NEUROMEDIA, INC.

Neuromedia Template Project
Functional Specifications

Jack Hodges
February 20, 2000

Abstract

This document serves to define the basic market and user requirements for the tools that will allow vRep™ developers to more quickly create, modify and fill in content for Hierarchy based vReps. This "template application" will actually consist of several "major tools" that will each have a high-level set of functionality and several functional "mechanisms" that will be combined into one Interactive Development Environment (IDE) which incorporates a consistent look and feel as well as a consistent usage model across the major tools and the major mechanisms.

The requirements for the major tools will be specified by the combination of the functional user interface (UI) elements, the use cases pertinent to each tool, the object model, the functional mechanisms in the tool, and the typical screen flows that describe how the primary scenarios of the use cases are accomplished with in each tool.

Major tools

The major tools that will be incorporated in to the application are listed here, a detailed description of each tool will be provided later in the document.

1) The "Organization Hierarchy Picker" tool
This tool allows the user to choose a complete starting hierarchy for the project at hand, from a repository/library of previously defined, possibly generic, hierarchies.

2) The "Domain Creation, Selection & Organization" tool
This tool allows the user to "edit" the hierarchy of domains that are being used in the project. It allows the user to add, remove, copy, paste, drag, and rename a domain, or domain subtree, around the hierarchy. It allows domains to be replaced by the selection of a new domain from a repository of previously defined, and possibly generic, domains.

3) The "Domain Topic Definition & Instantiation Wizard" (aka the "Thang Wizard")
This tool allows the user to define the structure of domains as it pertains to topics. That is, what are the generic properties of “this” domain that all subdomains will have, and what properties will it inherit from a less specific domain. This tool will also allow the user to create "instantiations" of the topics that are the specific instances of the generic definition. For example, vReps can talk about a company’s products. If the company manufactures and sells shoes then all the shoes have the same set of properties, like color, number of lace eyelets, sole material, upper material, price, weight, name, stock number etc. This is a generic domain definition. Now suppose the company has 17 different shoe models they make; for each shoe model all of the property values for the generic shoe must be filled in.
In object-oriented terms the generic domain definition is the "class" and the actual shoe models are 17 different "instances" that are instantiated from the class definition.

4) The "Functional Topic Wizard" (aka the "Topic Wizard")
This tool allows the user to construct simple topics from scratch, and to reuse functional topics from a repository/library of previously defined, and possibly generic, functional topics. Functional topics are topics that are not important for the domain of which they are associated, but are interesting because they show the user how to accomplish some kind of difficult functional task. These difficult functional tasks can

be things like connecting to a database to retrieve product or inventory information in real-time, or interfacing with other plug-ins. These difficult functional tasks can also be things that involve complicated scripting like how to parse a who question into first name and last name and then answer appropriately.

Major Mechanisms

5) The "Topic Propagation Mechanism"
This mechanism allows the user to propagate content and functional changes throughout the hierarchy, or some selected subset of the hierarchy, to "like" domains and topics. At the topic level, the propagation mechanism implements the proliferation of topics generated by the topic wizard in the Domain Topic Wizard (3) across domain instances. It also supports developer-discretion on how a particular topic is proliferated in the hierarchy.

6) The "Domain Censoring Mechanism"
This mechanism allows the user to "censor" and "un-censor" a given domain. It take the user through all of the things needed to be assured that the domain is censored properly.

Minor Mechanisms

Minor mechanisms are ones that may be incorporated into the higher level tools, or may need to operate differently within the context of the different tools.

7) The “Tree View Mechanism” enables the visualization of domain hierarchies and organization hierarchies, the manipulation of which is the foundation for most behavioral activity associated with this project. The tree view mechanism is currently implemented using a tree view object, such as Explorer, but may be changed to a graphical editor in future releases.

8) The "Domain, Pattern List & Subject Name Checking Mechanism"
This mechanism internally checks the validity of a domain name, subject name, and pattern list content when new domains are added into the hierarchy.

