawing points on layers in various shapes. |
| | |
| WebApplications\WebAR1 | Example webapplication stochastic generation of annual rainfall data |
| | |
| Winforms\Canvas | Visual representation of model linking |
| Winforms\Filesystem | Visual components for interacting with file systems |
| Winforms\Parameters | |
| Winforms\Processing | |

| | |
|---|---|
| Winforms\ReflectedItems | |
| Winforms\Resources | |
| Winforms\Time | Calendar/date selection |
| Winforms\ToolkitBadging | controls for Toolkit look & feel (splash screens, badging etc) |
| Winforms\Utils | Contains common tasks involving MessageBoxes. |
| Winforms\ViewEditing | Controls for users to manipulate maps & charts |
| Winforms\Weighting | Controls for weighing of data. Will be needed e.g. for Thiessen weighting and gap filling. |

CONTACT US

**t** 1300 363 400
+61 3 9545 2176
**e** enquiries@csiro.au
**w** www.csiro.au

YOUR CSIRO

Australia is founding its future on science and innovation. Its national science agency, CSIRO, is a powerhouse of ideas, technologies and skills for building prosperity, growth, health and sustainability. It serves governments, industries, business and communities across the nation.

FOR FURTHER INFORMATION

**Land & Water/Environmental Information Systems**
Ross Searle
**t** +61 738335606
**e** ross.searle@csiro.au
**w** http://my.csiro.au/en/Business-Units/Environment/Land-and-Water.aspx