SDE Group: Opportunities for Students

Project Course 7.5hp (lp3 and/or lp4), can do twice
M.Sc. thesis
Amanuens Position Paid (yearly) position to support research

(**1**) Automatically Inspecting and Transforming C/C++ code

ERICSSON

(**2**) Custom Bug Checkers with Logic Programming and Pattern Matching

(**3**) ExtendC: A C Compiler in JastAdd

(**4**) ExtendJ for Java 9 and later

(**5**) Advanced Bug Checking in ExtendJ with the latest program analysis frameworks

(**6**) Advancing the TEAL language for teaching program analysis (EDAP15)

(**7**) Executable Natural Semantics: Fast Language Prototyping in JastAdd

(**8**) Typing Rules: Fast Type System Prototyping in JastAdd

Amanuens: GANDER: Code Reviews & Eye Tracking

Amanuens: WARA-SW: Making Bug Detectors Useful

WASP WALLENBERG AI AUTONOMOUS SYSTEMS AND SOFTWARE PROGRAM

jastadd ExtendJ IntraCFG JavaDL/MetaDL

Project Course / M.Sc. projects

- If you have not taken EDAP15 or EDAN65, some projects may be challenging
- 2 students per project (can work solo, but I recommend working in pairs)
- **Project Course:**
  - Offered as **EDAN70** (first time) or **EDAN90** (second time)
  - 7.5 hp, credited in lp4
  - You can do both in sequence
  - Actual coursework can take place in lp3 / lp4
  - Weekly supervision meetings
  - Interested? Contact Christoph with e-mail subject line starting with "EDAN70" or "EDAN90"
- **M.Sc. thesis:**
  - 30 hp
  - Can be continuation: try idea in project course, deepen in thesis
  - Interested? Contact Christoph with e-mail subject line starting with "MSc Thesis Project"
  - Weekly supervision meetings
- NB: Supervision meetings may be remote, esp. in lp4

1. Inspecting and Transforming C/C++ Code

- **Background**
  - DMCE ("Did My Code Execute") is an Open Source tool for injecting C/C++ code into (nearly) arbitrary C/C++ code
  - Uses:
    - Profiling (where am I spending execution time?)
    - Logging
    - Debugging
  - Widely used (internally) at Ericsson
- **Project Goal**
  - Read up on DMCE and similar tools (e.g., `pin`)
  - Experiment with them to find strengths / weaknesses
  - Build and run experiments: which tool works best for industrial tasks?

Suitable for:	M.Sc., EDAN70/EDAN90
Supervisor:	Christoph Reichenbach, collaboration with Patrik Åberg (Ericsson)
Prerequisites:	Basic C or C++ experience
Tools:	DMCE (Ericsson)

ERICSSON

2. An intermediate language for MetaDL

- **Background**
  - MetaDL allows writing bug checks for Java ("no string comparison with =="):

$$<:\$x == \$y:>, \text{StringType}(\$x)$$

  - Combines syntax-based pattern matching with logical rules
  - Limitation: the following syntactic check will not match e.g. `1 / (0+0)`:

$$<:\$x/0:>$$

  - Must use logical rules instead to model semantics (clunkier to write)
- **Project**
  - Allow defining (abstract) semantic rules, e.g. $\$[z+0] \Rightarrow \$[z]$
  - Auto-generate logical rules from simplified syntax, e.g.: $<:\$x/\$[0]:>$

Suitable for:	M.Sc., EDAN70/EDAN90
Supervisor:	Alexandru Dura, Christoph Reichenbach
Prerequisites:	*Recommended:* EDAN65, EDAP15, or equivalent experience
Status:	One student has already expressed interest
Tools:	MetaDL (github), research paper

3. Initial ExtendC

- Build an initial (or complete) C compiler frontend in JastAdd
  - Parsing
  - Name analysis
  - Type analysis
  - C Preprocessor: integrate directly in AST (reusing prior research)

Suitable for:	M.Sc., EDAN70/EDAN90 (prototype only)
Supervisor:	Christoph Reichenbach or tbd
Prerequisites:	EDAN65 or other JastAdd experience
Tools:	jastadd

4. ExtendJ support for Java 9 and later

- **Background**
  - ExtendJ is a Java compiler built in JastAdd
  - Support for up to Java 8
- **Project**
  - Add support for language features from Java 9–17
  - Details depend on size of project (# of students etc.)

