Integrating the Text-Editor TEXMACS with the Proof Assistance System ΩMEGA using PLATΩ

Serge Autexier, Christoph Benzmüller, Andreas Franke,

Dominik Dietrich, Henri Lesourd, Marc Wagner

DFKI GmbH & CS Department, Saarland University, Saarbrücken, Germany

CHIT-CHAT'06, 21st December 2006

Nijmegen, Netherlands

Motivation

- Evolution of computer-supported mathematics
- Low user-friendliness of actual proof assistants
- Interfacing scientific text-editors instead of yet another proof assistant GUI

Source: Autexier, Wagner

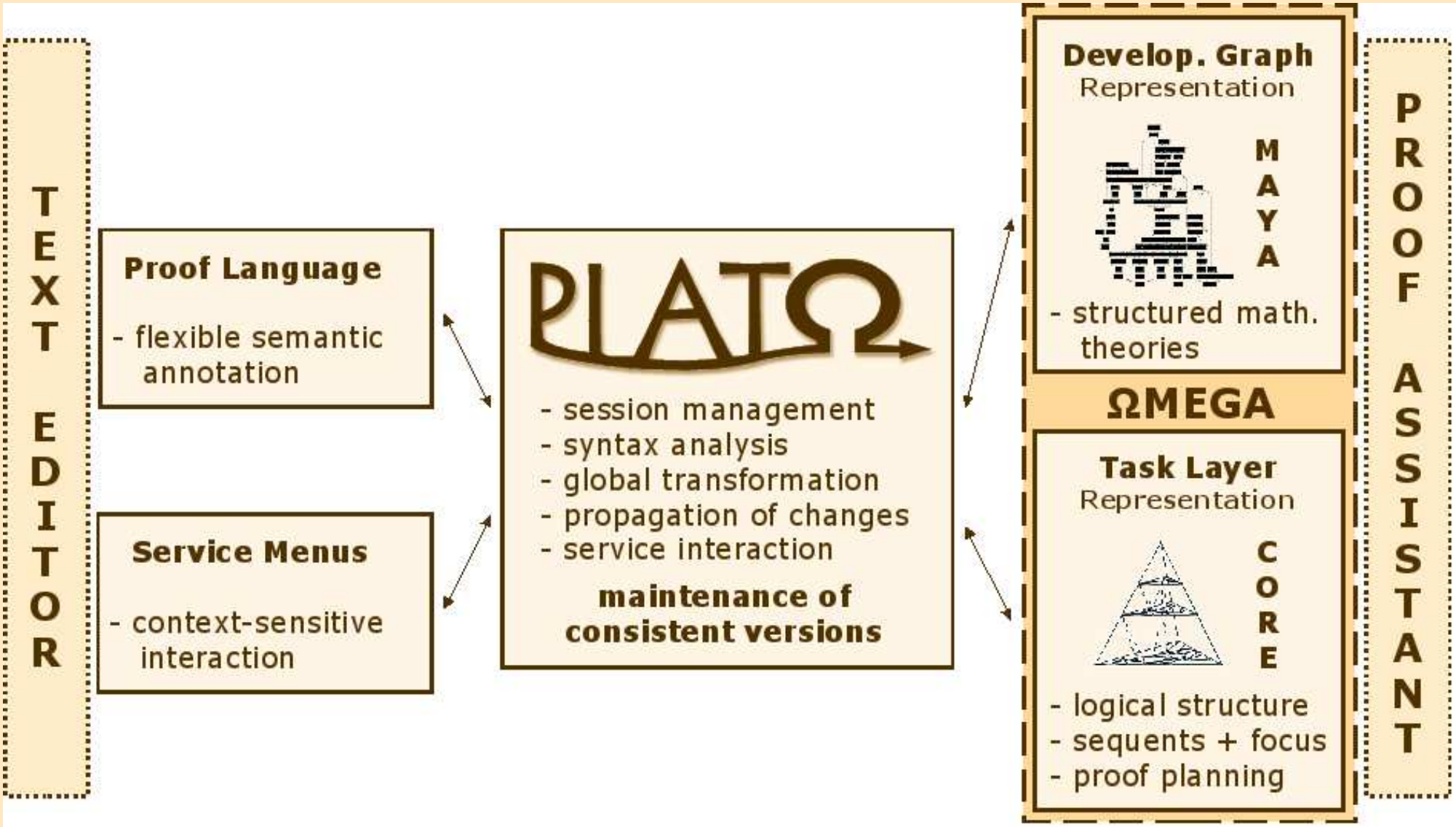
System Architecture

Source: Autexier, Wagner

The Mediator PLATΩ

Source: Autexier, Wagner

Mediation Problem



Source: Autexier, Wagner

Document Language

- **Semantic Annotation of Natural Language**
  - Macros in the text-editor
  - Individual layout style
- **Designed to support**
  - Textual structure of proofs
  - Flexible and multiple positioning
