ISAC
User Requirements Document

The *ISAC*-Team
see www.ist.tugraz.at/projects/isac

Revision

Contents

Part I

User Requirements Document

This document describes the user requirements for the ISAC-system.

By now the document captures at least those requirements in detail, which are a prerequisite for the *first phase of development, i.e. the development of the kernel of the math knowledgebase, of the indispensible tools for interaction on the knowledge, and of the tools for authoring the example collection.*

The design will try to meet these requirements while accepting the structure of the ISAC-math-engine, which is defined by mathematical reasons, and offers new functionality (calculations are done stepwise, the learner can input a formula or a tactic and receive feedback from the engine, the math-engine usually 'knows' the next step).

The User Requirements Document is structured along the different kinds of users envisaged. The Software Requirements Document, on the other hand, will be structured along the modules implementing the functionality. In order to establish comfortable tracing, the $m : n$ relation between user requirements and software requirements will be accurrately recorded in a double-linked way.

Terms marked by $\rightarrow$ are briefly described in appendix **??**.

Get this part from
`http://www.ist.tugraz.at/projects/isac/publ/urd.pdf`

Chapter 1

Kinds of *ISAC* users

There are several kinds of *ISAC* users which set the respective sections of requirements:

visitor (Besucher): occasionally drops into an *ISAC*-site and browses the respective math knowledge base and the example collection. May try to calculate some examples. Or some scientific content provider (wiki, scientific branding in an institution or an enterprise) includes occasional *ISAC*-services, in particular the possibility to interactively explore some specific calculations.

learner (Lernender): uses *ISAC* for learning and exercising, i.e. primarily calculates examples in the example collection by use of the math knowledge base. As a member of courses the learner is called a student.

math author (Mathematik-Autor): is an expert in computer mathematics who adapts and extends the mathematics knowledge base.

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

course designer (Kurs-Designer): adapts and extends the example collection which can be solved by a given math knowledge base, and adds explanations to items in the knowledge base. These tasks do not require special knowledge in computer mathematics.

course admin (Kurs-Administrator): is a person administering the use of *ISAC* for learning within a group of learners. This person is also regarded as a legal representative of the institution consuming *ISAC* services.

administrator (Administrator): this róle combines the system administrator installing the software, and the person who implements the overall design of an *ISAC*-site (introductory pages etc.). This person is also regarded as a legal representative of the institution hosting *ISAC* services.

These kinds of users are distinguished by the respective access-rights.

Table 1.1: Survey on the roles of ℐ𝒮𝒜𝒞 users

task	math author	dialog author	course designer	course admin
determine interaction in an expl.	by theories problems methods	by impl. of dialog patterns	by impl. of examples, of explanations to knowl.items	by setting content delivery, assessments

Chapter 2

General requirements

This section describes the requirements common to all users.

2.1 Users of *ISAC*

UR 2.1.1 ***ISAC is a multi-user system.***

UR 2.1.2 ***The users access ISAC via internet.***
The computing resources needed to run such a complex application exceed the computing-resources presently available to the average user. Moreover, organisation of centralised courses and curricula suggests separation of application and user-interface. An additional requirement is to keep expenses and effort for the average user at a minimum, in terms of computing power needed and installation effort. (Ideally, a standard web browser would suffice, but interaction required for the worksheet cannot be managed by a browser yet.)

UR 2.1.3 ***ISAC's data storage supports simultaneous access***
Data storage has to support locking and versioning.

UR 2.1.4 ***Users access ISAC with different roles***
Several possible roles when accessing *ISAC* dictate different rights and access to different modules of *ISAC* see p.5.

UR 2.1.5 ***Learners can be grouped into courses.***
There are groups of learners in order to support the adminstration of courses. The membership w.r.t. these groups determines the selections of examples in the example collecton (see UR.8.7.1), the selection of explanations in the knowledge (see UR.8.6.3)) and the initial setting of the dialog as captured in UR.4.4.11 and UR.4.2.8.

UR 2.1.6 ***One learner may be member in different groups***
One learner may be member in different groups but the settings for a session depend on exactly one group. This implies that multiple memberships have to be resolved at login time.

2.2 Calculations and Data Involved

UR 2.2.1 ***ISAC's math engine is given as an already-implemented module.***
ISACs mathematics engine(ME) is already given. Thus several requirements, looking like software requirements, are listed here. A calculation (see terms in the ISAC-project [iT02a]) undergoes three phases: the modelling phase, the specification phase and the solving phase, where the latter may contain these phases recursively (see the demonstration example in the usecases document [iT02b]).

Calculations are created interactively in steps. A step is initiated by the input of a learner: input of a tactic, of a formula or of a 'go-on' command. A tactic is applied to a formula and generates another (the derived) formula. The tactics specified so far are listed in appendix **??** An input formula during modelling phase completes a model, and during the solving phase an input formula is considered a derivation of the previous formula.

UR 2.2.2 ***ISAC stores 4 kinds of Mathematical Knowledge***

Theories

Problems

Methods

Examples

See [Neu01] for a detailed explanation.

UR 2.2.3 ***A Calculation undergoes 2 Phases***
A calculation undergoes two phases: the Specification Phase and the Solving Phase, where the latter may contain these phases recursively (see the demonstration example in appendix **??**).

UR 2.2.4 ***Specifying constructs a Model and a Specification.***
See UR.2.2.5 and UR.2.2.6.

UR 2.2.5 ***A Model consists of fields and items***
A Model consists of the fields 'given' containing the input-items, 'where' containing the pre-condition on the input-items, 'find' containing the output-items and 'relate' containing parts of the post-condition.

UR 2.2.6 ***A Specification consists of theory, Problem, Method***
Theory, Problem and Method provide for 3 pointers into the knowledge base (see Fig.**??** on p.**??**) referencing the knowledge required to solve the example specified by a model.

UR 2.2.7 ***Solutions are calculated interactively in steps.***
Stepwise calculation is done in the Solving Phase. A step is initiated by the input of a learner: input of a tactic, of a formula or of a request that ISAC take over the calculation. A tactic is applied to a formula and generates another (the derived) formula.

