Project title:	***Enabling Communities of Interest to Promote Heritage of European Regions***
Project acronym:	**Cipher**
Project number:	IST 2001 - 32559
Partners:	The Open University, UK (Project coordinator) Dublin Institute of Technology , Ireland University of Art and Design, Helsinki, Finland The Discovery Programme, Ireland Czech Technical University, Czech Republic Internet-Loesungen und Dienstleistungen RiS GmbH, Austria

Report title:	***DNAT – User's Manual***
Authors:	Jan Uhlíř, Petr Křemen, and Luboš Král E-mail: uhlir@labe.felk.cvut.cz
Deliverable:	D26/2
Version:	1.0
Report availability:	Internal
Date:	September 2004

Contents

DNAT – User's Manual

1. General Description

Dynamic Narrative Annotation Tool (DNAT) was designed to support users in creating printable knowledge-intensive content and the corresponding knowledge-base at the same time. DNAT allows creation of semantic annotations in Conceptual Graphs (CG). CG (developed by John Sowa) is a human readable notation for First Order Logic (FOL). DNAT enables users to link text fragments in the narrative and ontology classes to create instances in the CG and the corresponding knowledge-base. Then, CG represents easily readable model of narrative and shows the dependencies of narrative primitives.

The design structure of DNAT is strictly modular. It means that every piece of functionality is provided by a single plug-in. Therefore, for full annotation power, following plug-ins are needed: CG module (`cg.jar`), Ontology module (`onto.jar`), Narrative module (`narrative.jar`) and Marking module (`marking.jar`).

2. Installation Instructions

1. Download and install Java version 5.0 (J2SDK 5.0 RC).
2. Download actual release of DNAT and unpack it into any directory.
3. Edit `rundna.bat` before running DNAT. Batch file `rundna.bat` can be found in your DNAT installation directory.
   a. Set JAVA_HOME environment such that it points to your private Java Runtime Environment of your Java 5.0 installation.

      Example: `set JAVA_HOME=c:\j2sdk1.5.0\jre\bin`

   b. Change `cd` command such that it will change the current directory to your DNAT installation directory.

      Example: `cd "c:\Documents and Settings\uhlir\cvswork\Cipher\Dna\"`

   c. The last command shall run DNAT. You can increase heap size used by the Java Virtual Machine (JVM) by using -Xmx switch of JVM. You can also get rid of standard output messages by using `javaw.exe` rather then `java.exe`.

