A Qualitative Reasoning System for Behaviours in Interactive Systems

Simon Hartley and Marc Cavazza

{S.Hartley, M.O.Cavazza }@tees.ac.uk
School of Computing, University of Teesside
Middlesbrough TS1 3BA, United Kingdom.

Abstract

This paper describes an environment for the development of qualitative simulation applications in the context of interactive systems: *QPVis*. At the heart of *QPVis* is an implementation of Qualitative Process Theory (QPT) which is designed to be extensible and to allow the definition of new kinds of behaviours / interactions within the world. Another objective was to retain real-time and interactive behaviour whilst allowing multiple instances of the same processes to be active. The *QPVis* is a new platform for the extension of qualitative process theory into the area of interactive systems. We describe the foundations and ideas for *QPVis* and show how it extends QPT to the context of interactive systems, producing a new environment for qualitative reasoning research.

Introduction

In this paper, we present the *QPVis* system which has been developed to use qualitative simulation in interactive systems. The original motivation behind the development of *QPVis* was the modelling of virtual world behaviours. In particular, we wanted to create a system whose performance preserved the interactivity of the virtual world it helped simulating. Another requirement for the system was to provide tools for the authoring, maintenance and reuse of the libraries of qualitative processes developed for different environments. This work is based on Qualitative Process Theory (QPT [Forbus 1993]) and is similar in nature to [Erginac, 2000], an interactive semi-qualitative reasoning system for virtual environments that focuses on model reformation during runtime. On the other hand, the focus in our system is on the real-time performance of the system and its compatibility with interactive architectures using event-based systems. Other QPT Environments are VMODEL [Forbus et al, 2001] and Betty's Brain [Biswas *et al* 2001] which focus upon learning and educational applications. Another QR Environment is HOMER [Bessa Machado and Bredeweg, 2003], based upon VisiGarp [Bouwer and Bredeweg, 2001] (which are an authoring tool and a visualisation tool respectively). These systems where of particular inspiration during the authoring tools development, especially their creation of different diagrams, for state views, transitions, and the different level of views for the systems to aid the user in the modelling for the system (see below).

System Overview

QPVis relies on the synergistic combination of qualitative process theory with discrete systems to express the interactions of users in terms of the physical behaviours affected by them. The software structure for *QPVis* was inspired by [Collins and Forbus, 1989], a system for the production of thermodynamic models.

The *QPVis* system implements a generic framework for an extension of QPT, which added the constructs necessary to extend it towards interactivity. We designed a language specifying the *domain theory* and *scenario* in QPT terms using a syntax which express operations for a virtual machine. These have three main forms: Numerical (add divide), Structural (bind, exclude, own), and Qualitative (Influence Positive, Q-prop). Through this language and the framework primitives; we specify the relations, quantities and conditions under which entities exist. The virtual machine operates in two stages. The first constructs the entities and equations for the model from the domain knowledge. The second simulates the model by executing its equations. The main components of *QPVis* are the "*QPVis* Editor" used for the envisionment and construction of models and the "*QPVis* Simulator", which processes the qualitative equations and individuals data outputted from the editor (see Figure 1).

The *QPVis* Simulator relies on a "virtual machine" which interprets the qualitative syntax to form a scenario and perform simulation. The *QPVis* Simulator also presents in real-time the changes in variables and states in the system, through a series of interfaces. These interfaces display the data in a variety of ways, i.e. in graph form for parameters in the current scenario; it also allows the simulation to be paused and the simulation timescale to be modified. The system can be connected to interactive worlds via UDP sockets. Messages communicated to and from the system are also displayed on the interface. Figure 2 depicts the *QPVis* simulation interfaces running an example application.

The primary goal of interactive systems is to maintain a response rate which is acceptable for user interaction. Such applications in the field of 3D graphics rely on event systems that constantly discretise actions as they occur.

