

Multi-Media Instruction in Safe and Secure Systems

Bernd Krieg-Brückner, Christoph Lüth, et al.

4 June 2003

The MMiSS Project

- **Universität Bremen**
 - Bernd Krieg-Brückner, Hans-Jörg Kreowski, Arne Lindow, Christoph Lüth, Achim Mahnke, George Russell, et al.
- **FernUniversität Hagen, Universität Kaiserslautern**
 - Arnd Poetzsch-Heffter, et al.
- **Universität Freiburg**
 - David Basin, Jan-Georg Smaus, et al.
- **Universität des Saarlandes**
 - Serge Autexier, Dieter Hutter, Carsten Ullrich, Erica Melis, et al.
- **Ludwig-Maximilians-Universität München**
 - Rolf Hennicker, Martin Wirsing et al.

Document Structuring for Formal Methods

What Could Be Your Interest in This Talk?

- What have we learned from
 - Formal Methods, **CASL**, **development** support systems

What Could Be Your Interest in This Talk?

- What have we learned from
 - Formal Methods, **CASL**, **development** support systems
- Your future favorite way to
 - prepare, **reuse** and **manage** high-quality **L^AT_EX** slides
 - manage language / formalism **variants**
 - **re-use** and **adapt** course material from colleagues
 - **coordinate** slides with other documents
 - **integrate** CASL specifications with documentation
 - manage and **configure** results of research project

This Talk and Demo

- semantic interrelation via **ontology**
- **slides**, layout, animation
- **refinement** to/**abstraction** from article or book, **variants**
- “Literate Specification” with **CASL**, CATS tools

This Talk and Demo

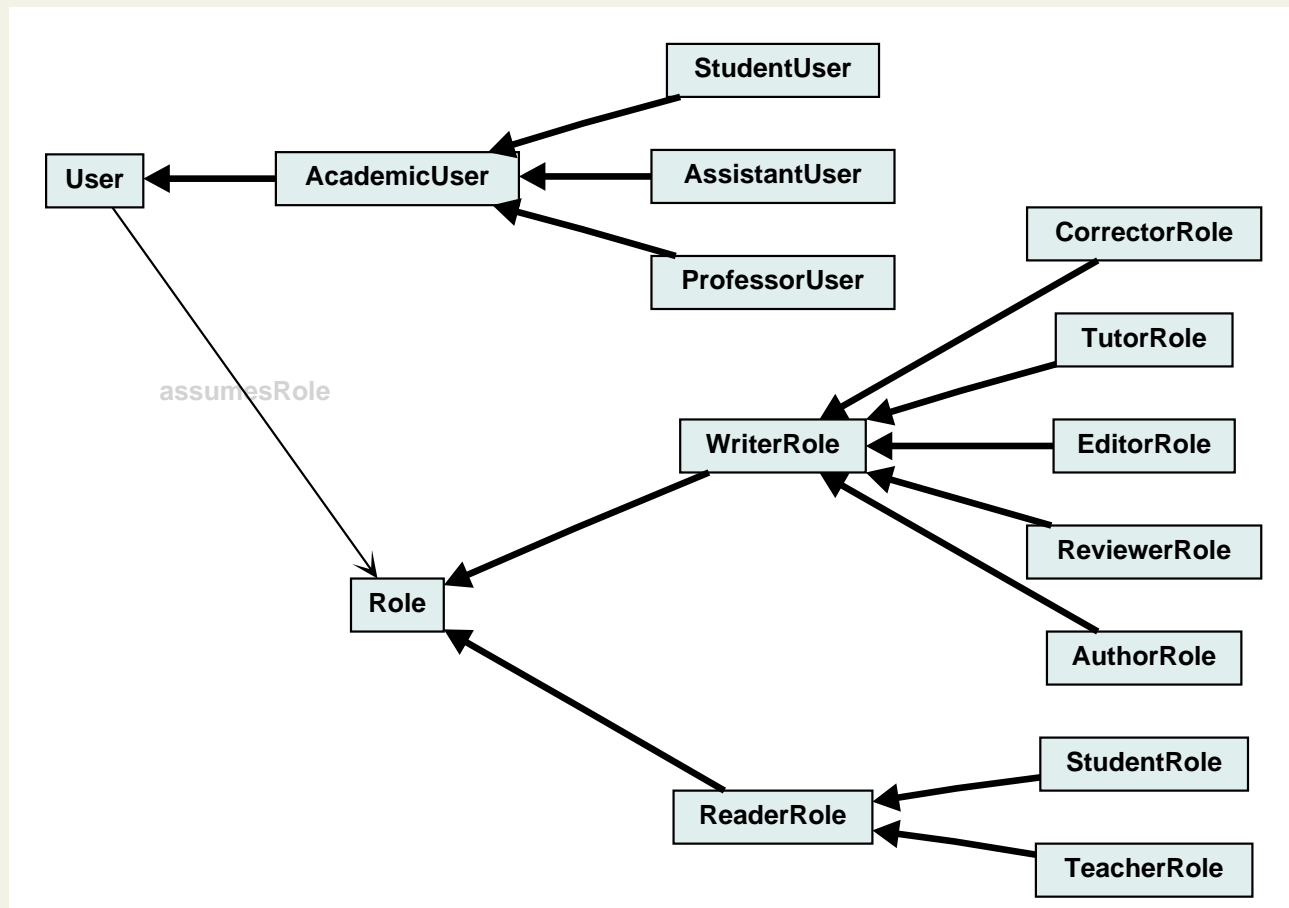
- semantic interrelation via **ontology**
- **slides**, layout, animation
- **refinement** to/**abstraction** from article or book, **variants**
- “Literate Specification” with **CASL**, CATS tools
- **structuring “in-the-large”** via Packages – ontology
- **structure graph**, CATS development graph
- authoring tools, version control, sessions in the **Repository**
- **re-use**, configuration management, change management

Why an Ontology?

