Eine neue Generation von Mathematik-Lernystemen Studiengangs-übergreifend entwickeln?

Walther Neuper

Institute for Software Technology, TU Graz


Slides:
http://www.ist.tugraz.at/projects/isac/publ/present.pdf
Outline

1. The ISAC Prototyping Project
   - The History
   - The Architecture
   - The New Generation + Demo

2. Preview: Learning Math with the New Generation

3. Challenges from ongoing Development in ISAC
   - Development by Students
   - Technologies used
   - Ideas for Subprojects
   - Project Engineering
1. **The ISAC Prototyping Project**
   - The History
   - The Architecture
   - The New Generation + Demo

2. **Preview: Learning Math with the New Generation**

3. **Challenges from ongoing Development in ISAC**
   - Development by Students
   - Technologies used
   - Ideas for Subprojects
   - Project Engineering
ISAC is an academic open source project in interdisciplinary cooperation largely promoted by student projects.

- 2000 – 2003 ISAC’s math-engine developed upon TP Isabelle
- 2003 – . . . ISAC front-end developed at TUG (IST, IICM) + cooperation with Cognitive Science (MEI:CogSci) Vienna
- 2005 – 2008 field tests at Austrian HTL successful
- 2011 Isabelle parallelized, ISAC broke partially
- 2012 field test in Signal Processing at TUG not successful: ISAC’s math-engine too complicated in programming!
- 2012 Participation in FP7 Proposal Deduction
- 2013 – . . . improve ISAC’s math-engine at RISC Linz

+ 2-3 years required to make the prototype practically useful.

---

1 Public repository: https://intra.ist.tugraz.at/hg/isac
History of ISAC

ISAC is an academic open source project in interdisciplinary cooperation largely promoted by student projects.

- 2000 – 2003 ISAC’s math-engine developed upon TP Isabelle
- 2003 – … ISAC front-end developed at TUG (IST, IICM) + cooperation with Cognitive Science (MEI:CogSci) Vienna
- 2005 – 2008 field tests at Austrian HTL successful
- 2011 Isabelle parallelized, ISAC broke partially
- 2012 field test in Signal Processing at TUG not successful: ISAC’s math-engine too complicated in programming!
- 2012 Participation in FP7 Proposal Deduction
- 2013 – … improve ISAC’s math-engine at RISC Linz

+ 2-3 years required to make the prototype practically useful.

\(^1\)Public repository: https://intra.ist.tugraz.at/hg/isac
ISAC is an academic open source project in interdisciplinary cooperation largely promoted by student projects.

- 2000 – 2003 ISAC’s math-engine developed upon TP Isabelle
- 2003 – . . . ISAC front-end developed at TUG (IST, IICM) + cooperation with Cognitive Science (MEI:CogSci) Vienna
- 2005 – 2008 field tests at Austrian HTL successful
- 2011 Isabelle parallelized, ISAC broke partially
- 2012 field test in Signal Processing at TUG not successful: ISAC’s math-engine too complicated in programming!
- 2012 Participation in FP7 Proposal Deducation
- 2013 – . . . improve ISAC’s math-engine at RISC Linz

+ 2-3 years required to make the prototype practically useful.

\(^1\)Public repository: https://intra.ist.tugraz.at/hg/isac
ISAC is an academic open source project in interdisciplinary cooperation largely promoted by student projects.

- 2000 – 2003 ISAC’s math-engine developed upon TP Isabelle
- 2003 – . . . ISAC front-end developed at TUG (IST, IICM) + cooperation with Cognitive Science (MEI:CogSci) Vienna
- 2005 – 2008 field tests at Austrian HTL successful
- 2011 Isabelle parallelized, ISAC broke partially
- 2012 field test in Signal Processing at TUG not successful: ISAC’s math-engine too complicated in programming!
- 2012 Participation in FP7 Proposal Deduction
- 2013 – . . . improve ISAC’s math-engine at RISC Linz

+ 2-3 years required to make the prototype practically useful.

\footnote{Public repository: https://intra.ist.tugraz.at/hg/isac}
History of ISAC

ISAC is an academic open source\(^1\) project in interdisciplinary cooperation largely promoted by student projects.

- 2000 – 2003 ISAC’s math-engine developed upon TP Isabelle
- 2003 – . . . ISAC front-end developed at TUG (IST, IICM) + cooperation with Cognitive Science (MEI:CogSci) Vienna
- 2005 – 2008 field tests at Austrian HTL successful
- 2011 Isabelle parallelized, ISAC broke partially
- 2012 field test in Signal Processing at TUG not successful: ISAC’s math-engine too complicated in programming!
- 2012 Participation in FP7 Proposal Deduction
- 2013 – . . . improve ISAC’s math-engine at RISC Linz

+ 2-3 years required to make the prototype practically useful.