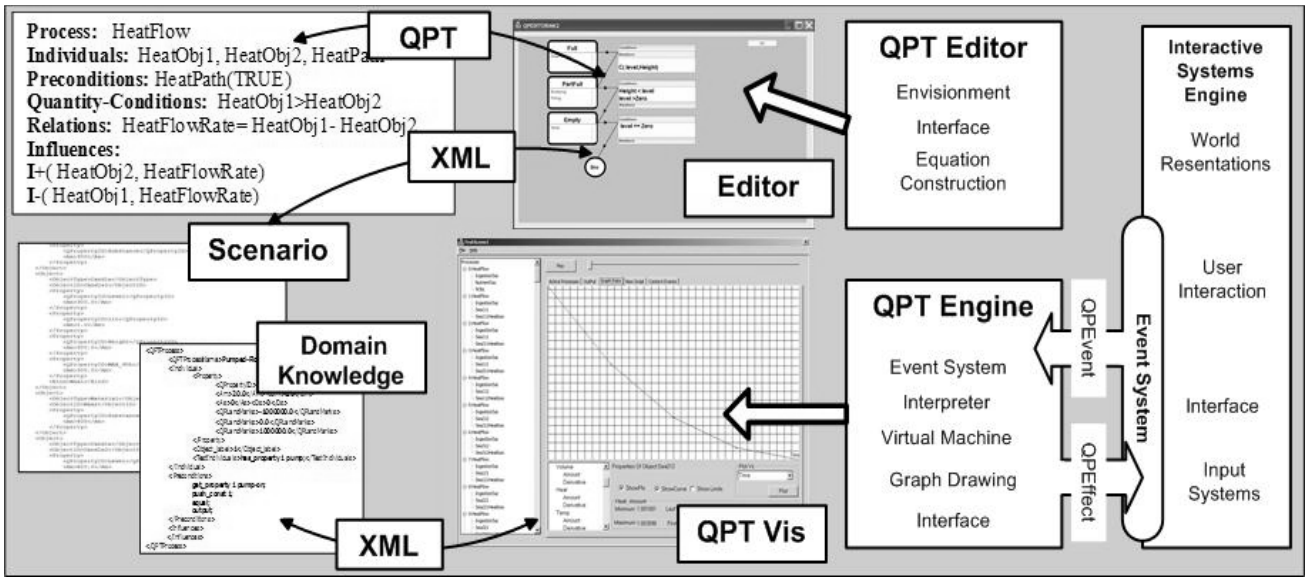

Figure 1: QPVis System

These elementary events (such as `hit`, `touch`, `untouch`, `enter` and `exit`) are also potentially extensible to define events with a higher level semantics, giving an avenue to incorporate an additional qualitative reasoning above them. Our *QPVis* system enhances QPT through the implementation of two new "interactive" primitives: the *qptevent* and *qpteffect,* which are qualitative parameters that encapsulate data, the former for received events, and the latter for generated effects. The *QPVis* System provides an API for interactive systems to use QPT-based simulation to create object behaviours.

Implementation of QP

The implementation of the system uses a generic framework for QPT. This framework is based upon a series of QPT-derived representations which have been expanded to include an event system. We begin by detailing the *basic primitives*, the lowest level building blocks used in the framework. We detail the representations which compose a primitive, the functions it has (both from QPT and our expansion), and the methods it uses to maintain its state. After the basic primitives, we describe the structural primitives which represent QPT, such as *process* and *object*.

Primitives. In QPT, numbers are traditionally represented through the use of numerical envelopes. These numerical envelopes have functions which are implemented in *QPVis* by three representations: historical, relational, and numerical that encapsulates the different behaviours for the qualitative parameter.

The first of these representations is used to encapsulate the histories for the parameter. In QPT, a history is composed of episodes, which occur over an interval of time, and events, which are instantaneous. An episode has a start and an end. In our implementation we have tried to preserve some of original intent for histories [Hayes, 1985] as an ontological primitive used for qualitative reasoning. This is due to the nature of our intended applications, interactive systems, where event systems play an important role and an ontological primitive which is inherently based upon events (and that can be used to form interactions [Bennett, B. and Galton, A 2001]) was naturally advantageous to our implementation. A history representation primitive can be used by the structural primitives to form a superposition of histories [Coiera, 1992(a) and Coiera, 1992 (b)]. The history primitive is intimately related to its associated parameter and landmarks through QPT. This primitive in our implementation is used to compose the structural primitive *qptscenario,* which is formed from the superposition of histories.

The history storage in the implementation represented an extensional challenge for the *QPVis* System. *QPVis* is intended for interactive systems with a high population of objects which can each have a number of processes active over an indefinite period. If the engine is to maintain a full history i.e. a parameters state at the beginning of every iteration cycle the system resources would rapidly be consumed (even for a simple real-time rate of 30 iterations per second). To avoid overflowing the system capacity, the history storage has different modes of operation depending upon the level of detail required.

Relations and Functions. Joined with the histories representation, we have qualitative relations between the parameter history, its quantity, and other parameters. These relations form conditions and mathematical equations. The qualitative relations are expressed by the user as domain knowledge, using our specific framework, and interpreted by a virtual machine. These equations depend on the state

of the parameters parent object, as in the system there are no independent parameters and all parameters must belong to a structural primitive. This structure allows us to identify and solve the problems which are associated with object inheritance creating behaviour exceptions.

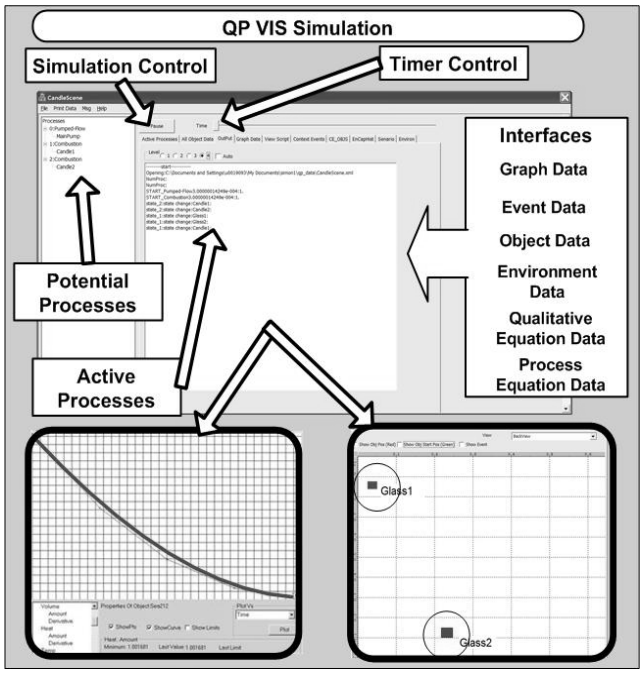

Figure 2: *QPVis* Software Interfaces

A parameter may have equations to represent the following functions: relations, events and effects. Relations are equations in QPT which relate this parameter to another within the parent object. The `qp_events` and `qp_effects` are the qualitative relations which are used to reason with a *qptevent*/*qpteffect* by the parameter. This allows the parameter to respond to different *qptevents* or to generate a different *qpteffect* dependent upon its internal state. For instance, a representation of substance in the system would need to change the relations between its parameters when changing its physical state.