  - Underspecification
  - Alternative proof attempts
- **Parameters**
  - (Sub-)Languages for formulas, definitions and references

```
[document]
└[theory]
  ├[context]
  │ └[reference]
  ├[axiom]
  ├[definition]
  ├[theorem]
  │ ├[precondition]
  │ │ └[formula]
  │ └[conclusion]
  │   └[formula]
  └[proof]
    ├[cases]
    │ ├[case]
    │ └[proof]
    ├[subgoals]
    ├[assumption]
    ├[fact]
    │ ├[by]
    │ └[obtain]
    └[done]
```

Source: Autexier, Wagner

Global Tranformation

- Normalizing the flexible semantic annotations
- Separating proofs from theory knowledge
- Transforming linear proofs into the tree-like proof structure

Proof Language (PL)

- Theorems (flexible)
- Proofs (flexible, linear)
- Natural language text

Intermediate Language (IL)

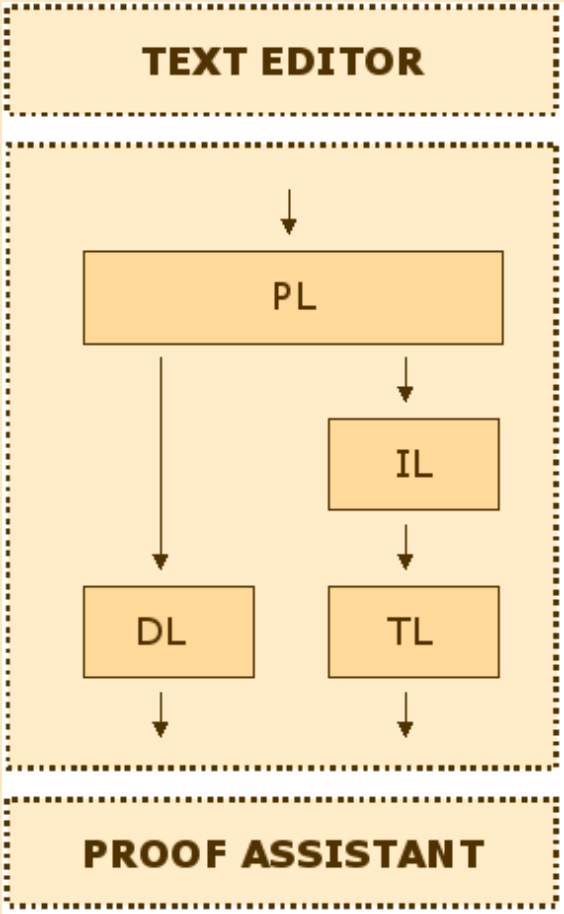
- Proofs (rigid, linear)

Development Graph Language (DL)

- Theorems (rigid)

Task Language (TL)

- Proofs (rigid, tree)