\(^1\text{Public repository: https://intra.ist.tugraz.at/hg/isac}\)
History of **ISAC**

**ISAC** is an academic open source project in interdisciplinary cooperation largely promoted by student projects.

- 2000 – 2003 **ISAC**’s math-engine developed upon TP Isabelle
- 2003 – . . . **ISAC** front-end developed at TUG (IST, IICM) + cooperation with Cognitive Science (MEI:CogSci) Vienna
- 2005 – 2008 field tests at Austrian HTL *successful*
- 2011 Isabelle parallelized, **ISAC** broke partially
- 2012 field test in Signal Processing at TUG *not successful*: **ISAC**’s math-engine too complicated in programming!
- 2012 Participation in FP7 Proposal *Deducation*
- 2013 – . . . improve **ISAC**’s math-engine at RISC Linz

+ 2-3 years required to make the prototype *practically useful.*

---

1 Public repository: [https://intra.ist.tugraz.at/hg/isac](https://intra.ist.tugraz.at/hg/isac)
History of ISAC

ISAC is an academic open source\(^1\) project in interdisciplinary cooperation largely promoted by student projects.

- 2000 – 2003 ISAC’s math-engine developed upon TP Isabelle
- 2003 – . . . ISAC front-end developed at TUG (IST, IICM) + cooperation with Cognitive Science (MEI:CogSci) Vienna
- 2005 – 2008 field tests at Austrian HTL successful
- 2011 Isabelle parallelized, ISAC broke partially
  - 2012 field test in Signal Processing at TUG not successful: ISAC’s math-engine too complicated in programming!
  - 2012 Participation in FP7 Proposal Deduction
  - 2013 – . . . improve ISAC’s math-engine at RISC Linz

+ 2-3 years required to make the prototype practically useful.

\(^1\)Public repository: https://intra.ist.tugraz.at/hg/isac
History of ISAC

ISAC is an academic open source \(^1\) project in interdisciplinary cooperation largely promoted by student projects.

- 2000 – 2003 ISAC’s math-engine developed upon TP Isabelle
- 2003 – . . . ISAC front-end developed at TUG (IST, IICM) + cooperation with Cognitive Science (MEI:CogSci) Vienna
- 2005 – 2008 field tests at Austrian HTL successful
- 2011 Isabelle parallelized, ISAC broke partially
- 2012 field test in Signal Processing at TUG not successful: ISAC’s math-engine too complicated in programming!
- 2012 Participation in FP7 Proposal Deduction
- 2013 – . . . improve ISAC’s math-engine at RISC Linz

+ 2-3 years required to make the prototype practically useful.

\(^1\) Public repository: https://intra.ist.tugraz.at/hg/isac
History of ISAC

ISAC is an academic open source project in interdisciplinary cooperation largely promoted by student projects.

- 2000 – 2003 ISAC’s math-engine developed upon TP Isabelle
- 2003 – . . . ISAC front-end developed at TUG (IST, IICM) + cooperation with Cognitive Science (MEI:CogSci) Vienna
- 2005 – 2008 field tests at Austrian HTL successful
- 2011 Isabelle parallelized, ISAC broke partially
- 2012 field test in Signal Processing at TUG not successful: ISAC’s math-engine too complicated in programming!
- 2012 Participation in FP7 Proposal Deduction
- 2013 – . . . improve ISAC’s math-engine at RISC Linz

+ 2-3 years required to make the prototype practically useful.

1Public repository: https://intra.ist.tugraz.at/hg/isac
**History of ISAC**

*ISAC* is an academic open source\(^1\) project in interdisciplinary cooperation largely promoted by student projects.

- 2000 – 2003 *ISAC*’s math-engine developed upon TP Isabelle
- 2005 – 2008 field tests at Austrian HTL *successful*
- 2011 Isabelle parallelized, *ISAC* broke partially
- 2012 field test in Signal Processing at TUG *not successful*: *ISAC*’s math-engine too complicated in programming!
- 2012 Participation in FP7 Proposal *Deducation*
- 2013 – . . . improve *ISAC*’s math-engine at RISC Linz

+ 2-3 years required to make the prototype *practically useful*.

---

\(^1\)Public repository: [https://intra.ist.tugraz.at/hg/isac](https://intra.ist.tugraz.at/hg/isac)
History of ISAC

ISAC is an academic open source project in interdisciplinary cooperation largely promoted by student projects.

- 2000 – 2003 ISAC’s math-engine developed upon TP Isabelle
- 2003 – . . . ISAC front-end developed at TUG (IST, IICM) + cooperation with Cognitive Science (MEI:CogSci) Vienna
- 2005 – 2008 field tests at Austrian HTL successful
- 2011 Isabelle parallelized, ISAC broke partially
- 2012 field test in Signal Processing at TUG not successful: ISAC’s math-engine too complicated in programming!
- 2012 Participation in FP7 Proposal Deduction
- 2013 – . . . improve ISAC’s math-engine at RISC Linz

+ 2-3 years required to make the prototype practically useful.