This implementation allows the user to define variations upon the quantity which is used. These derived variations being: the quantity, the set, the vector and the matrix. These quantities are used as a state vector for the system to allow the system to define a series of behaviours for the parameters object. For instance, the matrix quantity which represents a moment of inertia for a ridged body is given by a diagonal matrix. For the "moment of inertia" matrix, each element of the diagonals represents the ease of rotation about that principle axis. If the object was then to undergo a change in mass distribution by a process, filling for example, we could easily change the element for the moment of inertia and make the object "top heavy" or "bottom heavy". This concept was developed from the Qualitative State Vector ideas, used in [Forbus, 1987] for the motion of a ball (i.e. Region S3 velocity - left down). Where here we map the qualitative states to a matrix representations. The representation for a matrix quantity allows the user to specify viscous drag coefficients, electric charge and dipole moments. Thus, instead of creating a series of parameters for the system, we have encapsulated them in the single quantity making for an easier representation of objects behaviours. However, this extra expressive ability suffers from the same lack of compositionality which is common in state vector ontologies.

The vector parameter follows two different implementation paths: the first is based on qualitative vector analysis (QVA) [Weinberg, 1990]. However, event systems provide accurate vector position and velocity data for the 3D world from the event data. In order to avoid loss of data, we have another type which stores vector data, called vector quantity. This second implementation receives events and translates them into our vector quantity representation. This allows us to reason with magnitudes, create a QVA (if necessary) and still keep the vectors information.

Limit Points and Landmarks. The functions for the relations, events and effects for all the parameters are dependent upon the parent's state. However, a parameter maintains an internal state within these parent states which are governed by Limit Points. Passing a Limit Point changes a parameters internal state and generates a "state change" *qptevent* for that parameter. The Limit Points change in state for a parameter allows the parameter to define a new series of relations for the quantity. For instance, a parameter may change relations from monotonically increasing to monotonically decreasing when it passes a Limit Point. We have specified that Limit Points are a strictly ordered named set of values, which are static upon the domain of the quantity and cannot be added or removed. This is due to the dependence of the parameters internal state and hence qualitative equations on the Limit Point.

The other entity upon the numerical domain for the system is a *landmark*, which is not used in the control of the parameters internal state, but generates a "passed landmark" event for the system. Within the system, we treat a *landmark* as a quantity *sui generis*, although it is a temporary one (as a *landmark* is not a static value like Limit Points are). A *landmark* may have relations applied to it and may be dynamically added or removed from the parameters domain. This treatment allows a specific *landmark* to respond to *qptevents* in the system. We have two representations for *landmarks* in the implementation both the *landmark* representations can form relations between history and *qptevents* within the parameter. The first representation is the expected *landmark* which is affected by external parameters. For instance, in the implementation of a melting phase change under varying pressure conditions, one would need to calculate any changes in the melting point *landmark* as the pressure parameter varied. The second *landmark* has a more temporal relation in our representation. A relation we formed for this *landmark* type is the ability to form its own time representation dependent upon system elapsed time. This *landmark* is typically used for the representations in

encapsulated histories, where it is useful for the encapsulated history to maintain its own timescale. This method is used primarily to avoid the creation of "temporary objects". For instance, a landmark used in an explosion to control the debris of the explosion may wish to control the original object (destroyed by the explosion) for a time sample after the actual event. Since the object itself may no longer generate events, as it is no longer a valid object, we add the landmark as a new quantity, and its decay controls the decay of the debris.

Objects and Processes

In QPT the most significant representations are the object (defined as a collection of relations and parameters) and the process (the primary agent for parameter evolution). We have described a number of basic primitives and their relations. Within the system, these basic primitives are combined together by a series of "structural primitives" which are representations of the individuals within QPT. The base structural representations we have formed are *qptobject*, *qptprocess*, *qptEncapsulatedHistory*, and *qptscenario*. The *qptobject* and *qptprocess* are the base representations for object and process within the framework. *A qptEncapsulatedHistory* represents a standard QPT encapsulated history or a collection of *qptevents and qpteffects. A qptscenario* is the representation of a collection of histories within the system. Structural primitives, unlike basic primitives, can exist independently within a scene.

In QPT, the Individual is the most significant structural reasoning object in terms of forming object representations. This Individual is represented by the *qptobject* within the implementation. The *qptobject* is given by a hierarchy of structural objects which are representations for the object within the different parts of the system. The qualitative equations we implement for a *qptobject* are:

- Individual View (IV) qualitative equations which facilitate the changes in the objects internal state.
- `qp_event` and `qp_effects`, which perform the event reception/generation respectively. These are actually handling interactions for the object.

In addition, within the object representation we described an additional series of object logic conditions, which are expressed during authoring in order for the user to assess the current state of the object. These conditions we labelled Activity, which are statements about the state of an object. During runtime these can be created dynamically, where they are called *qptInferences*.

The Activity functions are a set of purely conditional operations forming qualitative conditions that we say the object possesses. These are used to enrich the state description. For Instance, a container may have the Individual views of "Empty", "Part-Full" and "Full" representing the discrete values of the `Amount of substance` contained in it. It may be required to further describe the "Part-Full" representation by adding the Activity Filling/ Empting for the Individual's Description. An Activity within the system is then not a complete state for the object but an additional statement which adds to the Individual View. An Activity can generate a *qpteffect* for the system allowing the additional information to be conveyed to the user.