UR 2.2.8 ***In the Solving Phase, every formula is justified by a tactic.***

UR 2.2.9 ***ISAC uses tactics as listed in appendix ??***

UR 2.2.10 ***A calculation has a tree-like structure***

UR 2.2.11 ***A Tactic may contain Error Schemes***

UR 2.2.12 ***There are fill-in patterns for Tactics.*** In addition to a Tactic being applied manually by the user or automatically by ISAC, a Tactic can be presented with parts left blank to be filled in by the user.

UR 2.2.13 ***There are fill-in patterns for items of the Model.***

UR 2.2.14 ***ISAC guarantees correct results.***
ISAC's display of a Calculation reflects the state of the Math Engine. The Math Engine does not produce any inconsistent state of calculation. Thus everything displayed by ISAC (apart from lines being edited by the user right now) is proven to be consistent with the Specification and the Knowledge Base. User input will be accepted only if it can be proven to be correct by a check with the underlying Math Engine.

UR 2.2.15 ***The the mode of the ME*** is *one* of the following:

step is applicable, result guaranteed
step is applicable, result is not guaranteed
(because the learner has input some strange step previously)
the ME is helpless, i.e. it cannot propose a next step
the step is not applicable, plus an error message, why not applicable.

UR 2.2.16 ***Consistent "look & feel" for all users.***
As long as the deductive part is left to Isabelle, the same is with the theory browser. ISAC will develop the "look & feel" of its own, and thus violate uniformity w.r.t. the theory browser.

2.3 Miscellaneous

UR 2.3.1 ***ISAC supports internationalization.***
The mother language should be used by the student. The language of math formulas is independent from other languages. Thus the names used in the knowledge base can simply be changed for each language (scriptures with different structure like Japanese will not be considered here). But there are other texts delivered by the system, like error messages, or the fields 'given,...' in a model; these should be implemented multilinqual.

UR 2.3.2 *ISAC supports cross-linking calculations and knowledge* Calculations may contain links into the Knowledge Base, e.g. to the definitions or proofs of tactics applied in the course of calculation or into underlying theories. The Knowledge Base may contain links to examples illustrating theoretical concepts. More specific requirements are in sect.4.4.

UR 2.3.3 *ISAC is open to data exchange with other tools.* To facilitate interfacing with other tools, *ISAC*'s objects support being exported to and imported from open data formats. XML formats are preferred.

UR 2.3.4 *Several users can watch the progress of one calculation.* Several users may watch the progress of a calculation, but there is exactly one user controlling the calculation and taking actions. This can be useful for instruction situations, especially teleteaching.

UR 2.3.5 ***A calculation can be edited with other tools*** Editing in *ISAC* is limited by UR.2.3.6. For publishing purposes, *ISAC*'s calculations can be exported for editing with standard publishing tools.

Guarantee of correct results is given by *ISAC*s calc-state in the ME. This guarantee can only be given, when each editing on the worksheet is mirrored in the calc-tree (eventually cutting certain branches, if an intermediate step is redone). Sometimes the learner might want to edit a calulation (shorten the calculation by cutting intermediate steps, etc.) without affecting the calc-tree any more.

UR 2.3.6 *ISAC guarantees correct results.* In this case the worksheet reflects the calc-state, which shall be indicated on the screen and on the printout; in the latter case this indication should be certificate which cannot be manipulated.

Chapter 3

Requirements of the visitor

Each user approaching an *ISAC*-site first time wants to know about the purpose of *ISAC* and about the contents of the site.

3.1 Browsers: overview – detail:

The contents of the knowledgebase and of the example collection are expected to become very large. Thus there need to be facilities to switch from an overview to a detail and vice versa. These facilities should be handled similarly for all the browsers. Primarily, there is no direct interaction between the visitor and the ME. Information for the visitor should be provided in statical HTML-Pages.

Visitors should get an overview over *all* the knowledge available at an *ISAC*-site, and *all* the explanations, and *all* the examples. Therefore, access to additional information should be gained if available.

UR 3.1.1 ***There are 4 kinds of data to be browsed: theories, problems, methods, examples.***

UR 3.1.2 ***All 4 kinds of data need a table of contents of variable detail.***

UR 3.1.3 ***Browse through statical HTML-Pages*** without e.g. matching a model with a problem.

3.2 Do some examples:

Visitors should get an overview over *all* the features available at an *ISAC*-site, and the most exciting feature is stepwise interactive calculating and reasoning guided by the system. This should be demonstrated by some examples, which also the visitor can execute.

However, the possibility for any visitor to start an interactive calculation, i.e. a Worksheet from a browser would have resulted in a too complicated and

unstable system. A respective architecture see [Gri03]. Now a visitor has to choices:

UR 3.2.1 ***A visitor can view a calculation.***
These calculations are displayed a a whole, with the possibility of folding and unfolding the hierarchical structure. There is no possibility of interactive stepwise construction of the calculation as described in sect.4.

UR 3.2.2 ***A visitor can download and run an interactive Worksheet.***
After downloading the launched front-end (containing the Worksheet) there is a possibility to connect to a server running the ISAC back-end performing the calculation and user guidance.

Chapter 4

Requirements of the learner

[1]

4.1 Start a calculation

There are two different ways for users to approach ISAC's facilities for learning: (a) the user may browse the knowledge base, and eventually calculate an example (illustrating a definition, a problem, etc.) and (b) the user calculates examples from the example collection, and while asking for justifications of steps in the calculation enters knowledge browsers.

UR 4.1.1 ***A calculation can be started for a pre-defined example from an example collection***

UR 4.1.2 ***A calculation can be started for a pre-defined example from the Knowledge Base*** This is to illustrate knowledge from the Knowledge Base by typical examples stored with the knowledge.

UR 4.1.3 ***A calculation can be started from scratch.*** In this case, the calculation must start with specifying, and ISAC's support is limited as described in UR.4.2.2.

UR 4.1.4 ***A calculation can start with specifying or solving.*** In order to emphasize exercising specifying *or* solving, which are very different tasks.

UR 4.1.5 ***A calculation can be done like in an algebra system*** by simple input of a function call like `solve`, `simplify`, `integrate` or the like. In this case the specifying phase is skipped.

