A Distributed System Dynamics-Based Framework for Modelling Virtual Organisations

Bryan Conneely, Jim Duggan and Gerard J. Lyons

National University of Ireland, Galway

Enterprise Computing Research Group,
Department of Information Technology,
National University of Ireland, Galway,
University Road, Galway, IRELAND

Phone: +353-91-750582
Fax: +353-91-750501
bryan.conneely@nuigalway.ie
jim.duggan@nuigalway.ie
gerard.lyons@nuigalway.ie

Abstract

With advances in distributed computing technology, the idea of a virtual organisation – a network of collaborators whose purpose it is to execute a business model – is now a reality. A core need of any business that conducts operations over a computer network is the availability of systems and tools to support their business process. Many excellent modeling products exist, however these have mostly been tailored to suit co-located planning activities, where group members typically build models within the same "four walls". The aim of this research is to leverage significant developments in distributed computing – XML and .NET in particular – to provide a "plug and play" vendor-neutral distributed framework, which can support dispersed actors in a virtual organisation. The problem used to focus the development efforts is the well-known "beer game". This paper presents the background to our approach, the overall system architecture is also shown, and future work is described.

1. Introduction

This paper describes ongoing research into the application of distributed computing technologies to system dynamics, with particular emphasis on the development of collaborative modelling and gaming environments to support actors in virtual organisations. It has evolved from initial research (Duggan 2002) based on a client/server architecture, to a proposed system that embraces: (1) emerging data exchange standards such the eXtendible Markup Language (XML) and (2) component and service based architectures such as .NET. These new technologies have also shifted the design focus of our research effort. Whereas previously, the design philosophy was to build a comprehensive environment, our thinking has now taken a more "componentised" perspective. Therefore we do not intend to "re-invent the wheel" by building detailed equation solving systems, but rather provide an enabling framework where proven systems can "plug-in" to our distributed framework, and therefore support collaborative modelling across organisational boundaries.

2. Collaborative Gaming and Distributed Computing

From a technological viewpoint, a collaborative gaming environment is a distributed and complex environment, where a number of core features have to be supported by a robust and fault-tolerant framework, namely:

- To allow input from a number of remote users where one input from every user represents a set of simulation parameters.
- To accurately progress the simulation with each new set of new inputs.
- To synchronise model output with adequate information to allow users to accurately formulate their next input.
- To maintain state across a number of different simulations.

In this section, we evaluate existing technologies and architectures that provide support for such an environment.

Client-Server

The client server model is the most mature and commonplace distributed model. The server is a process that implements a specific service and a client is a process that requests the service from the server. A server can typically provide any computing task that is common to many clients, who in turn use the service, e.g. a web server that responds to requests from many client browsers. Communication between the client and the server can be implemented by a number of protocols. With respect to a collaborative framework, where clients are simultaneously accessing one resource, there have to be measures in place to ensure that synchronisation and model validity are maintained.

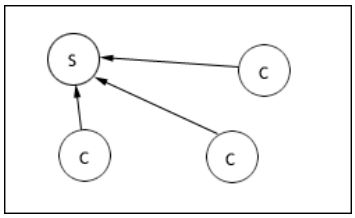

Figure 1. Client-Server Architecture

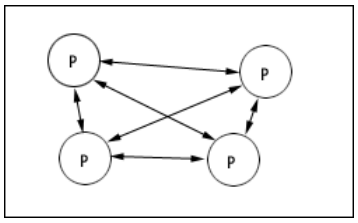

Figure 1. Peer-to-Peer Architecture

Peer-to-peer

A peer is a process that both listens and makes requests and as the name suggests, all peers are equal. Each node acts as a client and a server (O'Reilly 2001). In a peer-to-peer solution, all of the users would have a local private copy of the shared resource, and using a distributed event system, would update other collaborators' copies when changes are made locally. An implementation of a peer-to-peer distributive model is the most natural for a collaborative environment. (Berg 1999) describes "the inherently ad hoc nature of peer-to-peer technology makes it a good fit for user-level collaborative applications". The paper also mentions several technical challenges that make such implementations difficult, such as the location of other peers on the network, fault tolerance, access rights and security for the peers.