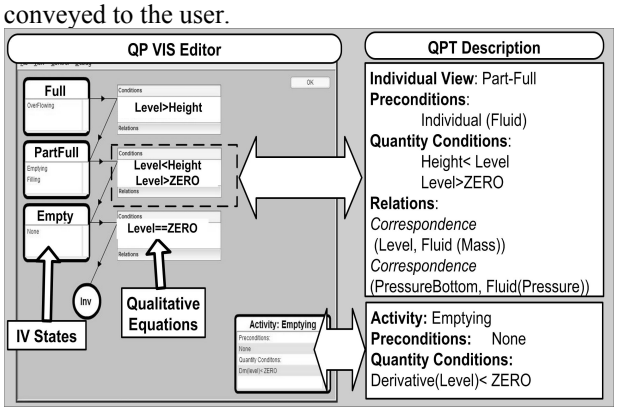

Figure 3: Individual View for Part Full Geometric Containers and in Editor.

The Activity function is our method for avoiding multiple inheritance hierarchies for objects. In the previous example we could choose to model the filling/emptying as states in an associated object derived from the base container. This adds more levels of indirection to the model. The Activity approach we have used does not directly affect the objects state. Since, the Objects states control the Individual Views the Activities impact is limited to statements and to affects on parameters.

In order to support interactivity, an object's preconditions are now given by two qualitative equations instead of the single QPT precondition. The system forms a new predicate for objects and parameters, the `qp_event` qualitative equation. This predicate is designed to perform qualitative operations on a received interaction event which has been transformed into a *qptevent*. The qualitative equation performs *qptevent* translation for an object by interpreting the *qptevent* parameters (Figure 4, (1)). Then the object broadcasts the modified *qptevent* to its individuals (Figure 4, (2)) and then to its parameters (Figure 4, (3)). Either of these stages may add new events or remove the original event. This allows any object to intercept any event before its parameters `qp_event` functions can respond to it. This system allows a chain of events to occur for an object as each of its parameters receives its broadcast (as shown in figure 4).

The second qualitative equation for a *qptobject* is its `preconditions`, (Figure 4, (4)), which tests whether the parameter within the event or a parameter flagged as external has satisfied the requirements. For instance, an `aligned` event is used to generate a "fluid path connection" between two containers, allowing a flow process to be triggered. A "Flow Path Object" would translate the

qptevent, setting the bidirectional nature of the connection, in the case of a tube, but only unidirectional for a valve.

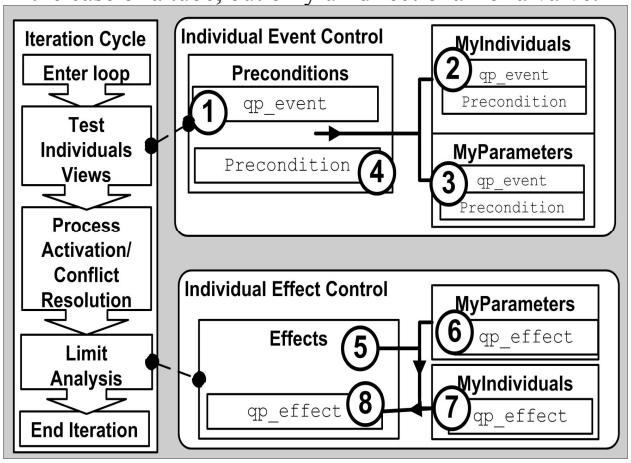

Figure 4: Event Control- Flow.

After simulation, `qp_effect` is used for event generation**.** The creation of the *qpteffect* takes place through the objects qualitative equation for `qp_effect`, which translate the *qpteffect* for a specific type of effect. As *qptobjects* are related to their parameters, the broadcasting of effects for an object begins with its parameters, (Figure 4, (6)). The parameters' effects are generated by a landmark, limit-point or by the `qp_effect` equation, and are then broadcast to any individuals in the parent object (Figure 4, (7)). The object calls its `qp_effect` to perform any modification to the *qpteffect* (Figure 4, (8)). For example, in visualisation of systems, it is often important to show the parameters values relative to one another. If the system ranks the object by a qualitative value, the effect generated would then be made relative by the object applying that value to the parameters generated effect.

An *qptobject* structural primitive may be given specific properties relating to the spatial nature of the object. An object can be said to be a "region" type which is a representation of the object for the spatial system. A region can have two types either distributed parameters System [Lundell, 1994] or Region Connections [Bitner and Stell, 2000]. The first system relies on *QSet* primitive that represents an aggregation of neighbouring points sharing the same parameter value. The sub-regions are represented by the QSet state which is formed from the aggregation of the closest points. In the second system, the primary logic is that the Regional Calculus controls the states and variable of the object, thus limiting which processes may affect the object. The Regions internal states in this system are given by one of the following states: `Disconnected`, `External-Connected`, `Partial Overlap`, `Tangential Proper Part`, `Non-Tangential Proper Part` and `Equal`. These are related to spatial reasoning between two connected objects.

The *qptprocess* primitive is the primary agent for change within QPT. The difference between our implementation and standard QPT implementations is the inclusion of the qualitative equation `qp_events` and `qp_effects`. Qualitative process relations may be represented in our system by either a purely qualitative set of equations describing the relations, or by a series of mathematical functions which describe the relations. The qualitative relations are expressed by the operators given in [Collins and Forbus, 1989], using constructs such as ordered-correspondence O*, whereas the mathematical expressions are inspired from the formalisms used in CML [Falkenhainer *et al*, 1994] developed from the KIF [Genersereth and Frikes, 1992] numerical descriptions. In our mathematical functions, we ignored the differential equations from CML/KIF which we assign to representation by processes, represented the complex space as a new quantity and implemented the other functions as instructions such as sine, cos in the virtual machine. We also expanded the mathematics to include common statistical operations such as standard deviation, harmonic mean. The reasoning behind inclusion of statistical operations is it enhances our distributed parameter spatial reasoning system. In the system, we assumed a semantic representation for the regions parameters. For example, high pressure concentration, low pressure concentration used to form regions. Using statistical analysis we form a qualitative state analysis of the data within in a region.

