Evolution of TIME

[1]**Rahman, J.M.**, J.M. Perraud, S.P. Seaton, H. Hotham, N. Murray, B. Leighton, A. Freebairn, G. Davis and R. Bridgart

[1]CSIRO Land and Water, E-Mail: **Joel.Rahman@csiro.au**

Keywords: *TIME, Model Development Frameworks, User Community, Spatial Modelling, Temporal Modelling, Catchment Modelling.*

EXTENDED ABSTRACT

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. TIME underpins the modelling products in the Catchment Modelling Toolkit (http://www.toolkit.net.au); a community developed collection of water quantity, water quality and related models.

TIME is founded on a compact architecture with a series of subsystem frameworks handling issues such as data IO, data visualisation and non-linear optimisation. Since the last congress, TIME has undergone significant functional evolution, although the major architectural elements remain largely intact. Key additions are in the area of spatial and temporal data analysis where a series of GIS-like tools and novel algorithms have been implemented. Additionally a series of generic, non-linear optimisation tools have been incorporated and used in several Toolkit products. A visual tool for the integration of models has been created based on ideas encompassed in ICMS. Additionally, the underlying framework has been extended to include the representation and visualisation of data uncertainty.

Much of the evolution of the framework and the libraries has been driven by an increasing stakeholder base of model developers and end users, which presents its own challenges for 'community' developed software systems. TIME is now the development platform for approximately 50 model developers from a range of technical backgrounds from professional software developers to PhD hydrologists. These developers approach the framework from one of several perspectives; as researchers implementing and testing model algorithms using the model testing tools; as software developers who create standalone modelling tools and as framework developers incrementally improving framework functionality. These users make use of a range of community resources, including shared source code access, an email discussion forum and formal training workshops.

TIME has proven to be an effective platform for the development of standalone modelling tools with high quality user interfaces. For example, the Stochastic Climate Library draws on the framework's inbuilt capabilities for visualisation, data handling and temporal analysis to create a polished modelling product producing stochastic climate replicates. The Stochastic Climate Library experience is a good case study of TIME's evolution, by illustrating how a product can be co-developed with researchers and professional programmers, and how direct user requirements for the end product can feed useful functionality back into the underlying modelling framework. This library includes a collection of models which already existed in various forms, and a number of approaches were used to bring them into TIME, including porting to newer dialects of the original language (Fortran), wrapping as DLLs, and porting to C#, a language with additional capabilities.

This paper builds upon the theoretical foundations of TIME to examine some of the practical issues in the use, adoption and evolution of the framework.

1. INTRODUCTION

TIME (Rahman *et al.* 2003) is a model development framework intended to support developers when coding new model algorithms, testing algorithms, and delivering models as customised, standalone applications.

TIME underpins many of the modelling products in the Catchment Modelling Toolkit (http://www.toolkit.net.au), including E2 (Perraud *et al.* 2005) and Rainfall Runoff library (RRL) (Perraud *et al.* 2003).

The high level conceptual structure of the framework has remained relatively stable over the previous years, although there has been significant expansion in terms of library capability, along with the number of active model and application developers. The concepts of model based metadata, and the use of these metadata to create 'model processing tools' (Rahman *et al.* 2004) remain central to the framework, and have been a useful teaching tool when training new TIME developers. The suite of metadata based tools has extended to include a graphical model integration tool, as well as tools for managing temporal and spatial-temporal data during model runs. Other improvements include a range of GIS-like spatial analysis and terrain analysis tools along with a capability for representing and visualising data uncertainty.

The range of capabilities added to TIME reflects the diversity of its user and developer base. While it was originally expected that TIME would support a relatively large number of 'algorithm developers', and a relatively small number of 'application developers', it has transpired that more than half of the TIME users undertake application development tasks within the framework. This has required some reassessment of where emphasis and resources should be placed when providing support to framework users.