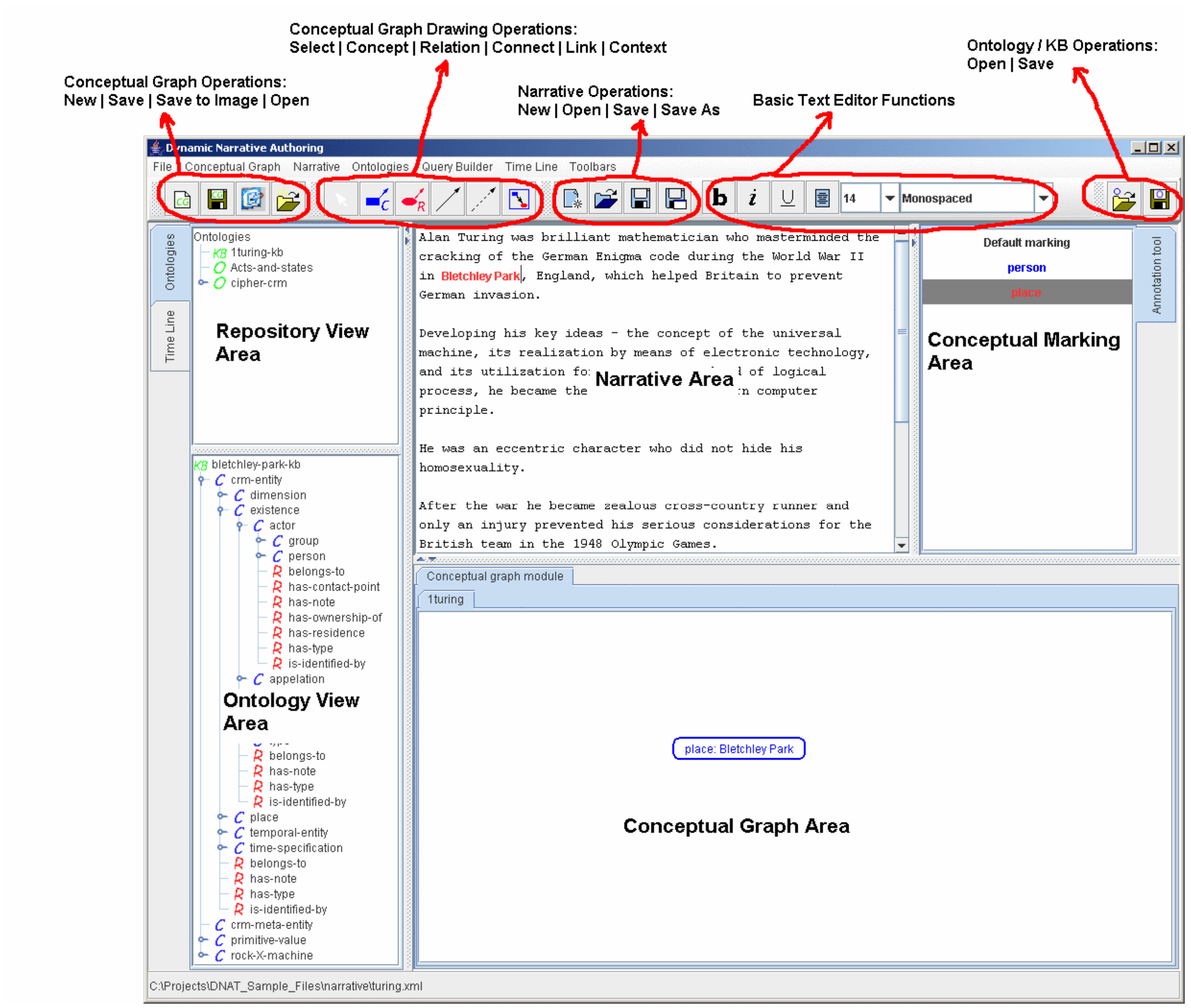
      Example: `%JAVA_HOME%\java -Xmx350m -jar Dna.jar en En`