Simulation Cycle

Qualitative Physics Instantiation. The *QPVis* instantiation for the structural primitives begins with object instances, and their associated individual views, it then instances any encapsulated histories or predefined processes. The second stage is for the dynamic elements, for which instantiation is performed by analysis of the individuals and comparison to individuals in the scenario. These elements are usually processes which have not been marked as requiring specific instantiation.

The dynamic element initialisation stage begins with `TestIndividuals`, where names are taken as denoting *individuals* which satisfy a set of predicates, either within a processes individuals list, or within a semantic structures individual list. The next stage in the instantiation of a process is a test upon the characteristic parameters for the object. This excludes individuals with certain combinations of parameters Limit Points or *landmarks*. The next stage for the semantic system is to alter the characteristic parameters to match its conditions**.** The difference in the expressions here being the nature of the envisionment performed for the object. A process by our definition cannot alter an object to match its predicates, whereas a semantic representation would add/remove the necessary parameters for a representation. This stage for the semantic representation would allow the object to participate in new processes which it previously would have been excluded from. However, a process may instantiate new individuals if their prototype was a dynamic type. This stage forms processes which previously could not occur. These two

systems can be combined to produce new envisionment. For example, the semantic system can be used to describe a kitchen oven as an "immovable infinite heat source" and a baking tray as a "movable heat sink". In the semantic description, we follow the logic for creation of heat paths to be immovable-movable, movable-movable but not immovable-immovable. This description for the heat flow paths allows the process to be generated for all categories of "heat object" and exclude the relations between immovable "heat objects", unless a specific connection is made between them. This logic is a useful construction tool; the drawback is the need to keep the synchronisation between objects in the qualitative simulation and the world. After this stage is completed, we have instantiated all of the process which may become active due to user interaction, and all processes which are initially active.

Simulation Phase. During the simulation phase, the constructed scenario is used to reason about the evolution of properties and spatial properties of the system.

In implementing QPT, the simulation phase will follow the same basic stages for any QPT system these are given by [Forbus, 1988] as:

1. Load Domain Model
2. Load Scenario
3. Find View Structures which are active
4. Find Processes which are active
5. Resolve Influences
6. Perform Limit Analysis

Our implementation expands upon the basic stages by adding the new stages for the qualitative equations for `qp_events`, `qp_effect` and *`Activities`*. In addition to these new qualitative equations, the implementation has modified the stage (3) for the *region* type *qptobject* structural primitives. For instance, spatial regions are handled differently as their states depend on the form of calculus we are now using. In the spatially distributed parameter, the region is a dynamic cluster of parameters which have similar values. Since we perform this stage after the calculation for individual views, we may either use the *qptobject* individual view or we may use the region and its calculus to alter the objects and recalculate an individual view before the processes qualitative reasoning.

The activation of a qualitative process stage (4) is achieved in two steps. The preconditions and quantity conditions are first tested, and then the system determines if any conflict has arisen upon the parameters. If no conflict has arisen by either direct or indirect means the process becomes active. Otherwise we form a conflict for stage (5).

Variable Updating. The first update occurs at the beginning of the simulation cycle when the *qptevents* are received. These events are propagated through the Object hierarchies event system(Figure 4) and affects/updates those parameters which are tagged as being external to the QPT System or creates new events for the system. The update stage is to determining the IV's for the objects. To determine any state changes within the system. The final update occurs after the resolution of influences stage, and is used by the system to generate the *qpteffects* within the system for parameters.

Termination Conditions. The termination conditions for a process are when it fails either preconditions or quantity conditions. In addition, the process will be made inactive if any of its individuals become invalid. An object may become invalid in a number of ways, the methods of invalidation for an object are:

1. Having Individual Views and failing them all, including its default individual view.
2. Its parameter reaching an invalid state i.e. INF or NAN, with no event to handle it.
3. Its parameter moving beyond a strict Limit Point i.e. ZERO for Non Positive or Max Value for Other parameters.
4. User Interaction "destroying" the object.

The conditions for an encapsulated history follow the same rules, with the encapsulated history being dependant upon the Iterations which it has been through (instead of the individual views).

Envisionment and History

The most common problem during the envisionment using QPT is the change in representations that can occur when different model fragments are combined. The *QPVis* aids the modeller in avoiding this in three different ways, the first being the object hierarchy (a standard method for solving such problems). The second method uses the classification of objects to advance the object hierarchy system with a model construction algorithm which uses a system of predicates similar to [Faulkenhainer and Forbus, 1990]. For example, with classification and `TestIndividuals` in the model construction, we form our version of the `consider` predicate. This combination allows a modeller to create a new level of granularity for the model. To achieve this, the modeller would introduce a new Individual that is the parent of the considered Individual. Since any Individual's state is dependent upon it parents state and qualitative equations are selected are dependent upon that state.

A modeller can then choose a simple method for the inclusion of both models within the system by simply selecting suitable criteria for the switching between the parents individual views. For instance, in combining systems which model phase changes by:

a) Unchanging Limit Points for the "Critical Point", Predicated upon the *simple-substance* object.

b) Modelling Gibbs phase rule in it simplest form; increase pressure always favours the formation of denser phase.