Source: Autexier, Wagner

Propagation of Changes

- **Semantic-based difference computation**
- **Efficient transformation of differences**
- **Preserve partial verifications in proof datastructure of PA**

Proof Language (PL)

- Theorems (flexible)
- Proofs (flexible, linear)
- Natural language text

Intermediate Language (IL)

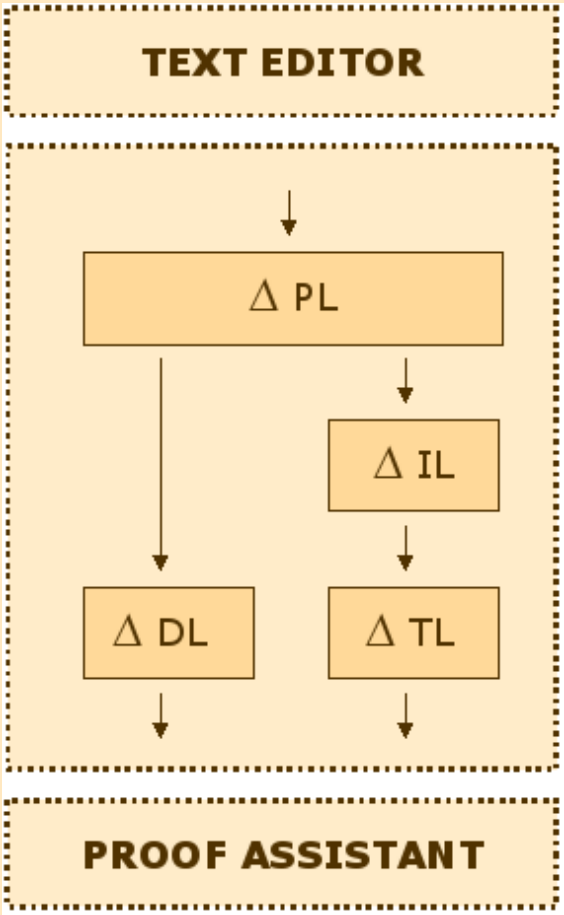
- Proofs (rigid, linear)

Development Graph Language (DL)

- Theorems (rigid)

Task Language (TL)

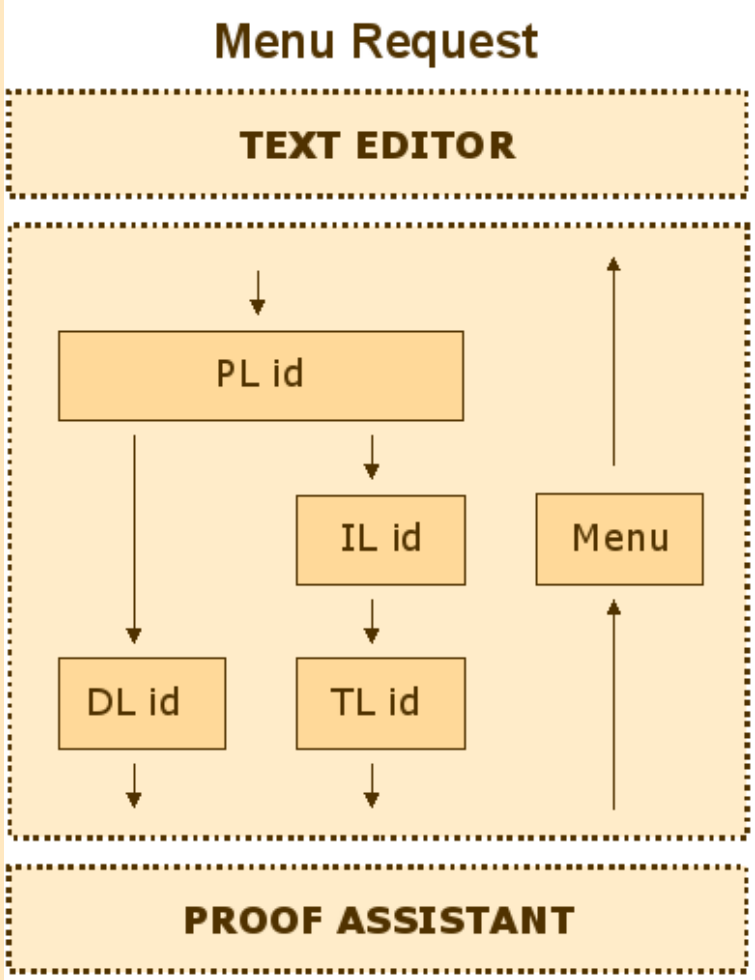
- Proofs (rigid, tree)