4. Consult project webpage for getting help.

3. Using DNAT

3.1 Creating annotations

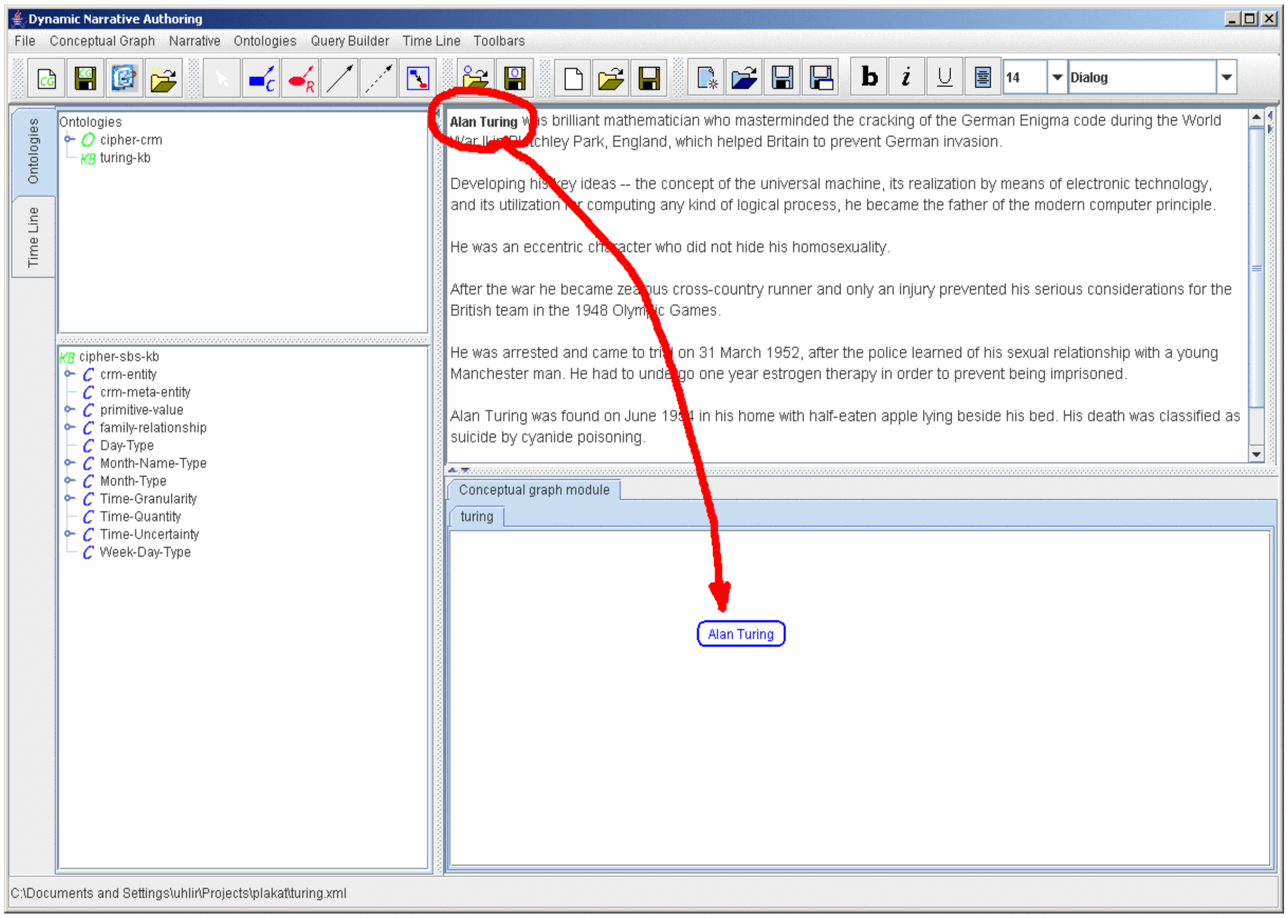
Basic features and GUI layout of DNAT are outlined in Figure 1. Before a narrative can be annotated, user must open at least one ontology in the DNAT. The repository view area of the DNAT shows the hierarchy of both opened ontologies and knowledge bases, while the ontology view area shows just one ontology that is selected in the repository view area. Selected ontology is displayed with all classes inherited from its super-ontologies.



```
Alan Turing was brilliant mathematician who masterminded the
cracking of the German Enigma code during the World War II
in Bletchley Park, England, which helped Britain to prevent
German invasion.

Developing his key ideas - the concept of the universal
machine, its realization by means of electronic technology,
and its utilization fo                     l of logical
process, he became the                     n computer
principle.

He was an eccentric character who did not hide his
homosexuality.

After the war he became zealous cross-country runner and
only an injury prevented his serious considerations for the
British team in the 1948 Olympic Games.
```

Figure 1: DNAT GUI Layout

While annotating a narrative, user seeks for important concepts in a narrative text. When some words are selected in the annotated text, they can be dragged over a CG area of DNAT and dropped there as shown in Figure 2. DNAT will then automatically draw concepts as nodes of the annotation graph. Each selected word appears in the referent field of a newly created node. It means that the instance of the most general conceptual type called *Entity* was created. However, some ontologies may use both different name and different structural properties for the most general conceptual type or they may have several concepts at the most general level. Therefore in this annotation step the newly created node does not automatically appear in the frame-based knowledge base that DNAT maintains besides the annotation graph.



Alan Turing was brilliant mathematician who masterminded the cracking of the German Enigma code during the World War II in Bletchley Park, England, which helped Britain to prevent German invasion.

Developing his key ideas -- the concept of the universal machine, its realization by means of electronic technology, and its utilization for computing any kind of logical process, he became the father of the modern computer principle.

He was an eccentric character who did not hide his homosexuality.

After the war he became zealous cross-country runner and only an injury prevented his serious considerations for the British team in the 1948 Olympic Games.

