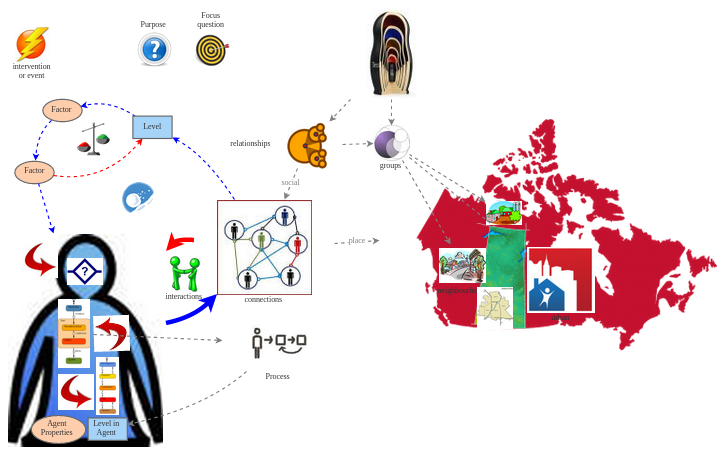
**Collaborative Qualitative  
Modeling Web Application**

**Document version 1.2**

**Motivation**

Qualitative modeling is increasingly widely performed by organizations seeking to make decisions in the context of great complexity.  These models consist of a variety of simpler graphical elements drawn in a similar place.  Despite the fact that such modeling is almost always conducted by larger – and often physically distributed – teams, the current software to support it is almost always desktop based, and uniformly supports use only by a single user at a time, and many teams just build physical models using materials such as yarn, pushpins, flipcharts, putty.  Either desktop software or a physical model impose problems both when modeling brainstorming sessions are conducted in a single room (when only one person serves as the “gate keeper” to the emerging model) but especially for the case when participants are joining from remotely.

It would be highly advantageous to have software that supports building up even simple diagrams in a collaborative fashion.  While the details of what is created is very different, the collaborative interaction envisioned would be similar at a high level to that offered by google docs, Etherpad, drawpile, etc.).



In this envisioned collaborative software, multiple users accessing the system through web browsers would be able to interactively and simultaneously add elements such as connections, pre-defined but resizeable and colorable shapes and pictures, text, and – critically – connections between them onto a diagram.  A given user would see the elements added by others as they are being positioned.

**Feature Set**

The envisioned solution would have a quite small (but very desirable and powerful) feature set

* Adding in text associated with a “V” icon (indicating a variable)
* Adding in text associated with a “P” icon (indicating a parameter)
* Placing “text” in chosen fonts
* Connecting items (including variables above, parameters, as well as items below) using a “link”.  can be curved.
* Saving a diagram in some way
* Loading a previously created diagram
* Adding in one of a small number of shapes and images
  + A person image icon (e.g., stick figure)
  + A stylized silhouette of a person's torso and head
  + Arbitrary image (e.g., PNG or JPG)
  + Network image
* Adding in UML-style statechart elements
  + Statechart entry point
  + States
  + Transitions
  + Sinks (final states)
* Adding in elements of a stock and flow diagrams.  Here, a given end of a flows can only be connected with stocks or with nothing (showing a cloud icon)
  + Stocks (a rectangle, including text labeling such)
  + Flows (a double line with arrows at the end, and a stylized geometric “valve” shape)
* Associating the regular links with polarities and possible perpendicular delay marks
* Placing clockwise and counterclockwise loops images, with a “+” or “-” indicated in the center to indicate the polarity of the loop
* Selecting and moving sets of the previous (with movement of a variable automatically moving along any links connected to that variable).
* Providing either of viewing or editing permissions to another user (before this is implemented, the system should allow anyone who has the link to view and edit the diagram).
* Setting the defaults for fonts, etc.
* To mark a given item with a “visibility level”, and to allow a given user to only view items of below a certain visibility level (e.g., visibility level 1).
* For states and certain other items to serve as containers for other items (i.e., states and transitions can be placed inside other transitions, can place items inside of the torso & head shape).
* Capacity to show version history and look specifically at earlier versions
* Picture
* undo functionality
* free-form sketching
  + ability to relocate text relative to an objects
* Disqus like comments on
  + elements
  + versions

Every item added except a comment has optional name associated with it

    originally, label is in a certain default place (beneath item)

\*

Undo

Persist to JSON or text framework

Print

Interventions

As a general rule, the above should be able to be resized and colored.

Notes on connections:  There are 3 types of connections referred to above:  Links (quite general), transitions (specific to states of a statechart), and flows (specific to stocks & flow elements).  Links should be able to connect up many types of things – variables, states of a statechart, stocks and flows, parameters, the icons indicating a network or other things. All types of connections (regular links, transitions between states of a statechart,  flows between stocks in a stock-and-flow diagram) need to be “sticky” in the sense that their ends are not merely visual overlapped with objects, but are instead actually connected to those objects (so if the object connected to the endpoint is dragged, the end of the connection is appropriately dragged with that object).

**Technologies**

Based on past research and commercial success, this effort will need to build atop well-established web-based collaborative software technologies that go by the name “Operational Transformation” libraries and technologies (see <https://operational-transformation.github.io/index.html>). These notably include javascript libraries such as ot.js, and node.js.  Naturally, supporting this and saving/loading of models will require a database-backed web application (with an associated virtual machine).

**Prioritization**

For reasons discussed in class, it is important that the project be rolled out in an incremental fashion, with higher-priority, easier to implement and potentially riskier items being explored first.

For this reason, the bulleted list provided above is given in the order of decreasing priority, with the understanding that the project may only reach partway through the list in the course of the semester.

**Client & Customer:**Nathaniel Osgood   nathaniel.[osgood@usask.ca](mailto:osgood@cs.usask.ca)

            Geoff McDonnell    gmcdonne@bigpond.net.au