Ontology

- Classes
- Relations
 - “typed” by source and target Classes
- subClasses, subRelations
 - inheritance of formal properties
- Objects
 - of a Class, related by Relations
- Axioms
 - Description Logic: OWL
 - CASL

Ontology of Users and Roles



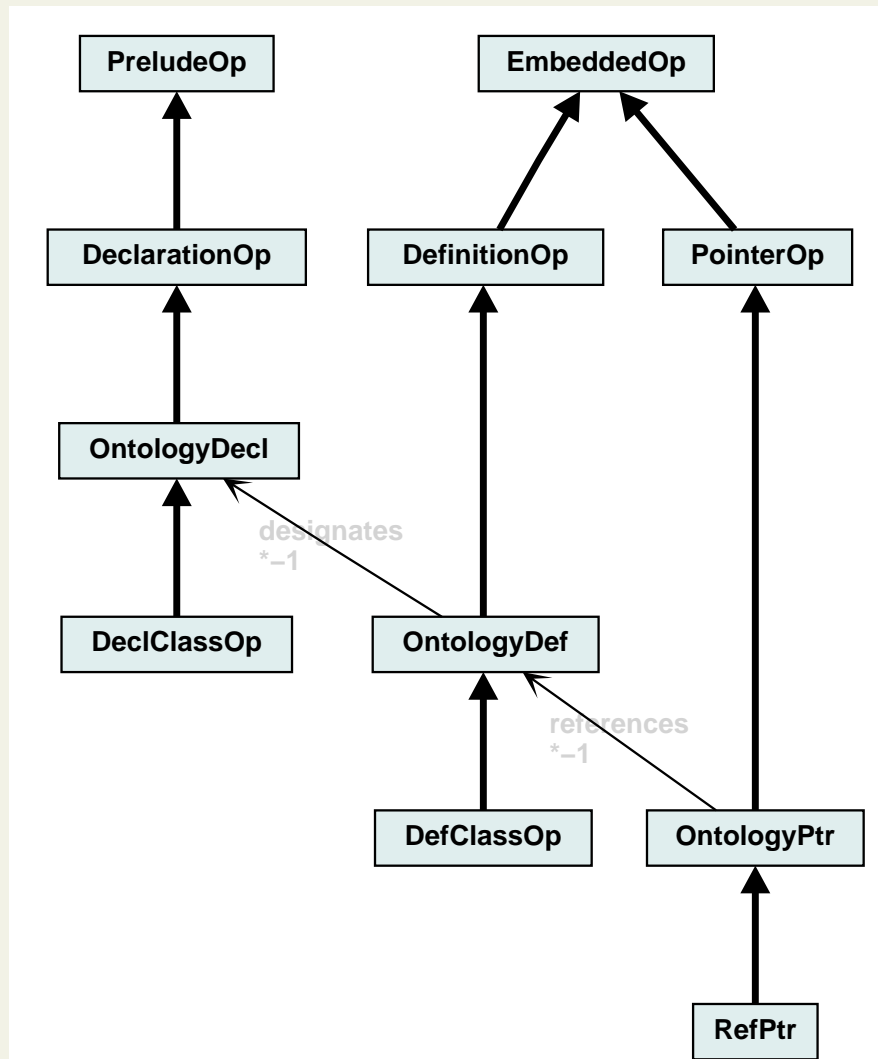
Ontology: Resolution of Ambiguity

a **Student** may assume the role of a **Student**

```
\DeclClass{User}{User}{}  
  \DeclClass{AcademicUser}{Reader}{User}  
    \DeclClass{StudentUser}{Student}{AcademicUser}  
\DeclClass{Role}{Role}{}{\Role}  
  \DeclClass{ReaderRole}{Reader}{Role}{\ReaderRole}  
    \DeclClass{StudentRole}{Student}{ReaderRole}{\StudentRole}  
  
  \DeclRel{*-*}{assumesRole}{assumesRole}{}  
  \RelType{assumesRole}{User}{Role}{\assumesRole}
```

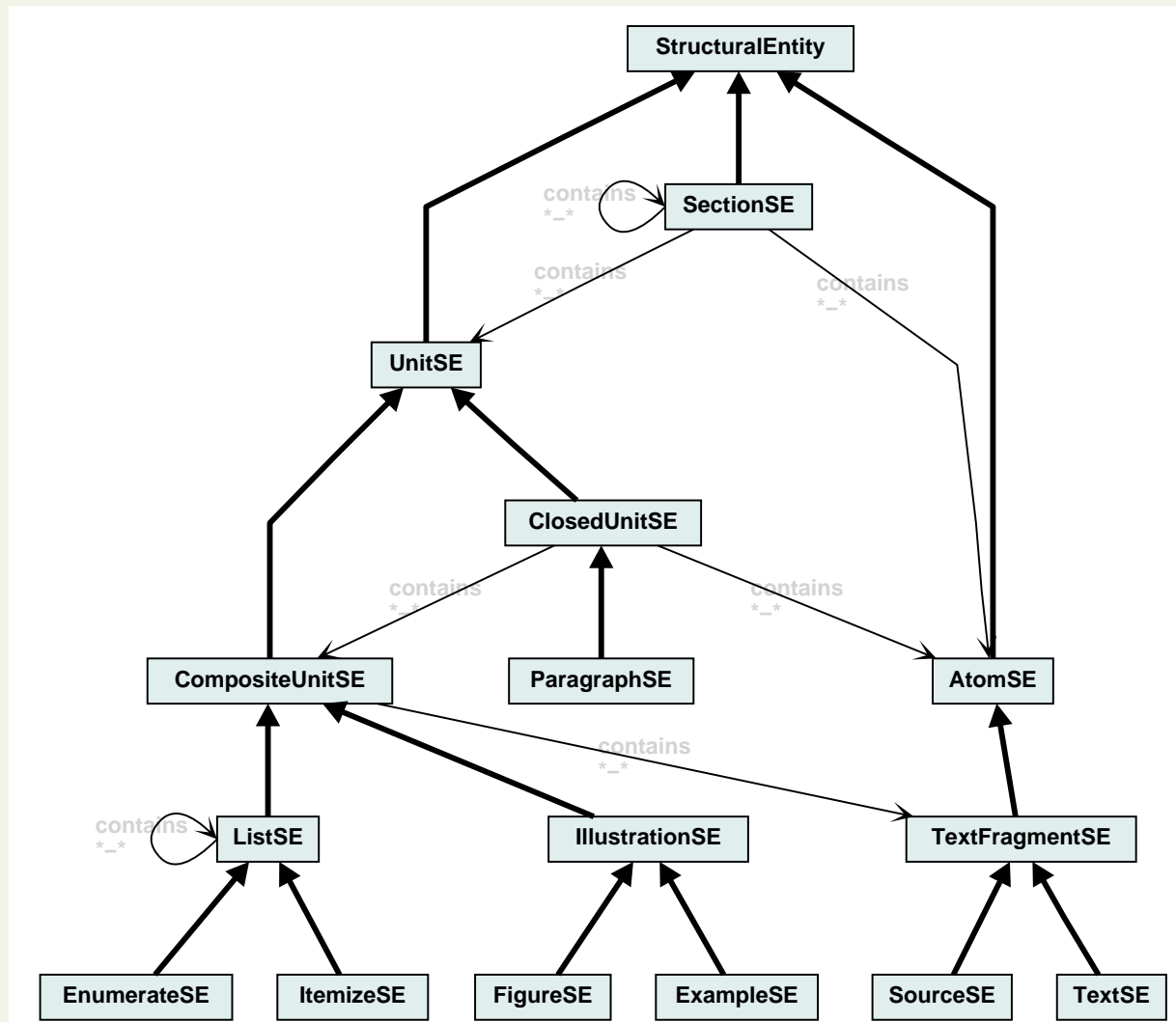
Ontology Declaration, Definition, Reference