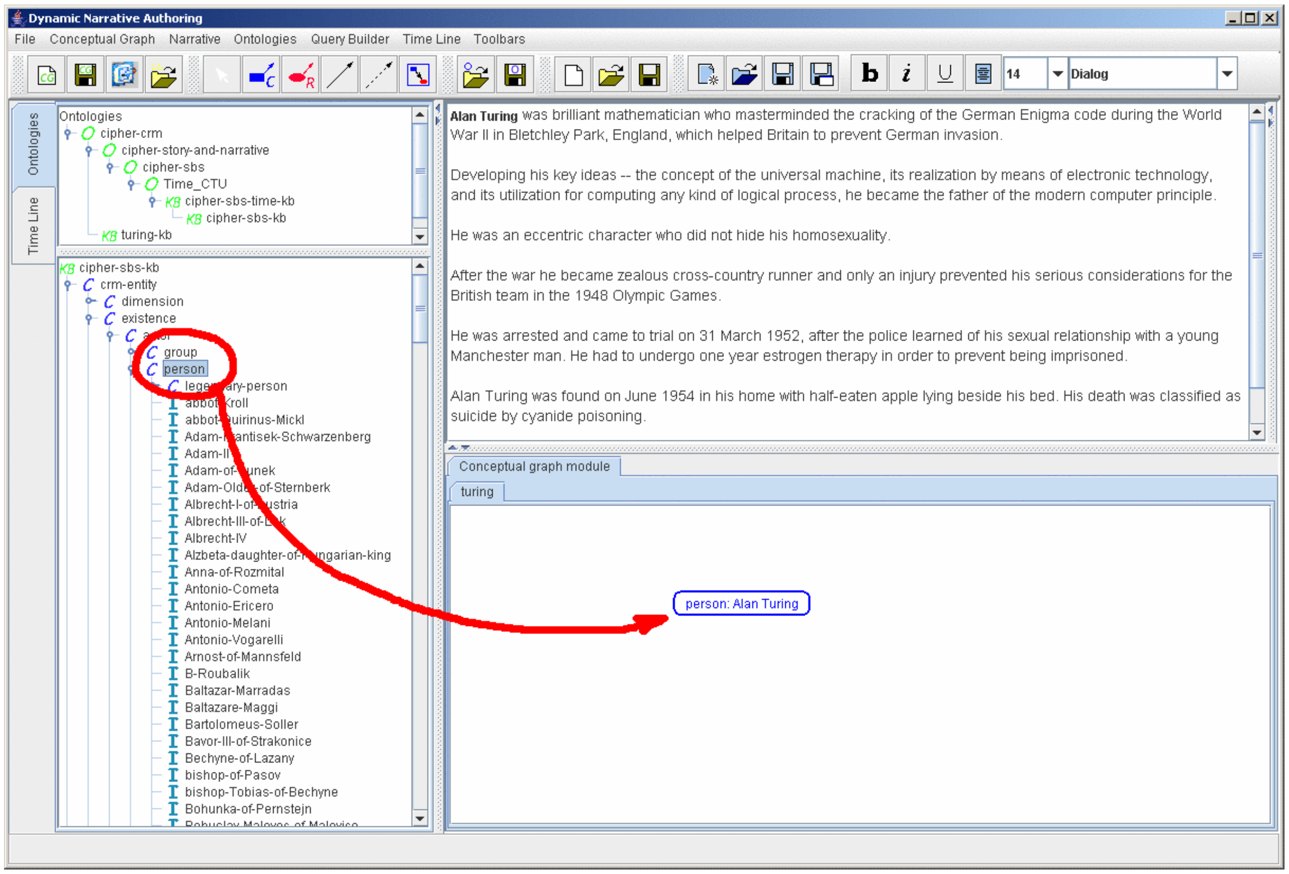
He was arrested and came to trial on 31 March 1952, after the police learned of his sexual relationship with a young Manchester man. He had to undergo one year estrogen therapy in order to prevent being imprisoned.

Alan Turing was found on June 1954 in his home with half-eaten apple lying beside his bed. His death was classified as suicide by cyanide poisoning.