Public repository: https://intra.ist.tugraz.at/hg/isac
Architecture for Prototyping

Walther Neuper
Architecture for Prototyping

Isaac Math-Engine

Theorem Prover (TP) “Isabelle”

Engineer
Architecture for Prototyping

Theorem Prover (TP) "Isabelle"

Java

SML

Isac Math-Engine

Programs Knowledge

Engineer
Architecture for Prototyping

Theorem Prover (TP) “Isabelle”

Isac Math-Engine

Isac Server

Dialogue

Java

SML

Engineer

Isac Math-Engine

Programs

Knowledge
Architecture for Prototyping

- Students
- GUI 1
- GUI 2
- GUI n
- Engineer
- Theorem Prover (TP) "Isabelle"
- Isac Server
- Isac Math-Engine
- Programs Knowledge
- Java
- SML
Architecture for Prototyping

Students

GUI 1

GUI 2

GUI n

Java

Isac Server

Dialogue

Knowledge

SML

Isac Math-Engine

Programs

Knowledge

Theorem Prover (TP) “Isabelle”

Engineer
Architecture for Prototyping

Walther Neuper

ISAC Prototype

History
Architecture
New Generation
Applications?

Development
Supervision
Technologies
Projects
Engineering

Theorem Prover (TP)
"Isabelle"
core 1
SML
Java
Isac
Math-Engine
Engineer
core 2
core n
Students
GUI n Dialogue
Isac
Server
Knowledge
GUI 2
GUI 1
Knowledge Programs
Theorem Prover (TP)
"Isabelle"
Architecture for Prototyping

Walther Neuper

ISAC Prototype

History
Architecture
New Generation
Applications?
Development
Supervision
Technologies
Projects
Engineering

Theorem Prover (TP) “Isabelle”

Java

SML

Gui 1
Gui 2
Gui n

Isac Math-Engine

Knowledge

Isac Server Dialogue

Programs Knowledge

Students

Engineer

core 1
core 2
core n
Architecture for Prototyping

Walther Neuper

Theorem Prover (TP) "Isabelle"

Java

GUI 1
GUI 2
GUI n
Students

Scala bridge

Isac Server
Dialogue

Knowledge

Isac Math-Engine
Programs
Knowledge

Students

Engineer

GUI 1
GUI 2
GUI n

Isac Engineer

core 1

core 2

core n

SML

Theorem Prover (TP) "Isabelle"
Architecture for Prototyping

Walther Neuper

ISAC Prototype
History
Architecture
New Generation
Applications
Development
Supervision
Technologies
Projects
Engineering

Theorem Prover (TP)
"Isabelle"

Java

Scala bridge

SML

Isac
Math-Engine

Engineer

Students

jEdit

GUI 1

GUI 2

GUI n

Isac Server
Dialogue

Knowledge Programs

Knowledge

Theorem Prover (TP)
"Isabelle"
Architecture for Prototyping

Walther Neuper

ISAC
⇝
FH?

ISAC Prototype
History
Architecture
New Generation
Applications?
Development
Supervision
Technologies
Projects
Engineering

Scala bridge
SML

Java

Isabelle

Knowledge
Programs

core 1
core n

Theorem Prover (TP)
"Isabelle"

Engineer

Students

jEdit
eclipse
browser

GUI 1
GUI 2
GUI n

Isac Server
Dialogue

Isac Math-Engine
Architecture for Prototyping

Walther Neuper

ISAC Prototype

History
Architecture
New Generation
Applications ?
Development
Supervision
Technologies
Projects
Engineering

Java

Isac Server
Dialogue

Knowledge

Scal a bridge

GUID 1

GUID 2

GUID n

jEdit

Students

Engineer

Isac Math-Engine

SML

Isabelle

Programs

Knowledge

Theorem Prover (TP) "Isabelle"

core 1

core 2

core n
Architecture for Prototyping

Theorem Prover (TP) "Isabelle"

Java

Scala bridge

SML

GUI 1

GUI 2

GUI n

Isac Math-Engine

Isac Server Dialogue

Knowledge

Programs Knowledge

Isac

Engineer

Students

jEdit

Scala bridge

Students

Engineer
Architecture for Prototyping
Features of New Generation

These features arise naturally from TP …

1. **check user input** automatically, **flexibly** and reliably: Input establishes a *proof situation* (for *automated* proving) with respect to the logical context.

2. **give explanations** on request by learners: All underlying mathematics knowledge is human readable, the system can be made **transparent** (is not a program code!)

3. **propose a next step** if learners get stuck: A specific kind of program computes a solution for a problem. The program is interpreted **step wise** by Lucas-Interpretation.
Features of New Generation

These features arise naturally from TP . . .

1. **check user input** automatically, **flexibly** and reliably:
   Input establishes a *proof situation* (for *automated* proving) with respect to the logical context.

2. **give explanations** on request by learners:
   All underlying mathematics knowledge is human readable, the system can be made **transparent** (is not a program code!)

3. **propose a next step** if learners get stuck:
   A specific kind of program computes a solution for a problem. The program is interpreted **step wise** by Lucas-Interpretation.
Features of New Generation

These features arise naturally from TP . . .

1. **check user input** automatically, **flexibly** and reliably: Input establishes a *proof situation* (for *automated* proving) with respect to the logical context

2. **give explanations** on request by learners: All underlying mathematics knowledge is human readable, the system can be made **transparent** (is not a program code!)

3. **propose a next step** if learners get stuck: A specific kind of program computes a solution for a problem. The program is interpreted **step wise** by Lucas-Interpretation.
Features of New Generation

These features arise naturally from TP . . .