Example

This section is to give a *single* example of each of the parts of the specification. The parts include:

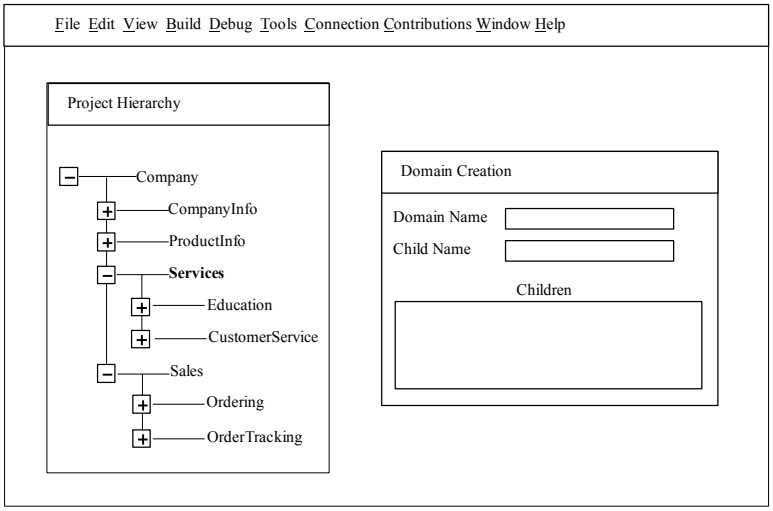
- Functional user interface (UI) elements
- The use cases
- The object model
- The functional mechanisms
- The typical screen flows (including basic screen examples).

Functional User Interface Elements

Functional UI Elements, provide details on key screen and user interface components. These items describe each element and it's functionality and provides an illustrative representation of the component. These drawings are only intended to convey the functionality needed. The look and feel as well as the method of implementation (dialog box, full screen, etc.) will be determined as part of a separate Look and Feel and usage model design effort.

2.1 Domain Selection and Organization

Description	Tool where user edits and organizes hierarchy
Functionality	Allows developer to create new domains Allows developer to remove domains Allows developer to move domains Allows developer copy domain subtrees and paste them Allows developer to move domain subtrees Allows developer to view all attributes of domains, domain families, and domains that have been affected by some operation
Applies To	Organization's hierarchy view
Key Screen Elements	Hierarchy view Creation dialog Message dialog Domain properties dialog
General Appearance (illustrative)	

	
Open	How do you allow user to find generic domains
Other Comments	
Implementation Priority	
Implementation Difficulty	
Implementation Notes	

The Detailed Use Cases

These talk about the use cases, for each tool that is specified and the primary and major secondary scenarios fore each use case.

Actors

- **VRep Developer (DEV):** Develops the vRep
- **VRep (VR):** The result of development, pre-hierarchical model
- **VRep Hierarchy (VH):** The pre-vRep structure
- **Hierarchical vRep (HV):** A hierarchically designed/developed vRep
- **NS Builder (NSB):** The compiler
- **NS Engine (NSE):** The interaction engine
- **NS Development Environment (NSDE):** The authoring environment

- **Excel File (Excel):** For tabbed hierarchy information
- **VRep Administrator (VA):** Post-development maintenance
- **Organization Library (OL):** Repository of HRs
- **Topic Library (TL):** Repository of topics organized functionally

Developer-Initiated (DEV) Use Cases

Load VR into HV

DEV ———————— Load ———————— VR

Precondition: A vRep created with the pre-hierarchical model exists

Flow of Events:

Primary Scenario

1) DEV selects file->load-from-nonhier

2) DEV selects vRep from file browser

3) The system will parse subjects into domains (hierarchically as much as possible)

4) DEV will graphically organize the remaining domains into a hierarchical collection

5) The system will reorganize the topics files according to the new domain structure

Secondary Scenarios

Postcondition: The vRep has been parsed into a hierarchical version

Load OL into HV

DEV ---------- Load ———————— OL

Precondition: An organization hierarchy exists

Flow of Events:

Primary Scenario

1) DEV selects open-project-from OL

2) The NSDE will display the OL in a tree view

3) DEV double clicks on the desired HV in OL

4) NSDE launches vRep creation wizard (mostly to select vRep home directory)

5) The NSDE copies the selected HV into local (text file in HV directory is copied into new home directory)

6) Hierarchy is parsed from text file, project files are loaded

Alternative path

7) DEV opens index bot

8) DEV asks for a bot that describes company

9) The NSDE selects closest match from OL and copies into local

10) NSDE launches vRep creation wizard (mostly to select vRep home directory)

11) The NSDE copies the selected HV into local (text file in HV directory is copied into new home directory)

12) Hierarchy is parsed from text file, project files are loaded

Secondary Scenarios

Postcondition: A new vRep copied from the OL has been created

Create VH from Excel

DEV ---------- Create ---------- Excel

Precondition: An Excel file representing a hierarchy exists

Flow of Events:

Primary Scenario

1) DEV selects create-hierarchy from Excel file

2) DEV selects Excel file to read from

3) NSDE parses Excel file into hierarchy

Secondary Scenarios

Postcondition: A hierarchy exists in NS

Create VH graphically

DEV ---------- Create ---------- VH

Precondition: NS is open

Flow of Events:

Primary Scenario

1) DEV selects create-new-vRep from file menu

2) VRep creation wizard does its thang (mostly sets the starting vRep directory)

3) NSDE opens empty tree view for vRep

4) DEV right-clicks create-domain in tree view

5) DEV types domain name into dialog (along possibly with children names)

6) DEV selects OK or CANCEL

7) If OK is selected, and if children were named, NSDE creates domain and children and displays in view

8) DEV continues until done creating

9) NSDE fills in parent and sibling information appropriately

Secondary Scenarios

Postcondition: A hierarchy exists in NS

Create HV graphically

DEV ---------- Create ---------- HV

Precondition: NS is open

Flow of Events:

Primary Scenario

1) DEV selects file->create-new-vRep

2) VRep creation wizard does its thang (mostly sets the starting vRep directory)

3) NSDE opens empty tree view for vRep

4) DEV right-clicks create-domain in tree view

5) DEV types domain name into dialog (along possibly with children names)

6) DEV selects OK or CANCEL

7) If OK is selected, and if children were named, NSDE creates domain and children and displays in view

8) DEV continues until done creating

9) NSDE fills in parent and sibling information appropriately

10) DEV selects build vRep from menu

11) NSDE fills in domain topics in file system, creates project folder, and builds vRep

Secondary Scenarios

Postcondition: Hierarchical vRep exists in NS

Save HV

DEV ---------- Save ---------- HV

Precondition: A hierarchical vRep has been created or loaded into NS

Flow of Events:

Primary Scenario

1) DEV selects file->save, or file->save-as for the open vRep

2) NSDE writes the hierarchy object to a text file in the vRep home directory

Secondary Scenarios

Postcondition: The HV is saved

Export HV to Excel

DEV ---------- Export ---------- Excel

Precondition: A hierarchical vRep has been created or loaded into NS

Flow of Events:

Primary Scenario

1) DEV selects file->export-to-excel for the open vRep

2) NSDE writes the hierarchy object to an Excel-formatted text file in the vRep home directory

Secondary Scenarios

Postcondition: An Excel-formatted text file of the hierarchy exists in the vRep home directory

Copy topics from VR

DEV ---------- Copy Topics ---------- VR

Precondition: A Hierarchical vRep is open and a non-hierarchical vRep exists

Flow of Events:

Primary Scenario

1) DEV selects domain to add topics to in tree view

2) DEV selects edit->copy-topics-from-nonhier-vrep for the open vRep

3) A browser is opened for selecting the vRep to copy from

4) DEV cntl-clicks files to copy topics from

5) NSDE copies topics into the current vRep's currently selected domain's topics file

Secondary Scenarios

Postcondition: The vRep has new topics in a domain

Copy components from HV2

DEV ---------- Copy Components ---------- HV

Precondition: A hierarchical vRep is loaded and a hierarchical vRep HV2 exists

Flow of Events:

Primary Scenario

1) DEV selects edit->copy-topics for the open vRep

2) The org-tree browser is opened for selecting the vRep to copy from

3) DEV selects the Org to copy from

4) NSDE loads the Org's tree-view

5) DEV selects the domain subtree to get topics from

6) NSDE recursively copies domain subtree to current vRep's domain

7) If a single domain was selected, the topics are copied into the terminal topics file

8) If a subtree was selected, the subtree is made a child of the current domain and the family members of the domain are updated

Secondary Scenarios

Postcondition: The vReo has new topics in a domain, or a new child subtree off domain

Move components in HV

DEV ---------- Move Components ---------- HV

Precondition: A hierarchical vRep has been created/loaded into NS

Flow of Events:

Primary Scenario

1) DEV selects edit->copy-topics for the open vRep

2) The org-tree browser is opened for selecting the vRep to copy from

3) DEV selects the Org to copy from

4) NSDE loads the Org's tree-view

5) DEV selects the domain subtree to get topics from

Secondary Scenarios

Postcondition: The vRep has been modified by moving components

Change properties of components in HV

DEV ---------- Change Properties ---------- HV

Precondition: A hierarchical vRep has been created/loaded into NS

Flow of Events:

Primary Scenario

1) DEV double-clicks on selected domain in tree view

2) The domain's dialog box is opened

3) DEV modifies editable attributes

4) NSDE checks domain names, subjects, and pattern lists for violations

5) NSDE propagates changes to family members

6) DEV iterates 1-5

Secondary Scenarios

Postcondition: The hierarchical vRep has had some domain attributes changed

Add topics from TL to HV

DEV ---------- Add Topics ---------- TL

Precondition: A topic library exists and a hierarchical vRep has been created/loaded in NS

Flow of Events:

Primary Scenario

1) DEV initially selects a domain from the vRep tree view

2) DEV selects tools->topic-wizard for the open vRep

3) Topic wizard dialog opens

4) DEV selects create-new, create from template, propagate new, or propagate from template from checklist

5) If DEV selected from template, the TL browser is opened

6) DEV selects the appropriate template from the browser

7) DEV selects OK or CANCEL

8) The topic wizard copies the template into the appropriate frames of the topic wizard

9) DEV inputs appropriate values into topic wizard dialog boxes

10) When done, if propagate was selected, DEV selects to-children, to-family, to subtree, or to selected from the final checkbox

11) If DEV selected 'selected' then DEV cntl-clicks selected domains in the tree view

12) Upon OK, instantiated topic template is propagated to the appropriate domains

Secondary Scenarios

Postcondition: A new topic or topics have been added to the vRep

Display Hierarchy

DEV ——————— Display Hierarchy ——————— HV

Precondition: NS is open and a hierarchical vRep is created/loaded

Flow of Events:

Primary Scenario

1) DEV selects view->display-hierarchy

2) The vRep tree view is displayed

Secondary Scenarios

Postcondition: The vRep tree view is displaying a vRep hierarchy

Display Family Properties

DEV ——————— Display Family Properties ——————— HV

Precondition: NS is open and a hierarchical vRep is created/loaded

Flow of Events:

Primary Scenario

1) DEV selects domain in tree view

2) DEV selects view->display-family

3) NSDE displays domain family members in different 'color'

Secondary Scenarios

Postcondition: The selected domain's family members are displayed in the tree view

Display Domain Properties

DEV ——————— Display Domain Properties ——————— HV

Precondition: NS is open and a hierarchical vRep is created/loaded

Flow of Events:

Primary Scenario

1) DEV selects domain in tree view

2) DEV selects view->display-domain-properties

3) NSDE displays domain-specific properties in tree view in different 'color'

Secondary Scenarios

Postcondition: Selected domain has terminal, censored, and num topics attributes displayed

Display Domain Orgs

DEV ——————— Display Domain Orgs ——————— OL

Precondition: NS is open and a hierarchical vRep is created/loaded, OL exists

Flow of Events:

Primary Scenario

1) DEV selects domain from tree view

2) DEV selects view->display-orgs

3) NSDE identifies orgs in OL that have a 'domain' domain

4) NSDE opens the orgs view for domain

Secondary Scenarios

Postcondition: A view of the organizations in OL that implement the domain is displayed

Display Changed

DEV ——————— Display Changed ——————— HV

Precondition: NS is open and a hierarchical vRep is created/loaded

Flow of Events:

Primary Scenario

1) DEV has made modifications to vRep

2) NSDE has propagated changes to hierarchy

3) NSDE flags domains that need to be edited and displays in different 'color'

Secondary Scenarios

Postcondition: Those domains which have been modified but whose topics have not been edited are shown in different way

NS Development Environment (NSDE) Use Cases

Create HV from VH

NSDE ———————— Create ———————— VH

Precondition: A Hierarchy exists but no Hierarchical vRep

Flow of Events:

Primary Scenario

1) NSDE proliferates the template domain topic file to each domain

2) NSDE modifies all references to subject based on domain name

3) NSDE writes domainname.n to .vsr file

4) NSDE writes HV information to .hier file

Secondary Scenarios

Postcondition: A hierarchical vRep exists, HV

Suggest domain names

NSDE ———————— Check Domain Names ———————— HV

Precondition: DEV has created a domain with a name that conflicts with existing domain names (generally a duplicate)

Flow of Events:

Primary Scenario

1) NSDE detects duplicate domain name after domain naming

2) NSDE presents dialog identifying clash (with what domain(s)) and suggests a name change that resolves the clash

3) DEV selects 'ok' or types a new domain name and 'ok'

4) NSDE continues check until problem is resolved

5) NSDE propagates change to family members

Secondary Scenarios

Postcondition: Duplicate domain name is resolved

Suggest new domains from examples

Precondition: VA has loaded new examples that do not fit in any HV domain

Flow of Events:

Primary Scenario

1) DEV selects create-hierarchy from Excel file

2) DEV selects Excel file to read from

3) NSDE parses Excel file into hierarchy

Secondary Scenarios

Postcondition: A new domain has been created and new topics inserted under it in HV

Check subjects

NSDE ———————— Check Subjects ———————— HV

Precondition: A modification on HV domains has been made

Flow of Events:

Primary Scenario

1) NSDE searches for and flags topics with 'subject' in them

2) If multiples are found, NSDE displays in domain dialog clashing domains

3) DEV can modify domain name or subject for current domain, or DEV can elect to create a tiebreaker

4) If DEV selects domain/subject name change

5) If DEV selects tiebreaker, then tiebreaker creation procedure is executed

Secondary Scenarios

Postcondition: Pattern lists in HV are unique or a tiebreaker has been created

Check pattern lists

NSDE ——————— Check Pattern Lists ——————— HV

Precondition: A modification on HV domains has been made

Flow of Events:

Primary Scenario

1) NSDE searches for and flags topics with 'pattern list' in them

2) If multiples are found, NSDE displays in domain dialog clashing domains

3) DEV can modify the pattern list for current domain, or DEV can elect to create a tiebreaker

4) If DEV selects pattern list name change, 3.7.5 check is reperformed

5) If DEV selects tiebreaker, then tiebreaker creation procedure is executed

Secondary Scenarios

Postcondition: Pattern lists in HV are unique or a tiebreaker has been created

Check domain names

NSDE ——————— Check Domain Names ——————— HV

Precondition: A modification on a HV domain name has been made

Flow of Events:

Primary Scenario

1) NSDE searches for and flags domains with 'domainname' in them

2) If multiples are found, NSDE displays in domain dialog clashing domains

3) DEV can elect the suggested domain name, or DEV can modify the domain name for current domain

4) If DEV selects domain name change, 3.7.6 check is reperformed

Secondary Scenarios

Postcondition: : Domain names in HV are unique

Proliferate changes to hierarchy

NSDE ——————— Proliferate Hierarchy Changes ——————— HV

Precondition: A modification on a HV domain name has been made

Flow of Events:

Primary Scenario

1) NSDE identifies all domains which are affected by the change, and groups them based on what type of affect the change produces

2) NSDE modifies domain family member objects in the hierarchy

3) NSDE modifies domain family member domain topics and terminal topics as appropriate

Secondary Scenarios

Postcondition: All appropriate domains/topics have been updated to accommodate the change

Proliferate changes to project

NSDE ———————— Proliferate Project Changes ———————— HV

Precondition: A modification on a HV domain name has been made

Flow of Events:

Primary Scenario

1) NSDE identifies all domains which are affected by the change, and groups them based on what type of affect the change produces

2) NSDE adds new domain .n files, removes old domain .n files from project, as appropriate

Secondary Scenarios

Postcondition: The project has been updated to account for HV domain changes

Proliferate changes to file system

NSDE ———————— Proliferate File System Changes ———————— HV

Precondition: A modification on a HV domain name has been made

Flow of Events:

Primary Scenario

1) NSDE identifies all domains which are affected by the change, and groups them based on what type of affect the change produces

2) NSDE modifies directory names and file names for domain name changes

3) NSDE adds/names new directories and files for new domains or domain subtrees

4) NSDE deletes domains or domain subtrees that have been removed

Secondary Scenarios

Postcondition: The file system has been updated to account for HV domain changes

Topic Library (TL) Use Cases

Save to TL from examples

TL ———————— Save ———————— VA

Precondition: VA has entered new a new example that doesn't exist in TL

Flow of Events:

Primary Scenario

1) VA imports examples from the contributor interface

2) NSDE parses examples into topic object / Neuroscript

3) NSDE/DEV integrates new topics into hierarchy

4) NSDE identifies topics as same/different from those in TL

NSDE saves 'new' topic objects to TL

Secondary Scenarios

Postcondition: The Neuroscript for the new topic is now integrated into the TL

Create new topic from example (new topic) to HV

TL ———————— Create ———————— VA

Precondition: DEV has launched the Topic Wizard on Domain in HV

Flow of Events:

Primary Scenario

1) DEV selects create-new checkbox in TW

2) DEV follows current procedure to create new topic

3) NSDE parses example into topic object / Neuroscript

4) NSDE integrates new topic into hierarchy as per DEV instructions

5) NSDE identifies topics as same/different from those in TL

NSDE saves 'new' topic objects to TL

Secondary Scenarios

Postcondition: A new topic has been created from scratch and placed into HV

Create topic from TL (template topic) to HV

TL ——————— Create ——————— HV

Precondition: DEV has launched the Topic Wizard on Domain in HV

Flow of Events:

Primary Scenario

1) DEV selects create-from-TL checkbox in TW

2) NSDE opens TL browser

3) DEV double-clicks best-suited template in browser

4) NSDE copies/instantiates topic template into topic object / Neuroscript

5) DEV follows current procedure to create new topic

6) NSDE integrates new topic into hierarchy as per DEV instructions

7) NSDE identifies topics as same/different from those in TL

NSDE saves 'new' topic objects to TL

Secondary Scenarios

Postcondition: A new topic has been created using the TL and placed into HV

Proliferate topic(s) to HV domains

TL ——————— Proliferate ——————— HV

Precondition: DEV has created a new topic in the Topic Wizard

Flow of Events:

Primary Scenario

1) DEV completes topic creation in TW

2) DEV selects integration type checkbox (current domain, domain subtree, domain siblings, domain selections)

3) NSDE copies topic to specified domains, replacing domain names as appropriate

Secondary Scenarios

Postcondition: The newly created topic exists in desired domains

Create topic families based on HV domains

Precondition: DEV has created a domain in HV, terminal topics library (TTL) exists

Flow of Events:

Primary Scenario

1) DEV creates new domain in HV

2) DEV selects tools->TTLWizard

3) NSDE opens the TTL Wizard dialog

4) DEV selects copy-from-existing domain in checkbox, or selects create from template

5) If DEV selects former, the domain->orgs view is opened

6) DEV selects org to copy topics from

7) Topics are copied/modified into current domain

otherwise

8) If DEV selects 'from template', the 'domain' topics from the TTL are copied/modified into current domain