Some algorithms for peer locating are described in (Milojicic et al. 2002). The *Flooded Request Model* is based on peers broadcasting all over the Internet for other peers. This has limitations as other peers may be inside firewalls. The *Central Directory Model* uses a server to hold location and other information on all the peers, similar to MSN Messenger (Microsoft 2002). In a hybrid system (Milojicic et al. 2002) like MSN, the peer consults the server only when loading, and communicates directly to the other peers thereafter. The server *can* however hold information to meet the other technical challenges, such as access rights and peer security.

In the traditional peer-to-peer model, peers append changes to a single resource; however, the gaming nature of a collaborative environment sees users synchronically competing with each other by appending changes to a single resource. It is not suitable to have several copies of the model because inconsistencies may arise in the game state amongst different version of the model. These inconsistencies arise due to distributed synchronisation, real-time constraints and network performances, and because of these limitations, the client-server approach where users connect to a server is most suitable for our proposed framework.

Architecture

Our proposed collaborative framework is far too complex to be modelled by a client-server model alone, and, because of this, the server must have an appropriate architecture to accommodate its range of functions. It is to be implemented using a service-orientated architecture (SOA). An SOA is "a way of designing a software system to provide services to either end-user applications or other services through published and discoverable interfaces" (Brown et al. 2002). The published service operates as an entry point for the system. Our service architecture is component based. A component (FEA-PMO and SAWG 2002; McInnis 1999) is the implementation of a business function that is exposed through a deployed interface. A Component Based Architecture uses a suite of components, that call each other to provide an end-to-end solution. It offers developers the potential to assemble applications rapidly. Figure 3 shows how these technologies are used together.

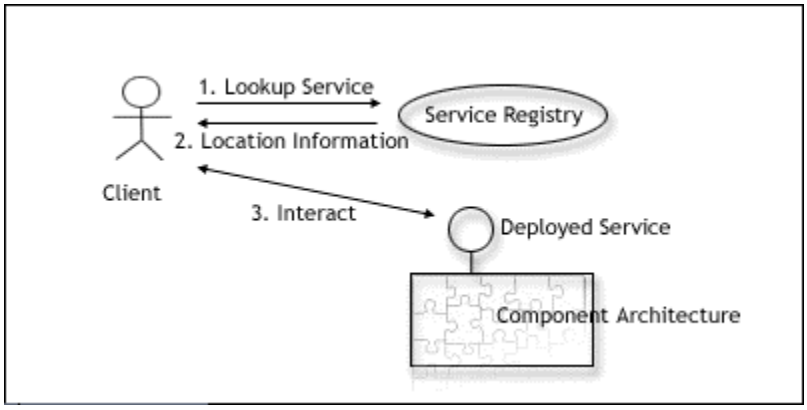

Figure 2. Service-oriented and component-based architectures

An example of a service-orientated, component-based development environment is Microsoft's .NET (Microsoft 2002) platform.

Microsoft .NET

Microsoft's .NET (Microsoft 2002) platform was introduced as "an emerging platform used to define, build, deploy and execute XML Web Services, components, applications and business solutions. It provides a standard based, multi-language environment to integrate 'today's' existing applications with 'tomorrows' next generation solution and business applications" (FEA-PMO and SAWG 2002). Some of its basic design objectives (Microsoft 2003) include full compliance with W3C standards, extensibility and a component-based architecture. With these design goals in mind, the .NET platform makes the ideal choice for the rapid development of distributed, component and service orientated, data-centric applications. And despite the fact that the .NET platform is an emerging technology and relies on emerging standards, it has large community of skilled developers, it is heavily supported, it integrates seamlessly into the Windows (Microsoft 2000) environment and is the chosen platform for the collaborative simulation environment.

XML

XML (W3C 1998) is a mark-up language for documents containing structured information. It is a plain-text representation of data and is an open standard being used by applications to store and transmit data. XML is a natural object representation and it is easily distributable. It is the technology behind SOAP (W3C 2000), which is a standard for data interchange. SOAP is an XML syntax for exchanging messages and is both language and platform independent, and because SOAP typically transferred over HTTP, it is also firewall friendly. SOAP is the enabling technology for Web Services.