1. **check user input** automatically, **flexibly** and reliably:
   Input establishes a *proof situation* (for *automated* proving) with respect to the logical context

2. **give explanations** on request by learners:
   All underlying mathematics knowledge is human readable, the system can be made **transparent** (is not a program code!)

3. **propose a next step** if learners get stuck:
   A specific kind of program computes a solution for a problem. The program is interpreted **step wise** by Lucas-Interpretation.
Features of New Generation

These features arise naturally from TP . . .

1. **check user input** automatically, *flexibly* and reliably:
   Input establishes a *proof situation* (for *automated* proving) with respect to the logical context

2. **give explanations** on request by learners:
   All underlying mathematics knowledge is human readable, the system can be made *transparent* (is not a program code!)

3. **propose a next step** if learners get stuck:
   A specific kind of program computes a solution for a problem. The program is interpreted *step wise* by Lucas-Interpretation.
Features of New Generation

These features arise naturally from TP . . .

1. check user input automatically, flexibly and reliably:
   Input establishes a proof situation (for automated proving) with respect to the logical context

2. give explanations on request by learners:
   All underlying mathematics knowledge is human readable, the system can be made transparent (is not a program code!)

3. propose a next step if learners get stuck:
   A specific kind of program computes a solution for a problem. The program is interpreted step wise by Lucas-Interpretation.
Features of New Generation

These features arise naturally from TP …

1. **check user input** automatically, **flexibly** and reliably:
   Input establishes a *proof situation* (for *automated* proving)
   with respect to the logical context

2. **give explanations** on request by learners:
   All underlying mathematics knowledge is human readable,
   the system can be made **transparent** (is not a program code!)

3. **propose a next step** if learners get stuck:
   A specific kind of program computes a solution for a problem.
   The program is interpreted **step wise** by Lucas-Interpretation.
The ISAC Prototype in Action

Demonstration
1. The ISAC Prototyping Project
   - The History
   - The Architecture
   - The New Generation + Demo

2. Preview: Learning Math with the New Generation

3. Challenges from ongoing Development in ISAC
   - Development by Students
   - Technologies used
   - Ideas for Subprojects
   - Project Engineering
Features of New Generation

These features arise naturally from TP . . .

1. **check user input** automatically, **flexibly** and reliably:
   Input establishes a *proof situation* (for *automated* proving) with respect to the logical context

2. **give explanations** on request by learners:
   All underlying mathematics knowledge is human readable, the system can be made **transparent** (is not a program code!)

3. **propose a next step** if learners get stuck:
   A specific kind of program computes a solution for a problem. The program is interpreted **step wise** by Lucas-Interpretation.

The features allow software for:

- **step-wise solving** problems in applied mathematics
- learning interactively like with a **chess-program**
- **self-explanatory** systems.
Features of New Generation

These features arise naturally from TP . . .

1. **check user input** automatically, **flexibly** and reliably:
   Input establishes a *proof situation* (for *automated* proving) with respect to the logical context

2. **give explanations** on request by learners:
   All underlying mathematics knowledge is human readable, the system can be made **transparent** (is not a program code!)

3. **propose a next step** if learners get stuck:
   A specific kind of program computes a solution for a problem. The program is interpreted **step wise** by Lucas-Interpretation.

The features allow software for:

- **step-wise solving** problems in applied mathematics
- learning interactively like with a **chess-program**
- **self-explanatory** systems.
Features of New Generation

These features arise naturally from TP . . .

1. **check user input** automatically, **flexibly** and reliably:
   Input establishes a *proof situation* (for *automated* proving) with respect to the logical context.

2. **give explanations** on request by learners:
   All underlying mathematics knowledge is human readable, the system can be made **transparent** (is not a program code!)

3. **propose a next step** if learners get stuck:
   A specific kind of program computes a solution for a problem. The program is interpreted **step wise** by Lucas-Interpretation.

The features allow software for:

- **step-wise solving** problems in applied mathematics
- learning interactively like with a **chess-program**
- **self-explanatory** systems.
Features of New Generation

These features arise naturally from TP . . .

1. **check user input** automatically, **flexibly** and reliably:
   Input establishes a *proof situation* (for *automated* proving) with respect to the logical context.

2. **give explanations** on request by learners:
   All underlying mathematics knowledge is human readable, the system can be made **transparent** (is not a program code!)

3. **propose a next step** if learners get stuck:
   A specific kind of program computes a solution for a problem. The program is interpreted **step wise** by Lucas-interpretation.

The features allow software for:

- **step-wise solving** problems in applied mathematics
- learning interactively like with a **chess-program**
- **self-explanatory** systems.
Some foreseeable Applications

- \textit{independent learning (MOOCs)}: system proposes next step, is transparent
- \textit{exploratory, trial & error learning}: input checked flexibly and reliably
- \textit{gaming} — \textit{interactive math problem solving}
- …?!? …

- joining theory and practice in academic math courses:
  course in 1\textsuperscript{st} semester $\leftarrow$?\rightarrow laboratory in 7\textsuperscript{th} semester
- complement of GeoGebra in formal calculations (highschool math is already mechanised in Isabelle !)
- academic online courses
  see Harvard University \url{https://www.edx.org/}
- wikipedia’s math content + interactive example calculation
- …?!? …
Some foreseeable Applications