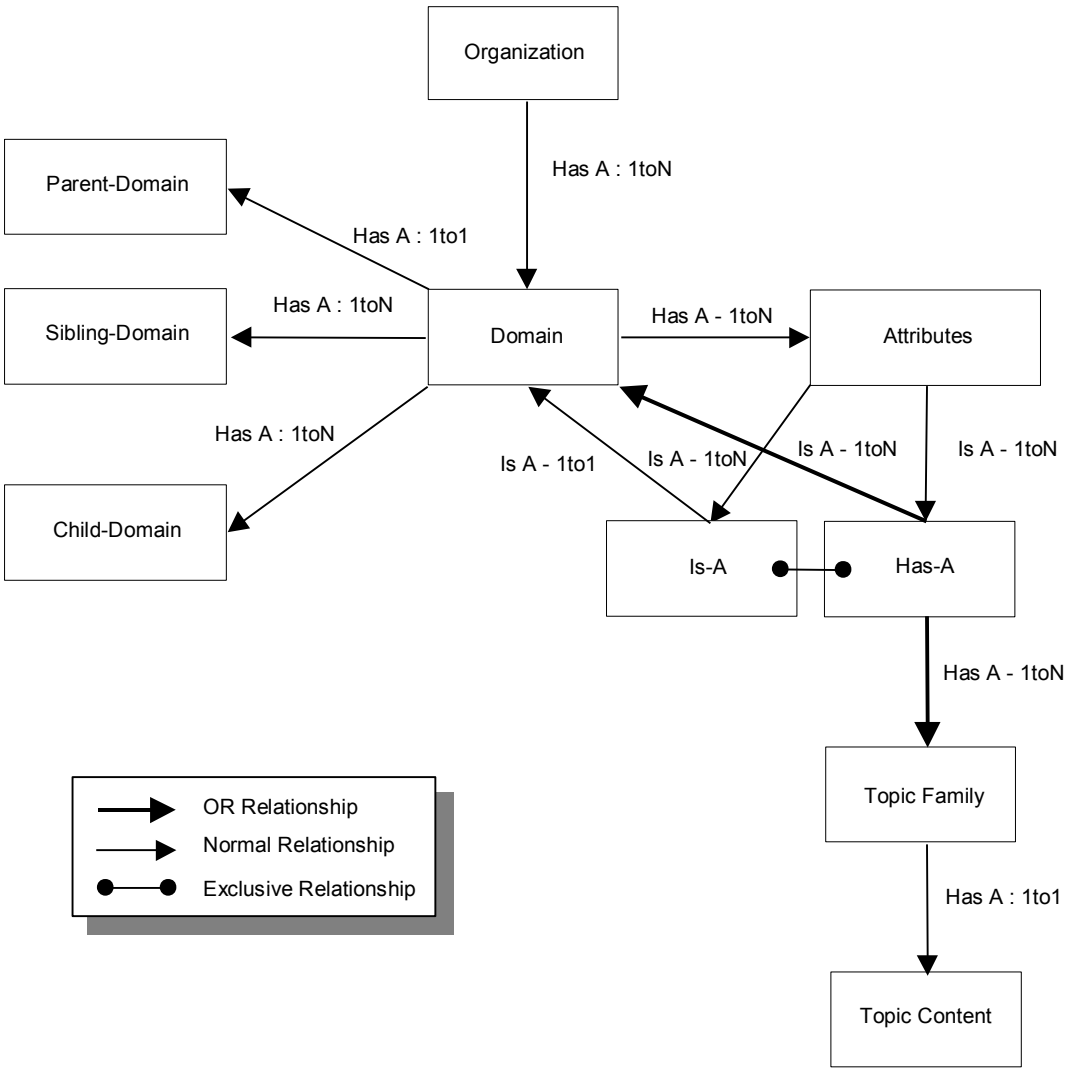
Secondary Scenarios

Postcondition: The newly created domain has terminal topics from TTL

The Object Models

The object models should show a graphical representation of the classes defined in each tool in an "object diagram" and the associated properties. It should also start to talk about the data elements and properties of each object.

Object Diagram - Hierarchy Editor



The Functional Mechanisms

Functional mechanisms identify the mechanisms and their interactions that support the intended functionality. This takes the form of a textual description.

There is also a diagram or several diagrams that show how all the functional mechanisms or groups of functional mechanisms interact with one another and with other parts of the system.

Example

- Generic Domain Retrieval Mechanism - presents the user with the selection of sample domains from the repository/library of (possibly) generic domains. Once the choice of domain is made, the mechanism retrieves the domain and inserts an instantiation of it

into the hierarchy as a child of the user's currently selected domain in the current hierarchy.

- Domain Selection Mechanism - this is the mechanism that allows the user to indicate a domain within the hierarchy that they want to make the currently selected domain. This is accomplished with a mouse click over a domain in the hierarchy viewer. The domain is then highlighted as the selected item.

Typical Screen Flows

Typical Screen Flows, outline the day-in-the-life-of a vRep developer using the tool and primary decision flows for a simple example use case. They illustrate the screen relationships and flows users will follow to accomplish typical tasks.

The user launches NeuroServer in the usual way and sees the following screens:

- Project window (with last vRep project)
- VRep window (for last vRep project)
- **File**/**Edit**/**View**/**Build**/Debug/**Tools**/Connection/Contributions/**Window**/Help menus, with those appropriate to the Template Project boldened.

An example set of windows is described for creating a hierarchically-organized project:

- The user pulls down the *File* menu and selects *New vRep Project*. This is the same as before the Template Project.
  - The vRep creation wizard is displayed. The user may select to load a vRep from the Organization Library as a template for the new project, or to create one manually.
  - The vRep project and .vsr windows are displayed after the initial build, along with a hierarchy display window. If the vRep came from a template, then that template's hierarchy is now displayed, otherwise it is empty.
- The user can pull down the Edit menu and select *Add Domain* (there are also now *Remove* and *Edit Domain*, and *Copy Domain from vRep*).
  - The domain creation dialog is displayed. The user enters a name for the domain and, possibly, children names. When 'ok' is selected, the domain and children names are checked for validity in the current hierarchy and then added, as children to the currently selected domain, if there is one.
  - The user may select to Save to Files for the vRep, at which time any changes made to the hierarchy editor are proliferated to both the project file and to the topic files.