Source: Autexier, Wagner

Service Interaction: Requesting a Menu

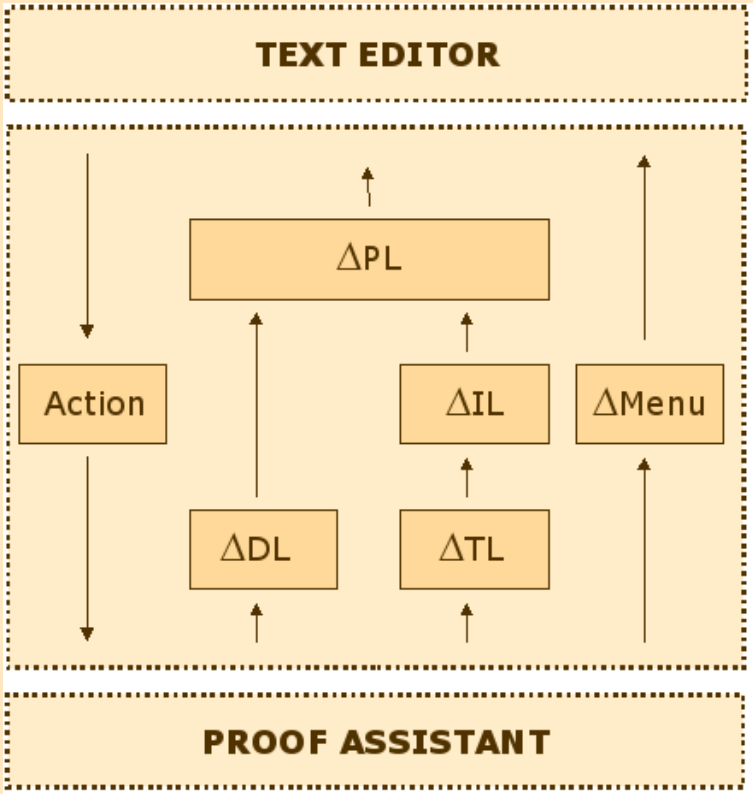
- The user selects object in the text-editor
- Lookup all related objects in the maptable of the transformation
- Request a menu from the PA for these objects
- Display the menu in the text-editor



Source: Autexier, Wagner

Service Interaction: Executing Actions

- User selects an action and its arguments in the menu
- Evaluate this action in the proof assistance system
- Evaluation result:
  - Patches to the menu differences (nested evaluation)
  - Patches to the DL and TL documents (top level evaluation)
- Transformation of DL/TL patches into PL patches



Source: Autexier, Wagner

The Mediator PLATΩ

- XML-RPC server as interface for the text-editor
- Connection to proof assistant in Lisp
- Operating as online webservice or local plugin

Source: Autexier, Wagner

The Plugin in TEXMACS

Source: Autexier, Wagner

Rôle of the Plugin

- Key and session management
- Patch applications on the TEXMACS document
- Resolve and check references

