ISAC
Appendices to
the Analysis and Design Documents

Andreas Griesmayer, Alan Krempler, Walther Neuper
{agries,akremp,neuper}@ist.tugraz.at

Revision : 1.1.1.1
Contents
Part VII

Appendices
Appendix A

List of terms used in the ISAC-project

**Browser-Window (Browser-Window)** is the presentation of data generated by the browser, which also handles the http requests generated by the browser window. (The browser-window is that what usually is called a 'browser').

**Browser (Browser):** There are browsers for theories, problems, methods and examples. The browser generates the html-representation of the respective browser-window, and handles the respective http-requests.

**Calculation (Rechengang)** leads from the description of an example via the modeling phase, the specification phase and the solving phase to the result.

**Calculator (Rechner)** is that part of the SML-kernel which does single steps of calculation without touching the proofstate.

**Current formula (aktuelle Formel)** is the unique formula marked on the worksheet. If another item on the worksheet is marked, the formula closest (.. to be specified) to the marked item is the current formula.

**Course admin (Kurs-Administrator)** is a person administering the use of ISAC for learning within a group of learners.
Course designer (Kurs-Designer) edits the example collection which can be solved by a given math knowledge base (edited by a mathematics author) and/or edits explanations within the math knowledge base.

Decorated knowledge (Erweitertes Mathematik-Wissen) is the mathematics knowledge(base) plus explanations.

Description (Beschreibung) of an example consists of formulas, eventually of text and/or a figure. The modeling phase transforms the description into a model.

Dialog atom (Dialog-Atom) is a predefined, minimal unit of interaction between the learner and ISAC. These atoms are symmetrical w.r.t. the two dialog partners.

Dialog pattern (Dialog-Muster) is assembled from dialog atoms such that it can adapt to certain situations in a dialog, e.g. if the learner produces many errors. Dialog patterns are designed and implemented by a dialog author.

Dialog mode (Dialog-Modus) is assembled from dialog patterns and supports certain learning strategies, e.g. exploratory learning, written examination etc. Dialog patterns are designed and implemented by a dialog author.

Dialog profile (Dialog-Profil) defines certain dialog modes for examples in the example collection for a certain course. A dialog profile is defined by a course designer and set/reset for a specified duration during a course by the course admin.

Dialog author (Dialog-Autor) an expert in learning theory who adapts and extends the dialog guide.

Dialog guide (Dialog-Komponente) is a component of ISAC which combines dialog atoms to dialog strategies in order to adapt ISAC’s behaviour to different user models and learning situations.

Example (Beispiel) is a unit to be calculated and solved separated from others. In general, they are prepared by an author in an example collection. Within such a collection the example has a unique identifier.

Example browser (Beispiels-Browser) is an interactive representation of the example collection within the front-end.
Example collection (Beispielsammlung) contains examples, each of them consisting of formulas, of a hidden formalization and specification, and eventually of text and a figure.

Example profile (Beispiels-Profil) describes the structure of an example collection; this structure provides data for the dialog guide.

Explanation (Erklärung) is an optional addon (text, formulas, figures, movies, links, examples and any combination of these) to elements of the math knowledge base.

Formalization (Formalisierung) contains the formulas in a minimal structure necessary for automated solving the example (together with a specification).

Formula (Formel) consists of variables, constants and functions constants (for logical, algebraic etc. operators); all these parts, however are not yet structured as a (typed) term.

Front-end (Frontend) consists of the worksheet, the theory browser, the problem browser, the method browser and the example browser.

Interpreter (Interpreter) comprises the modules math engine, the calculator and dialog guide.

Isabelle is the name of one of the most successful interactive theorem provers; Isabelle provides the theories containing the deductive part of IS4C’s knowledge base.

Item (Item) of a model, which can be an input item (in the field ‘given’), a precondition (in the field ‘where’), an output item (in the field ‘find’) or a relation (in the field ‘relate’). ‘Given’, ‘find’and ‘relate’ may be input by the user, where ‘where’ is supplied by the system. An item consists of the item-description and the item-data.

Item-data (Item-Daten) are the formulas following the item-description.
**Item-description (Item-Beschreibung)** is an identifier heading each item in the fields 'given', 'find' and 'relate'. It indicates the kind of data to be input to the respective item by the users, serves typechecking of the data etc.

**KE-base (KE-Basis)** is the decorated math-knowledge plus the example collection.

**Kernel (SML-Kern)** comprises the interpreter and the knowledge base, all written in SML.

**Knowledge base** mathematics knowledge base

**Knowledge browser** is one of the theory brosers, problem browser, method browser.

**Learner (Lernender)** a user of \( \text{ISAC} \), who uses \( \text{ISAC} \) for learning and exercising, i.e. who calculates examples by use of the math knowledge base.