[1] Begin of copy from [Kre05] p.36-38

4.2 User Guidance

UR 4.2.1 *ISAC* ***provides User Guidance if the problem to be solved has been specified.*** A problem known to the system implies that the system knows a method to solve the problem. Thus *ISAC* can solve the problem automatically or propose the next step to be done.

UR 4.2.2 *ISAC* ***can assist in calculating examples unknown to the system*** Without knowing the Problem, *ISAC* cannot propose steps or solve the Problem automatically. Still, *ISAC* can apply Tactics chosen by the user to a formula. With a theory specified, *ISAC* can check formulas input by the user for correctness and consistency with previous steps in the calculation. At all times, *ISAC* ensures the calculation displayed is consistent and error-free as detailed in 2.3.6.

UR 4.2.3 *ISAC* ***can offer a list of actions meaningful in the current state*** of the calculation. 'Meaningful' is weaker than 'applicable', see UR.4.2.4

UR 4.2.4 *ISAC* ***can offer a list of actions applicable to the current state*** of the calculation.

UR 4.2.5 *ISAC* ***can propose the next action to be taken.***

UR 4.2.6 *ISAC* ***can do one or more steps automatically.***

The flow of interaction shall be adapted to the learner. The learner might be bored because the system offers too little challenge (by using less active dialog atoms – see UR.6.2.1, or by proceeding in too little steps of calculation) – or, in contrary, the user might be frustrated by too high challenges (in activity and/or stepwidth).

UR 4.2.7 ***The 'activity' of the dialog atoms adapts to the learner.***

UR 4.2.8 ***Varying stepwidth with tactics, rule sets and subproblems.***

The dialog regards presettings of the course admin (see Sect.7 below), when guiding the flow of interaction, and the dialog regards the ongoing interaction with the student.

UR 4.2.9 ***The system records examples done/not done by a learner*** regardless which course the learner is logged in.

UR 4.2.10 ***The dialog regards the performance*** in calculations done by the learner in the current session. The performance is measured by response times, errors, difficulty of examples done, requests into the knowledge base, active-passive behaviour. [2]

[2] The completion of this list is up to a future phase of development, and the evaluation of these data as well.

UR 4.2.11 ***The dialog regards the knowledge*** touched by the learner in the current session.

UR 4.2.12 ***The dialog regards the history*** of performance and knowledge touched in previous sessions.

UR 4.2.13 ***The amount of user guidance is configurable.*** The amount can be set by the user according to his preferences or by a course designer to match requirements of the course. For exam purposes, the amount of user guidance can be limited.

UR 4.2.14 ***The learner can override the Dialog behaviour chosen by the system.*** This includes the settings for dialog activity, stepwidth and display filtering rules. The conflicts between this UR and UR.4.2.13 need to be resolved by an indepth design lateron.

4.3 Help in the specify phase

According to UR.2.2.3 this phase comprises modeling (i.e. translating an example into a formal representation, the so-called Model) and specifying (i.e. relating the model to the mathematics knowledge available — that means, identify an appropriate theory, a problem and a method).

In the model phase the learner generally has to input the →items of a →model (see the example for reference in chapter **??** p.**??**): input and output items can be 'correct', 'incomplete', 'missing', 'superfluous' or have a 'syntax error'; preconditions can be 'correct' or 'false' (see UR.4.3.1) — these properties should be clearly indicated. This help is only possible, if the learner choose an example (with a hidden →formalization and →specification) from an →example collection, or if he had specified a problem.

If users want to calculate an example *unknown* to the system they have two choices:

1. specify a problem in order to get help from the system. Again, there are two possibilities:
   (a) first specify the appropriate problem (which contains the →item-descriptions), and then input the items. The item-descriptions help the user to provide the appropriate item-data.
   (b) first input the items (the item-descriptions are to be looked up in a specific theory), and then specify the problem — already assisted by the system (see UR.4.3.2 and UR.4.3.3).
2. do the calculation independently and without relying on assistance by the system (i.e. without specifying a problem, thus the system cannot check for completeness, and cannot assign a →method). In this case the input of items in the 'given'-field is sufficient; the users can calculate the example by applying tactics (after manual input).

UR 4.3.1 ***Visualization of the feedback on input to a model.*** The user gets immediate feedback on data entered into the Model by means of an item-status. The item-status can be: 'correct', 'incomplete', 'missing', 'superfluous', 'syntax error' or 'false'.

Retrieve a matching problem: Given a model initiating a calculation, or as a subproblem within a calculation, the learner has to determine a problem matching this model. This involves information retrieval from a large problem-hierarchy. There the learner may get lost, and thus he should get help: make the math-engine find a matching problem; visualize the problem found within the hierarchy.

In case of an exam, the user is forced to find the correct problem with no or limited help. Therefore it is nessesary to adjust the amount of help given by the browsers (e.g. only "match" or "nomatch" instead of a detailed listing of all conditions)

UR 4.3.2 ***ISAC can retrieve and match a model to a problem.***

UR 4.3.3 ***ISAC can help by automated refinement of a problem.***

4.4 Views into the knowledge

ISAC is a 'transparent system' by providing access to all the knowledge the system requires for (automated) problem solving. The knowledge is structured within a 3-dimension universe as stated in UR.2.2.2 on p.8. Needless to say that the knowledge has a highly complex structure, and that it is a demanding challenge to help the student not to get lost in this structure.

Principally, there are two ways to acces the knowledge, (1) access along the structure inherent to the respective knowledge, and (2) access from a concrete calculation, where the knowledge is required for solving the problem. These two kinds of access are separated below.

4.4.1 Surveys on related knowledge

As example collections (respective requirements see sect.8.2, sect.8.3 and sect.8.4) are very close to the other three parts of knowledge, theories, problems and methods, we include them in this subsection.

UR 4.4.1 ***Each element of the knowledge belongs to either theories, problems or a methods, or to the example collection.*** The elements of belonging to theories are

- theorems
- rule sets, i.e. sets of theorems or other rule sets, which are applied as long as they can be applied to a certain formula.

- rewrite orders which are required to terminate rule sets which contain theorems on commutativity etc. TODO
- computations involving sml-code (the only exception to rewriting). TODO

UR 4.4.2 ***Examples, theories, problems and methods all have a hierarchical structure*** and each element of the knowledge base has a unique position in this hierarchy.