- independent learning (MOOCs): system proposes next step, is transparent
- exploratory, trial & error learning: input checked flexibly and reliably
- gaming — interactive math problem solving
- …?!? …

- joining theory and practice in academic math courses: course in 1\textsuperscript{st} semester $\leftarrow$?\ $\rightarrow$ laboratory in 7\textsuperscript{th} semester
- complement of GeoGebra in formal calculations (highschool math is already mechanised in Isabelle !)
- academic online courses see Harvard University https://www.edx.org/
- wikipedia’s math content + interactive example calculation
- …?!? …
Some foreseeable Applications

- **independent learning (MOOCs):**
  system proposes next step, is transparent

- **exploratory, trial & error learning:**
  input checked flexibly and reliably

- **gaming — interactive math problem solving**

- ... ?!? ...

- joining theory and practice in academic math courses:
  course in 1\(^{st}\) semester $\leftarrow$ ? $\rightarrow$ laboratory in 7\(^{th}\) semester

- complement of GeoGebra in formal calculations
  (highschool math is already mechanised in Isabelle !)

- academic online courses
  see Harvard University [https://www.edx.org/](https://www.edx.org/)

- wikipedia’s math content + interactive example calculation

- ... ?!? ...
Some foreseeable Applications

- *independent learning (MOOCs)*: system proposes next step, is transparent
- *exploratory, trial & error learning*: input checked flexibly and reliably
- *gaming — interactive math problem solving*
- …?!? …

- joining theory and practice in academic math courses: course in 1\textsuperscript{st} semester $\leftarrow$?$\rightarrow$ laboratory in 7\textsuperscript{th} semester
- complement of GeoGebra in formal calculations (highschool math is already mechanised in Isabelle !)
- academic online courses see Harvard University [https://www.edx.org/](https://www.edx.org/)
- wikipedia’s math content + interactive example calculation
- …?!? …
Some foreseeable Applications

- *independent learning (MOOCs)*: system proposes next step, is transparent
- *exploratory, trial & error learning*: input checked flexibly and reliably
- *gaming — interactive math problem solving*
- ... ?!? ...

- joining theory and practice in academic math courses: course in 1\textsuperscript{st} semester $\leftarrow$ ? $\rightarrow$ laboratory in 7\textsuperscript{th} semester
- complement of GeoGebra in formal calculations (highschool math is already mechanised in Isabelle !)
- academic online courses see Harvard University [https://www.edx.org/](https://www.edx.org/)
- wikipedia’s math content + interactive example calculation
- ... ?!? ...
Some foreseeable Applications

- **independent learning (MOOCs):** system proposes next step, is transparent
- **exploratory, trial & error learning:** input checked flexibly and reliably
- **gaming — interactive math problem solving**
- ...?!? ...

- joining theory and practice in academic math courses: course in 1\textsuperscript{st} semester $\leftarrow$? $\rightarrow$ laboratory in 7\textsuperscript{th} semester
- complement of GeoGebra in formal calculations (highschool math is already mechanised in Isabelle !)
- academic online courses see Harvard University \url{https://www.edx.org/}
- wikipedia’s math content + interactive example calculation
- ...?!? ...
1 The ISAC Prototyping Project
   The History
   The Architecture
   The New Generation + Demo

2 Preview: Learning Math with the New Generation

3 Challenges from ongoing Development in ISAC
   Development by Students
   Technologies used
   Ideas for Subprojects
   Project Engineering
Development by Students

Facts: ISAC comprises

- 32 student projects produced 23 master theses
- successfully accomplished by 27 different students
- supervised by 7 different experts
- from 6 different univ.institutes (Graz, Vienna, Linz)

Methodology in ISAC:

- agile software development
- test-driven development (TODO: Graphical user interface testing)
- extreme programming

Links:
http://www.ist.tugraz.at/isac/Current_ISAC_Team
http://www.ist.tugraz.at/isac/Publications_and_Theses#Theses
http://www.ist.tugraz.at/isac/Credits
Development by Students

Facts: ISAC comprises
- 32 student projects produced 23 master theses
- successfully accomplished by 27 different students
- supervised by 7 different experts
- from 6 different univ.institutes (Graz, Vienna, Linz)

Methodology in ISAC:
- agile software development
- test-driven development (TODO: Graphical user interface testing)
- extreme programming

Links:
http://www.ist.tugraz.at/isac/Current_ISAC_Team
http://www.ist.tugraz.at/isac/Publications_and_Theses#Theses
http://www.ist.tugraz.at/isac/Credits
Development by Students

Facts: ISAC comprises

- 32 student projects produced 23 master theses
- successfully accomplished by 27 different students
- supervised by 7 different experts
- from 6 different univ.institutes (Graz, Vienna, Linz)

Methodology in ISAC:

- agile software development
- test-driven development (TODO: Graphical user interface testing)
- extreme programming

Links:

http://www.ist.tugraz.at/isac/Current_ISAC_Team
http://www.ist.tugraz.at/isac/Publications_and_Theses#Theses
http://www.ist.tugraz.at/isac/Credits
Technologies used in ISAC
Technologies used in ISAC

