

Topologos project and software – An Introduction

Author

Patrick Chénais
University of Bern
Informatikdienste
chenais@id.unibe.ch - <http://www.id.unibe.ch/~chenais/>

Date

30.7.2004 (27.8.2004 for the English version)

Software

Name

Topologos

Team

Patrick Chénais
Daria Spescha
and sporadic but efficient assistance of Alex Bergel and Stéphane Ducasse

Version

V0.1

Keywords

Modeling, process modeling, object modeling, visual modeling, data integration, activity diagram, class diagram, monitoring, process monitoring, UML, Unified modeling language, Squeak

The software will be a priori freeware, but this point remains to be discussed with the direction of the university.

Supply and Download

This document	TOPO_ESUG_2004_E.PDF
Software source	Topologos.st
Examples of models	m-esugUni.xml m-esugNsSub.xml
ZIP File, contains all supplies	http://www.topologos.unibe.ch/topologos.zip
Download page	http://www.topologos.unibe.ch/

Resources / requirements

- Squeak 3.6
- **NCCconnectorMorph** (Connectors 1.9)
& **SIXX** should be installed before installation of Topologos.st
(SIXX is only used for loaded and saving models).

Context of the development

To summarize, administrative data processing crossed two great phases: The era of system analysts followed by the phase of accumulation of data. A short review of the context of these two periods: ■ In the Eighties, the computer finds a niche in companies which are absolutely not ready to accommodate it. It is necessary to adapt the tool in order to introduce it in the organization of the company. The potentials are identified and developments are performed on an ad-hoc basis. It is the golden age of the analysts and the development of a Top Down method: the *Structured Analysis*, and its related technique the *Data Flow Diagrams* DFD.

■ As the size of the hard discs is increasing, begins the generalized accumulation of data: *data bases, data banks, data warehouses...* more and more data, but redundant, non-coherent data. Soon nobody, at the level of the company, any longer knows exactly who does what, what are the data available, produced or consumed. The situation becomes even more difficult as the developments are widely entrusted to external companies.

Faced with this situation, we come to regret the nice old-fashioned DFD. Actually, this type of diagrams reappears today as process diagrams (Activity Diagrams UML, for example), the good hurdy-gurdy *Top Down* approach makes its comeback. The processes are identified, analyzed, described, and the necessary data, applications and user interfaces are determined, (let's be up to date: using use cases UML). A bit further and we would return to the '*executable specifications*' concept with the MDA paradigm (Model driven architecture).

One problem remains: how to ensure the transparency of data with the Top down approach? The issue of transparency of processes, data, applications and their security is now at the heart of the problems of administrative data processing.

■ In an industrial control system, there is no chance of relying on human intervention to counteract a modeling deficiency. Modeling must thus be comprehensive and controllable, therefore transparent, hence two conclusions: 1/. **The concerns of administrative data processing meet the issues formerly specific to industrial data processing (and now mostly solved), 2/. Industrial data processing (and the research in Data processing, in particular the field of artificial intelligence) which was the source for the emergence of new concepts, can bring solutions to the current problems of administrative data processing.**

Topologos (in short *Topo*)

A tiny software with great ambitions: ensuring transparency of data and processes, by an unified representation of processes and objects. Topo can be seen as an evolution of *Workflows*, but also as an extension of *Data Flows*, however in the processes described by Topo neither documents nor data circulate, but only objects do. Topo allows the representation of what we might call *Object Flow Diagrams* (OFD). The origin of Topo is however mainly to be found in the QOBJ (*Queuing Objects*), a data-processing concept developed in the nineties with my friend A. Stagno then doctorate at the EPFL.

Background

Responsible for the development of an elevator group control system, I was faced with the problem to integrate three models: an **object model** (appropriate to represent the physical configuration of an elevator group: elevator shafts, cabins, floors, call panels and buttons, displays, etc), a **queuing network model** for the representation of dynamic processes (movement of cars, door operation, passenger movements, etc.) as well as a **multi agent organization** to ensure a distributed control of the group. The data-processing and -modeling solution consisted in working out the concept of QOBJ. A QOBJ is, in short, an object provided

with 1/ a **queue** and able to receive other QOBJs, 2/ a **pilot** (another QOBJ as well) and 3/ a **service** (a list of instructions describing the operations to be carried out by the pilot).