2. FRAMEWORK FOUNDATIONS

TIME (Rahman *et al.* 2003) is a .NET based model development framework intended to support three key aspects of model development:

1) Coding new model algorithms in a high level language such as C# or Visual Basic,

2) Testing and applying new models using a range of visualisation and analysis tools, and

3) Delivering models as standalone applications.

Model developers using the framework may make use of all of these aspects at various times, although it is common for individual users to focus on just algorithm development and testing (aspects 1 + 2) or primarily application development (aspect 3). While all three aspects rely on the same libraries of software components, a user's focus will determine which subsystems they encounter as end users (interacting with components at runtime) or as developers (writing code that consumes components).

2.1. Layered Architecture

The high level architecture of TIME is reproduced in figure 1 as a layered approach. Each layer contains software components which consume components from the lower layers. In this respect, a user focussed on algorithm development will write components within the 'Models' layer. These model components will consume components in the data and kernel layers, while at runtime the user will be able to interact with their model component using components provided in the Visualisation and User Interface layer. By contrast, an application developer will create a standalone software product, which consumes components from all layers. TIME based applications include TIME's visualisation components which can be customised and used in the same way as standard Windows user interface components such as buttons and text boxes.

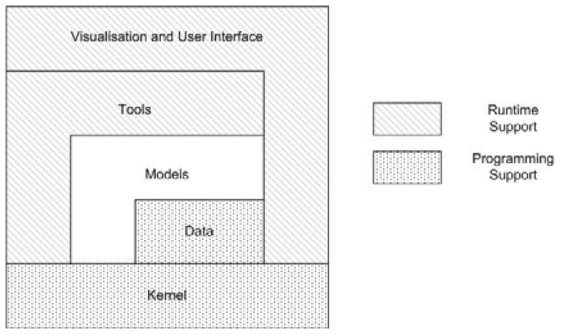


Figure 1: Main architectural layers of TIME

The ability for developers to make use of the framework at a variety of levels is a key advantage of TIME, making it useful for a range of model development tasks.

2.2. Component Metadata

TIME includes a system for marking-up components with structured metadata attributes (Rahman *et al.* 2004). These attributes are used to categorise model variables (eg Input, Parameter, State, Output) and describe constraints such as units and numeric ranges. The metadata is implemented using the .NET Custom Attribute capability (ISO, 2003), and is retained within the compiled components and is subsequently available at runtime to tools within TIME. The metadata capability has previously been used to coordinate the execution of temporal models (Rahman *et al.* 2003) and for the development of generic, non-linear optimisation components (Perraud *et al.* 2003). This capability continues to be developed and exploited, and has more recently been used to underpin a river network based catchment framework within TIME (Perraud *et al.* 2005) and a graphical tool for model integration.

3. KEY DEVELOPMENTS

The high level concepts of TIME are relatively stable, and much of the ongoing development focuses on the functionality of the data analysis libraries and user interface components. Additionally, there is a focus on several model support components, including a system for representing and visualising data uncertainty and a tool for visual model integration.

There has also been significant effort in the development of a customised framework for catchment modelling, using TIME as a base (Perraud *et al.* 2005) as well as a framework for the parallel execution of TIME models using grid computing (Davis *et al*. 2005).

3.1. Data Analysis Libraries and Tools

TIME has a library of spatial and temporal data analysis routines, including Boolean and mathematical operations on data, time series analysis, digital terrain analysis and GIS-like spatial analysis routines.

While these routines are auxiliary to the underlying architecture of the framework, they represent a significant amount of reusable functionality that makes development within TIME more efficient.