Web Services

Web Services are becoming the new official standards for application integration. The reason for its popularity is that it is made up of several industry-accepted standards, namely SOAP, WSDL (Web Services Description Language) and UDDI (Universal, Description, Discovery and Integration. WSDL is a format for describing network services as a set of endpoints and the nature of the message passed between them, either as a document transfer or an RPC call. It describes the Web Services interface as IDL describes a CORBA (OMG) interface. WSDL can be automatically generated with development tools and SOAP toolkits.

UDDI is a standard for publishing information about Web Services in a global registry. Queries can be made to a UDDI registry for a company or service, which will provide sufficient information to locate, dynamically bind to and consume the Web Service. Web Services is a core feature of Microsoft's new .NET framework. The use of web-services to create complex or secure applications has been a matter for concern but *Web Service Enhancements* (Microsoft 2003) offers a means to extend basic web services functionality. Using WSE, security, routing and attachment handling are integrated, making Web Services a more suitable choice for document passing and RPC.

In summary, our view is that this emerging service-based component architecture can be used to extend the capabilities of system dynamics tools and modelling environments. Our choice of Microsoft .NET is also informed by the ease of integration with existing desktop applications such as spreadsheets, databases and word processing, as we would view our framework as being

co-dependent on these other forms of technology. Before introducing our system, we provide an overview of the state of the art in distributed and component approaches to system dynamics.

3. Current Distributed and Component Approaches for System Dynamics

Here, we examine distributed and collaborative system dynamics applications. Simulation is traditionally a stand-alone, centralised task but emergence of the Internet has offered vendors a new means to execute models. *Netsim* and *SableNet* models are distributed using the Internet. *Sable* and *SableNet* introduce the concept of multiple user interaction with system dynamic models in both centralised and distributed environments respectively. *Molecules* highlight the idea of substructures existing within models; substructures that can be assembled together, like components, to build larger structures that can be further assembled to create models.

NetSim Creator

NetSim *Creator* (HPS 2002) enables you to transform a STELLA or ithink model into a web-based simulation and provides the server components needed to distribute the simulation using a web server.

When using NetSim *Creator* to publish a model on the Internet, the model must be in either a STELLA or ithink model format. Along with the publication of interactive models, *Creator* also publishes the websites server framework (i.e. several server pages, a navigation structure and input and output controls). So when deploying a model on the Internet, *Creator* interprets the model file and will generate the framework of pages required. It is up to the developer to add extra content to the website. When the model is run on the Internet, a component that sits on the web-server executes the model and generates numerical output. NetSim *Creator*'s focus is to enable a user to run a simulation remotely and enabling the user to interact with the model to run scenarios. However, every user connects to a new instance of the model. There is no collaboration.

Saple

Sable (Ventana 2001), the Vensim Application Builder, allows you to design and create graphical user interfaces for Vensim models. Gaming is an interesting feature of Vensim and Sable as it enables a user to interact with simulations by allowing them make assumptions throughout the run of the game. These assumptions directly manipulate the state of the model and affect the output giving the user power to control the ultimate outcome of the game. A user should be able to control the simulation to maximise their benefit in the long run. *Saple* is a standalone technology. Vensims *SapleNet* technology is its distributive counterpart.

SapleNet

SableNet (Ventana 2002) enables Vensim developers to publish a model on the Internet. Several participants can simultaneously interact with the model. When a model is published to the Internet, the model is loaded into a component that is controlled via the *Sable* interface, which is distributed to the client computers and viewed through a web browser. Users can view/input assumptions, run, interact (game) and evaluate model outcomes.

Vensim Molecules

Molecules (Ventana 2000) can be regarded as the building blocks of a larger system dynamics model. They are made of primitive stocks, flows and auxiliary elements and are, in turn, the building blocks of complete models. *Molecules* and their organisation provide a framework for presenting important and commonly used elements of model structure. This concept has strong parallels to the component approach of our system.

4. Proposed Framework

The overall system structure and behaviour is now illustrated in figure 4 below.