**Match (Matchen)**: a model matches a problem, or not. This kind of matching is different from the matching-algorithm of symbolic computation: it checks if all items are input, and evaluates the predicates in 'where'.

**Math engine (Mathematik-Maschine)** provides for all functions doing calculations: for applying tactics, for input formulas, for calculating resulting formulas, for proposing the next tactic, and for doing calculations automatically; it maintains a proofstate for each calculation.

**Mathematics author (Mathematik-Autor)** an expert in computer mathematics who adapts and extends the mathematics knowledge base.

**Mathematics knowledge base (Mathematische Wissensbasis)** is stored in three SML-datastructures, in an acyclic graph of theories, in a hierarchy of problems, and in a hierarchy of methods. It is extensible by math authors and can be both, read by learners and interpreted by \( \text{ISAC} \). See also decorated math-knowledge.
Method (Methode) contains a script describing the algorithm for calculating the result, and a guard structured like a problem in order to inhibit inappropriate application of the script.

Method browser (Methoden-Browser)

Model (Modell) consists of items. It matches a problem, or not.

Model context (Modell-Kontext) comprises all the information handled in the modeling phase as represented on the worksheet. This is at least one of the following items: a model (of a problem and/or of a method), a problem, a theory, a problem, a method.

Modeling phase (Modellierungs-Phase) is the initial phase in problem solving. In this phase either the system automatically transforms a formalization of an example into a model or the user inputs the items into the model.

Parsing (Parsen) is the process of transforming an ‘plain’ formula into a typed term. Parsing requires the specification of a theory containing information about infix position of operators etc.

Problem browser (Problem-Browser)

Problem (Problem) consists of patterns of items which can be matched with a model during the specification phase.

Proofstate (Beweiszustand) is given by an internal prooftree (partially represented on the worksheet) and a current formula.

Rewriting (Rewriting) transforms a formula into a new one by application of a theorem. ŁS4C provides conditional as well as ordered rewriting.

Script (Skript) describes the algorithm solving a particular problem; a script contains tactics, expressions for guiding the flow of evaluation, and eventually subproblems.
Selection-tool (Auswahls-Tool) displays the contents of either the example collection, or the dependency graph of theories, or the hierarchy of problems, or the hierarchy of methods; and it allows to select a respective item for detailed display.

SML-kernel kernel

Solving phase (Lösungs-Phase) is the final phase in problem solving, which generates the solution from the model and the specification; this phase may comprise all problem solving phases for one or more subproblems.

Specification (Spezifikation) relates a model to the knowledge base while determining a theory, a problem and a method.

Specification phase (Spezifikations-Phase) is the second phase in problem solving, which determines the theory, the problem and the method.

Step ((Rechen-) Schritt) propagates a calculation and involves both partners once, i.e. the learner and the dialog guide. A step is represented by one of the dialog atoms.

Tactic (Taktik) is applicable or not to the current formula within the current proofstate, and generates a new formula accordingly.

Term (Term) is an Isabelle term (simple typed lambda calculus) generated from a formula by parsing.

Theorem (Theorem) is a predicate proven true by Isabelle w.r.t. certain preconditions. Theorems are applied by rewriting.

Theory (Theorie) is the part of the math knowledge base which defines (function) constants and axioms. Within a theory usually the related theorems are being proven by Isabelle and stored.

Theory Browser (Theorie-Browser)
User (Benutzer) of ISAC may be one of the following: visitor, learner, math author, dialog author, course designer, or course admin.

Visitor (Besucher) a user of ISAC, which occasionally browses an ISAC-site, i.e. the knowledge base and the example collection.

Worksheet (Arbeitsblatt) contains the calculation of an example eventually leading to a result.
Appendix B

\textsf{\textit{ISAC}s tactics}

\texttt{Init.Proof.Hid} (dialogmode, formalization, specification) transfers the arguments to the math engine, the latter two in order to solve the example automatically. The tactic is not intended to be used by the student; it generates a proof tree with an empty model.

\texttt{Init.Proof} generates a proof tree with an empty model.

\texttt{Model_Problem problem} determines a problemtype (eventually found in the hierarchy) to be used for modeling.

\texttt{Add_Given, Add_Find, Add_Relation formula} inputs a formula to the respective field in a model (necessary as long as there is no facility for the user to input formula directly, and not only select the respective tactic plus formula from a list).

\texttt{Specify_Theory theory, Specify_Problem problem, Specify_Method method} specifies the respective element of the knowledgebase.

\texttt{Refine_Problem problem} searches for a matching problem in the hierarchy below 'problem'.

\texttt{Apply_Method method} finishes the model and specification phase and starts the solve phase.

\texttt{Free_Solve} initiates the solve phase without guidance by a method.