UR 4.4.3 ***Each element of the knowledge base is displayed in the related browser.*** This requirement has to be met in particular, if such an element is referenced by a link from another browser: where-ever such a link is located, the element is displayed in the browser it belongs to, a theory-element in the theory-browser etc.

UR 4.4.4 ***Links can go from any element to any element.*** That means, an explanation for a problem can have a link to a method solving it, or to a theorem (in a theory) important for this problem; an explanation for a theorem can have a link to a problem, which uses the theorem in a certain way, etc.

UR 4.4.5 ***There are specific links which start an example.*** Such a link may be located in any part of the knowledge; if the link is activated, the respective example is displayed in the example-browser (UR.4.4.3) together with the examples location in the example hierarchy (UR.4.4.7) **and** a worksheet is opened for this example.

UR 4.4.6 ***Links outside ISAC's knowledge base open the standard browser.*** i.e. links within the knowledge base may point anywhere, and if the destination is outside ISAC's knowledge, it is displayed outside ISAC, too.

UR 4.4.7 ***An element is always displayed togehter with the respective location in the hierarchy.***. This requirement has to be met in particular, if the element is displayed following a link (indenpendently, from which browser).

UR 4.4.8 ***Each element has a certain position in the respective hierarchy.*** The hierarchy is the means for systematic search by the user. This does *not* mean, that this position can serve as a unique identification over time, see UR.8.6.4 !

UR 4.4.9 ***With each element in the knowledge base, explanations can be stored.*** Every element in the Knowledge Base can provide explanations illustrating its meaning, giving theoretical background information, referencing related topics or giving examples of use. See sect.7.1 p.27 and also UR.8.6.2.

4.4.2 Context-related views

The context is given by the state of a calculation (the so-called 'calc-state') on a Worksheet. UR.4.3.2 and UR.4.3.3 is one case, the other is displaying a method with the tactic marked which just has been applied in a calculation (this is well-known from debuggers). And there are other dynamic views into the knowledge base.

Context to all parts of the math-knowledge

UR 4.4.10 ***ISAC's use of math knowledge can be watched by the user.*** The user can look up the elements of the Knowledge Base currently used by ISAC to do a certain step in a calculation. This includes showing

- Tactics used in the calculation in their context in the Knowledge Base; particularly interesting are rewrite-tactics. The context to the calculation allows to display the rewrite, the assumptions generated, the rewrite-order used etc.
- Methods being applied to solve the current problem with indication of the tactic being currently applied, of the method's guard etc.
- Problems currently being solved in their context in the hierarchy of problems. This particularily helpful just before and after having ISAC refine a problem.

UR 4.4.11 ***On request, ISAC provides additional information on parts of the calculation.***
Additional information in a calculation can be provided at any time on request of the learner. This feature comprises more detailed views onto the calc-state, as well as explanations according to the following table:

`'detail'` on	element	yields
whole formula	//	intermediate steps
whole formula	//	tactic applied or applicable
whole formula	//	applicable tactics
whole formula	//	associated assumptions
whole formula	//	accumulated assumptions
formula	function-constant	definition in the theory
formula	floatingpoint-no	precision of this no
evaluated predicate	//	derivation
evaluated assumption	//	derivation
tactic	theorem	theorem instantiated
theorem instantiated	//	animation of matching
`'detail'` on	calchead	yields
specification	theory	file of the respective theory
specification	problem	model instant. this problem
specification	problem	inst. problem in the hierarchy
specification	method	guard instantiated by the model
specification	method	script in the hierarchy

UR 4.4.12 ***The context is a certain formula in a certain calculation*** as displayed on the respective 'worksheet'.

########################## WN060704 end update! ##########################[3]

[3] Begin of work extracted from [Kom07].

UR 4.4.13 ***A context is the current position on the worksheet last touched.*** The formula at this position is highlighted.

UR 4.4.14 ***Usually there is a context to any of the 3 parts of the math-knowledge for a position. See also UR.4.4.15 and UR.4.4.16***

UR 4.4.15 ***If there is no Worksheet open, then there is no context.***

UR 4.4.16 ***There may be NO context for an element of the math-knowledge.*** The reasons are specific to theories, problems and methods.

UR 4.4.17 ***The default for <Context on/off> is <on>*** - as soon as a Worksheet has been opened.

UR 4.4.18 ***If the KnowledgeBrowser is opened the first time and there is no context, it displays the element at the root of the respective hierarchy.***.

UR 4.4.19 ***Any formula in a calculation has a context (with different details to different parts of the math-knowledge).***

UR 4.4.20 ***The user can switch the context to a calculation on or off.***

UR 4.4.21 ***The context of an element is displayed in the respective browser-window.*** This is analoguous to UR.4.4.1

UR 4.4.22 ***A context is displayed as soon as*** the window is activated or the respective button <theory> <problem> <method> is pushed – if the user has NOT decided for .

UR 4.4.23 ***The contents of the hierarchy can be filtered (due to a UserModel).***

UR 4.4.24 ***The content of the hierarchy remains unchanged during a session (SR: thus the hierarchy is loaded*** **once** ***at the beginning of a session)***

Context to examples

UR 4.4.25 ***The context of an example is the respective worksheet.*** If <Context on> is set and a worksheet is brought to top, the respective example (i.e. the description of the example) is displayed in the example browser.

UR 4.4.26 ***A worksheet started by <New> has no context.*** It is handled according to UR.4.4.18

UR 4.4.27 ***<Context on/off> is the only choice for examples.*** This choice is always available. It is never changed by the system.