We can now hope to transfer and extend the results obtained into the field of administrative data processing. Topologos is a first step in this direction.

Getting started and use of the software


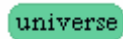

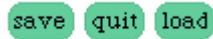
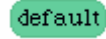
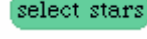
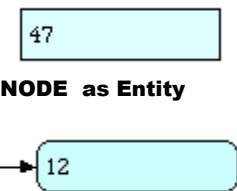




Getting started:

dummy



'dummy' is the name of the current model (It can be edited S. 'save' below.)

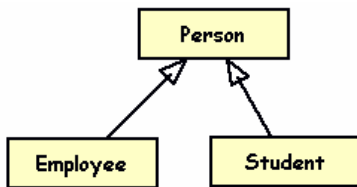
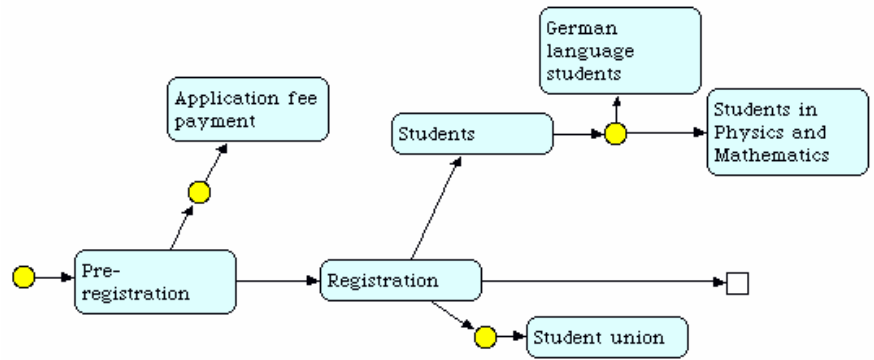
1. Install Connectors 1.9 and SIXX
2. Copy models (xml files) into Squeak folder
3. File List / install / Topologos.st
4. Workspace / OUniverse new / do it
5. Click right to open menu
 - click default, a default model will be created
 - or load a model, e.g.: *m-esugUni.xml*
'esugUni' will replace previous empty 'dummy' model

Elements	Mouse left	Mouse right	Notes
	Change display mode: Nodes, circles, stars...	Open / Close menu, hide all	Try it
	Check syntax and refresh model, display all valid elements. Toggle UML / not UML display.	>> idem	UML display is in development
	Create element NODE, ARC or STAR	Create element DOT, ARC or CIRCLE	
	Save, quit, or load model 'save' uses the current name of the model (that can be edited) to generate the filename: 'm-', modelName, '.xml'	>> idem	Load f.i. m-esugUni.xml or m-esugNsSub.xml
	Create default model	>> idem	
	Display only selected stars e.g. [:x x name = 'John']. Block must return a boolean.	>> idem	
 NODE as Entity NODE as Activity	Create ARC	OPEN SUB MODEL with name written between //. f.i. <i>Natural sciences subjects /esugNsSub/</i> It appears a small square. Close previous model and click right or left on the small square.	A NODE with arcs is an activity (rounded). A NODE with an empty name becomes (irreversibly) a DOT
 DOT	Create ARC	Toggle DOT / GATE	
 GATE	Create ARC	Toggle DOT / GATE	'Gate of life'
 STAR	Create relation to node, STAR, ARC or NODE	Create a root (red) relation to STAR, ARC or NODE	A STAR can only have one root
 CIRCLE	Create relation to node, STAR, ARC or NODE	Create a root (red) relation to STAR, ARC or NODE	A STAR with a name starting with a . (point) is a CIRCLE

Main concepts and examples

The figure opposite (Topologos screen shot) is a traditional representation of process. However, two remarks can be made:

■1 *There is no formal distinction between classes (or objects) as for example 'Students' and activities of processes, for instance 'Registration'.*

