ValueMatrix Implementation

Latest: 20170520

Background

We need the ability to *rate* various contributions to the knowledge garden in terms of their *apparent value* to the garden. But, *value* of any kind is always context dependent. Much of what follows entails *game design* in the sense that games rely on scoring mechanisms, among other things.

Our interest here lies in the design of a *ValueMatrix* which is a many-dimensioned artifact which stores values associated with any node in the knowledge garden. A trivial example is given here to illustrate what we have in mind. This example is a *JSON* object, with keys and their values.

```
{
        “KeyA”:”10”,
        “KeyB”:
                [ 10, 29, 3 ],
        “KeyC”:
                [ { “keyAA”:”20”,
                    “keyBB”:”30”}
                ]
}
```

Our interest lies in the design of that structure.

What we will do is look at some scenarios. In each scenario, there are actors, and relationsengaged.

Actors:

- An entity which creates a node
- Nodes

Relations

- Relations between the created node and any other node

Terms

- Actor is an entity
    - A human
    - A guild
    - A software agent
- Node is a *topic* in the topic map
- *TotalValue* is the total value accumulated by a given node
    - Remains to define the algorithm to compute that
- Bucket-Brigade algorithm is a way of distributing value along a chain of linked nodes, with a declining value given to nodes further away from the origin

Before we look at scenarios, let us imagine some scoring rules.

Scoring Rules

These rules are just here for the sake of conversation. Each is associated with a particular *use case*, so use cases are important to us.

Use Case 1 Original node creation

- Creating a node is worth 10 points as an initial value to the node
- Creating a node is worth the TotalValue (dynamic) to the node's creator

Analysis

Some keys:

- InitialValue
- TotalValue

Use Case 2 Re-using an existing node by *transclusion*

- Transcluding a node, to the entity which performs the transclusion, is worth the transcluded node's *total value*.
- Transcluding a node in a conversation affects the value of the entire parent-child tree above the transcluded node
    - Here, we imagine a *bucket-brigade* algorithm in play

Analysis

Some keys:

- Transcludes
    - Compound list value:
        - Locator of node to which transcluded
        - value from that node

Use Case 3 Re-using an existing Tag by simply applying that tag to other nodes

- Each time a tag is re-used, the tag itself has a *reuse* value incremented by 10 points
- Each time a tag is reused, the tag's creator gets an additional 10 points (which key to increment remains to be determined)

Use Case 4 Re-using an existing Bookmark by adding another annotation to it

- Creation of the annotation is the same as Use Case 1
- Adding an annotation to a bookmark is worth 10 points to that bookmark
    - It may well be that this is a dynamic value: it's always worth the TotalValue of the annotation

Scenarios

Scenario 1 Joe create an *annotation*

In this scenario, Joe has used the TQPortal (GTI) bookmarklet to annotate a web page he discovered while researching a topic of interest at the condo. This scenario has three components in play, besides Joe:

- The *bookmark* for the chosen web page (URL)
    - Joe could be the first person to create that bookmark
    - Joe might simply be adding an annotation to an existing bookmark
- The *annotation* itself
- Any *tags* Joe choses to add to the annotation
    - Joe could be the first person to create one or all of the tags
    - Joe might simply be re-using existing tags

Analysis

Some keys\

- ChildValues (used in game moves and annotations -- all conversations)
    - Compound list value:
        - Locator of node to which transcluded
        - value from that node

Scenario 2 Guild creation of a Game Move (1)

In this scenario, a particular guild creates a Game Move in which a small *tree* of claims and justifications -- several new nodes -- are made public and attached to a chosen *issue*. This scenario is the simplest of all possible game play scenarios in a GTI role-playing game; all nodes are originated in the guild itself:.

- Each node is marked as *created by* the guild itself.
- Each node in the game tree is subject to dynamic TotalValue of its subtree(s)

Analysis

Some keys\

-

Scenario 3 Guild creation of a Game Move (3)

This scenario is complex in the sense that the guild creates a tree of claims and justifications, some of which are original to the guild, and some of which are nodes transcluded from elsewhere in the condo.

- The scenario simply adds the complexities of scoring transclusions (Use Case 2) to scoring

Analysis

Some keys\

-

Scenario 4 Joe *crosslinks* two nodes

In this scenario, a new node which is an *instance* of a particular RelationType is created, and two nodes -- a source and a target -- are then coupled by that new relation node.

- There is a creation event
- That creation event adds value (dynamic?) to the RelationType node itself
- Potential to share TotalValue from each linked node to the relation node

- Potential to share TotalValue from relation node to each related node
    - Relation node, itself, can have a dynamic TotalValue -- see Scenario 5

Analysis

Some keys\

- For Relation Node
    - SourceValue (Compound list)
    - TargetValue (Compound list)
- For related nodes
    - RelationValue (Compound list)

Scenario 5 Relation node as a root in a conversation

Each topic map relation is, itself, a topic. As such, it can be *decorated* with context; that is, each relation has a biography, which, itself, is a topic.Also, since a relation occasionally represents a *debatable claim*, it can serve as a root in a conversation. Thus, the relation node, itself, has a dynamic TotalValue

- All the usual parent-child use cases are in play here

Implementation Concepts

We believe that the *InfoBox* feature of SubjectProxy objects (nodes) in the topic map are ideal for ValueMatrix. An InfoBox is a *named* JSON object as described above. Each node can have many different InfoBoxes. For example, a Guild's game activities are recorded in them.

TopicMap InfoBox

In InfoBox is simply a named *map* (collection of key/value pairs, where values can be singletons, or collections). They are *named* because a topic node (SubjectProxy) can have as many of them as necessary. The InfoBox concept is borrowed from MediaWiki, as used at WikiPedia. TQPortal already uses one to record game state for guild nodes.
<todo>

Scoring System

scoringengine.js has been added to <tqportal>/apps/rpg where it is just a shell for now.
<todo>