- **Declare** in OntologyPrelude (“signature” of a Package) structured via Packages (renaming etc.), language variants
`\DeclClass{StudentRole}{Student}{ReaderRole}`
- **Define** embedded in text at defining occurrence
`\DefClass{StudentUser} Student`
`\DefClass{StudentRole} Student`
- **Reference** embedded in text
`\StudentRole{} Student, \Ref{StudentRole} Student`
or `\Ref[student author]{AuthorRole} student author`

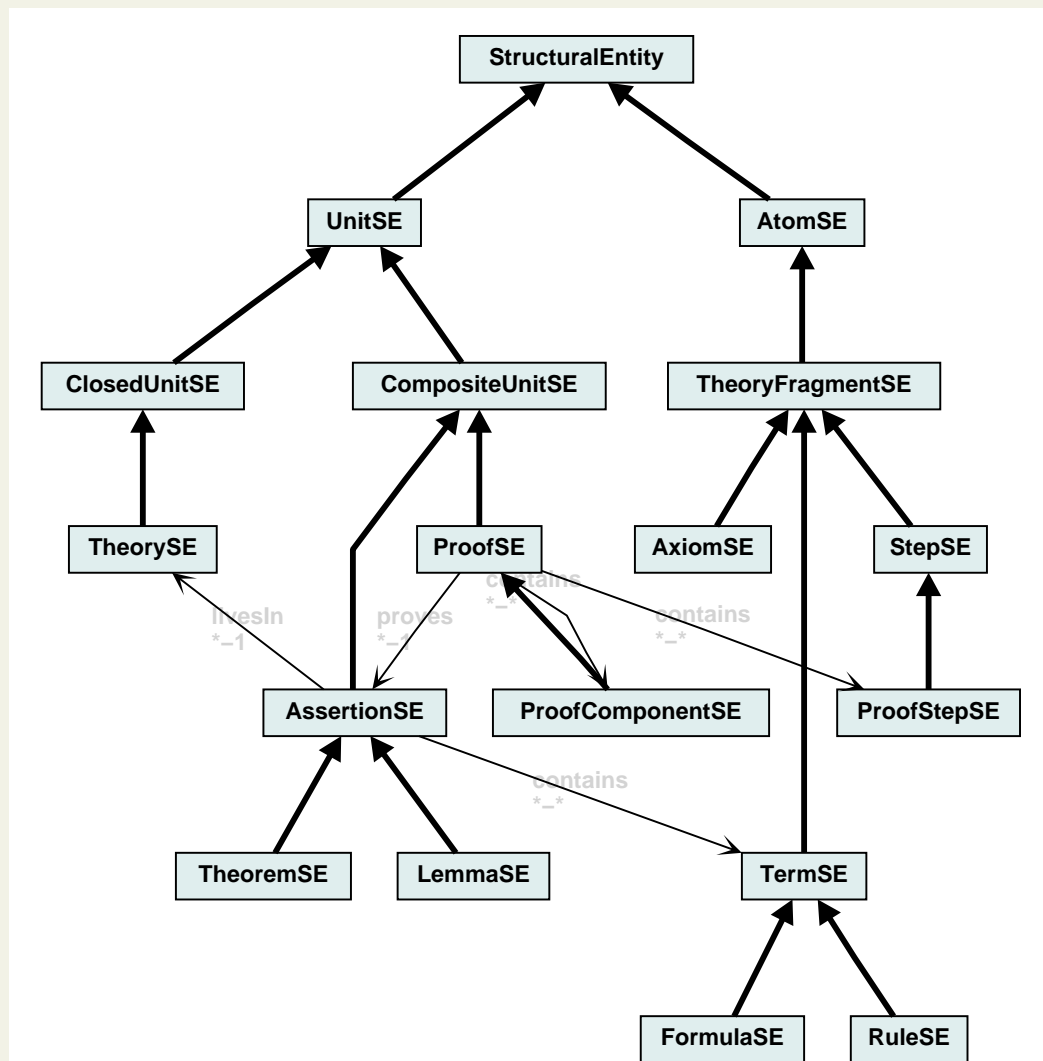


- **Declare**
in **OntologyPrelude**,
- **Define**
at defining occurrence
- **Reference**
in text
- **pointsTo** relations
 - Def **designates** Decl
 - Ref **references** Def

Conceptual and Formal Structural Entities



- **Package** contains
 - Preludes
 - Sections
- **Sections**
 - nested
- **Unit**
 - smallest editing context
 - should fit on one page
- **Atom**
 - no structure
 - tools



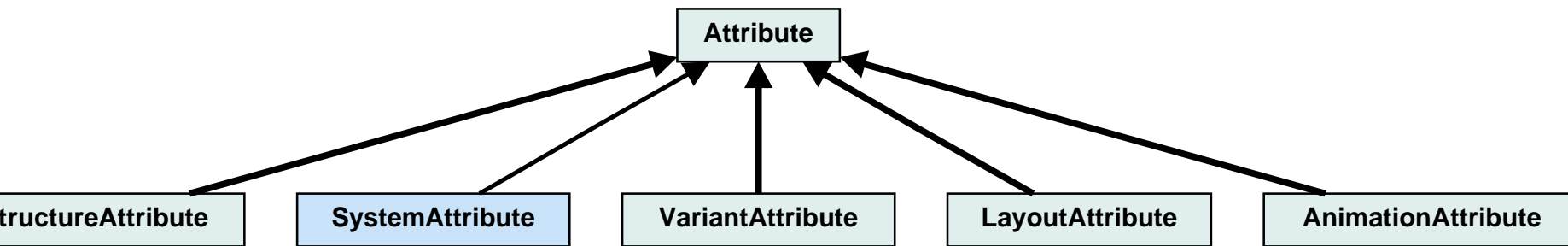
- reliesOn relations
 - Package imports Package
 - Section after Section
 - Theorem livesIn Theory
 - Proof proves Theorem

Extensions

Future Extensions

- macros as in \LaTeX , parameterised
 - `\Macro{\Future}[1]{\NoBold{#1}}`
`\Future{Future} Extensions`
Future **Extensions**
- extension of the **System Ontology**
 - new Structural Entity
 - new Embedded Operation
 - new Relation (?)
- **Generics**