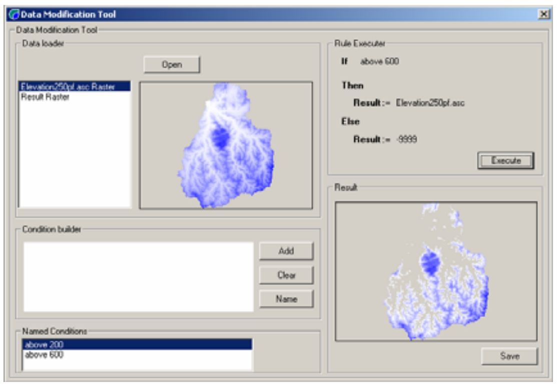
A Boolean 'Rule Engine' allows users to construct operations that derive new data sets based on Boolean combinations and operations on existing data sets. For example, a user might represent a change in land use by deriving a new land use map with the following rule:

```
WHERE elevation is between 600m and 800m
AND current land use is grazing or broadacre agriculture

SET new land use to forest

ELSE set new land use to current land use
```

User's construct rules, which are made up of a predicate, a success action and an unsuccessful action (else action). Predicates can be made up of a nested combination of basic predicates, which operate on a data set (eg elevation) and a test (eg =, <, <>). Rules are evaluated at each 'element' of a source data set to produce the corresponding element of a resultant data set. As all data sets in TIME can be evaluated using a single, one-dimensional element indexing system, the rule engine can be applied equally well to raster data sets (as in the land use example), time series (for example, to represent an ecologic response function based on flow and water quality) or other data types. Users construct predicates and rules using a graphical user interface (figure 2). The Rule Engine GUI is developed as a control which can be deployed in a variety of applications.



```
If    above 600
Then
   Result := Elevation250pf.asc
Else
   Result := -9999
```

Figure 2: Rule engine

TIME includes a range of digital terrain analysis components, including D8 and D-infinity (Tarboton 1997) flow routing, derivative terrain properties such as slope mapping and wetness indices, and watershed delineation. These components are widely used within Catchment Modelling Toolkit products, and reduce the need for Toolkit users to pre-process model data in a GIS. TIME also includes a wider range of GIS-like components for spatial and zonal analysis, including tools for merging and processing polygon coverage and converting between raster and polygonal representations. The inclusion of sophisticated spatial operations raises a philosophical question about the extent to which a spatio-temporal modelling environment such as TIME should seek to emulate a commercial GIS, or even open source GIS efforts such as GRASS (GRASS 2005). We have made a decision to not develop a comprehensive GIS capability *upfront*, but rather to develop routines and components as are required for inclusion in Toolkit products, or requested by TIME users. This approach is complemented by the ability to interchange raster and vector data with common GIS systems such as Arc™ (ESRI 2005) and MapInfo™ (MapInfo 2005).

3.2. Data Uncertainty

The quantification, representation and reporting of uncertainty is a recurrent concern in modelling, irrespective of the discipline (Jolma and Norton 2005). The increasing need to make management decisions based on a quantified risk will only make this concern more pressing. Quantifying the uncertainty of model outputs is non-trivial theoretically, and difficult practically.

TIME currently supports the representation and visualisation of uncertainty for its data. The uncertainty is described at the level of the abstract class Data, parent of any data in TIME, e.g. time series or raster. Data uncertainty is defined as the representation of the uncertainty for each of its items, i.e. by assigning a probability density function (PDF) for each data item. Data uncertainty may thus range from being simply described as a Gaussian PDF centred on each item value and with a standard deviation that is a fraction of this item value, or a complex PDF derived from running a Monte-Carlo simulation and summarising the characteristics of the realisations for each data item.

Visualisation of data uncertainty is currently supported for time series data principally. Even in situations where fully-fledged uncertainty estimation cannot be performed it is an important communication tool to convey it to decision makers. The uncertainty can be displayed by showing a confidence interval around the predicted time series, or using a "fuzzy box" around the prediction based on the PDF for each data item (figure 3). The Stochastic Climate Library (SCL), case study later in this paper, uses the data uncertainty estimations of TIME to determine a confidence interval on the cumulative probability functions of stochastically generated rainfall.