Figure 2: Identifying Concepts in Narratives

In the next annotation step, user finds an appropriate conceptual type in the ontology browser of DNAT and assigns it to the newly created concept by means of drag and drop (see Figure 3). By doing so, the type field of the concept node is set. In this moment, the drag and drop event triggers automatic creation of an instance of the selected conceptual type in the frame-based knowledge base. Instances stored in the frame-based knowledge base can be seen in the ontology view area of DNAT. Slots defined in ontology can be used to describe semantic relations (i.e. extrinsic relations) between concepts used in a narrative and other concepts that are not explicitly stated in the same narrative. DNAT allows yet making these extrinsic relations explicit in a narrative annotation written in CG as they may carry contextual information important for story understanding.

DNAT provides a conceptual marking feature that simplifies previous two annotation steps. The User first selects several conceptual types that are most often used during the annotation process. It can be done by dragging a concept from the ontology view area and dropping it in the conceptual marking area, which appears on the right relative to the narrative area. Different colour can be selected for each conceptual type in the conceptual marking area of DNAT. By pre-selecting one conceptual type in the conceptual marking area the user defines both a conceptual type and a colour for each term that is highlighted in the narrative area and dropped in the conceptual graph area. Thus, each term highlighted in the narrative area gets colour of the selected conceptual type and each new instance dropped into the annotation graph has already defined its conceptual type. This simplified procedure can be used until the user pre-selects different conceptual type or the pre-selection is cancelled by selecting 'Default marking' option from the conceptual marking area. For better idea about the conceptual marking feature of DNAT see Figure 4.



Alan Turing was brilliant mathematician who masterminded the cracking of the German Enigma code during the World War II in Bletchley Park, England, which helped Britain to prevent German invasion.

Developing his key ideas -- the concept of the universal machine, its realization by means of electronic technology, and its utilization for computing any kind of logical process, he became the father of the modern computer principle.

He was an eccentric character who did not hide his homosexuality.

After the war he became zealous cross-country runner and only an injury prevented his serious considerations for the British team in the 1948 Olympic Games.

He was arrested and came to trial on 31 March 1952, after the police learned of his sexual relationship with a young Manchester man. He had to undergo one year estrogen therapy in order to prevent being imprisoned.

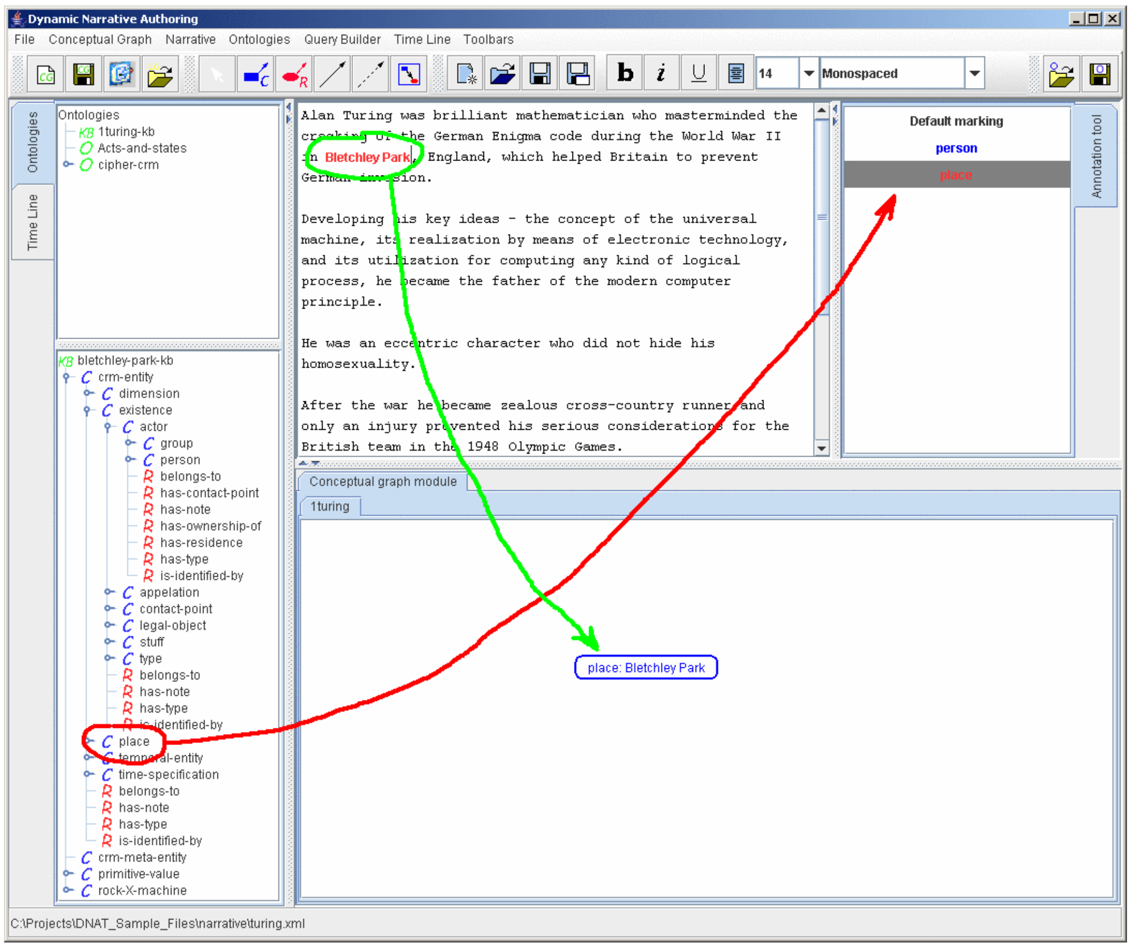
Alan Turing was found on June 1954 in his home with half-eaten apple lying beside his bed. His death was classified as suicide by cyanide poisoning.