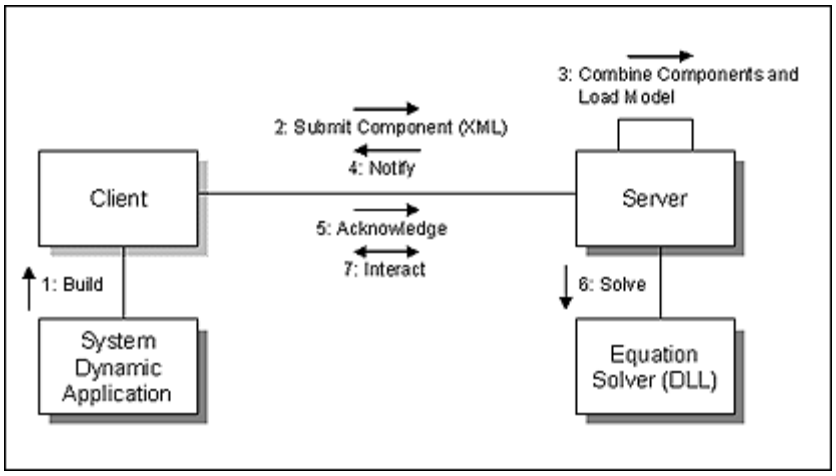

Figure 3. Structure and behaviour of Collaborative Framework

The sequence of steps is:

Step 1: The user builds a model using a system dynamics application that supports the concept of components, and this model is then converted XML format file. As of yet, support for XML version of models is not present in the main vendors systems but it is likely that future version will have this capability. For our initial version of the system, it is expected that we will implement a simple model builder, but our intention is to provide seamless integration with all System Dynamics vendor applications.

Step 2: This model component is submitted to the Server

Step 3: The server, having knowledge of all the partners in the supply chain, combines this model (with reference to the integration points), and then loads this into memory.

Step 4: The server notifies all clients that the model is loaded.

Step 5: The clients acknowledge they are ready to run the model.

Step 6: The model is solved using a problem-solving engine (most probably provided by a System Dynamics vendor – for the first version of the system we will use the Vensim DLL, but because of the modular design, this will be open to other vendors also).

Step 7: The clients simultaneously interact with the model.

In order to focus the development activity, we decided to base the first version of the system on the "beer game" scenario. In this case, we assume that each node in the virtual organisation has the same basic stock, flow and decision rule structure, which is extended from the model

proposed by (Sterman 2000). Within this structure the standard heuristic for inventory ordering is used (adjustment for stock + expected demand). As the basic component does not take the supply line into account, the expected results from this model will mimic the overall beer game with significant stock oscillations resulting from a small increase in demand.

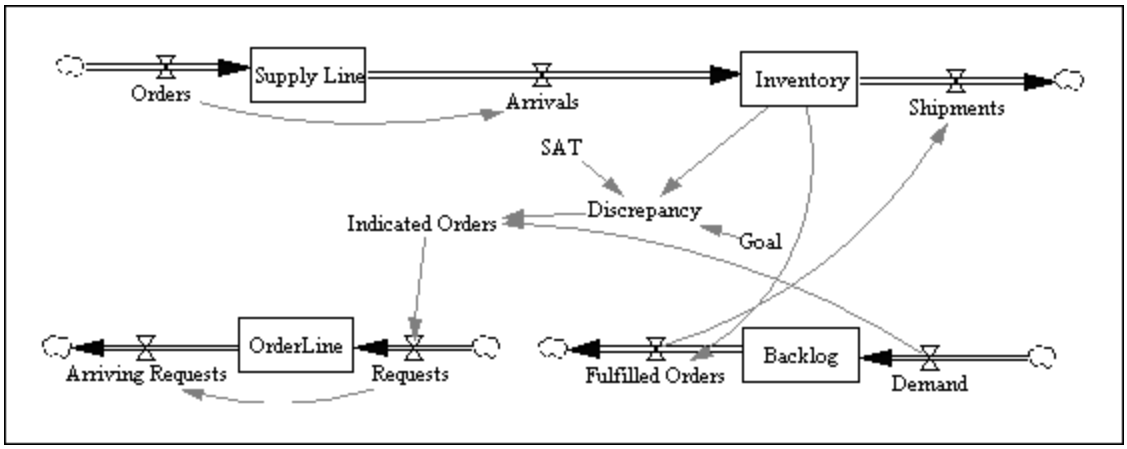

Figure 4. A Model Component for the Beer Game

We view this common structure as a model component – similar to the molecule idea presented earlier. This model component will be expressed using XML – an independent representation format that is application-neutral. It will represent the set of differential equations, and also specify "integration points" in the model, namely those variables that will link other model components. These models must be comprehensive enough to show an accurate portrayal of the users activity, as discussed by ((Yogesh 2000).The simulation servers parse these integration points and this will ensure that the virtual supply chain is totally connected from end–to-end.

