Deploying environmental software using the Tarsier modelling framework

Fred Watson1,3, Joel Rahman2,3, Shane Seaton1,3

SUMMARY: The Tarsier modelling framework is a modular collection of Windows software that enables fast development and deployment of a wide variety of environmental computing tools. These tools include simulation models, data storage and analysis tools, and visualization systems. The system supports many structures for the organization of quantitative environmental information, including: gridded maps, networks, time series, and simple lists of geographic locations. Upon these are built analytical tools covering topics such as interpolation, statistics, sampling, and data transformation. At the top level are modules that implement a variety of simulation models, from cellular automata to stream pollutant routing models to large-scale spatial catchment hydrology models.

Tarsier was developed with the aim that a user can quickly learn to connect modules together in arbitrary ways to form their own customized analytical, visualization, and modelling systems. Further, it is intended that developers can quickly learn to develop their own fully functional modules that become seamlessly integrated with the existing functionality provided within the framework.

One of the test beds of this approach is the south east Queensland (SEQ) Environmental Management Support System (EMSS) being developed at the Cooperative Research Centre for Catchment Hydrology (CRCCH). This system implements the stream networks of SEQ as the backbone of a modelling and visualization system intended to facilitate the management of water quality in the region. The region is divided into catchments, represented by a selection of catchment models of varying levels of detail. A stream model receives inflows from the catchment models and routes pollutants down toward the ocean. Reservoir models intercept these flows at certain points. The complete integrated system is visualised during simulation using a variety of additional, independent modules. We describe the SEQ EMSS operating within the Tarsier modelling framework.

THE MAIN POINTS OF THIS PAPER

• A modelling framework can make model development and use easier, and lead to better models.
• Features such as modularity, data sharing, and visualisation have enabled a sophisticated, integrated model of runoff and water quality to be developed for southeast Queensland within a relatively short time frame.

1. OVERVIEW

It is increasingly apparent that most environmental systems are complex arrangements of inter-connected physical processes operating at a variety of spatio-temporal scales, and spanning a number of traditional scientific ‘disciplines’. Land managers demand understanding of these systems, and the often far-reaching consequences of their perturbations. Researchers collect ever more data to support these demands, and those who are modellers attempt to simulate the behaviour of systems displaying everything from chaos to equilibria. We rely on computers as the only tools that can cope with the amount of data and process understanding associated with a given area of study. We use them to store information, to display it, and to organise our understanding of inter-connected processes.

The problem is that although we need computers for these tasks, their own complexity places significant additional burden on the environmental scientist. This paper describes the Tarsier environmental modelling framework, a computer tool that seeks to eliminate the overhead involved in using computers for environmental science.

2. WHAT SORT OF MODELS ARE NEEDED FOR STREAM MANAGEMENT?

Streams are central to human existence. If managed incorrectly, water either becomes un-potable or unavailable. Streams are the meeting place of a number of processes and the complexity of their interactions results in consequences of management actions that are difficult to predict. Simulation models facilitate better understanding and prediction of management actions.

The stream system is influenced by climate processes, watershed and groundwater processes, geomorphic and hydraulic processes, transport and biogeochemical processes, and social pressures. In formulating a stream modelling system, the importance of each of these sub-systems must be considered with respect to its potential influence on the objectives of the work. A watershed model may be required in order to simulate inputs of water and nutrients to the stream. A groundwater model may be required to predict exchange of water between stream and aquifer. Geomorphic models can account for variance in parameters such as substrate size, upon which may depend the lives of organisms simulated by models of

1 California State University, Monterey Bay, California, United States
2 CSIRO Land and Water, Canberra, ACT, Australia
3 Cooperative Research Centre for Catchment Hydrology
aquatic ecology. Hydraulic and transport models are used to simulate the movement of water and entrained material down stream networks. Sources, sinks, and transformations of essential nutrients are the domain of biogeochemical models and govern the overall trophic state of aquatic systems. Finally, socio-economic models deal with the valuation of stream resources, the social cost of their degradation, and the likely trajectory of land use change.

Numerous models simulate these systems in relative isolation, and while many are well-formed representations of real physical systems, much can be done to improve their utility through better software practices. Integrated systems, involving a number of connected models are starting to emerge (Watson et al., 2001), but again, their potential remains underdeveloped. Many are inaccessible and poorly communicated, which leads to further problems. Poorly communicated models cannot be submitted to appropriate peer and public scrutiny, and so are likely to contain biases and errors.

3. WHAT IS TARSIER

Tarsier is an environmental simulation modelling framework that aims to achieve better modelling by making it it easier to develop, use, and communicate environmental models. It is implemented as a Microsoft Windows executable application and a number of semi-autonomous modules that “plug in” to the application to form an integrated modelling system.

The plug-ins contain models, data representations, and tools for analysis, interfacing, control, and display. They are are implemented as “dynamically linked libraries” (DLLs) within Windows, and can be linked into the core application by a user at run time. Independent developers can develop their own modules that may be loaded into the system alongside existing modules.