UR 4.4.28 ***A example may be displayed and*** **not** ***be allowed to calculate (according to the UserModel).***

Context to elements of theories

UR 4.4.29 ***The context comes exactly from the current position on the worksheet.***

UR 4.4.30 ***<Context on/off> is always available.***

UR 4.4.31 ***<To Worksheet> is available if:***

1. <Context on> is selected *and*
2. the worksheet on top is in the specify phase (and a CalcheadPanel is open)

UR 4.4.32 ***<To Worksheet> is* not *available:***

1. if <Context off> is selected, as described in UR.4.4.36
2. if no worksheet is open
3. if the worksheet on top is in the solve phase (and *no* CalcheadPanel is open)

Context to problems

UR 4.4.33 ***The context of a formula to the problems is given by the headline of the calc-head on the next higher level in the calculation.***

UR 4.4.34 ***The context of problem concerns the model of the current position and the modelpattern of a problem.*** Thus there is always a context, only exception is UR.4.4.18

UR 4.4.35 ***<Context on/off> is always available.***

UR 4.4.36 ***If <Context off> is selected, then <Refine> and <To Worksheet> are not available.***

UR 4.4.37 ***<Refine> is available if <Context on> is selected.***

UR 4.4.38 ***<Refine> is* not *available:***

1. if <Context off> is selected, as described in UR.4.4.36
2. if no worksheet is open

UR 4.4.39 ***<To Worksheet> is available if:***

1. <Context on> is selected *and*
2. the worksheet on top is in the specify phase (and a CalcheadPanel is open)

UR 4.4.40 ***<To Worksheet> is* not *available:***

1. if <Context off> is selected, as described in UR.4.4.36
2. if no worksheet is open
3. if the worksheet on top is in the solve phase (and *no* CalcheadPanel is open)

Context to methods

UR 4.4.41 ***The context of a formula to the methods is given by the headline of the calc-head on the next higher level in the calculation.***

UR 4.4.42 ***The context of a method concerns the guard and the script.*** The script in context show the tactic which calculated the currents position; the guard is matched with the model of the problem at the current position.

UR 4.4.43 ***<Context on/off> is always available.***

UR 4.4.44 ***<To Worksheet> is available if:***

1. <Context on> is selected *and*
2. the worksheet on top is in the specify phase (and a CalcheadPanel is open)

UR 4.4.45 ***<To Worksheet> is* not *available:***

1. if <Context off> is selected, as described in UR.4.4.36
2. if no worksheet is open
3. if the worksheet on top is in the solve phase (and *no* CalcheadPanel is open)

[4]

[4] End of work extracted from [Kom07].

Chapter 5

Requirements of the math author

All the authoring of math knowledge is still be done on the SML toplevel, i.e. immediately on the datastructures holding the knowledge. This part of the task is described in the 'interfaces for authors of math knowledge' [iT02c].

UR 5.0.46 ***Remote access to the sml-kernel*** is required. However, knowledge modfied need not yet imported to the production (i.e. tutoring) system.

Another result of authoring math knowledge, however, will be tools for the visitor and the learner to view the knowledge generated. This part of the task is described here.

UR 5.0.47 ***Automatic linking-tool*** supports setting links to other occurrences of an item: the system suggests links, the author accepts or rejects. Automated linking is done between the following items:

item	in ...	linked to occurrence in
predicate	theorem	definition in theory
	pre/postcondition	definition in theory
theorem	tactic	definition in theory
TODO [1]		

UR 5.0.48 ***Exchange data with other ISAC sites.***
TODO

UR 5.0.49 ***Exchange data with other knowledge bases.***
TODO

Copyright is important as ISAC's development will depend on the efforts of many, many authors.

UR 5.0.50 ***Copyright on any substantial item or part of math knowledge.*** Such items are

- a theory, and within theories
  - a rule set
- a problem
- method

See UR.8.1.3.

UR 5.0.51 ***An item can have more than author*** according to the ISAC-charta.

Authoring theories
The generation of the theory browser is already implemented by Isabelle. Within phase 1 of development, ISAC will take this component without any change.

Authoring problems
TODO

Authoring methods
TODO

Chapter 6

Requirements of the dialog author

The dialog author focuses on learning theory; administrative aspects of dialogs are discussed in chap.7 for the course admin.
[1]

6.1 User Profiling

UR 6.1.1 *ISAC* ***records examples done by the user*** *ISAC* keeps a per-user record of examples done and the user's performance in doing the example. The record is independent of the course the user has been logged into when doing the example.

UR 6.1.2 *ISAC* ***records items in the Knowledge Base viewed by the user.*** This information can be used to base the Dialog Guide's behaviour on information supposedly known to the user.

UR 6.1.3 *ISAC* ***records the user's success and errors.*** This extends to application of single Tactics, fill-in patters and error-patterns as well as whole examples or courses.

UR 6.1.4 *ISAC* ***records the user's time performance.*** In the future, assumptions about the user's familiarity with certain topics could be derived from these data.

UR 6.1.5 *ISAC* ***records the user's activity.*** In this context, activity means the ratio of steps done by the user to the steps the user had done by *ISAC*.

[1] Begin of copy from [Kre05] p.38-40

6.2 Flexible Dialog Behaviour

UR 6.2.1 ***ISAC's Dialog behaviour is constructed from Dialog Atoms*** We hope that it is possible to develop a language which allows to define Dialog Patterns as combinations of Dialog Atoms already implemented and Dialog Strategies sequencing these atoms. By means of such a language learning strategies could be described, and this description could be interpreted in reaction to a dynamic dialog state and according to a knowledge profile.

To do such 'dialog programming' is considered a comprehensive task, which in general exceeds the knowledge of a course designer or a course admin. On the other hand, a course admin can be expected to associate courses with dialog profiles, and a course designer can be expected to select Dialog Strategies within process of time in a course.

UR 6.2.2 ***The Dialog's behaviour can be configured.*** The Dialog's behaviour in terms of Dialog Atoms to be used can be preset by the course designer and the user to match the requirements of the situation in style and complexity. The probability of asking the user a question is an example of such a preset.

UR 6.2.3 ***The number of calculation steps taken at a time can be configured.*** This extends to taking several steps at a time and doing whole rulesets or subproblems in one step.

UR 6.2.4 ***The amount of information displayed can be configured.*** As with the Dialog's behaviour, the amount of information can be preset to meet the requirements of the learning situation. This could mean displaying or not displaying the Tactics used in the calculation or hiding specific steps in the calculation considered too complex or too trivial.