- **Obj.Oriented**
- **functional**
- **Java**
- **SML**

- **GUI 1**
- **GUI 2**
- **GUI n**

- **Students**
- **browser**

- **Isac Math-Engine**
- **Programs Knowledge**

- **Isac Server Dialogue**
- **Theorem Prover (TP) “Isabelle”**

- **core 1**
- **core 2**
- **core n**

- **Technologies used in ISAC**
Technologies used in **ISAC**

- **Obj.Oriented**
- **NetBeans**
- **Eclipse**
- **Mercurial version.manag.**
- **Java**
- **SML**
- **functional**
- **Students**
- **GUI 1**
- **GUI 2**
- **GUI n**
- **browser**
- **Isac Math-Engine**
- **Knowledge**
- **Programs Knowledge**
- **Theorem Prover (TP) “Isabelle”**
- **core 1**
- **core 2**
- **core n**

**Walther Neuper**

**ISAC Prototype**
- History
- Architecture
- New Generation

**Applications ?**

**Development**
- Supervision
- Technologies
- Projects
- Engineering
Technologies used in ISAC

- Java
- Java RMI
- SML

- GUI 1
- GUI 2
- GUI n

- Students
- Browser

- Obj.Oriented
- NetBeans
- Eclipse
- Mercurial
- "Isabelle"

- Isac Math-Engine
- Programs
- Knowledge

- Theorem Prover (TP)
  "Isabelle"

- core 1
- core 2
- core n
Technologies used in ISAC

ISAC
⇝
FH ?

Walther Neuper

ISAC Prototype
History
Architecture
New Generation

Applications ?
Development
Supervision
Technologies
Projects
Engineering

Students

Obj.Oriented
NetBeans
Java Swing
Eclipse
Mercurial
version.manag.

Java

Isac
Knowledge

Browser

GUI 1
GUI 2
GUI n

Isac
Math-Engine

Isabelle
XML-RPC
DOM Parser

XML Parser

Java Swing
Java RMI

functional

SML

core 1
core 2
core n

Theorem
Prover (TP)
"Isabelle"
Technologies used in \textit{ISAC}
Technologies used in ISAC

- Obj.Oriented
- NetBeans
- Java Swing
- Mercurial version.manag.
- SML
- Eclipse
- Java
- Isac
- Math-Engine
- core 1
- core 2
- core n

- Students
- GUI 1
- GUI 2
- GUI n

- browser
- Drools expert sys.
- SQL database
- JDBC

- Isac Server Dialogue
- Knowledge
- Isac Math-Engine
- Programs
- Knowledge

- Java RMI
- XML-RPC
- DOM Parser
- XML Parser

- Thm Prover (TP)
- “Isabelle”

- functional
- Java Webstart
- Java Keystore

- Java Server Faces
- Struts
- XML-RPC
- DOM Parser
- XML-RPC
- DOM Parser

- Drools expert sys.
- SQL database
- JDBC

- XML Parser
- XML Parser
Subprojects ↔ FH Programmes

- 1. Formula Editor, various Frontends for Cloudservices
- 2. Improve and Extend Rule-Based Dialogues
- 3. Mathematics as a Game (like Chess)?
- 4. Parallelize the ISAC Server for Cloudservices
- 5. Embed ISAC into Learning Management Systems
- 6. Provide ISAC with Authoring Tools
- 7. A Secure System for Exams Written in ISAC?
- ...
Subprojects ↔ FH Programmes

- 1. Formula Editor, various Frontends for Cloudservices
  Software Engineering, Mobile Computing

- 2. Improve and Extend Rule-Based Dialogues
  Software Engineering, Communication & Knowledge Media

- 3. Mathematics as a Game (like Chess)?
  Media Technology and Design

- 4. Parallelize the ISAC Server for Cloudservices
  Hardware-Software-Design

- 5. Embed ISAC into Learning Management Systems
  Communication and Knowledge Media

- 6. Provide ISAC with Authoring Tools
  Communication and Knowledge Media

- 7. A Secure System for Exams Written in ISAC?
  Secure Information Systems

- ...
1. Formula Editor, Frontends for Cloudservices

- **Formula input: most urgent for usability**
  - ? formula editor ??? handwritten formula recognition ?

- **Remote access:** ISAC front-ends connected by Java RMI
  web access for ISAC under construction,
  see http://www.ist.tugraz.at/isac/Current_ISAC_Team#Tom_Zillinger

The next years will bring new handheld devices!
1. Formula Editor, Frontends for Cloudservices

- Formula input: most urgent for usability?
  - formula editor ??? handwritten formula recognition ?

- Remote access: ISAC front-ends connected by Java RMI
  - web access for ISAC under construction, see http://www.ist.tugraz.at/isac/Current_ISAC_Team#Tom_Zillinger

The next years will bring new handheld devices!
1. Formula Editor, Frontends for Cloudservices

- Formula input: most urgent for usability
  - formula editor ??? handwritten formula recognition ?

- Remote access: ISAC front-ends connected by Java RMI
  - web access for ISAC under construction,
  - see http://www.ist.tugraz.at/isac/Current_ISAC_Team#Tom_Zillinger

The next years will bring new handheld devices!
1. Formula Editor, Frontends for Cloudservices

- Formula input: most urgent for usability? formula editor ??? handwritten formula recognition?

