TECHNISCHE UNIVERSITÄT
KAISERSLAUTERN

Software-Engineering Seminar, Summer 2016

AG Softech
FB Informatik
TU Kaiserslautern

Goals

- Learn an interesting topic in SE
- Read and understand scientific papers explaining the topic
- Learn how to present the topic

Your tasks

You get one topic based on an existing paper from a good conference or journal.

- Read and understand the paper
- Search for additional material on the topic
- Write a paper
  - Language: English
  - 10-15 pages, LNCS template
  - Easy to read for average master student
  - Present the problem and motivation of the work
  - Present the solution
  - You may add critique
- Review 2 papers from other students
- Presentation
  - 20-30 minutes presentation
  - about 15 minutes discussion and questions (know your topic!)

How to fail a seminar?

- Plagiarism
- Late submissions
- Not attending final presentations
- Poorly written paper
- Bad presentation
- Poor review
- Never talk to your supervisor
- Do not use a spell checker

Students

Accepted:

- Mahwash Makhdoom
- Syed Arij Hussain
- Arunava Chaki
- Jay Bonkile
- Vivek Jain
- Abhigyan Mahajan
- Dalbir Singh
- Shruthi Lalka

Waiting list:

- Fahad Mehboob
- Eid Muhammad
- Pascal Stahl
- Hafiz Ahsan Raza
- Ranjith K.B
- Shivaji Yadav
- Devina Vyas

For students on the waiting list: Write a mail to Peter Zeller, which explains why you need/want to do the seminar this semester. In particular: Are other seminars feasible? Last semester?

Choosing topics

- Today: topics are presented
- Until February 26th: Find 3 or more topics which are interesting for you
- We will assign topics based on your preferences

Please note: We will present more topics than there are available places in the seminar. We will assign topics not only based on your preferences, but also according to the supervisors.

Putting consistency back into eventual consistency

Authors: Valter Balegas, Sérgio Duarte, Carla Ferreira, Rodrigo Rodrigues, Nuno M. Preguiça, Mahsa Najafzadeh, Marc Shapiro:
Published in: Proceedings of the Tenth European Conference on Computer Systems (EuroSys 2015)

Supervisor: Arnd Poetzsch-Heffter

Designers of distributed systems have to choose between strong consistency and weaker forms of consistency for storing the data. This paper presents a system, which can help with these decisions, by automatically finding operations which might violate application level invariants when executed under weak consistency. The paper also presents techniques for fixing theses issues in a system.

Global Sequence Protocol: A Robust Abstraction for Replicated Shared State

Authors: Sebastian Burckhardt, Daan Leijen, Jonathan Protzenko, Manuel Fähndrich
Published at: 29th European Conference on Object-Oriented Programming (ECOOP 2015)

Supervisor: Arnd Poetzsch-Heffter

The Global Sequence Protocol is an operational model, which shows how to manage shared data in a replicated system. The protocol is the foundation of so called “Cloud Types” which are used in Microsofts TouchDevelop platform and allow programmers to easily manage data which is shared between multiple devices and users.

Coordination Avoidance in Database Systems

Authors: Peter Bailis, Alan Fekete, Michael J. Franklin, Ali Ghodsi, Joseph M. Hellerstein, Ion Stoica
Published in: Proceedings of the Very Large Data Base (VLDB) Endowment, Volume 8 (2014)

Supervisor: Houssam Abdoullah

Avoiding coordination between replicas in a Geo-replicated data store is vital to improve availability, performance and latency. However, some application correctness criteria need coordination to be preserved. This paper presents invariant confluence analysis which checks whether coordination is needed to preserve some application level correctness criteria (Invariant).

The Homeostasis Protocol: Avoiding Transaction Coordination Through Program Analysis

Authors: Sudip Roy, Lucja Kot, Gabriel Bender, Bailu Ding, Hossein Hojjat, Christoph Koch, Nate Foster, Johannes Gehrke
Published in: Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data (SIGMOD 2015)

Supervisor: Houssam Abdoullah

This paper tries to avoid coordination between replicas by exploiting the semantic of the program to only synchronize when necessary. It uses program analysis to derive the conditions under which coordination is not needed and presents "the homeostasis protocol" which allow the replicas to work without coordination as long as these conditions are maintained.

Declarative programming over eventually consistent data stores

Authors: K. C. Sivaramakrishnan, Gowtham Kaki, Suresh Jagannathan
Published at: Proceedings of the 36th ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI 2015)

Supervisor: Houssam Abdoullah or Peter Zeller

Distributed databases often provide the user with different consistency levels, where stronger consistency means less performance and less availability, while weaker consistency means harder to program applications. For programmers it is not easy to choose the right consistency level for applications, since the consistency requirements on the application level do not always map directly to consistency levels on the database. The approach in this paper lets programmers describe the required consistency on the application level and uses automated theorem provers to automatically map those requirements to the database level.

For this topic it is recommended to know Haskell (or a similar language) and have a good understanding of formal logic.