6.3 Adaptation to Individual Users

UR 6.3.1 ***The activity of the Dialog Guide adapts to the learner.*** In a learning situation, active participation of the student is one key to acquiring and consolidating knowledge and skills. The Dialog Guide will support this by asking questions or letting the user decide what to do in the next step. On the other hand, with growing expertise, once thrilling questions become trivial and boring. The Dialog Guide adapts his strategy by dropping challenges the user has alredy mastered a few times. Whenever possible, the dialog adapts its behaviour, i.e. the choice of Dialog Atoms, to challenge the user without frustrating him. The choice is based on the present and past actions of the individual user.

UR 6.3.2 ***The Dialog adapts the amount of information displayed to the learning situation.*** As with the Dialog's behaviour, the amount of information displayed adapts to the requirements of the current situation.

UR 6.3.3 ***The Dialog regards the performance of the user.*** The performance is measured by response times, errors, difficulty of examples done, requests into the Knowledge Base and Dialog Activity.

UR 6.3.4 ***The Dialog regards the knowledge touched by the user in the current session.***

UR 6.3.5 ***The Dialog regards the history of the user.*** In addition to the of performance and knowledge touched in the current session, the history of the user's previous sessions is regarded as well.
[2]

[2]End of copy from [Kre05] p.38-40.

Chapter 7

Requirements of the course admin

Authoring in ISAC comprises various tasks: authoring mathematics knowledge and authoring the dialog have been described in Chapt.5 and Chapt.6; these both tasks require more special knowledge than the others. Chapt..8 describes a kind of knowledge ('explanations') which may change from one course to the other, and wich does not require special knowledge on computer mathematics or dialog design.

The latter part of authoring is done by the course designer preparing as course in advance (see Chapt.8). Some of the knowledge prepared in advance underlies time constraints, which are managed by the course admin. The requirements of the course admin are separated in this section (despite the fact that the course designer and the course admin are one and the same person, the lecturer or teacher of the course).

7.1 Groups of learners

There are groups of learners in order to support the adminstration of courses. The membership w.r.t. these groups determines the selections of examples in the example collecton (see UR.8.7.1), the selection of explanations in the knowledge (see UR.8.6.3)) and the initial setting of the dialog as captured in UR.6.2.4 and UR.4.2.8.

UR 7.1.1 ***Learners can be grouped into courses.*** There are groups of learners in order to support the adminstration of courses. The membership in these groups determines the selections of examples in the example collecton (see UR.8.7.1), the selection of explanations in the knowledge (see UR.8.6.3)) and the initial setting of the dialog as captured in UR.4.4.11 and UR.4.2.8.

UR 7.1.2 ***One learner may belong to different groups*** but only to *one* group within a session. For instance, a student of TUG can be member of

Analysis semester 1, of Signal Processing semester 5, Analysis semester 2.

UR 7.1.3 ***Administrative information for groups*** is part of any login, and should be available to the course admin for review at any time during a session. [1]
[2]

7.2 Restrictions

Restrictions will apply when using ISAC as a learning or tutoring tool, especially during exams. When being used as a calculation tool, restrictions are relaxed to gain access to the full power of the Math Engine.

UR 7.2.1 ***Restrictions are individual to a user or group.***

UR 7.2.2 ***Groups of examples may be invisible.*** Parts of the example collection may be inaccessible for specific groups of learners.

UR 7.2.3 ***Access to items in the Knowledge Base can be restricted.***

UR 7.2.4 ***The amount of User Guidance may be restricted.***

UR 7.2.5 ***The use of Dialog Patterns may be restricted.***

UR 7.2.6 ***Restrictions may be overridden.*** Depending on the settings provided by the course-designer and the user's access rights, some restriction may be overridden by the user. Overriding e.g. allows to look at explanations and examples for other courses.

UR 7.2.7 ***Restrictions may apply within time limits.*** Some restrictions may apply only within certain time limits, e.g. during an exam or during class. Time limits can be given by start and finish or by their duration.

UR 7.2.8 ***Restrictions may depend on the user's learning progress.*** Some restrictions may apply until certain examples have been solved or the user has mastered certain aspects of knowledge.
[3]

Time limits for delivering course material are a general requirement for educational systems, and it applies for example collections as well. This requirement provides the course admin to distribute workload over time and to focus the attention of the learners. Sometimes it may be desirable to have some examples finished within a certain time limit. Also examples (for instance prepared for an examination) may be invisible for learners.

[1] End of copy from [Kre05] p.38-40
[2] Begin of copy from [Kre05] p.41
[3] End of copy from [Kre05] p.41

UR 7.2.9 ***There are 2 kinds of time limits:*** (1) given by start and finish, and (2) given by a duration (where start and finish are recorded with the user).

UR 7.2.10 ***Groups of examples may be locked*** for groups of learners for certain time limits. (SR: attention with links from the KB !)

UR 7.2.11 ***Groups of examples may be invisible*** for all users except the example author for certain time limits. (SR: examples have an author !)

UR 7.2.12 ***Restricted help for the learners during a exam*** i.e. particular links into the knowledge base are restricted for a particular group for a limited time.

UR 7.2.13 ***Access-rights during an exam-session*** are particularily restricted (a user might get unwanted information by opening another session as a member of another group).

UR 7.2.14 ***Restrictions may be overridden or not*** depending on the setting by the course-designer. Overriding e.g. allows to look at explanations and examples for other courses.

7.3 Survey on the progress of the learners

i.e. of the students in a course. Such surveys are indispensable requirements for a course admin. Thus all the surveys below are related to a specific course.

UR 7.3.1 ***There is a statistics-tool*** for the learners progress

UR 7.3.2 ***Statistics are drawn from*** the following data: TODO
Examples for statistics:

- Which examples have been done by whom with which performance
- Sorts w.r.t. the examples: frequency of touched, not touched, solved, unsolved (, evaluation of performance, as soon as evaluation functions have been designed) over all students of a course
- Sort w.r.t. the students: likewise over all examples in specified groups
- Which lookups into the knowledge have been done
- Sorts w.r.t. the destination: likewise over students
- Sorts w.r.t. the students: likewise over the knowledge.

For preliminarily checking the usage of the system (as well as for preliminary monitoring of the system – see sect.9.1) the following requirement is stated, until the respective requirements analyses are done.