When executed, Tarsier does not appear unlike Excel or Word or any other large Windows application (Figure 1). It has a main window, within which may be situated multiple child windows, each representing a particular module or modules. A main menu allows to user to open and control the child windows and invoke various operations in a manner analogous to any commercial spread-sheeting program.

Linkage between modules is achieved by two simple, yet powerful concepts: sharing and messaging. Data are implemented as autonomous objects that can be shared amongst a number of user objects (e.g. models). User objects can pass messages to each other to communicate changes in the system, thus achieving tightly coupled, dynamic integration.

The framework was developed by Watson (2001) in the late 1990s as a vehicle for the generalisation of a collection of modelling code and concepts accumulated from modelling efforts dating back to the 1980s. The present authors comprise the current coding development team. The work now involves occasional improvements to the core, but is mainly

![Figure 1. A Tarsier modelling session. The main window represents the Tarsier Windows application. In this example, there are four sub-windows, representing the four visual modules that are currently open. The example shows a spatial climate and snow model (top right), a time series of predicted and observed snow pack (bottom right), a perspective view of snow pack distribution (top left), and a view of the vegetation cover that governs snow pack distribution (bottom right).](image)
focused on testing the framework by developing models within it for various “real world” applications.

4. FOUNDING OBSERVATIONS

The framework is a response to the following common observations that experienced model developers might make:

- All models use some specific types of data, either as input or output. These types do not vary greatly between models, and there are only a few types in use. Examples include: scalars, time series, points/sites, polygons/vectors, and raster maps.
- All modellers benefit by being able to visualise their data easily. Visualisation using external packages can be cumbersome. Continually writing internal visualisation code is also time-consuming.
- All models need to read and write their data to and from disks. Repeatedly writing the code to do this is time-consuming.
- All scientific models grow beyond the intended capability of their user interface. Keeping pace with this growth by writing supporting code for interfacing, control, and accounting of variables is time-consuming, and often neglected to the point of diminishing scientific integrity.

If one were to account for every hour of time spent developing a typical environmental simulation model, the development effort might be distributed as shown in Figure 2. It is very common for auxiliary tasks to occupy the overwhelming majority of development time, leaving only a small fraction for the primary tasks: such as when the model is initially conceived, when its equations are coded, and when useful “output” runs are executed.

It is not uncommon for the novel, useful portion of model development to occupy, say, only 6% of total development time – with the other 94% being devoted to repetitive tasks that play only a supporting role to the useful effort. Auxiliary tasks include negotiating information transfers with GIS and spreadsheet programs, massaging data for suitable input to models, writing code to control, observe, and save models, and to aggregate model results into a useful form. Although models are perhaps the only means we have to convey the complexity of Earth, scientists either avoid getting involved with their development, or a relater heard to cry: “It took for ever to get the interface to work. I’ll never do it again.”

The aim behind Tarsier is to work towards the optimal distribution of effort shown in Figure 2, where auxiliary tasks become trivial, the whole process takes less time, and the majority of effort is spent on productive innovation.

5. FEATURES OF THE APPROACH

Within Tarsier, all types of model data are represented as standard “classes”. The instances of these classes are semi-autonomous data objects managed by a central kernel within the framework. Client modules can request to ‘use’ a data object, which is then supplied from the central kernel. For example, a stream routing model might request the kernel to supply a stream network data object (from whichever module supplies such objects). The central kernel always ensures that only one copy of a given object is loaded at any time, and that it is shared amongst all users. This avoids time-consuming data transfers. It means that models that are developed within the framework run just as fast as ones developed outside it.

Primary visualisation and editing support for each data class is also provided through standard classes, the instances of which are child windows within the main application window (e.g. Fig. 1). Multiple ‘viewer’ windows can view the same data. Viewers are considered ‘users’ of data.

Models are also ‘users’ of data. They read it and change it and, from the perspective of the framework, are no different from viewer/editors. These concepts generalise to two groups of object classes: users and...
usees. Users include viewers and models. Usees include data and, as it turns out, models themselves.

Linkage between users and usees is implemented by a system of message passing. Users of a given usee can send messages to each other via the usee itself. A common message is: “Data has changed”. This system enables surprisingly complex integration of modelling components. The outcomes of effective linkage include run-time control and visualisation of models in a manner where the models do not need to be ‘aware’ of or burdened with the fact that these things are occurring. Another outcome is that models can be linked to form super-models, again without having to be ‘aware’ of each other.

A system of linked models is shown in Figure 3. At the bottom are the data objects used by the system. Each has a corresponding ‘view’ (top) that responds to messages about changing data. In the middle is a spatial model, say a model for routing water down a stream network. It has its own ‘view’, which might provide an interface to its parameters and controls. The spatial model uses a sub-model, perhaps a watershed model, which has its own ‘view’.