  . . . by [Definition of $\subseteq$] . . .
- Communicates with PLATΩ by fully annotated documents
- Manually writing a fully annotated document is tedious
  - Have a context sensitive menu to write the annotations
  - May be acceptable for the user to write large structures (begin definition, theorem, proof, etc.)
  - But certainly not for formulas . . .
    - To obtain: $x \in U \Rightarrow x \in V$
    - Write:

```
\imp{\in{\V{x}}{\V{U}}}{\in{\V{x}}{\V{V}}}
```

Writing Formulas

- **Use a parser for formulas**
- Allow the user to define its own syntax for any concepts

```
\notation{for=in}{
Let \declare{x} be an element and \declare{A} be a set.
Then we write \denote{x \in A}, \denote{x is in A},
\denote{x is element of A}, or \denote{A contains x}.}
```

Notation. Let **x** be an element and **A** be a set. Then we write **x** $\in$ **A**, **x is in A**, **x is element of A**, or **A contains x**.

- **Procedure:**
  - Scan all notation definitions, convert automatically into parser grammar rules
  - Using parser generator create (document) specific parser
  - Use that parser to parse formulas

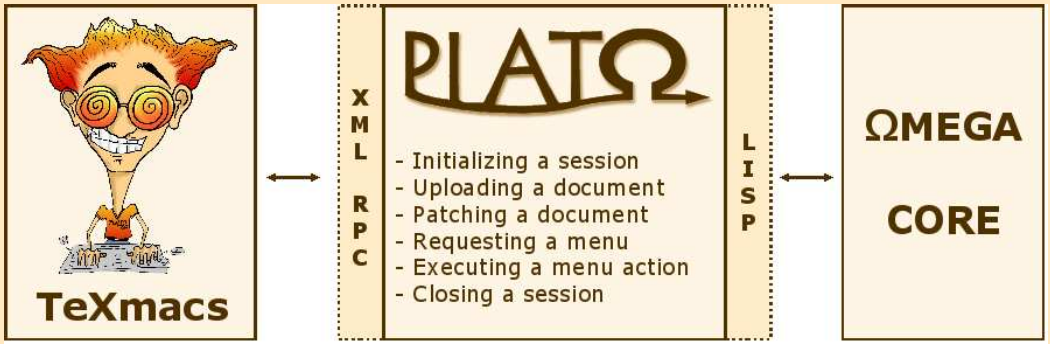
Source: Autexier, Wagner

The Parser Generator

- Standard LALR(1) parser generator implemented in Scheme
- Creates parser that generates all possible readings
- Allows specification of external call-backs to use to single-out invalid readings.
  Example: Could be used to integrate a “refiner”
- Sophisticated specification mechanism for precedences of operators