- Remote access: ISAC front-ends connected by Java RMI
web access for ISAC under construction,
see http://www.ist.tugraz.at/isac/Current_ISAC_Team#Tom_Zillinger

The next years will bring new handheld devices!
1. Formula Editor, Frontends for Cloudservices

- Formula input: most urgent for usability
  - formula editor ??? handwritten formula recognition ?

- Remote access: ISAC front-ends connected by Java RMI
  web access for ISAC under construction,
  see http://www.ist.tugraz.at/isac/Current_ISAC_Team#Tom_Zillinger

The next years will bring new handheld devices!
1. Frontends for Cloudservices

Walther Neuper

ISAC Prototype
History
Architecture
New Generation
Applications?
Development
Supervision
Technologies
Projects
Engineering

Theorem Prover (TP) "Isabelle"
Scala bridge
SML
Java
Isac Math-Engine
Engineer
core 1
core 2
core n

Knowledge Programs
Isac Server Dialogue
GUI 1
GUI 2
GUI n

Students
jEdit

Engineer

Knowledge
1. Frontends for Cloudservices
1. Frontends for Cloudservices
1. Frontends for Cloudservices
2. Rule-Based Dialogues

Preparing rule-based dialogue guidance ISAC integrates

- *drools*, a free, commercial(?!): rule-based system
  

- an open source data base logging user interactions
  

and a feasibility study

- "Implementation of Error-Patterns Guiding ISAC Dialogues"
  
2. Rule-Based Dialogues

Figure: Interaction with ISAC
2. Rule-Based Dialogues

Preparing rule-based dialogue guidance *ISAC* integrates

- *drools*, a free, commercial(?!?) rule-based system
  

- an open source data base logging user interactions
  

and a feasibility study

- ”Implementation of Error-Patterns Guiding *ISAC* Dialogues”
  

Demo in *ISAC*

The number of rules is still minimal (stubs only).

The integration of (commercial!) *drools* needs to be reconsidered!
2. Rule-Based Dialogues

Preparing rule-based dialogue guidance \textit{ISAC} integrates

- \textit{drools}, a free, commercial(?!?!) rule-based system
  
  \url{http://www.ist.tugraz.at/projects/isac/publ/mkienl_bakk.pdf}

- an open source data base logging user interactions
  
  \url{http://www.ist.tugraz.at/projects/isac/publ/fkober_bakk.pdf}

and a feasibility study

- ”Implementation of Error-Patterns Guiding \textit{ISAC} Dialogues”
  
  \url{http://www.ist.tugraz.at/projects/isac/publ/gdaroczy-ma.pdf}

Demo in \textit{ISAC}

The number of rules is still minimal (stubs only).

The integration of (commercial!) \textit{drools} needs to be reconsidered!
2. Rule-Based Dialogues

Preparing rule-based dialogue guidance ISAC integrates
- *drools*, a free, commercial(?!?) rule-based system
- an open source data base logging user interactions
and a feasibility study
- ”Implementation of Error-Patterns Guiding ISAC Dialogues”

Demo in ISAC

The number of rules is still minimal (stubs only).

The integration of (commercial!) *drools* needs to be reconsidered!
3. Mathematics like Chess

Learn

- by doing some **steps** (checked by the computer)
- in an adverse situation: go back some **steps** and try others
- in an unclear situation: change roles and watch the **steps** of the computer

**Steps** in

- **chess** are moves on the chessboard
- **math** are input of a formula/rule within a calculation

???????????? Ideas of gaming, of collaboration ????????????
3. Mathematics like Chess

Learn

- by doing some **steps** (checked by the computer)
- in an adverse situation:
  go back some **steps** and try others
- in an unclear situation:
  change roles and watch the **steps** of the computer

**Steps in**

- **chess** are moves on the chessboard
- **math** are input of a formula/rule within a calculation

?????????? Ideas of gaming, of collaboration ????????????
3. Mathematics like Chess

Learn

- by doing some **steps** (checked by the computer)
- in an adverse situation:
  go back some **steps** and try others
- in an unclear situation:
  change roles and watch the **steps** of the computer

**Steps in**

chess are moves on the chessboard

math are input of a formula/rule within a calculation

??????????? Ideas of gaming, of collaboration ????????????
3. Mathematics like Chess

Learn

- by doing some **steps** (checked by the computer)
- in an adverse situation: go back some **steps** and try others
- in an unclear situation: change roles and watch the **steps** of the computer

**Steps** in

- **chess** are moves on the chessboard
- **math** are input of a formula/rule within a calculation

??????????? Ideas of gaming, of collaboration ?????
3. Mathematics like Chess

Learn

- by doing some **steps** (checked by the computer)
- in an adverse situation:
  go back some **steps** and try others
- in an unclear situation:
  change roles and watch the **steps** of the computer

**Steps** in

- **chess** are moves on the chessboard
- **math** are input of a formula/rule within a calculation

???????????? Ideas of gaming, of collaboration ????????????
4. Parallelize the ISAC Server
4. Parallelize the ISAC Server
4. Parallelize the ISAC Server
The *ISAC* prototype already has the knowledge represented in XML and transforms XML to HTML with specific tools. You can start the front-end from a web link (Java Web Start).