Figure 3: Assigning Conceptual Type

Conceptual relations related to the selected concept (i.e. represented as slots of ontology concepts) can simply be selected from the table of all available slots and then connected to other concepts in the graph. This is visualised in Figure 5. When user wants to delete a concept from the annotation graph, it is deleted from the corresponding frame-based knowledge base as well. Thus, instances of concepts are created in the knowledge base and visualised in the annotation graph. CG node represents a concept used in the narrative and at the same time associates a concept from the narrative with a formal definition of this concept (possibly instance) in ontology.

We keep both ontologies and knowledge bases in *Apollo CH* format. Apollo CH is an ontology editing environment created and maintained by the Information and Knowledge-Based Systems Group of the Gerstner Laboratory at the Czech Technical University in cooperation with the Knowledge Media Institute at the Open University, Milton Keynes. Export from internal *Apollo CH* format to other formalisms is provided by means of export plug-ins to *Apollo CH*, which work in DNAT as well. Resulting annotation can be exported into many different formalisms intended for semantic annotations supported by DNAT; export to OCML, RDF, DAML-OIL, and OWL is currently available by means of I/O Plug-ins.

When appropriate conceptual type or the entire domain ontology is missing, an instance of the most general concept entity is created. Such instance can still play role in many conceptual relations and its specific conceptual type can be defined later. Therefore the annotation graph remains valid though semantics of annotated content is captured with less accuracy.



```
Alan Turing was brilliant mathematician who masterminded the
cracking of the German Enigma code during the World War II
in Bletchley Park, England, which helped Britain to prevent
German invasion.

Developing his key ideas - the concept of the universal
machine, its realization by means of electronic technology,
and its utilization for computing any kind of logical
process, he became the father of the modern computer
principle.

He was an eccentric character who did not hide his
homosexuality.

After the war he became zealous cross-country runner and
only an injury prevented his serious considerations for the
British team in the 1948 Olympic Games.
```