Attributes



Structure Attributes

```
\begin{Section} [Label=Attributes, Title=Attributes,  
    Authors={Bernd~Krieg-Br{"u}ckner}]  
    \begin{Paragraph} [Title=Layout,  
        Authors={Jan-Georg~Smaus, Markus~Roggenbach}]  
    \end{Paragraph}  
    \begin{Paragraph} [Title={Variant Attributes}]
```

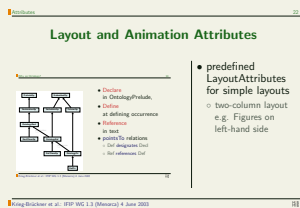
- most recent Authors Attribute,
stack of PriorAuthors Attributes
- Attributes are **inherited** downwards, may be overwritten

Layout and Animation

Layout and Animation Attributes

Layout and Animation Attributes

Layout and Animation Attributes



- predefined LayoutAttributes for simple layouts
 - two-column layout
 - two-column layout e.g. Figures on left-hand side

- predefined LayoutAttributes for simple layouts
 - two-column layout e.g. Figures on left-hand side

Layout and Animation Attributes

Variant Attributes

33

	<i>Hyper</i>	<i>Replay</i>	<i>Interactive</i>
Contents skeleton	Hyper-Medium	Replay in Tool	Interaction
Outline abstracts			
Lecture presentation in class	laptop browsing during lecture	demonstrate use of tool	experiment in tool
Lecture Notes annotated after presentation	offline browsing personal annotation	demonstrate examples	experiment with examples
Course self-contained for self-study	personal navigation dynamic?	model solutions	personal solutions

Krieg-Brückner et al.: IFIP WG 1.3 (Menorca) 4 June 2003

MiS

- predefined
LayoutAttributes
for simple layouts
 - n by m matrix
(as list of lists)

Layout and Animation Attributes

- predefined `LayoutAttributes` for simple layouts
 - two-column layout, e.g. for Figures on left-hand side
 - n by m matrix (as list of lists)
- predefined `AnimationAttributes` for simple animations
 - rollout of list items
 - traversal of matrix: left-to right or top-to-bottom
 - sequential build-up of Figures
 - pointers via arrows
- avoid tedious \LaTeX commands or pause levels

Lectures Held so Far

- Bremen
 - Information security 1 (in German)
 - Object-oriented Software Development
 - Techniques of Correct Software Development
 - Safety-Critical Systems 4: Engineering of Embedded Software
 - Logic (in German)
- Freiburg
- Berlin
- Hagen/Kaiserslautern
- Munich

Lectures Held so Far

- Bremen
- Freiburg
 - Computer-Supported Modeling and Reasoning
 - Automata-Based System Analysis
- Berlin
- Hagen/Kaiserslautern
- Munich
- Saarbrücken

Lectures Held so Far

- Bremen
- Freiburg
- Berlin
 - Distributed Algorithms (in German)
 - Model Checking
- Hagen/Kaiserslautern
- Munich
- Saarbrücken

Lectures Held so Far

- Bremen
- Freiburg
- Berlin
- Hagen/Kaiserslautern
 - Advanced Aspects of Object-Oriented Programming (in German)
 - Software Architecture (in German)
- Munich
- Saarbrücken

Lectures Held so Far

- Bremen
- Freiburg
- Berlin
- Hagen/Kaiserslautern
- Munich
 - Temporal Logic
 - Formal Object-Oriented Software Development
 - Foundations of System Development
 - Model Checking
- Saarbrücken

Lectures Held so Far

- Bremen
- Freiburg
- Berlin
- Hagen/Kaiserslautern
- Munich
- Saarbrücken
 - Security (in German)
 - Formal Software Development (in German)

Rules Discharging Assumptions

Derivation trees are built using **rules**, e.g. \wedge -I.

Rules Discharging Assumptions

Derivation trees are built using **rules**, e.g. \wedge -I.

Each rule application can be associated with the **discharging** of an assumption.

Rules Discharging Assumptions

Derivation trees are built using **rules**, e.g. \wedge -I.

Each rule application can be associated with the **discharging** of an assumption.

A

Rules Discharging Assumptions

Derivation trees are built using **rules**, e.g. \wedge -I.

Each rule application can be associated with the **discharging** of an assumption.

$$\frac{A}{B \rightarrow A} \rightarrow\text{-I}^2$$

Rules Discharging Assumptions

Derivation trees are built using **rules**, e.g. \wedge -I.

Each rule application can be associated with the **discharging** of an assumption.

$$\frac{\frac{[A]^1}{B \rightarrow A} \rightarrow\text{-I}^2}{A \rightarrow B \rightarrow A} \rightarrow\text{-I}^1$$

Rules Discharging Assumptions

Derivation trees are built using **rules**, e.g. \wedge -I.

Each rule application can be associated with the **discharging** of an assumption.

$$\frac{\frac{[A]^1}{B \rightarrow A} \rightarrow -I^2}{A \rightarrow B \rightarrow A} \rightarrow -I^1$$

- Number tags are administered using symbolic labels, making it easy to compose trees.
- Brackets are synchronised with the application of the rule.

Alternative Animations

Sometimes we desire a more sophisticated synchronisation:

$$A \rightarrow B \rightarrow A$$

Want to prove $A \rightarrow B \rightarrow A$.

Alternative Animations

Sometimes we desire a more sophisticated synchronisation:

$$A$$

$$\frac{}{A \rightarrow B \rightarrow A} \rightarrow\text{-I}$$

Want to prove $A \rightarrow B \rightarrow A$.

Should use $\rightarrow\text{-I}$. Assumption A should **eventually** be discharged.

Alternative Animations

Sometimes we desire a more sophisticated synchronisation:

$$\frac{\frac{A}{B \rightarrow A} \rightarrow -I^2}{A \rightarrow B \rightarrow A} \rightarrow -I$$

Want to prove $A \rightarrow B \rightarrow A$.

Should use $\rightarrow -I$. Assumption A should **eventually** be discharged.

Should use $\rightarrow -I$. No assumption B involved.

Alternative Animations

Sometimes we desire a more sophisticated synchronisation:

$$\frac{\frac{[A]^1}{B \rightarrow A} \rightarrow -I^2}{A \rightarrow B \rightarrow A} \rightarrow -I^1$$

Want to prove $A \rightarrow B \rightarrow A$.

Should use $\rightarrow -I$. Assumption A should **eventually** be discharged.

Should use $\rightarrow -I$. No assumption B involved.

Have derived $B \rightarrow A$ assuming A . Now we can apply $\rightarrow -I$.