Source: Autexier, Wagner

The Interface of the Proof Assistant



Source: Autexier, Wagner

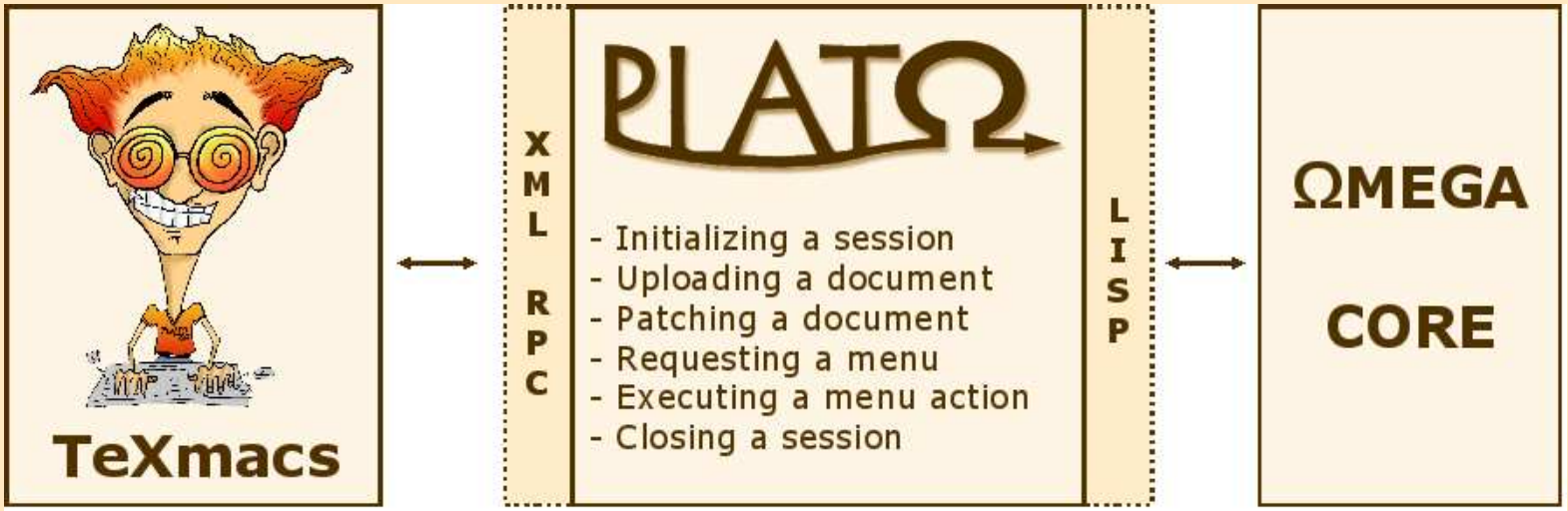
Features of the Proof Assistant

- Create inference rules from axioms/lemmas
  Example: $\forall A, B : set.(A \subseteq B \wedge B \subseteq A) \Rightarrow A = B$

$$\frac{A \subseteq B \qquad B \subseteq A}{A = B}$$

- Proof construction on that level (assertion level)
- Requires proof representation that allows encoding of proof continuations
- Every (additional) feature of the PA is immediately available on the corresponding text part.
  Example: Automatic Theory Exploration System MATHSAID

System Demo



- Definition of concepts and their notations
- Writing Axioms, Conjectures using pre-defined and user-defined notations
- Proof support: interactive and automatic

Source: Autexier, Wagner

Related Work

- **Automath, Mizar, Isar:**
  - Balanced compromise between machine processable and human readable
- **Grammatical Framework:**
  - Framework to define grammar for an abstract and a concrete syntax
- **PCOQ:**
  - Schematic quasi-natural language output
- **Nuprl, Clam, Omega/P.Rex:**
  - Natural language processsing technique to generate proof descriptions

Source: Autexier, Wagner

Related Work

- **Theorema:**
  - Strictly separated formal and informal parts
- **Mathlang:**
  - Top-down from natural language
  - Use annotations for structure, no parser as well
  - Still even more far away from PAs
- **ProofGeneral:**
  - Top-down processing of documents
  - Documents are input format of PA rather than of some typesetting program.

Source: Autexier, Wagner

Conclusion

- **Have a stable connection between TEXMACS and ΩMEGA**
  - PLATΩ deals with all mediating (translation, consistency, patching, relationship between parts of text and formal parts in PA, menus)
  - Clean interface to text-editor
  - Parameterized over language for formulas, definitions and references
- **Either side can be enhanced without affecting the mediator**
  - PLATΩ plugin:
    - User-definable notation used when parsing formulas
    - Add more NL analysis to automatically annotate text(parts)
    - Add NL generation (incremental)
  - ΩMEGA:
    - Added theory exploration, classical ATPs are available
    - Increase proof search automation