Figure 4: Pre-selecting the most often used conceptual types

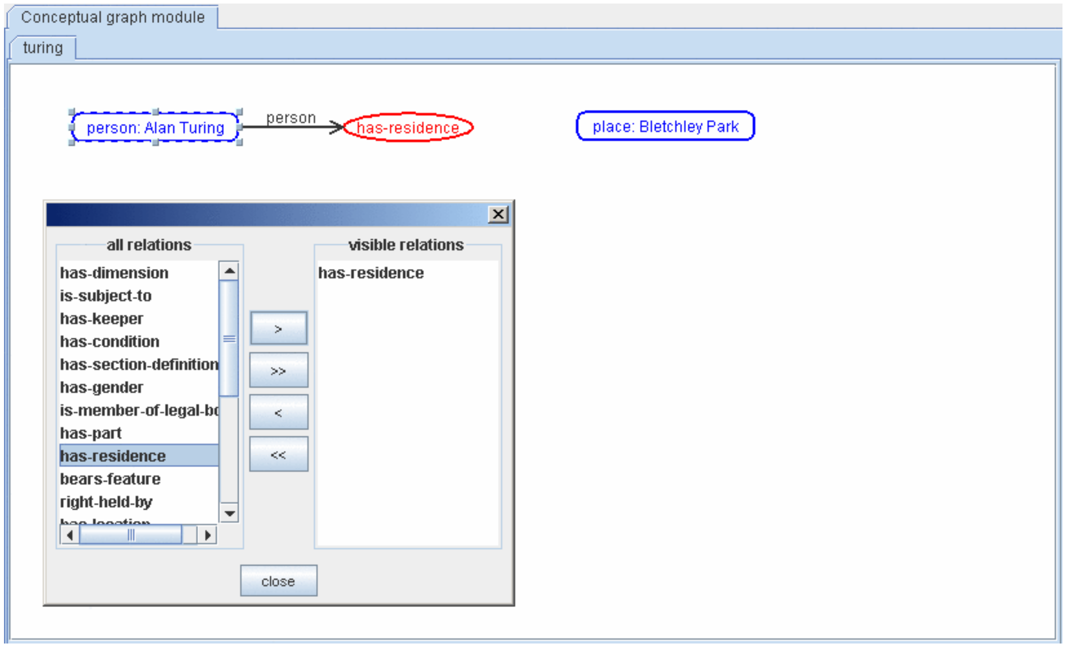


Figure 5: Connecting Concepts in Annotation Graph

3.2 Description of available functions

Functions available in both DNAT and its modules are outlined in Table 1 for those functions accessible through the main application menu, and in Table 2 for those functions accessible also through application tool bar.

Menu items			Icon	Description
File				
	→	Open Project		NOT IMPLEMENTED YET
	→	Save Project		NOT IMPLEMENTED YET
	→	Exit		Shuts down the application.
Conceptual graph				
	→	New CG	📄	Creates new conceptual graph.
	→	Save CG	💾	Saves current conceptual graph. The binary format is provided.
	→	Export to Image	🖼	Exports conceptual graph to the image. Available image formats depend on the used JDK.
	→	Open CG	📂	Opens saved conceptual graph. The binary format is provided.
	→	Close CG		Closes current conceptual graph.
	→	Rename CG		Opens rename dialog for current conceptual graph.
Narrative				
	→	New Document	📄	Creates new narrative document.
	→	Open Document	📂	Opens narrative document either in RTF or in saved XML narrative format.
	→	Save	💾	Saves narrative document into XML format.
	→	Save as	💾	Saves narrative document under different name in XML format.
Ontologies				
	→	Open Ontology	📂	Opens ontology in any of these formats - XML, meta.
	→	Save Ontology	💾	Saves current ontology into any of these formats – XML, meta, CommonLISP (cls), CLOS, OCML, OIL, RDFS.
	→	Close Ontology		Closes current ontology.
Query Builder				
	→	Configure Server		Configures server for the query builder.
Time Line				
	→	Execute Query		Executes timeline query.
	→	New Time Line	📄	Creates new time line.
	→	Open Time Line	📂	Opens saved time line.
	→	Rename Time Line		Renames current time line.
	→	Close		Closes current time line.
	→	Save	💾	Saves current time line.
	→	Save as		Saves current time line under different name.
	→	New Event		Creates new event in current time line.
	→	Edit Event		Opens edit dialog for current event.
	→	Delete Event		Deletes current event.
	→	Sort by Name		Sorts events by name.
	→	Sort by Date		Sorts events by date.
	→	Sort by Type		Sorts events by type.