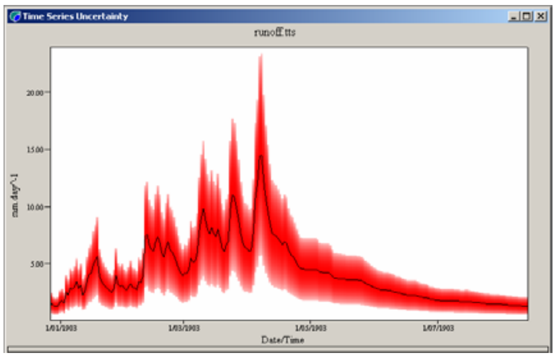


Figure 3: Time series uncertainty visualisation

3.3. Model Integration Tool

The canvas tool is a component of TIME that gives an application the ability to visually link models together and run the result. The tool provides a rapid and simple way to develop new modelling functionality and is accessible to users who are not programmers. Model testing is enhanced through the use of data outputs from other models as inputs to the test model. Automated processing of sequential data analysis can be achieved, and the visual construction can provide a user a clear picture of how a model really works, compared to traditional programming methods.

The canvas makes use of TIME model attributes to ensure the validity of composite models. For example, the Input and Output attributes are used to guarantee inputs never connect to outputs. The use of attributes is not mandatory for the canvas; they simply provide guidance for the user.

The Model Integration Tool is a proposed project that will use the canvas at its heart, and provide a much richer feature set than that currently available from just the canvas. Features planned include, the ability to save composite models which can then be used in further composite models, professional user interfaces, and potentially, connections to data stores for easy access to data.

Figure 4 is a screenshot of a prototype that uses the canvas. The process shown consists of a catchment delineation process, sending its resulting sub-catchment map to a raster extraction routine. This extracted sub-catchment is then sent to a model that converts the raster into a polygon.

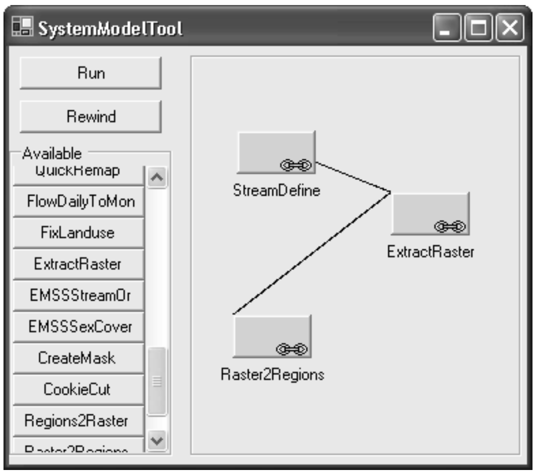


Figure 4: System Model Tool

4. USER BASE

Use of TIME is expanding, predominately amongst Australian natural resource management organisations. There are currently approximately 50 active users of TIME. This includes the developers of fifteen TIME-based Catchment Modelling Toolkit products.

TIME developers come from a range of technical backgrounds, and focus on quite different components within the framework. TIME developers stay in touch through a combination of training workshops, face to face meetings and an email discussion group.

4.1. User Profile

The TIME users range from senior research scientists and PhD students, to professional software developers. While these users are predominately from Australian organisations, they are drawn from a range of academic institutions, government agencies and private industry. The modelling needs of these users vary across a range of spatial and temporal modelling domains.

It is possible to subjectively categorise TIME users as having a primary focus on algorithm development, a primary focus on application development, or a focus on both algorithm and application development. Referring to the three aspects of model development supported by TIME (Section 2), algorithm developers focus on coding algorithms using the TIME data libraries, and testing algorithms using the model testing tools. Application developers, also use the data libraries and testing tools, but create standalone products by developing custom user interfaces that consume TIME's user interface and visualisation components. It was originally expected that only a small number of TIME users would undertake application development, with most users only creating and testing underlying algorithms. It has transpired that the majority of current TIME users do create custom user interfaces and standalone applications. This has resulted in more development emphasis being placed on reusable user interface components as a key value proposition to new users. This also required a reassessment of the training material to include information on making programmatic use of the TIME visualisation system.

The wide focus on application development also reflects an early immaturity in the model testing tools, which required developers of most complex models to create a user interface for the model. There has been a recent resurgence of TIME users who predominately build algorithms, reflecting a growing maturity in the model test tools including the automatic user interface generator, thereby allowing algorithm developers to build and test more sophisticated models without reverting to custom user interface development.