6. PHILOSOPHIES

The framework attempts to embody the following philosophies:

6.1 Use of RADs and optimised compilers

The best application software developments available today are PC-based rapid application developments (RADs) such as Borland C++ Builder and Microsoft Visual C++. These are available for a wide range of base languages including C++, Pascal, and Java. They allow visual drag-and-drop development and compile very fast code that is unlikely to ever be matched by any custom modelling language implemented within GISs and modelling frameworks.

6.2 Portability through PC-based modelling

A further advantage of using PCs is the physical portability they offer as laptop computers. This promotes better communication between model developers, model users, and the various audiences at whom model results are aimed. Many existing environmental models have only ever been seen ‘in the flesh’ by those who developed and applied them. This compromises their scrutiny, utility, and validity.

6.3 Modularity and re-use

All models that do not run within a modelling framework implement a certain amount of functionality that is in common with almost all other models. Examples include saving and opening data, control, and parameter files; performing initialisation and manipulation on data; and displaying feedback on model operation to the user. Within the framework, as many as possible of these repetitive operations have been identified, characterised in general terms, and implemented in a manner that model developers can utilise. Some of these are implemented within a central class hierarchy that is common to all Tarsier modules, and others are implemented as independent modules or classes.

6.4 Modular autonomy

A requirement of the modelling framework was that model developers should be able to develop models independently of the framework itself, and then ‘plug in’ those modules at run time. A further desire was that the modules should not need to rely on any

Figure 3. A system of linked models, data objects, and visualisation components with Tarsier.
‘knowledge’ of the modules to which they might be linked. The use of Windows DLLs that act as clients to a core application satisfies this requirement, although it does introduce a modicum of device-dependence.

6.5 Linkage through sharing and messaging
Linkage or communication between modules is accomplished solely through the sharing of data and passing of messages between co-users of the same data or ‘usee’. By generalizing the idea of ‘model’ and ‘data’ to the concept of ‘user’ and ‘usee’, the sharing-messaging system becomes a vehicle for complete interaction amongst autonomous modules.

6.6 Visualisation and control
The inability to completely observe and influence model function is a frustration experienced by many modellers. The solution to this is visualisation and control. Every concept in a model should have a visual analogue. This, every functional module in Tarsier has a corresponding visual module that provides dynamic feedback on the state of its respective functional module. Visualisation of specific concepts is supported by an array of customised components that sit on the drag-and-drop design palette during the coding process. These include arrows, tanks, sliders, ‘VCR’ controls for sequencing through time, and ‘usee controls’ for directing modules to use different usees (Figs 1 and 6).

6.7 Open software
There is little point in retaining a modelling framework such as Tarsier as a commercial entity, and much to be gained by adopting an open software principle. The framework will gain strength as more modules are developed within it. Developers will be more encouraged to create modules if they have access to the source code of both the kernel and key framework modules.

6.8 Minimal reliance on third parties
Incorporating third party components into a software application can rapidly improve performance and utility. However, there are longer-term tradeoffs to be made that, ultimately, have resulted in a policy of zero third party reliance. In particular, we avoid reliance of the framework upon application software such GISs and programming software such as 3rd party custom components. The former are slow, limited, monopolizing and can be prohibitively expensive. The latter are either free and flawed, or commercial and not open-source.

7. APPLICATION TO S.E. QLD
A current application of the framework is the development of the Southeast Queensland Environmental Management Support System (SEQ EMSS), an array of linked hydrologic models and scenario-management modules that predict pollutant loads to coastal waters (Vertessy et al., 2001; Cuddy et al., 2001; Scanlon & Chiew, 2001). The system represents SEQ as a collection of sub-watersheds defined on a raster map, and a digital stream network that connects the watersheds to the ocean via the 14 major rivers in the region (Figure 4 shows the Brisbane River sub-watersheds).

The surface hydrology and pollution load of each sub-watershed is represented by the Colobus model. Colobus is a pair (model and viewer) of autonomous modules (Fig. 5) using time series data for both input and output. Each Colobus shares an output time series with a node of the stream network, which treats it as an input. The Marmoset runoff routing model accepts this input into the stream network and simulates the movement of storm waves downstream. A reservoir...
Development within structured software frameworks leads to better models. There are speed and utility advantages to both developer and user. This allows the focus to return back to the science in models. The use of tightly integrated autonomous modules for data, viewer, and model objects adds value to environmental models by making them more effective and accessible vehicles for the conduct and communication of science.

5. ACKNOWLEDGEMENTS

Thanks go to our colleagues who have fostered Tarsier development and application: Rob Vertessy, Peter Hairsine, Murray Peel, Lars Pierce, Joseph Coughlan, Aukje Rolofson, Mark Angelo, Bob Garrott, Don Watson.

6. REFERENCES


4. CONCLUSIONS

Figure 6. The Marmoset viewer/controller window, showing an execution control and several parameter controls.

Figure 7. Two frames from a dynamic visualization of stream routing. The left frame shows a storm peak that has just formed in the headwaters, and will soon meet the main stem of the River. The right frame was captured a few days later. The peak has flowed into the main stem, and its tail remains in the tributary streams.