Table 1 Menu items description

Toolbar	Icon	Description
Conceptual graph		
	📄	Creates new conceptual graph. The same function is available through menu item Conceptual graph > New CG.
	📂	Opens conceptual graph saved in the binary format. The same function is available through menu item Conceptual graph > Open CG.
	💾	Saves current conceptual graph. The same function is available through menu item Conceptual graph > Save CG.
	🖼	The same function is available through menu item Conceptual graph > Export to Image. Exports conceptual graph to the image. Available image formats depend on the J2RE used.
Conceptual graph elements		
	↖	Sets selection mode. The selection can be made by pressing and dragging mouse in the graph.
	■	Sets concept add mode. A new concept can be placed by single mouse click in the graph.
	●	Sets relation add mode. A new relation can be placed by single mouse click in the graph.
	↗	Sets connection mode. A new connection between a concept and a relation can be established by pressing mouse on a concept and releasing on a relation.
	⇢	Sets co-reference link mode. A new co-reference link between two concepts can be created with this tool.
	▢	Sets context create mode. In this mode, new context is created by selecting desired cells.
Narrative toolbar		
	📄	Creates new narrative document. The same function is available through menu item Narrative > New Document.
	📂	Opens narrative document either in RTF or in saved XML narrative format. The same function is available through menu item Narrative > Open Document.
	💾	Saves narrative document into XML format. The same function is available through menu item Narrative > Save Document.
	💾	Saves narrative document under different name in XML format. The same function is available through menu item Narrative > Save As… Document.
	b	Selects bold font.
	i	Selects italic font.
	U	Selects underline font.
	≡	Aligns current line to the centre.
Ontology toolbar		
	📂	Opens ontology in any of these formats - XML, meta. The same function is available through menu item Ontology > Open.
	💾	Saves current ontology into any of these formats – XML, meta, Common LISP (cls), CLOS, OCML, OIL, RDFS. The same function is available through menu item Ontology > Save.
Timeline toolbar		
	📄	Creates new time line.
	📂	Opens saved time line.
	💾	Saves current time line.

Table 2 : Toolbar items description

Plug-in menus and toolbars are written in italic – these are shown only in case that the corresponding plug-in is loaded.

3.3 Why to Use DNAT ?

- Easy to learn, easy to use. Experiments proved that people are able to read Conceptual Graphs, and the knowledge editors who write semantic annotations as Conceptual Graphs can learn to write it in a few hours because with DNAT the process is similar to using colour markers for highlighting key concepts in documents. DNAT guides users through the process of connecting concepts by offering context-sensitive sets of applicable semantic relations. DNAT users need not to be familiar with syntax of semantic mark-up languages.

- Automated complex computer-oriented representation tasks. During the knowledge extraction process, DNAT automatically builds a knowledge base, which can be translated to many knowledge representation formalisms such as (RDF, OWL, OCML, etc.). After the knowledge has been translated to CG with human assistance, the unambiguous semantic interpretation of the text was provided in a formal and explicit way. It was achieved through the assignment of conceptual types with well-defined semantics to terms and their interlinking via conceptual relations. Therefore, the semantic ambiguities are eliminated by the system, which enforces a single definition for every term. As a result, the system can automatically translate CG to and from logic and various computational languages.

- Document annotations. People can read CG without special training, and a computer system can translate it automatically to different natural languages when multilingual ontologies are available. Thus, CG can be used for semantic search and reasoning regardless the language of documents used for extracting the knowledge.

- Multilingual presentation abilities. The Conceptual Graph output from the DNAT can also be presented directly as an annotation to the original source documents in a preferred language. They can serve as humanly readable comments. This is only limited by availability of multilingual ontology, which provide basis for the annotation process.

- Versatility in knowledge representation. DNAT allows the annotation to be represented in frame-based formalisms (e.g. OCML, CLOS), semantic mark-up languages (e.g. RDF, OWL), and logic-based languages such as CG. The open plug-in architecture of DNAT enables developers to easily extend its I/O capabilities by means of other plug-ins.

Acknowledgement:

This work is funded by the CIPHER project (Enabling **C**ommunities of **I**nterest to **P**romote **H**eritage of **E**uropean **R**egions) supported by the European Commission - IST RTD program

POWERED BY CIPHER

DNAT was created in the Information and Knowledge-Based Systems Group of the Gerstner Laboratory, Department of Cybernetics, Czech Technical University, Prague.

Project homepage:
http://labe.felk.cvut.cz/~uhlir/DNATWeb/DNAThome.html

laboratory Gerstner