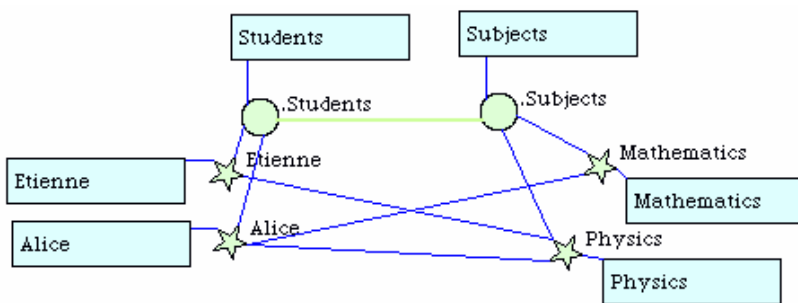


Explanation - Attributes are specific to class diagrams, but they are most of the time only necessary because the processes at the origin of the attributes remain unknown. In the figure opposite a 'Student' is a subclass of the class 'Person'. This is an inaccurate interpretation; it would be more precise to say that **'A student is a person who enters the process of studying'**. Actually a person can be involved in a multitude of processes without losing his identity. Topologos by

managing the links between classes and processes brings a simple solution to the problem of instantiation (How to instance a person if he or she is employed and student at the same time?) and multiple inheritances. **Managing the links between classes (and objects) and process activities solves the question of inheritance of attributes, it allows to get rid of an arbitrary distinction: Finally a class is a container of individuals, and an activity contains a [subset of individuals, there is no more reason to make a distinction between both.** Note - Topologos automatically operates a graphical transformation of a NODE (either a class or an activity) by rounding the corners when encapsulated within a process.

■2 *Some activities are separated by small circles (DOTS), others are not, why so?*

A process is crossed by individuals, for example a student is first in the phase 'Pre-registration', then in the phase 'Registration' and cannot be in two phases simultaneously. In this case there will be no DOT inside the process flow. On the other hand a student can be in the process 'German language students' (here reduced to one NODE) and simultaneously in the stage '(Registered) Students', in this case his identity will be duplicated. This duplication is specified by a DOT. Note that a similar representation, a small square, is used to depict the gate of life (e.g. transformation of an object into another).