Design

The framework is designed using the Client-Server model where a thin client connects to the server. The server implements a service-orientated, component-based architecture. The framework has three high-level components and three high-level services as shown in figure 6.

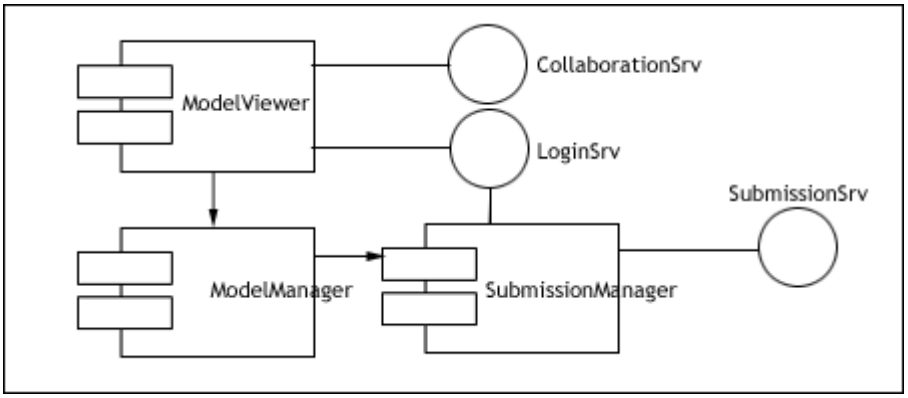

Figure 5. Server Component Diagram

- *SubmissionManager*: This component administers the submission of models from the different clients. When all the models are submitted and validated, all of the individual models are assembled into a single model and prepared for a collaborative simulation. This SubmissionManager implements the *LoginSrv* and *SubmissionSrv* interfaces.

- *ModelViewer*: This component is distributed to the client machine over the web. This can be deployed in two ways:

    1. The *ModelViewer* will be downloaded as an ActiveX object. The user will then only have to point the web browser to the appropriate URL. The functionality is then downloaded. The advantages of this are that there is little distribution overhead, and that there is one central copy of the component. Any updates to the components can be easily redistributed.
    2. The alternative is to package the component as an application that is installed onto the client machine. Although this method will allow a greater flexibility, its shortcomings are that the user is limited to using machines that have the application installed, and higher distribution overheads. If changes are made to the component, it is up to the user to update the changes.

    The ModelViewer component is, in essence, a proxy for the client's portion of the model being executed on the server,

- *ModelManager*: This component will reside on the server and act as a container for the model. It will coordinate input from all the connecting clients, progress the simulation and synchronise output. This client uses the CollaborationSrv interface to interact with the simulation.

- *LoginSrv* is an interface, which allows a client to login and be authenticated.

- *SubmissionSrv* is an interface, which allows clients to submit their models. It can also provide the client with information on the submission status of other clients, and feedback when the model results have been accumulated.

- *CollaborationSrv* is an interface, which enables the client to interact with the simulations (submitting inputs, starting stopping and restarting the simulation).

Presently, the distributed element of the framework is complete, and the model manager is under construction. It is expected to have a first iteration of the system developed by July 2003.

5. Conclusions and Future Work

This paper has proposed the development of collaborative modelling and gaming environments, based on system dynamics, to support actors in virtual organisations. The emergence of data exchange standards, architectural styles and development platforms are broadening the boundaries of distributive system development and enriching the environments that utilise them. Our view is that participants from across the virtual organisation contributing to a 'componentised' system dynamics model. These are aggregated within a server, and the simulation "broadcast" to all participants. Although its framework is still in an early stage of development it has significant scope for development.

1. *Horizontal Development:* Enhancement of the collaborative modelling environment framework offering a wider range of basic functions.
   - Developing a model description that will support the automatic submission of current well-known System Dynamics vendor model formats, i.e. Vensims .MDL format or Stellas .STM format. When a participant is submitting their model, they can bypass the step that involves converting the model into our internal XML model representation format by selecting their model format and calling an automatic converter. The intention is to provide seamless integration with all System Dynamics vendor applications.