\texttt{Rewrite theorem} applies 'theorem' to the current formula and transforms it accordingly (if possible – otherwise error).

\texttt{Rewrite_Asm theorem} is the same tactic as 'Rewrite', but stores an eventual assumption of the theorem (instead of evaluating the assumption, i.e. the condition)

\texttt{Rewrite_Set ruleset} similar to 'Rewrite', but applies a whole set of theorems ('ruleset').
Rewrite_Inst (substitution, theorem), Rewrite_Set_Inst (substitution, ruleset) similar to the respective tactics, but substitute a constant (e.g. a bound variable) in 'theorem' before application.

Calculate operation calculates the result of numerals w.r.t. 'operation' (plus, minus, times, cancel, pow, sqrt) within the current formula.

Substitute substitution applies 'substitution' to the current formula and transforms it accordingly.

Take formula starts a new sequence of calculations on 'formula' within an already ongoing calculation.

Subproblem (theory, problem) initiates a subproblem within a calculation.

Function formula calls a function, where 'formula' contains the function name, e.g. 'Function (solve 1 + 2x + 3x² = 0 x)'. In this case the modelling and specification phases are suppressed by default, i.e. the solving phase of this subproblem starts immediately.

Split_And, Conclude_And, Split_Or, Conclude_Or, Begin_Trans, End_Trans, Begin_Sequ, End_Sequ, Split_Intersect, End_Intersect concern the construction of particular branches of the prooftree; usually suppressed by the dialog guide.

Check_elementwise assumptions w.r.t. the current formula which comprises elements in a list.

Or_to_List transforms a conjunction of equations to a list of equations (a questionable tactic in equation solving).

Check_postcond: check the current formula w.r.t. the postcondition on finishing the respective (sub)problem.

End_Proof finishes a proof and delivers a result only if 'Check_postcond' has been successful before.
Appendix C

Abbreviations

BG  browser generator
BR  browser, for theories, problems, methods or examples
DG  dialog guide
KB  mathematics knowledge base
ME  mathematics engine
Appendix D

Development environment

Development environments: There are two environments, (1) SML for the knowledge interpreter and the meth knowledge base, which both are based on Isabelle written in SML, and (2) Java for networking, the dialog and the front-end.

Dev D.0.1 SML version is Standard ML of New Jersey, Version 110.0.7, September 28, 2000

Dev D.0.2 SML library for HTML is smlnj-110.9.1/src/smlnj-lib/HTML

Dev D.0.3 Isabelle version is "Isabelle99: October 1999"

Dev D.0.4 Java version is "1.3.0_01", Java(TM) 2 Runtime Environment, Standard Edition (build 1.3.0_01), Java HotSpot(TM) Client VM (build 1.3.0_01, mixed mode)

Dev D.0.5 Java applets ???

Dev D.0.6 Development environments are linux version ... and Unix ...

Standards and components: A major part of the components concern networking because the knowledge interpreter and the knowledge base reside on a server for two reasons: (1) for efficiency reasons: both need major computer resources, and (2) for administrative reasons: ISAC will be used in courses initially.

Access to the server is established by standard internet protocols, formats and browsers as much as possible; this is not trivial because important standards are still under development.

Dev D.0.7 Standard for knowledge and example collections due to IEEE Learning Technology Standards Committee (LTSC), Learning Object Metadata, Working Draft Document 3 (approved 1999-11-27). TODO: OmDoc, LMML, IMS?
Dev D.0.8 *The middle ware layer is realized by Dinopolis* version …

Dev D.0.9 *The Browser is Mozilla* 1.2 - Released November 26, 2002. There should be as few additional software components within the browser as possible.

Dev D.0.10 *Plugins must not be required by the browser.* ???

Dev D.0.11 *The SML-kernel* comprising the knowledge interpreter and the knowledge base is under development. The interface to the SML-kernel is documented in ???isac/doc/interfaces/me-calc.dvi.

**Documentation and revision control:** All documentations and sources are under one revision control.

Dev D.0.12 *Doxygen* is used for all java sources.

Dev D.0.13 *cvs* contains all documentation, sml sources, the config file of doxygen and the java sources. There are the following directories:

```
cvs/doc             UseCases, URD-SRD, ADD-SDD, math-eng
                   etc
lib/               settings
    icons
    scripts
    tools/
src/ dox-config
    java/  … package 1/ … klasse 1
            … klasse …
            … package n/ … klasse 1
            … klasse …
    jars
sml
```

Dev D.0.14 *Directories* for all files of the project are the following:

```
bin
    cvs  … as above …
doxygen
html/    thy  html-representation of Isabelles theories
    pbl  XML-representation of ZSAC’s hierarchy of problems
    met  XML-representation of ZSAC’s list of methods
    exp  XML-representation of ZSAC’s example collection
```

147