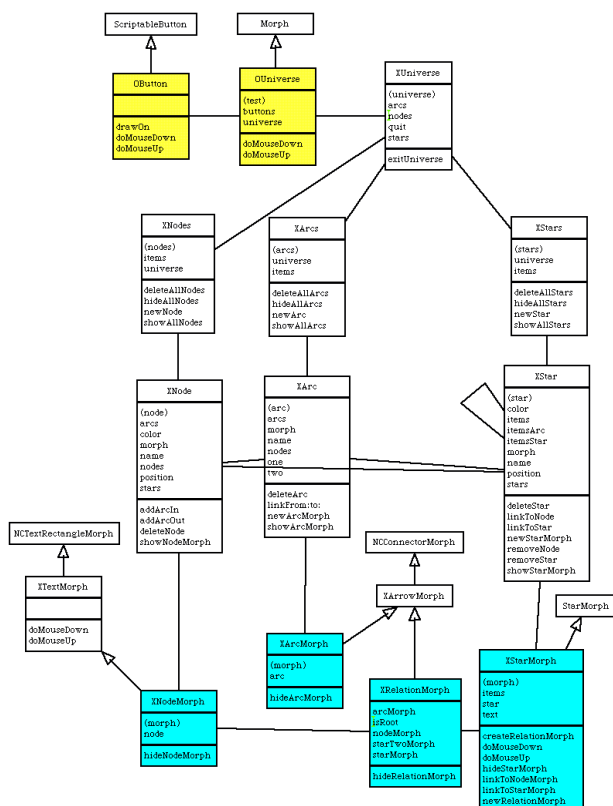
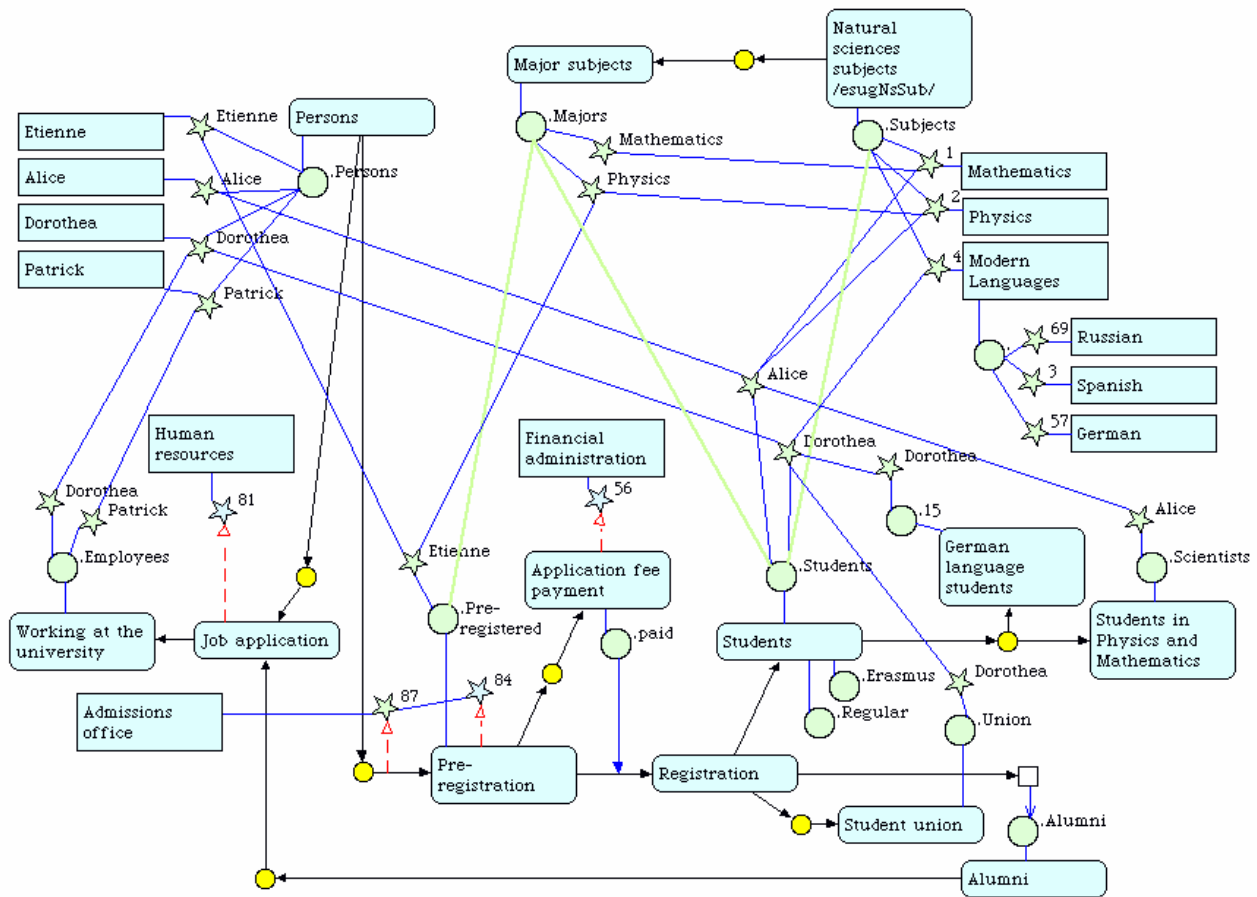
Two elements can be allocated to a NODE (recall either a class or an activity): ■1 *STARS represent the identity of entities.* For example a STAR 'Alice' in the figure opposite (screen shot) represents the entity NODE 'Alice'. We may have as many STARS 'Alice' as necessary, but only one NODE 'Alice' because Alice is unique, but may participate to several processes or may

have several type of relationship with other entities. ■2 *CIRCLES contains a collection of STARS (implemented and displayed above as links to these STARS).* For instance the CIRCLE 'Students' contains all instances of students, i.e. Etienne and Alice. Having both have STARS and CIRCLES, a NODE, can be at the same time container and contents.

■ **Relationships** between classes are established between the CIRCLES, because relationships may depend on the activity, e.g. a student in the pre-registration stage has no minor subjects. In the example above, relation n-m: A student is following several subjects, and a subject is followed by several students.

Appendix

■ A complete, but small-scale, model is given below (Topologos screen shot). This representation may seem complex, but notice that it covers class diagrams, process flows and instantiation and monitoring of objects (not discussed in this short paper).



■ The graphical representation is entirely generated starting from a non-graphical model (to which is added only information on the position and the color of the objects). This allows an easy saving in an XML file via SIXX. The software architecture is given opposite (you will need good eyes).

■ Further works

- Make user interface as simple as possible.
- Monitoring real processes and data through Web Services / SOAP

■ More information...

<http://www.topologos.unibe.ch/>