To combine these fragments, we define a parent object to *simple-substance* used in fragment a) which would include the quantity condition logic to model the changes, only in the varying pressure conditions. The operating assumptions for the system would then be expressed though operations upon the new object. This object would include a new quantity *CriticalPoint* used by the process "Gibbs Phase

Rule". In doing this, the new parent would need to intercept the now superseded event pass Limit Point for the simple-substance generating the effect upon passing *CriticalPoint*.

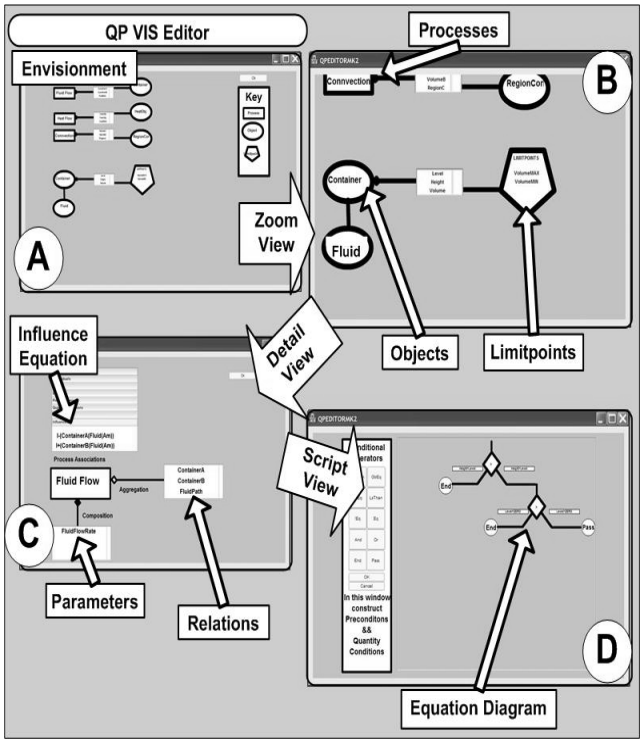

Figure 5: Visualisation of QP Envisionment within Editor

The final way *QPVis* aids in the prevention of envisionment problems is to aid the modeller by providing methods for the automatic depiction of models within the authoring tool. The depictions of states and transitions within the editor, which are generated as a fragment, is constructed to allow the modeller to quickly assess the gain size of the methods used. This is achieved by multiple levels of view in the editor, allowing the view of the scene (Figure 5:A) which allows the user to zoom to the area of interest (Figure 5:B) and to select Individuals to give a detailed view of the individuals relations (Figure 5:C). Form here, logic diagrams of Equations can be shown (Figure 5:D).

Authoring

The creation of authoring tools for the system requires methods for visualising the entities, as well as editors for the creation of qualitative equations in ways which are quick to understand and easy to use. These authoring and creative stages are preformed in a separate entity, the *QPVis* Editor System, which uses an interface written in ActionScript and MFC to create diagrams and give interactive methods for the creation of the qualitative structures. These qualitative structures can be written to an XML file for use in the simulation system or imported as fragments.

The system produces four XML Files; each one encapsulates a particular knowledge or behaviour. The first file is a Domain knowledge file which contains the qualitative equations and definitions for the processes, objects and encapsulated histories including the qualitative structural equations for their instantiation. The next is a scenario file, which contains all the information for the instances within the environment, such as default values, overridden parent behaviour, and any semantic properties added to the object. The third file defines the semantic properties and relations for an object. The final file produced is for formatting messages and contains specific messages for creation as qptevents/ qpteffects.

The creation of entities within the system is approached by first defining an Individual, usually a *qptobject*. The *QPVis* Editor allows the generation of the qualitative equations and IV's for the *qptobject* via a series of visual representations. The interface draws a logical representation for the qualitative equations which are purely conditional. For instance, within the individual view for a container, we construct a qualitative equation for the "Part-Full" condition as shown in Figure 6.

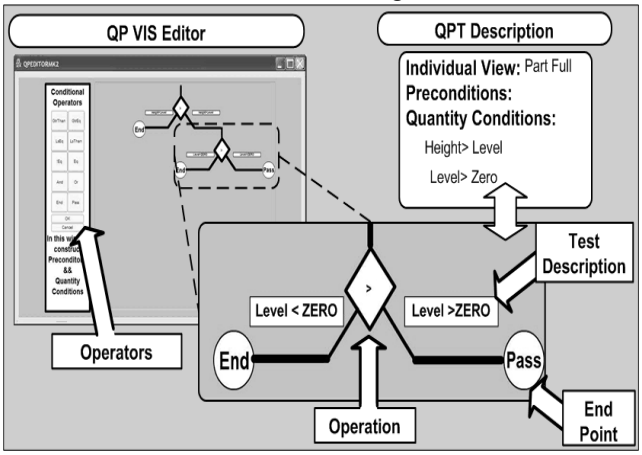

Figure 6: Logic Construction for "Part-Full"

After the construction of the individual view equations, the system draws a state representation designed to be similar to the state diagrams in UML for the qualitative equation. For instance, geometric containers preconditions are represented in qualitative process theory and diagrammatically in figure 3: IV for "Part-Full".

The system provides similar interfaces for all qualitative equations in the system. The construction of qualitative processes within the system uses similar interfaces. The interface shows the completed process as an individual and allows the user to browse a process and the Individuals it has relation with, as shown in Figure 7.

To create a model, the user can specify a *Semantic Object* or an *Instance Object*. For an *Instance Object*, the user selects a prototype from the list of QP prototypes objects which have been previously defined. Then, the user can either assign the default values to the object or select a series of semantic characteristics for the system from a tree of semantic properties. To visualise the entire scenario the system automatically draws a diagram representation of the

individuals in the system. The system draws a representation of all the individuals and shows the first level of their associations using a different depiction for each individual type. The user can interact with these depictions creating new associations or showing more detailed views. The diagrams are drawn for the instances as well as the knowledge base which allows the user to browse the entire scene.