UR 7.3.3 ***A text-file records the usage of the system.*** The records contain the following basic data for each user logging in during the life-time of the session-manager.

- at login
  - user name (if there are anonymous users, record the mail address)
  - time of login
- for each example called (no recording of access to the knowledge base !)
  - example-ID
  - time of starting the example (i.e. the respective worksheet)
  - time of removing the example (i.e. the respective worksheet)
- at logout
  - user name (if there are anonymous users, record the mail address)
  - number of examples
  - time of login and logout

Special requirements may be raised by field studies on learning mathematics. *ISAC* shall be open for such special research. I.e. queries over examples, lookups, students *and courses*. Anonymous evaluation has to be regarded.

UR 7.3.4 ***Anonymous evaluation of statistics.***

7.4 Written examinations

can be done due to requirements stated elsewhere: hidden examples UR.7.2.11, restricted access to knowledge UR.7.2.12, and adapted dialog UR.6.2.1.

UR 7.4.1 ***The course admin can force a student to exam-mode:*** That includes, that there is no way for this user, to open a session with a user-model which would undermine the exam-mode. A student, in spite of being a member of a group (or several groups), might be examined separated from other members of the group (in case of illness at the time of the group's exam etc.)

UR 7.4.2 ***The course admin can force a group of users to exam-mode:*** Sideconditions as in UR.7.4.1.

UR 7.4.3 ***The exam-mode is described as follows:*** Preliminarily, the buttons <next> and <auto> are not available, neither in the specification phase nor in the solve phase.

UR 7.4.4 ***The exam-mode can be quitted in several ways:***

1. by the learner any time

2. predifined by settings (given by the course admin) the system closes the exam-mode due to the time constraints set (and kindly warning the learner in time)

3. by the course admin at any time (in case of cheating, etc)

UR 7.4.5 ***The couse admin gets several views on the results of an exam:*** Required are views to individual results ("Prüfungseinsicht"), to groups, to several groups for comparison, etc. These views can be printed out ("Prüfungsliste" only with "Matrikelnummer" for public announcement, etc).

Chapter 8

Requirements of the course designer

8.1 General

A course desinger prepares the learning materials and exercises for a course according to the goals of that course, or according to the learners' level. If ISAC is being used for a course, specific explanations may be added to the knowledge [1] and examples will be prepared within the example collection.

UR 8.1.1 ***Explanations from other ISAC-sites can be imported.***

UR 8.1.2 ***Examples from other ISAC-sites can be imported.***

These requirements are important, because ISAC will be an open system in all respects. In particular, each ISAC-site shall go public with the knowledge and the examples available at this site — for information of individual learners as well as a starting point for exchange of course-content between educational institutions.

UR 8.1.3 ***Copyright for the author of explanations.*** See UR.5.0.50.

UR 8.1.4 ***Copyright for the author of examples.***

8.2 Appearance of example collections

One application of ISAC will be to mechanize tutoring on existing example collections, as found for instance in traditional textbooks. Thus the example collection must copy the structure already given (the enumerations, page breaks, assembly on a page etc.) in order to allow the learner to find a particular

[1] It is the task of a *mathematics author*, however, to edit the *contents* of a math knowledgebase !

example (e.g. given as homework). Traditional textbooks use arbitrary labels for their chapters and sections, the levels are nested arbitrarily deep, and there are arbitrary labels for the examples.

For copy-right reasons it also may happen, that the example itself (i.e. text, formulas, figures) is not displayed, only the respective label. In this case the label should be located exactly at the same position on a virtual 'page' on the screen as the original position in the page of the textbook.

UR 8.2.1 ***The labels of examples are defined by the author***

UR 8.2.2 ***The layout can model textsbooks.*** i.e. the structure of the underlying system must be strong enough to model the structure of textbooks (but also exceeds this structure, e.g. with links)

8.3 The structure of example collections

is a hierarchy of groups of examples. A group of examples consists of the part visible to the learner (which may be copied from a textbook) and meta data for ISAC to suggest examples of a level appropriate to an individual learner. There are already requirements concerning examples, UR.7.2.10 and UR.7.2.11, which apply to groups of examples for convenience.

UR 8.3.1 ***There is only one example collection in the system*** in analogy to *one* problem-hierarchy and *one* method hierarchy, see UR.4.4.2. This collection, however, even may comprise several textbooks.

UR 8.3.2 ***There are groups of examples*** with common properties, see UR.7.2.10 and UR.7.2.11.

UR 8.3.3 ***A group and/or an example are weighted*** w.r.t. properties to be defined in a later phase of development (at least the properties 'difficulty' and 'length').

UR 8.3.4 ***A tool for selecting examples*** w.r.t. the weights (and the user-model etc.) is required. There may be also administrative concerns:

- A limit of the number of solved examples may be defined for a group; if the limit is touched, this group has been mastered successfully by a student within a certain course. An evaluation-function for the performance will be advantageous here in the future.
- There might be obligatory examples in a group; i.e. such an example must be solved by a student in order to have the group mastered successfully.

8.4 Edit examples in the example collection

An example can be described by verbal text, by formulas, and by figures (and eventual by movies). Additionally, each example contains data hidden from the visitor and the learner.

UR 8.4.1 ***Explanatory data can be embedded into examples*** An example can be described by verbal text, by formulas, by figures and possibly by movies. Additionally, each example contains data hidden from the visitor and the learner.

UR 8.4.2 ***A formalization is a list of formulas.***

UR 8.4.3 ***A specification is a triple*** of three pointers to a theory, a problem and a method respectively.

UR 8.4.4 ***Each example is combined with a list of pairs of*** a formalization and a specification respectively. The pairs of Formalization and Specification are used for user guidance while specifying a calculation and are remain to users in a learning situation.

UR 8.4.5 ***Formalizations and specifications are hidden*** from the learner.

UR 8.4.6 ***An example may contain Error Schemes*** An Error Scheme modelling typical errors and providing explanations and specific user guidance for resolving the errors may be stored with an example.An Error Scheme may be paired with an explanation (see UR.8.6.1. . . UR.8.6.2 below).

8.5 Checks for example collections

For each example or for groups of examples is necessary to run particular checks before delivery to the learners.