4.2. Community Resources

As TIME users move from being beginner framework users through to expert developers, they make use of a range of support services which broadly constitute the *TIME community*. These services include training material (available at http://www.tookit.net.au/) and practical workshops, an email discussion list, face to face meetings and source code access to the underlying framework.

Most new TIME developers are introduced to the framework with a workshop lasting between two to four days. The shorter workshops focus on algorithm development with TIME, and testing new algorithms with generic tools such as the user interface generator and non-linear optimisers. These workshops accommodate developers from a wide range of technical backgrounds, from professional programmers, to modellers with very minimal programming background. The longer workshops add material on developing standalone applications with the TIME user interface and visualisation components, along with material on customising existing Catchment Modelling Toolkit products such as the Rainfall Runoff Library (Perraud *et al.* 2003) and E2 (Perraud *et al.* 2005).

Workshop participants receive a pre-compiled version of TIME which is sufficient for developing and testing algorithms, but they are encouraged to seek access to the source code to make the most of the framework. Source code access is provided to framework users using a community license agreement requiring them to contribute work back to the main code base. All source code users receive an account on a source code control server which gives them access to the source, along with incremental updates. Additionally, they are provided a workspace on the server where they can store their code, and share it first with the core TIME development team and subsequently with wider community as needed. The source code control system supports access controls which allow individual users to have read-write, read-only and no-access to different parts of the code base. This control is used to minimise the amount of code an individual user needs to download and to prevent accidental corruption of critical framework components by inexperienced users. There is, however a ‘tightrope’ to walk to ensure a user has neither too much nor too little access. The source code control system is supported by a growing suite of automated unit tests, run twice daily, which monitor the integrity of core components.

5. CASE STUDY: STOCHASTIC CLIMATE LIBRARY

The Stochastic Climate Library (SCL) is a collection of single and multi site stochastic data generators, for generating rainfall and other climate variables to catchment models. Stochastic data is often used to examine the effects of climate variability on model predictions. These studies can then be used to perform risk based analysis of various systems, such as a water resources or agricultural system.

SCL is a TIME based product available in the Catchment Modelling Toolkit. It has evolved from a tool providing point based (single site) climate sequences, to include the stochastic generation of spatial daily rainfall. The original models were available from a variety of sources, but were typically encapsulated in command-line driven applications. In order to make the tool available to a wider audience, and to incorporate important visual and statistic measures of output quality, it was decided to produce a graphical Windows application (figure 5).

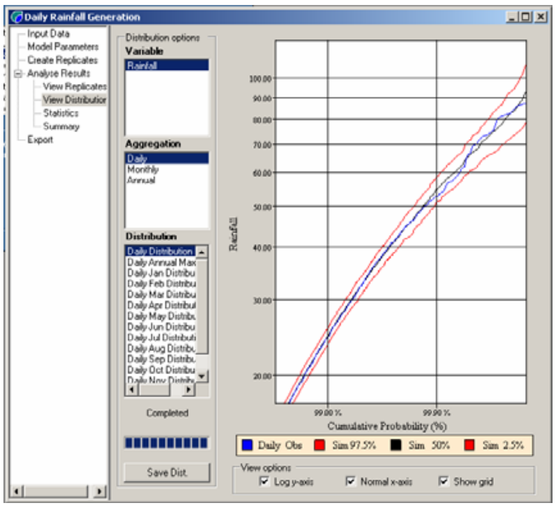


Figure 5: The Stochastic Climate Library

The development of the SCL is the result of a collaboration between stochastic climate researchers and professional programmers. The development of the library included the implementation of the algorithms (shared between the researchers and the programmers), the development of a graphical user interface (undertaken by the programmers and utilising the TIME visualisation and data handling components) and the extension of TIME to include various mathematical tools, such as random number generators for non-uniform distributions.

The original algorithms existed in various forms of procedural Fortran. As TIME models are implemented as classes in .NET languages, there was a mismatch between the current implementations and the desired forms. However, as there were a number of self contained models to be included, it was possible to use a variety of techniques to bring them into the TIME framework.