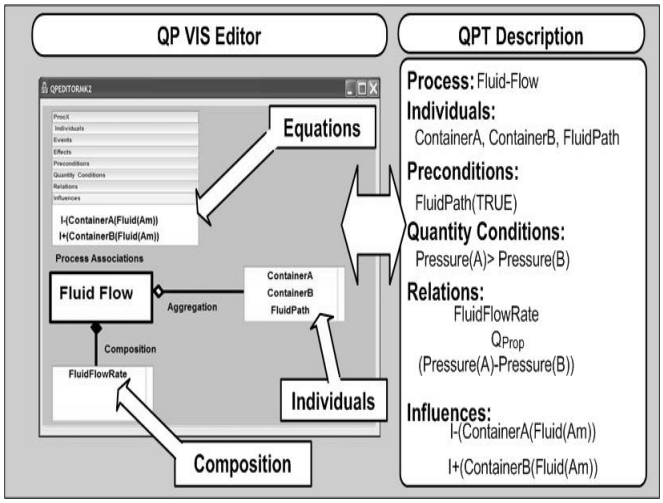

Figure 7: Browsing a Process in Editor.

Debugging

In the current implementation of the *QPVis* system, there are a number of ways to debug and trace a scenario as it is simulated. The interface provides a number of methods to alter the structure of a scenario as it runs. The first being an input window tool to alter the value of or derivatives of a parameter(the changes are applied before the limit analysis but after simulation). This allows the user to change the values for any parameter in the system and see the qualitative results as they propagate into the system. For instance, the user can change pressure variables and assign paths which would allow fluid flow process activation. The user can also form a limited Activity equation (via *qptinference*) which behaves like a watch statement within the system.

To aid in debugging the system, we have provided a timer tool. The system has a mode of operation which allows the iterations per second to be altered up to a maximum value or for the user to define a time step these methods allow the user to speed through the simulation to find the outcomes of changes in the system.

The interface assists the debugging process by displaying a graph which shows the evolution of the amount, or derivative for the selected parameter. The graphs can plot two values so a comparison can be made between their changes. This provides a useful series of visual tools for the user to be able to quickly visualise the evolution of the parameter (Figure 8).

To aid the user to find errors in the envisionment, the system implements a message window which shows any system events such as invalid object, individual view states, or parameter overflow.

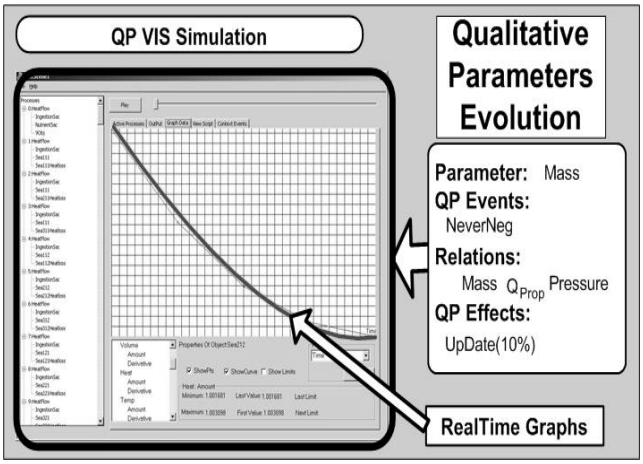

Figure 8: Parameter Graphs in *QPVis* Simulator.

Results

The *QPVis* System has been used in a number of implemented scenarios, for instance in artificial life applications [Hartley *et al*, 2005]. The different implemented briefs are using some of the typical QPT Simulation fragments(such as thermodynamic properties, fluid flow properties and concentration/ mixing properties and combustion) and the envisionment. These typical simulations where expanded for the use in the QPVis Simulator via the effects and events in order to display simulation results [Hartley *et al*, 2004]. The fragments used include: Container Objects (as shown in Figure 3), Heat Objects and Fluid-flows. In these systems, the QP *Landmarks* and Limit points are used to visualise the effects of the simulation through [Cavazza et al, 2003].

Our artificial life simulation involves 53 objects which are both standard objects and spatial objects including Individual Views and parameter relations. This scenario has 22 processes defined which give over 150 envisioned potential processes for the single artificial creature. In this environment, the simulation time for one iteration is about 5.68 ms averaged over 10000 Iterations.

An example of the *QPVis* simulation system showing a typical simulation using QP features appears on Figure 9. In this example, the environment modelled is a creature ecosystem based on a liquid environment with convection current processes and mixing processes which alter the concentration of nutrients within the environment. The QPVis system also simulates a chemotactic response for the creature giving it sensors for the concentration of nutrients. The simulation then starts the Locomotion processes within the creature add selects the target region which has the highest concentration of nutrients.

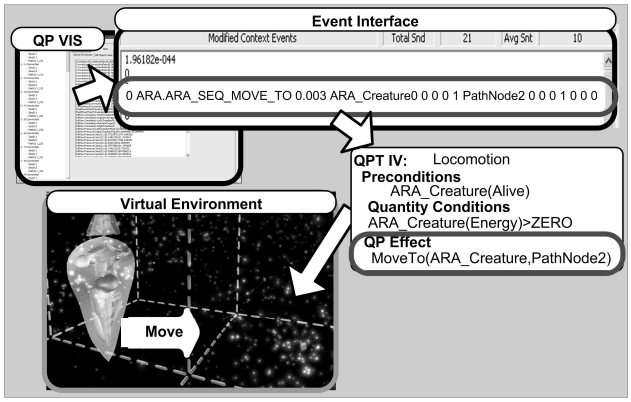

Figure 9: *QPVis Scenario* Simulation Effects Example

Conclusions

The QPVis platform combines qualitative process theory and real-time event systems. Thus it proposes a novel type of tool to experiment with qualitative systems in applications such as virtual environments or robotics. The system has successfully been used to implement a library of processes and behaviours used in different virtual environments and supports multiple model fragments, however the main limitations concern its lack of flexibility when mixing primitive representation. Currently, we are investigating algorithms to automate the integration of different grain model fragments, using different types of primitive representation.