Animation of Proof Trees

- \LaTeX proof-tree style
- pause levels
 - based on pause command provided by PPower4
 - text fragment (“subtree”) appears at specified pause level range
 - original pause levels are restored afterwards

(Black)Board Presentation, Notes

- overhead screen + script/annotations on laptop

Layout and Animation Attributes

Thumbnail of slide 23 content:

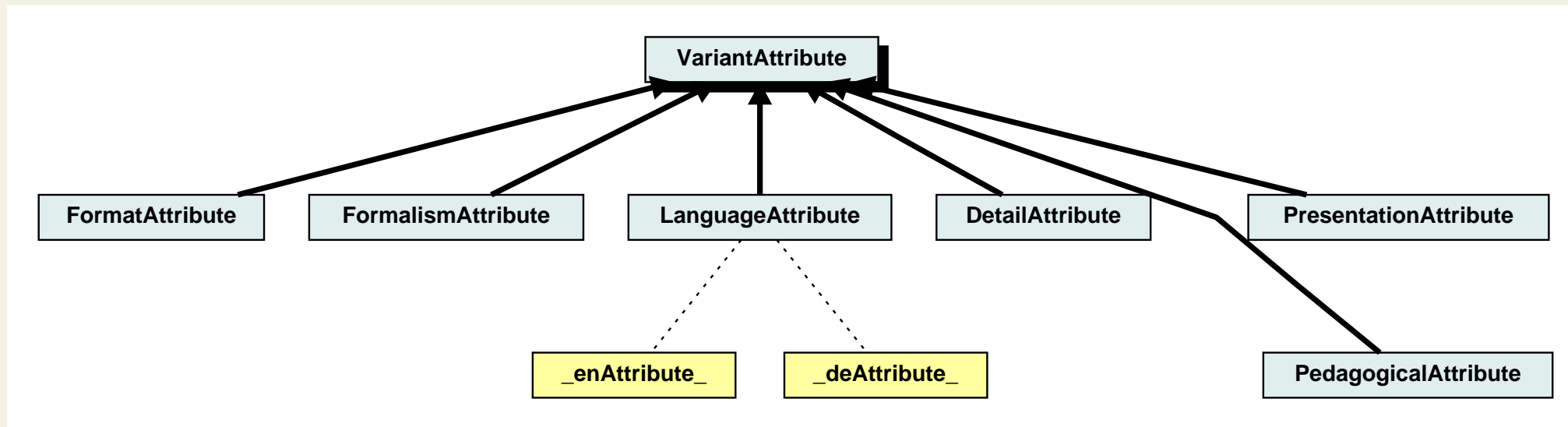
- Layout and Animation Attributes
- predefined LayoutAttributes for simple layouts
 - two-column layout e.g. Figures on left-hand side

Layout and Animation Attributes

- predefined LayoutAttributes for simple layouts
 - two-column layout, e.g. for Figures on left-hand side
 - n by m matrix (as list of lists)
- predefined AnimationAttributes for simple animations
 - rollout of list items
 - traversal of matrix: left-to right or top-to-bottom
 - sequential build-up of Figures
 - pointers via arrows
- avoid tedious \LaTeX commands or pause levels

Variant Attributes

Variant Attributes Ontology



	<i>Paper</i>	<i>Board</i>	<i>Hyper</i>
Contents	<i>Text+Pictures</i>	Manual	<i>Hyper-Medium</i>
skeleton			
Outline			
abstracts			
Lecture			
presentation	handout	black/white board	laptop browsing
in class	before lecture	during lecture	during lecture
Lecture Notes			
annotated	handout	annotated	offline browsing
after presentation	after lecture	manuscript	personal annotation
Course			
self-contained	course script	integrated	personal navigation
for self-study	personalized?	(manu)script	dynamic?

	<i>Hyper</i>	<i>Replay</i>	<i>Interactive</i>
Contents	<i>Hyper-Medium</i>	<i>Replay in Tool</i>	<i>Interaction</i>
skeleton			
Outline			
abstracts			
Lecture			
presentation	laptop browsing	demonstrate	experiment
in class	during lecture	use of tool	in tool
Lecture Notes			
annotated	offline browsing	demonstrate	experiment with
after presentation	personal annotation	examples	examples
Course			
self-contained	personal navigation	model	personal
for self-study	dynamic?	solutions	solutions

Variants

- **one structure graph** for all variants
- **different variants** for one node with the same Label
- can be edited **side-by-side**, or
- for a given **selection** of VariantAttributes
a **projection** of the structure graph
can be used for specialised editing
- **variants** are perhaps the **most important** novelty

Change Management

Inheritance for Relations in Ontology

```
\DeclRel{*-*}{comprises}{comprises}{relatesDocConstructs}  
  \DeclRel{<-}{contains}{contains}{comprises}  
\DeclRel{>}{reliesOn}{reliesOn}{relatesDocConstructs}  
  \DeclRel{}{imports}{imports}{reliesOn}  
  \DeclRel{}{livesIn}{livesIn}{reliesOn}  
  \DeclRel{}{proves}{proves}{reliesOn}  
  \DeclRel{}{after}{after}{reliesOn}  
\DeclRel{->}{pointsTo}{pointsTo}{relatesDocConstructs}  
  \DeclRel{}{designates}{designates}{pointsTo}  
  \DeclRel{}{references}{references}{pointsTo}  
\DeclRel{->}{variantOf}{variantOf}{relatesDocConstructs}
```

Consistency and Completeness

- **reliesOn** relations
 - strict order (irreflexive, transitive)
 - **completeness**: at least one target,
e.g. no dangling **Theorem livesIn Theory**
 - **consistency**: **reliesOn** relations internally consistent
e.g. **A after B** and **B after C** implies **A after C**
 - **consistency**: **reliesOn** relations consistent with each other
e.g. **A livesIn B** implies **A after B**

Literate Programming, Literate Specification

Literate Specification

- complete specification in extra Section (e.g. appendix)
 - complete Theory with internal structure, down to Atoms
 - Labels allow sharing, e.g. IncludeAtom in document
- use of tools, e.g. for CASL
 - internal structure converted to special annotations in CASL
 - CATS can be applied to ASCII format
 - original structure is recovered from special annotations
 - additional \LaTeX variants are generated,
associated one-to-one to ASCII variants in Repository
 - revisions are only made in source

Introducing: the Repository

- What is a **repository**?
 - central database of **all documents** and their **history**

Introducing: the Repository

- What is a **repository**?
 - central database of **all documents** and their **history**
- What does it do?
 - **Version control**
 - **Configuration management**
 - **Change management**

Introducing: the Repository

- What is a **repository**?
 - central database of **all documents** and their **history**
- What does it do?
 - **Version control**
 - **Configuration management**
 - **Change management**
- **Well-known** from Software Engineering
 - RCS, CVS, make, . . .