Suitable for:	M.Sc., EDAN70/EDAN90
Supervisor:	tbd
Prerequisites:	EDAN65 or other JastAdd experience
Tools:	jastadd, ExtendJ

5. Supporting advanced program analysis for Java

- **Background**
  - The SDE group's ExtendJ compiler for Java includes a number of program analyses that can find bugs and point to the problematic source code
  - There are more advanced analyses available (SOOT, OPAL), but those analyse Java bytecode
    $\Longrightarrow$ Can't always find the right program locations
- **Project**
  - Translate ExtendJ's Java representation into SOOT or OPAL intermediate representations
  - Examine how much better we can make bug reports from these framweorks

Suitable for:	M.Sc., EDAN70/EDAN90
Supervisor:	Christoph Reichenbach + tbd
Prerequisites:	EDAN65, EDAP15, or equivalent experience
Tools:	jastadd, EXTENDJ, soot, OPAL

6. Extensions to the TEAL teaching language

- **Background**
  - EDAP15 uses a special programming language, TEAL, to teach program analysis techniques
  - TEAL is implemented in JastAdd
  - To improve the course, we want to improve TEAL and its tooling
- **Project**
  - Multiple options:
    - Graph visualiser over source code (via JavaScript)
    - JastAdd-based TEAL API cleanup
    - Incorporate INTRACFG framework for data flow analysis

Suitable for:	M.Sc., EDAN70/EDAN90
Supervisor:	Christoph Reichenbach + tbd
Prerequisites:	EDAN65, EDAP15, or equivalent experience
Tools:	jastadd, TEAL

7. JastAdd + Natural Samantics

Given Natural Semantics specifications, generate a JastAdd-based interpreter:

$$\dfrac{\dfrac{\dfrac{1 \in nat}{E_\emptyset \vdash \texttt{1} \Downarrow 1}\,(nat_I) \quad \dfrac{2 \in nat}{E_\emptyset \vdash \texttt{2} \Downarrow 2}\,(nat_I)}{E_\emptyset \vdash \texttt{1 + 2} \Downarrow 3}\,(add_I) \quad \texttt{a} \in id \quad \dfrac{\dfrac{\texttt{a} \in id}{E_\emptyset[\texttt{a} \mapsto 3] \vdash \texttt{a} \Downarrow 3}\,(var_I) \quad \dfrac{\texttt{a} \in id}{E_\emptyset[\texttt{a} \mapsto 3] \vdash \texttt{a} \Downarrow 3}\,(var_I)}{E_\emptyset[\texttt{a} \mapsto 3] \vdash \texttt{a + a} \Downarrow 6}\,(add_I)}{E_\emptyset \vdash \texttt{let a = 1 + 2 in a + a} \Downarrow 6}\,(let_I)$$

Suitable for: M.Sc., EDAN70/EDAN90
Supervisor: Christoph Reichenbach
Prerequisites: Recommended: EDAN65, EDAP15, or equivalent experience
Tools: jastadd

8. JastAdd + Typing Rules

Given a set of typing rules, generate a JastAdd specification that performs type analysis

$$\texttt{true} : \text{Bool}$$

$$\texttt{false} : \text{Bool}$$

$$\frac{v \in nat}{v : \text{Nat}} \; (t\text{-}nat)$$

$$\frac{e_1 : \text{Nat} \quad e_2 : \text{Nat}}{e_1 = e_2 : \text{Bool}} \; (t\text{-}eq\text{-}nat)$$

$$\frac{e_1 : \text{Bool} \quad e_2 : \text{Bool}}{e_1 = e_2 : \text{Bool}} \; (t\text{-}eq\text{-}bool)$$

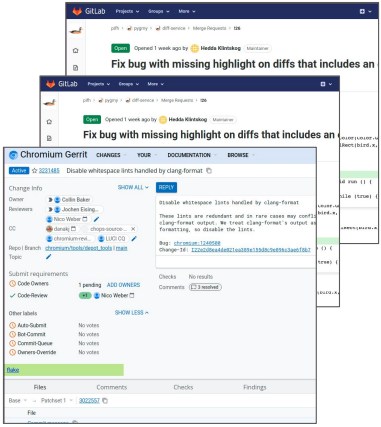
Suitable for: M.Sc., EDAN70/EDAN90
Supervisor: Christoph Reichenbach
Prerequisites: Strongly recommended: EDAN65, EDAP15, or equivalent experience
Tools: jastadd

Positions as Amanuens

- Paid position (monthly salary, pension contribution etc.)
- Typical commitment: 1 day per week
- Contract typically over one year
- Support ongoing research projects
- Positions for two separate projects:
  - GANDER (Emma Söderberg)
  - WARA-SW (Christoph Reichenbach)
- Contact us with e-mail subject line starting with “Amanuens position”

GANDER: Evolve eye tracker, understand SW dev. needs

GANDER: Gazing at Code Review(s)

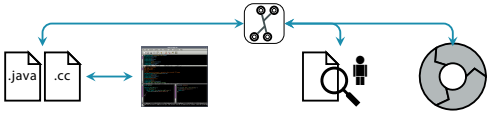


ELLIIT
Excellence Center at Linköping – Lund in Information Technology

WARA-SW: Build framework for evaluating software tools

WASP WALLENBERG AI, AUTONOMOUS SYSTEMS AND SOFTWARE PROGRAM

- Which bug checkers work best for what?
- How can we integrate novel software tools easily into code review, IDEs, CI?
- How can we showcase software technology research across Sweden to Swedish industry?
- WARA-SW is collaboration with: KTH, Ericsson, Saab, possibly more soon



Core Software Engineering Tools

WP5
WP4

WP8
IDE Tools

WP7
Review Tools

WP6
CI Tools

WP9
Courses

Topic Area: *Software Tools*

Topic Area: *Correct by Construction* (Saab)

Topic Area: tbd

...