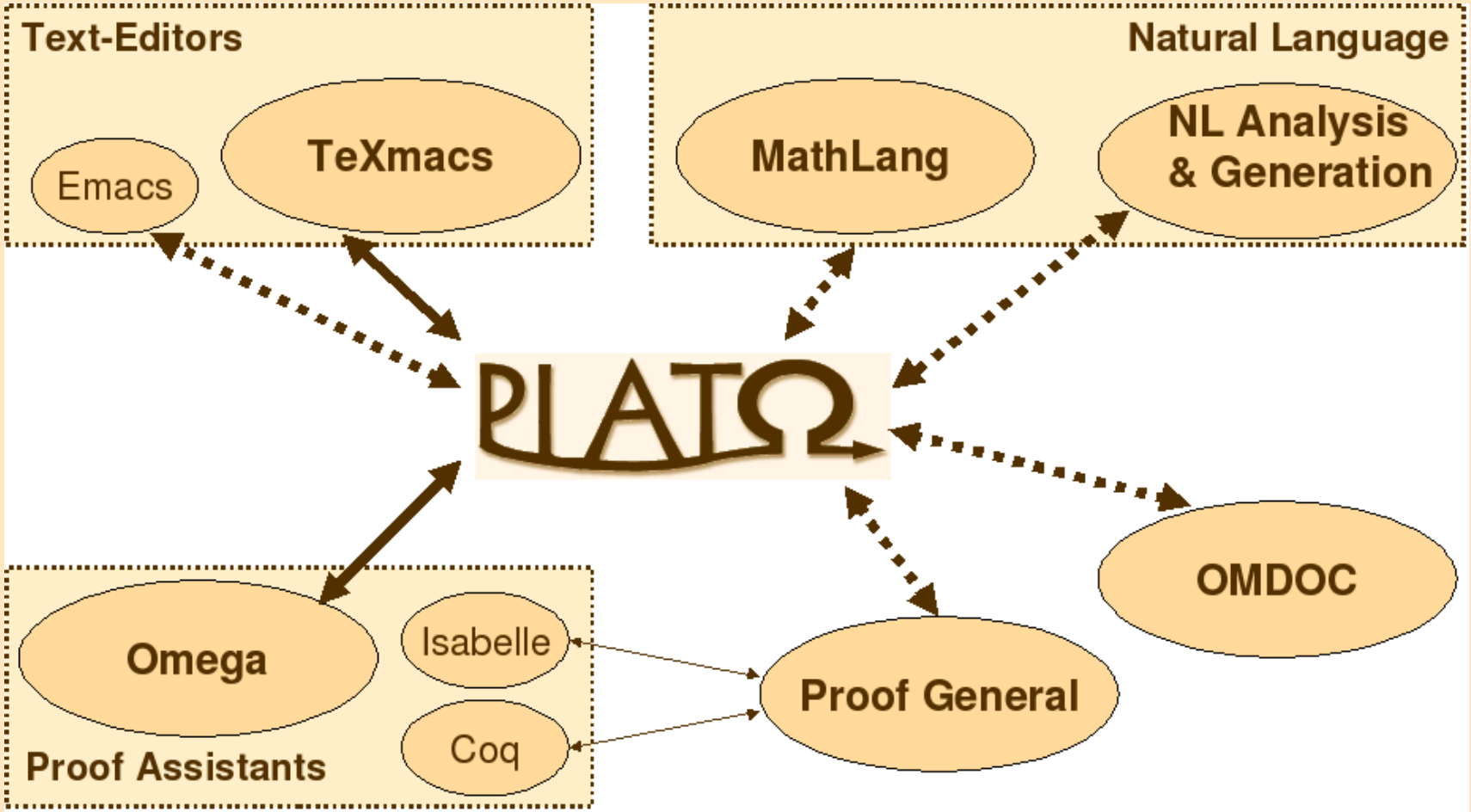
Source: Autexier, Wagner

Future Work

- **Technicalities:**
  - XML-RPC Interface for PAs (OMDOC based)
  - Asynchronous communications
- **Editing:**
  - Library mechanism
  - Dependent types (Scunak, ΩMEGA'07, others are welcome)
  - Support overloading
  - More natural language analysis and NL generation
  - Collaborative editing
- **Use it to formalise mathematics**

Source: Autexier, Wagner

Future Work



Source: Autexier, Wagner