Introducing: the Repository

- What is a **repository**?
 - central database of **all documents** and their **history**
- What does it do?
 - **Version control**
 - **Configuration management**
 - **Change management**
- **Well-known** from Software Engineering
 - RCS, CVS, make, . . .
- **New**: application to MMiSS documents

The MMiSS Repository

- Fine-grained approach.
- Objects: **Packages, Units** and **Atoms**
 - sections, definitions, theorems, text fragments . . .
- **Folders** allow grouping of objects.
- **Relations** between objects:
 - contains (structural links, explicit or implicit)
 - includes (sharing)
 - . . . relations from the ontology
- Aims: **Change Management**
 - keep track of dependencies
 - maintain consistency and completeness

User Interaction with the Repository

- Different versions \rightsquigarrow **version graph**
- Dependencies between objects \rightsquigarrow **structure graph**
 - `contains`, `livesIn`, `references`

User Interaction with the Repository

- Different versions \rightsquigarrow **version graph**
- Dependencies between objects \rightsquigarrow **structure graph**
 - contains, livesIn, references
- **Visualisation** of repository as a **graph**
 - dag, starting with **root folder**
 - user-defined **views**
- Editing by **direct manipulation**

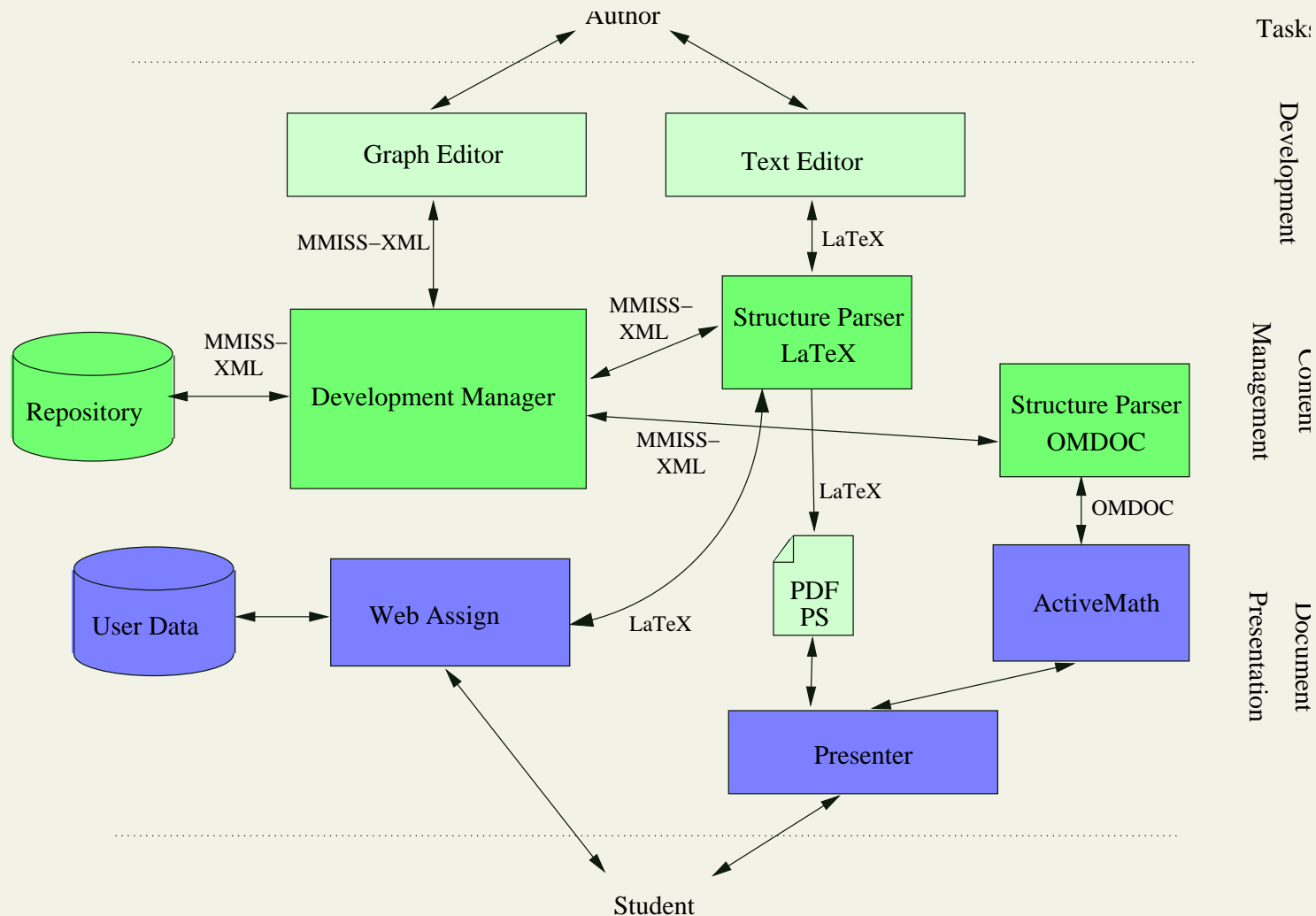
Tool Interaction with the Repository

- Repository exchange format: MMiSS-XML
 - XML format based on MMiSS Document Structuring
- User does not want to see XML but:
 - \LaTeX with text editor
 - WYSIWYG tools
 - Method-specific tools