Several models, including an annual rainfall generator, were migrated to a Fortran 95.NET compiler and recast as a collection of Fortran ‘types’, which have the same internal representation as a class within other .NET languages. This approach was most efficient with models implemented in a modern dialect of Fortran (90 or 95) and structured in such a way that the core algorithm remained separate from IO and other peripheral code.

A sub-daily stochastic rainfall generator was deemed inappropriate for translation to .NET for reasons of efficiency and for the complexity of the original code. In this case, the Fortran code was compiled as a Win32 DLL, rather than as .NET byte code, and a .NET based wrapper DLL was created in Fortran 95.NET.

The remaining models where translated from their original form into C# code. This involved significant recasting of the original structure into a class structure that took advantage of object-oriented concepts such as inheritance and polymorphism. While this involved separating the code from the original model implementation, the object oriented structure included design efficiencies by making commonalities across models explicit.

6. CONCLUSIONS

The evolution and maturation of TIME represents a largely stakeholder driven development. Much of the functionality has been dictated by the needs of Catchment Modelling Toolkit products, and this partly explains the greater adoption by users focussed on the development of model applications. Nevertheless, the functionality that now exists in areas such as data analysis, optimisation, model integration and uncertainty representation is a significant drawcard to a growing user base. TIME is currently moving forwards in areas such as distributed model execution (Davis *et al.* 2005) and as the platform for the E2 catchment modelling framework (Perraud *et al.* 2005).

7. ACKNOWLEDGMENTS

TIME is developed as a joint project between the Cooperative Research Centre for Catchment Hydrology (CRCCH) and the Commonwealth Scientific and Industrial Research Organisation (CSIRO).

8. REFERENCES

Reed, M., Cuddy, S. M.; Rizzoli, A. E. (1999), A framework for modelling multiple resource management issues—an open modelling approach, Environmental Modelling and Software with Environment Data News, 14, November, 1999, pp. 503-509

Davis, G.P. Bridgart, R.J. Stephenson, T.R. and Rahman, J. (2005), Adding Grid Computing Capabilities to an Existing Modeling Framework, Proceedings of MODSIM 2005.

International Organization for Standardization (2003), ISO/IEC 23271:2003(E) Information technology - Common Language Infrastructure, 10.6 Custom Attributes, http://www.iso.org, Last Accessed August 2005.

Jolma, A., Norton, J. P., (2005), Methods of uncertainty treatment in environmental models, Environmental Modelling & Software, 20, 979–980

Perraud, J.-M., Podger G. M., Rahman J. M., Vertessy R. A. (2003), A new rainfall-runoff software library, Proceedings of MODSIM 2003, (4), 1733-1738

Perraud, J.-M., Seaton, S. P., et al. (2005), The architecture of the E2 modelling framework, Proceedings of MODSIM 2005

Rahman J. M., Seaton, S. P., et al. (2003), It's TIME for a new environmental modelling framework, Proceedings of MODSIM 2003, (4), 1727-1732

Rahman, J.M., Seaton, S. P., and Cuddy, S. M. (2004), Making frameworks more useable: using model introspection and metadata to develop model processing tool, Environmental Modelling and Software, 19, March, 2004, pp. 275-284.

Environmental Systems Research Institute, Inc. (ESRI), http://www.esri.com/software/arcgis/index.html, Last Accessed August 2005

Geographic Resources Analysis Support System (GRASS). http://grass.itc.it/ Last Accessed August 2005.

MapInfo Corporation (2005), www.mapinfo.com, Last Accessed August 2005

Srikanthan, R., Chiew, FHS and Frost, A. J. (2005). Stochastic Climate Library User Guide, CRCCH Toolkit, www.toolkit.net.au/scl, Last Accessed August 2005

Tarboton, D. G. (1997) A new method for the determination of flow directions and upslope areas in grid digital elevation models, Water Resour. Res., 33(2), 309-320, 1997.