Acknowledgements

This work has been funded in part by the European Commission through the ALTERNE project (IST-38575).

References

Bessa Machado, V. and Bredeweg, B. 2003. *Building Qualitative Models with HOMER: A Study in Usability and Support*. In Proceedings of the 17th International Workshop on Qualiative Reasoning, QR'03. P.Salles and B. Bredeweg (eds), pp. 39-46, Brasilia, Brazil, August 20-22 May 17-19, 2003.

Biswas, G., Schwartz, D., Bransford, J., and The Teachable Agents Group at Vanderbilt. 2001. *Technology Support for Complex Problem Solving: From SAD Environments to AI*. In Smart Machines in Education. Forbus, K. and Feltovitch P. (Eds.) Pp72-97. AAAI Press MIT Press, Menlo Park California, USA.

Bittner, T. and Stell, J. G. 2000. *Approximate Qualitative Spatial Reasoning*, Spatial Cognition and Computation, vol(2),pp 435-466

Bouwer, A. and Bredeweg, B. 2001. *VisiGarp: Graphical Representation of Qualitative Simulation Models*, In G. Biswas(Ed.), Proceedings of the 15th International Workshop on Qualiative Reasoning, pp. 142-149 , San Antonio, Texas, USA , May 17-19.

Bennett, B and Galton, A. 2001. *A Versatile Representation of Time and Events*, Fifth Symposium on Logical Formalizations of Commonsense Reasoning (Common Sense 2001), New York, May.

Cavazza, M., Hartley, S., Lugrin, J.-L. and Le Bras, M. 2002. *Alternative Reality: Qualitative Physics for Digital Arts*, Proceedings of the 17th International Workshop on Qualitative Reasoning 2003, Brasilia, Brasil.

Coiera, E.W., 1992. *Qualitative superposition*. Artificial Intelligence,56: 171-196.

Coiera, E.W., 1992. *The qualitative representation of physical systems*, The Knowledge Engineering Review 17(1) 55-77. 28

Collins, J. and Forbus, K. 1989. *Building qualitative models of thermodynamic processes*. In Proc. 3rd Int. Workshop on Qualitative Physics, Stanford, CA, May.

Erignac, C. 2000. *Interactive Semi-Qualitative simulation*, proceedings of 14th international workshop on qualitative reasoning, Morelia, Mexico. June.

Falkenhainer, B.; Farquhar, A.; Bobrow, D.; Fikes, R.; Forbus, K.; Gruber, T.; Iwasaki, Y.; & Kuipers, B. 1994 *CML: A Compositional Modeling Language*. Knowledge Systems Laboratory, September.

Falkenhainer, B and Forbus, K.D. 1990 *Setting Up Large-Scale Qualitative Models.* Qualitative Reasoning About Physical Systems, 553-558, Morgan Kaufman.

Forbus, K.D., Carney, K. , Harris, R., and Sherin, B.L. 2001 *A qualitative modelling environment for middle-school students: A progress report.* In G. Biswas(Ed.), The 15th International Workshop on Qualiative Reasoning, pp. 142-149 , San Antonio, Texas, USA , May 17-19.

Forbus, K.D. 1993. *Qualitative process theory: Twelve years after*. Artificial Intelligence, 59, pp. 115-123.

Forbus, K. D. 1988. QPE: *Using Assumption-based Truth Maintenance for Qualitative Simulation* Int. journal AI in Eng, pp. 200-215.

Forbus, K.D. 1987. *The qualitative process engine*: A study in assumption-based truth maintenance. In Qualitative Reasoning Workshop Abstracts. Qualitative Reasoning Group, University of Illinois at Urbana-Champaign.

Genersereth, M.R and Fikes, R.E. 1992 *Knowledge Interchange format,* V3 Reference Manual. Logic Group Report logic -92-1, Computer Science Department, Stanford University, June.

Hartley, S. Cavazza, M. Lugrin, JL. Le Bras, M. 2004. *Visualisation of Qualitative Processes.* Proceedings of the 18th international workshop on Qualitative Reasoning 2004, Chicago (Evanston), USA. August 1-4.

Hartley, S. Cavazza, M. Bec, L. Lugrin, Simon Hartley, Marc Cavazza, Louis Jean-Luc Lugrin, and Sean Crooks. 2005 *Qualitative Simulation of an Artificial Life Ecosystem.* 19th Workshop on Qualitative Reasoning, Graz, Austria. May 18-20.

Hayes, P. J. 1985. T*he second naive physics manifesto*. In Formal Theories of the Commonsense World, J. R. Hobbs and B. Moore, Eds. Ablex, pp.1-36

Lundell, M. 1994. *Qualitative Reasoning with Spatially Distributed Parameters.* International Workshop on Qualitative Reasoning about Physical Systems. Nara, Japan.

Weinberg, J.B., S. Uckun, and G. Biswas. 1990. *Qualitative Vector Algebra* Fourth International Workshop on Qualitative Physics. Lugano, Switzerland: p. 82-96.