8.5.1 Check of format

Like the other items of ISAC's knowledge base, examples are stored in XML-format. The data in the XML-files are filtered and converted to HTML for presentation to the user.

UR 8.5.1 ***A tool for checks on the conversion from XML to HTML*** for groups of examples is required, and also for particular examples found to be buggy.

8.5.2 Check of solvability

The course designer addresses items of the mathematics knowledge (theories, problems, methods) prepared by math authors. It may be, that particular examples cannot be solved automatically by means of the knowledge addressed.

UR 8.5.2 ***A tool for batch-processing examples*** is required.

UR 8.5.3 ***Errors for batch-processing*** are: TODO

8.6 Edit explanations in the knowledge base

As already mentioned, explanations are multi-media add-ons to the math-contents of the knowledge base. The latter is generated automatically from the SML-representation in the SML-kernel (where the knowledge in the SML-kernel is due to authoring by math-authors, see chap.5.).

UR 8.6.1 ***An explanation may consist of*** text, formulas, figures, movies, and links into ISAC's knowledge base itself, into the web, to an example in the resident site. Most HTML-editors, as presently available, meet the requirements for editing explanations.

UR 8.6.2 ***Each item in a knowledge base can have an explanation.*** This should also be possible within formulas, e.g. an explanation to a predicate in a 'where'-field of a model etc. . See also UR.4.4.9.

UR 8.6.3 ***Explanations are course-specific.*** Each course might have different explanations according to the contents of the course, according to the specific example collection, and according to the learners' level.

UR 8.6.4 ***The location of an item in the hierarchy is not persistent over time.*** For instance, a problem like 'linear equation' (together with its children) can be shifted around in the hierarchy of equations. As the knowledge comes from 2 sources, (1) exported from the SML-kernel after authoring by math-authors and (2) manually addition of explanations by course-designers, the identification of a primary key requires particular attention.

UR 8.6.5 ***There are tools to shift hundreds of explanations together with their items.*** This may happen with the above example for 'linear equation' together with its children, and with explanations for several user groups.

UR 8.6.6 ***There are tools for searching the knowledge base.*** Not only the learner (see sect.4.4) but also the author depends on requirements for searching the knowledge; these requirements are different.

8.7 A knowledge profile

for each course results from the explanations following the requirements from above, plus from the detail knowledge is used in, from error schemes and from fill-in patterns for theorems.

UR 8.7.1 ***An example group and/or an example specifies the details*** as defined in UR.4.4.11.

UR 8.7.2 ***There are error schemes,*** eventually several for one theorem in a particular method, or for specific tactics in a particular method. An error scheme is paired with an explanation (UR.8.6.1. . . UR.8.6.2).

An Error Scheme modelling typical errors and providing explanations and specific User Guidance for resolving the errors

UR 8.7.3 ***There are fill-in patterns*** for theorems, eventually several for one theorem in a particular method. Such patterns are used for the dialog atoms UR.6.2.1, e.g. **??** or **??**.

In addition to a tactic being applied manually by the user or automatically by *ISAC*, a tactic can be presented with parts left blank to be filled in by the user.

8.8 A dialog profile

can be preset according to the students' level. These predefined setting can be overridden by the students in most cases, but not in all cases. The dialog profile will be more elaborated as soon the 'dialog programming' and the user model habe been clarified in a future development phase.

UR 8.8.1 ***A dialog mode restricts the dialog atoms*** (UR.6.2.1) to be used by the learner within certain time limits (e.g. during the time of a written examination).

Chapter 9

Requirements of the administrator

The administrator has to install the system and to monitor it. Moreover his duty is to implement the overall design of an *ISAC*-site. This includes introductory pages as well as an basic overall design (Corporate Design, links to "home"...)

9.1 Install and monitor the system

TODO
preliminary requirements:

UR 9.1.1 ***A text-file records the data exchange with the sml-kernel.*** This file serves as a primitive way to check if the sml-kernel is alive as well as for debugging.

UR 9.1.2 ***The system can be asked for the number of users logged in.***

UR 9.1.3 ***The system can be asked for the number of calculations under construction.***

9.2 Customize the appearance in the web

TODO
preliminary requirement:

UR 9.2.1 ***Basic parameters to adapt the HTML-outtput to a corporate design (e.g. colors, font, . . .)***

.

Bibliography

[Gri03] Andreas Griesmayer. Architecture and Knowledge-Represenation of the Web-based Math-Learning-System *ISAC*. Master's thesis, University of Technology, Institute for Softwaretechnology, Graz, Austria, Oct 2003.
http://www.ist.tugraz.at/projects/isac/publ/da-griesmayer.pdf.

[iT02a] *ISAC* Team. *ISAC* appendices to the analysis and design documents. Technical report, Institute for Softwaretechnology, University of Technology, 2002.
http://www.ist.tugraz.at/projects/isac/publ/appendices.ps.gz.

[iT02b] *ISAC* Team. *ISAC* use cases. Technical report, Institute for Softwaretechnology, University of Technology, 2002.
http://www.ist.tugraz.at/projects/isac/publ/use.ps.gz.

[iT02c] *ISAC* Team. *ISAC*, interfaces for developers of math knowledge and tools for experiments in symbolic computation. Technical report, IICM, Institute for Softwaretechnology, University of Technology, 2002.
http://www.ist.tugraz.at/projects/isac/publ/mat-eng.pdf.

[Kom07] Georg Kompacher. Context-based access to *ISAC*s knowledge base. Software-Projekt und Bakk.-Arbeit B, Oct 2007. Graz University of Technology, Institute for Softwaretechnology.

[Kre05] Alan Krempler. Architectural design for integrating an interactive dialogguide into a mathematical tutoring system. Master's thesis, University of Technology, Institute for Softwaretechnology, Graz, Austria, March 2005.
http://www.ist.tugraz.at/projects/isac/publ/da-krempler.pdf.

[Neu01] Walther A. Neuper. *Reactive User-Guidance by an Autonomous Engine Doing High-School Math.* PhD thesis, IICM - Inst. f. Softwaretechnology, Technical University, A-8010 Graz, 2001.
http://www.ist.tugraz.at/projects/isac/publ/wn-diss.ps.gz.