Tool Interaction with the Repository

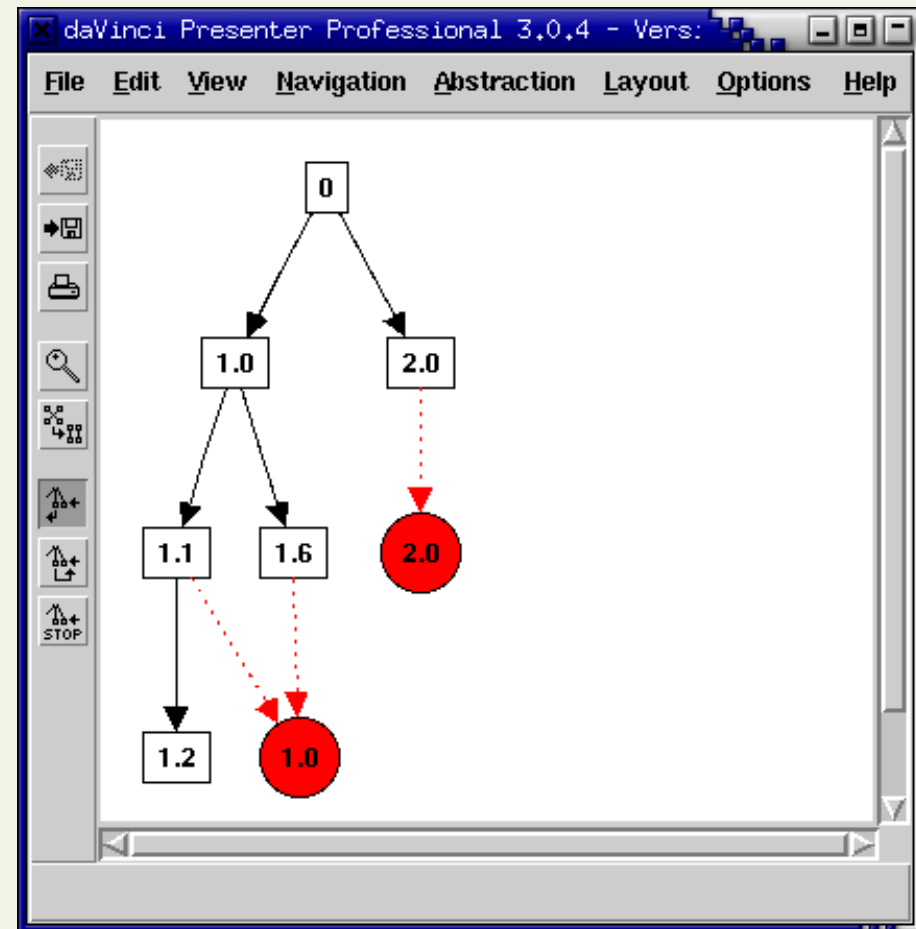
- Repository exchange format: MMiSS-XML
 - XML format based on MMiSS Document Structuring
- User does not want to see XML but:
 - \LaTeX with text editor
 - WYSIWYG tools
 - Method-specific tools
- Structure parser for external exchange formats.
- External exchange formats:
 - \LaTeX , OMDoc (planned), . . .

MMiSS Workbench System Architecture



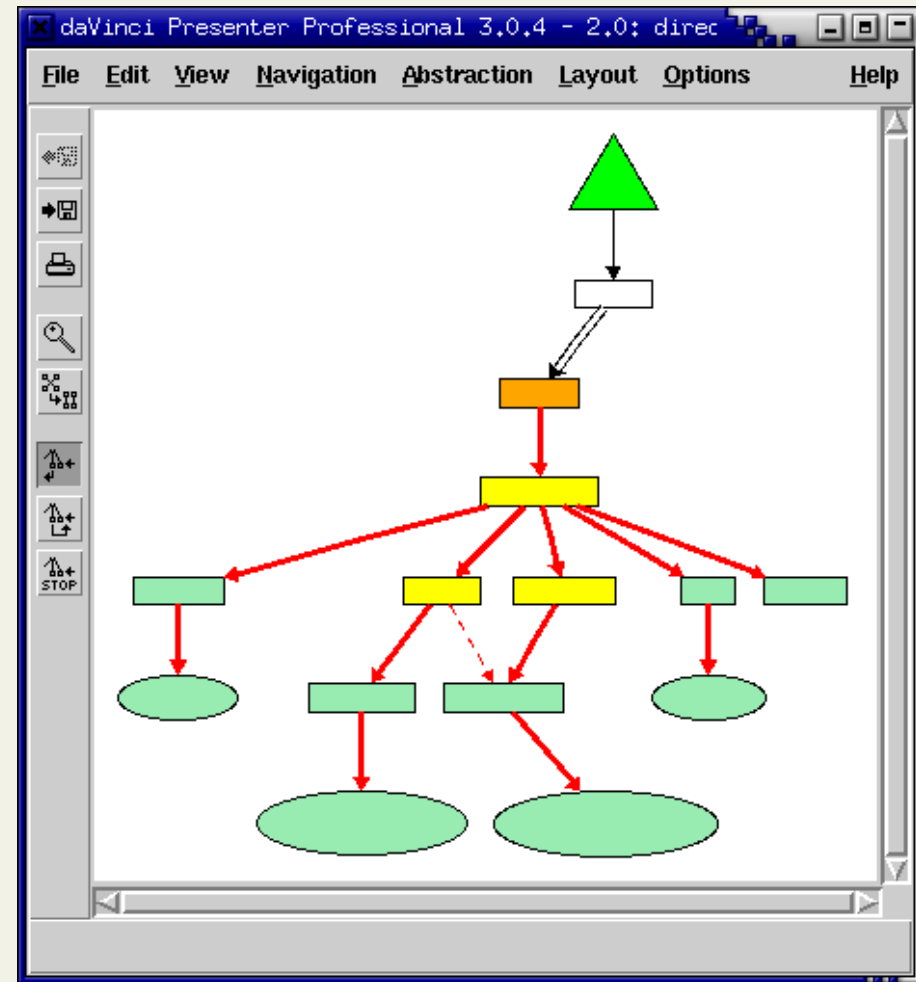
Version Graph

- Checkout to edit
- Commit back to repository
- Merge
- Shared View



Structure Graph

- **Folders** contain other objects
- Operations per **object**
 - **Create** or **import** objects
 - **Edit** objects
 - **Export** and **preview**
 - Display and select **variants**
- **Private** View
- One per checked-out version