   - Adding value to the end-user experience by allowing a larger range of collaborative options during scenarios. For example, not only could the simulation be stopping and restarted; a client could roll the simulation back up. There could also be *chat* or *meeting facility* support.

2. *Vertical Development*: Applying the collaborative framework as a platform for the basis of other functional components to be built
   - Introduction of the role of a host, as described in (Werden et al. 2002) and corresponds to the endpoints of the supply-chain. The Host can have some inputs into a model that are outside the scope of the internal components. For example, in the *Beer Game*, the Host can fluctuate demand. In respect to its business value, the Host can represent the individual/corporation building the system to see how the models components interact with each other.

   - Provide feedback on model performance and a substitution process to the Host. This is to allow weak components to be substituted for a more appropriate component with hope that the change will improve the models performance.

7. References

Berg, Eelco van den. 1999. Collaborative Modelling Systems.

Brown, Alan, Simon Johnston, and Kevin Kelly. 2003. *Using Service-Orientated Architecture and Component-Based Development to Build Web Service Applications* 2002 [cited February 2003].

Duggan, Jim. 2002. A distributed computing approach to system dynamics. *System Dynamics Review* 18 (1):87-98.

FEA-PMO, and SAWG. 2002. *Component Based Architecture Guidance and Recommendations* (January). Fereral Enterprise Architecture Program Management Office 2002 [cited 2003 2002]. Available from http://www.feapmo.gov/resources/CBA_White_Paper_Working_Draft_v1.3.pdf.

NetSim Creator 2.0. High Performance Systems Inc.
McInnis, Kirby. 2003. *Component-based Design and Reuse*. Web Services Interoperability Organisation 1999 [cited March 2003]. Available from www.cbd-hq.com/PDFs/cbdhq990715km_CBDreuse.pdf.
Windows 2000 (Operating System). Microsoft, Seattle, WA.
Microsoft .NET 1.0.3705. Microsoft, Seattle, WA.
MSN Messenger 5.0. Microsoft, Seattle, WA.
Microsoft. *Design Goals for XML in the .NET Framework* 2003 [cited. Available from http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpguide/html/cpcondesigngoalsforxmlframeworkinnet.asp.
Web Service Enhancements 1.0. Microsoft, Seattle, WA.
Milojicic, Dejan S, Vana Kalogeraki, Rajan Lukose, Kiran Nagaraja, Jim Pruyne, Bruno Richard, Sami Rollins, and Zhichen Xu. 2002. *Peer-to-Peer Computing* 2002 [cited January 2002]. Available from www.hpl.hp.com/techreports/2002/HPL-2002-57.pdf.
OMG. 2002. *CORBA* [cited July 2002]. Available from http://www.omg.org/.
O'Reilly, T. 2001. Remaking the peer-to-peer meme. In *Peer-to-Peer: Harnessing the Power of Disruptive Techologies*, edited by A. Oram. Sebastopol, CA: O'Reilly & Associates.
Sterman, John D. 2000. *Business Dynamics, Systems Thinking and Modelling for a Complex World*: McGraw-Hill Higher Education.
Modelling with Molecules 1.4. Ventana Systems, Inc.
Sable 5.1. Ventana Systems, UK.
SableNet. Ventana Systems, UK.
W3C. 2002. *XML*. W3C 1998 [cited September 2002]. Available from http://www.w3.org/XML/.
———. 2002. *Simple Object Access Protocol (SOAP) 1.1*. W3C 2000 [cited September 2002]. Available from http://www.w3.org/TR/SOAP/.
Werden, Scott , Colleen Evans, and Marc Goodner. 2003. *WS-I Usage Scenarios*. Web Services-Interoperability Organization 2002 [cited February 2003]. Available from http://www.ws-i.org/SampleApplications/SupplyChainManagement/2002-11/UsageScenarios-1.00-CRD-02a.pdf.
Yogesh, Joshi V. 2000. Information Visibility And It's Effect On Supply Chain Dynamics, Mechanical Engineering, Massachusetts Institute Of Technology.

Back to the Top