Thus technology is ready to tackle challenging tasks like:

- administrate dialogue settings for groups
- provide groups with specific parts of knowledge
- hide certain problems until written exams
- . . .
5. Embed ISAC into LMS

The ISAC prototype already has the knowledge represented in XML, transforms XML to HTML with specific tools, and can start the front-end from a web link (Java Web Start).

Thus technology is ready to tackle challenging tasks like:

- administrate dialogue settings for groups
- provide groups with specific parts of knowledge
- hide certain problems until written exams
- ...

...
5. Embed ISAC into LMS

Theorem Prover (TP) "Isabelle"

Java

Scala

Bridge

SML

Isac

Math-Engine

Engineer

Students

GUI 1

GUI 2

GUI n

Knowledge

Dialogue

Isac Server

Programs

Knowledge

Core 1

Core 2

Core n

Engineer

jEdit

Scala bridge

Projects

Development

Supervision

Technologies

Projects

Engineering

ISAC Prototype

History

Architecture

New Generation

Applications
5. Embed ISAC into LMS
5. Embed ISAC into LMS

Walther Neuper

ISAC Prototype
History
Architecture
New Generation
Applications?

Development
Supervision
Technologies
Projects
Engineering

5. Embed ISAC into LMS

Theorem Prover (TP) "Isabelle"

Isac Math-Engine

moodle courses

Dialogue Author
Course Author

Mathematics Author

Java

Isac Server
Dialogue

Knowledge

jEdit

Engineer

GUI n

Students

GUI 1

GUI 2

core 1

core 2

core n

Core 1

Core 2

Core n

Java

Isac
Math-Engine

Programs
Knowledge

Knowledge

Course Author

Dialogue Author

Mathematics Author

moodle courses

GUI 1

GUI 2

GUI n

GUI 1

GUI 2

GUI n

Scalabridge

Theorem Prover (TP) "Isabelle"

“Isabelle”
6. Various Authoring Tools

*ISAC*'s knowledge needs adaption for different courses.

The mathematics knowledge is represented twice

1. as SML in the math-engine for efficient access
2. as XML and HTML for inspection within the GUI

Keeping the representations in sync during adaption is laborious, see
http://www.ist.tugraz.at/isac/Extend_ISAC_Knowledge

Thus authoring tools are really needed!
6. Various Authoring Tools

**ISAC**’s knowledge needs adaption for different courses.

The mathematics knowledge is represented twice

1. as SML in the math-engine for efficient access
2. as XML and HTML for inspection within the GUI

Keeping the representations in sync during adaption is laborious, see
http://www.ist.tugraz.at/isac/Extend_ISAC_Knowledge

Thus authoring tools are really needed!
6. Various Authoring Tools

**ISAC**’s knowledge needs adaption for different courses.

The mathematics knowledge is represented twice

1. as SML in the math-engine for efficient access
2. as XML and HTML for inspection within the GUI

Keeping the representations in sync during adaption is laborous, see

http://www.ist.tugraz.at/isac/Extend_ISAC_Knowledge

Thus authoring tools are really needed!
ISAC’s knowledge needs adaption for different courses.

The mathematics knowledge is represented twice

1. as SML in the math-engine for efficient access
2. as XML and HTML for inspection within the GUI

Keeping the representations in sync during adaption is laborious, see

http://www.ist.tugraz.at/isac/Extend_ISAC_Knowledge

Thus authoring tools are really needed!
6. Various Authoring Tools
Written exams on *computer systems* raise *additional challenges!*
Challenges in Project Engineering

50,000 LoCs from 32 subprojects since more than 10 years ;-) (plus the same amount of LoCs for ISAC’s math-engine in SML)

Particularly urgent are:

- migration from Java 1.6 > 1.7 > 1.8 (Java is not upwards compatible in all details!)
- push Java RMI over subnet boundaries
- present and input formulas like $\frac{x+1}{x^2-1}$
- review the zoo of technologies
- review and unify code
- ...

... as preparations for successful master projects!
Challenges in Project Engineering

50,000 LoCs from 32 subprojects since more than 10 years ;-) (plus the same amount of LoCs for ISAC’s math-engine in SML)

Particularly urgent are:

- migration from Java 1.6 > 1.7 > 1.8
  (Java is not upwards compatible in all details!)
- push Java RMI over subnet boundaries
- present and input formulas like $\frac{x+1}{x^2-1}$
- review the zoo of technologies
- review and unify code
- ...

... as preparations for successful master projects!
Challenges in Project Engineering

50.000 LoCs from 32 subprojects since more than 10 years ;-) (plus the same amount of LoCs for ISAC’s math-engine in SML)

Particularly urgent are:

- migration from Java 1.6 > 1.7 > 1.8
  (Java is not upwards compatible in all details!)
- push Java RMI over subnet boundaries
- present and input formulas like $\frac{x+1}{x^2-1}$
- review the zoo of technologies
- review and unify code
- …

…as preparations for successful master projects!
Thank you for attention!

Meeting for installation of ISAC next week
Thank you for attention!

Meeting for installation of ISAC next week