Authoring Tools: XEmacs

- **MMiSS \LaTeX** mode
 - Insert **MMiSS \LaTeX** elements, commands, etc.
 - cf. **AuCT \LaTeX**
 - Parsing on-the-fly
 - Driven from workbench
- **Active buttons** for elements
- **Synthesis** of **structured** and **textual** editing

```

emacs: IntroAlgSpec
File Edit View Cmds Tools Options Buffers Command LaTeX MMSSTeX I
IntroAlgSpec
[IntroAlgSpec:
\begin{Section}
[Label={IntroAlgSpec},
Title={Signatures,
Terms and Algebras}]
[Abstract]
[Intro:
\begin{Introduction}
[Label={Intro},
Title={Why Formal Specification?}]
[Reasons]
\end{Introduction}]
[Signatures]
[Algebra]
[Summary]
\end{Section}]
ISO8--**XEmacs: IntroAlgSpec (MMSSTeX Fill)---
Not over a window.

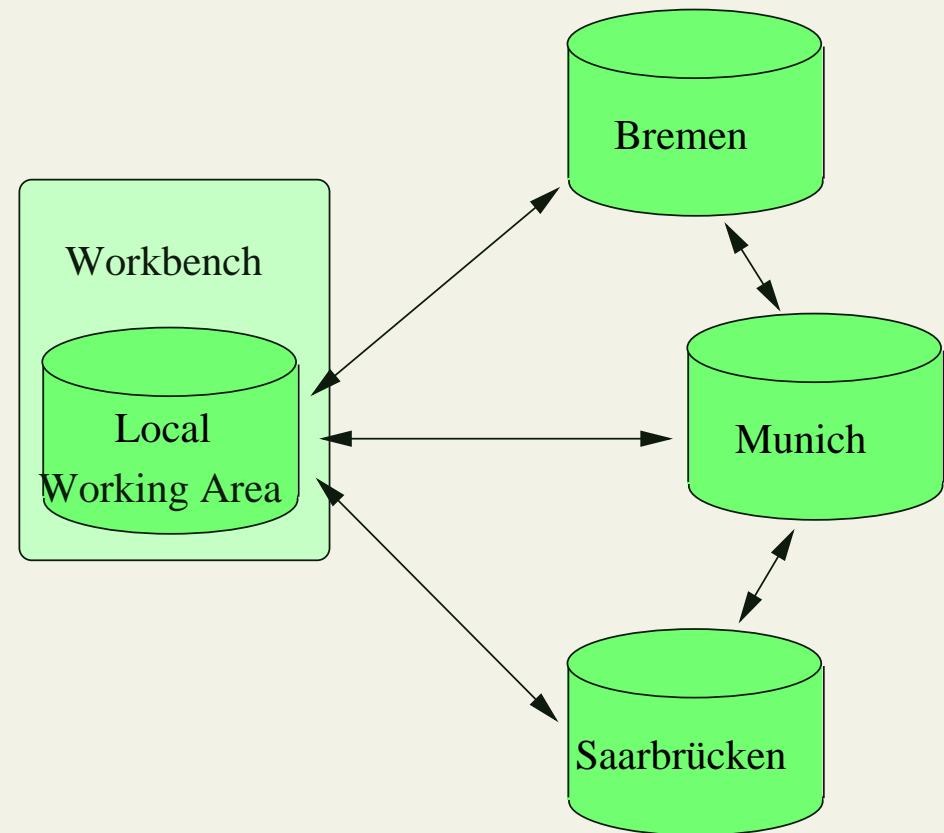
```

The MMiSS Workbench at Work

Start Workbench

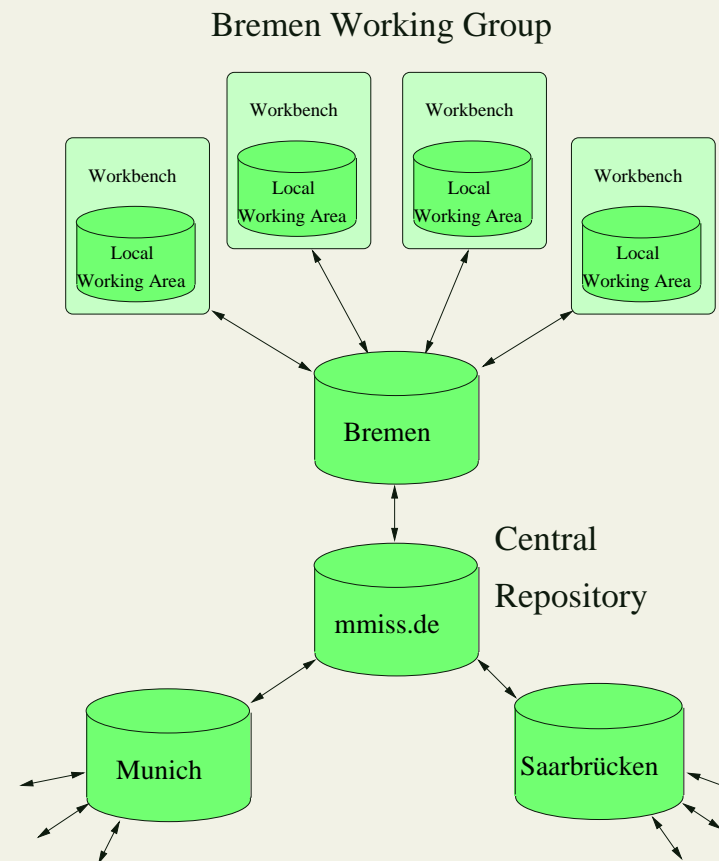
Distributed Development

- Client connects to **remote servers**
 - Communication via Sockets
- Client contains **local working area**
- **Copy versions** between
 - local working area and servers
 - servers
- Flexible setup
- Demonstrate. . .



Distributed Development

- Client connects to **remote servers**
 - Communication via Sockets
- Client contains **local working area**
- **Copy versions** between
 - local working area and servers
 - servers
- Flexible setup
- Demonstrate. . .



The Implementation

- Implemented in **Haskell** (80 Kloc)
- Based on **events** as central notion of interaction
- **Generic** content model (other XML formats)
- Uses
 - Tcl/Tk for user interface (HTk)
 - BerkeleyDB (<http://www.sleepycat.com>) as database
 - HaXML for generic content model
 - daVinci for graph visualisation
- Runs on Linux and Solaris

Summary

- Repository holds all **documents** and their **history**.
- Provides
 - configuration management and version control,
 - **change management**: consistency and completeness,
 - conversion between different formats,
 - distributed development
- XEmacs with MMiSS \LaTeX mode
 - Textual **and** structural editing
 - Closely coupled to repository
- Please **download** it at <http://www.mmis.de>.
- Wanted: **user feedback**.

Conclusion

Goals of the MMiSS Project

- develop **slides**, lecture notes, **courses**
 - DetailAttribute
- work on the **board**, with slides, interactively with **tools**
 - PresentationAttribute
- embed mathematical formulae, **programs**, etc.
 - L^AT_EX, Interactive use of tools
- manage **English** and **German** documents in parallel
 - variants, LanguageAttribute

Goals of the MMiSS Project

- publish **complete** and **consistent** “packages”
 - versions, configuration management
- (partially) **re-use** the slides of a colleague
 - re-use by structural sharing, views
- be made aware of the **changes** made by your colleague
 - sustainable development via change management
- agree with your colleague on a **uniform terminology**
 - ontology

Join the
International MMiSS Forum
www.mmiss.de
Open Source!

UDoc — We Help

What We Have Learned From Formal Methods

- slide as **refinement** to / abstraction from article or book
- **structuring “in-the-large”** of Packages, of ontology
- formalisation of **consistency** and **completeness**
- **structure graph**, CATS development graph (subsumed)
- **conservative extension** for configuration management
- formal specification of **ontology**

Potential IFIP WG 1.3 Tasks

- standard
 - (?) support **UDoc** L^AT_EX slide and document standard
- content
 - re-usable **courses**
 - **coordinated** slides with papers
 - language **variants**
 - formalism **variants**: programming / specification languages
- research issues
 - “Literate Specification, Proofs, ...” with **CASL**, CATS tools
 - launch CASL in the **ontology** area