Composite Replicated Data Types

Authors: Alexey Gotsman, Hongseok Yang
Published at: Programming Languages and Systems - 24th European Symposium on Programming (ESOP 2015)

Supervisor: Houssam Abdoullah or Peter Zeller
Replicated data types are used with eventually consistent databases and help programmers to ensure convergence and some level of consistency in their application, without programming application specific merge functions for every use case. This paper presents a formal proof system, which can be used to prove the correctness of a replicated data type, which is composed of smaller data types.
For this topic it is recommended to have a good understanding of formal logic, proofs, and programming language semantics.

Patterns in Property Specifications for Finite-State Verification

Authors: Matthew B. Dwyer, George S. Avrunin, James C. Corbett
Published in: Proceedings of the 1999 International Conference on Software Engineering (ICSE' 99)

Supervisor: Peter Zeller

Temporal logic formulas are often used in system specifications and can then be automatically checked by tools. However, users often have problems to specify the requirements of their application in temporal logics. This paper explores common patterns in real-world specification and then extracts them into a more approachable language for specification.

Hybrid Concolic Testing

Authors: Rupak Majumdar, Koushik Sen
Published at: 29th International Conference on Software Engineering (ICSE 2007)

Supervisor: Peter Zeller

Concolic executions are a combination of concrete and symbolic executions. With symbolic execution, a constraint solver can be used to exactly generate the values that cause a certain path to get executed. However, in practice this often fails for more complex examples, because the generated conditions get too big. The hybrid approach presented in this paper combines concolic executions with random testing in order to get a better coverage than either approach on its own.
Students are expected to also research the related work on concolic testing and use one of the available tools with an own example.

A metamodel of access control for distributed environments: Application and properties

Authors: Clara Bertolissi, Maribel Fernández
Published in: Information and Computation, Volume 238 (November 2014), Special Issue on Security and Rewriting Techniques

Supervisor: Mathias Weber

Various different access control models exist, most of them designed from scratch. These models have commonalities which can be abstracted into a metamodel of access control models. For distributed environments one has to specify how to connect the access control models on different sites. The paper presents such a metamodel and introduces a formalization which allows to proof properties of access control policies.

Federated Access Management for Collaborative Network Environments: Framework and Case Study

Authors: Carlos E. Rubio-Medrano, Ziming Zhao, Adam Doupé, Gail-Joon Ahn
Published at: Proceedings of the 20th ACM Symposium on Access Control Models and Technologies (SACMAT 2015)

Supervisor: Mathias Weber

Organizations which want to share information with each other still want to control the access to this shared information. The paper introduces an approach for access control to shared data based on attributes. The organization need not agree an a common set of attributes, but can instead specify policies based on foreign attributes published by the organization owning the data.

Transaction Chopping for Parallel Snapshot Isolation

Authors: Andrea Cerone, Alexey Gotsman, Hongseok Yang
Published at: Distributed Computing - 29th International Symposium (DISC 2015)

Supervisor: Deepthi Akkoorath

Serializable transactions in Geo-replicated databases are not scalable. Hence weaker consistency models such as Parallel Snapshot Isolation were proposed. However when transactions are long, probability of conflicts is more and thus more aborts occur. Transactional chopping is a way of splitting long transactions into small transactions without changing the expected behavior. This paper discusses when a transaction can be chopped if the consistency provided by the database is parallel snapshot isolation.

Transactional Interference-less Balanced Tree

Authors: Ahmed Hassan, Roberto Palmieri, Binoy Ravindran
Published at: Distributed Computing - 29th International Symposium (DISC 2015)

Supervisor: Deepthi Akkoorath

Using Software Transactional Memory, programmers can safely access shared memory without explicit locks or other synchronization mechanisms. Similar to database transactions, exclusive access to the shared memory is handled by STM transactions. If any conflicts occur, transactions are aborted and restarted. Generic mechanisms offered by STM often do not exhibit good performance when used for implementing large data structures such as maps or trees. This paper discusses the design of a transactional friendly balanced tree.

JErlang: Erlang with joins

Authors: Hubert Plociniczak, Susan Eisenbach
Published at: Coordination Models and Languages, 12th International Conference (COORDINATION 2010)

Supervisor: Deepthi Akkoorath

Erlang is a programming language suitable for programming distributed systems, based on actor model. Processes communicate with each other through message passing, and there is no shared memory between processes. Hence synchronization among multiple processes is difficult. This paper proposes an extension to Erlang using "join" as synchronization primitive.

Generating Safe Template Languages

Authors: Florian Heidenreich, Jendrik Johannes, Mirko Seifert, Christian Wende, Marcel Böhme
Published at: Proceedings of the 8th international conference on generative programming and component engineering (GPCE 2009)

Supervisor: Malte Brunnlieb

Template-based code generators enable easy specification as well as maintenance of code to be generated. However, a closer look reveals serious problems faced by template developers. At template development time there is often no tool support indicating syntax and type issues of the generated code (in object-language). There are template languages covering such analysis for a single object-language. This paper introduces a mechanism to extend any language description to support general template language constructs.

For this topic it is recommended to have a good understanding of language engineering, template-based generation, and compiler construction.

The JastAdd Extensible Java Compiler

Authors: Torbjörn Ekman, Görel Hedin
Published at: Proceedings of the 22nd annual ACM SIGPLAN conference on Object-oriented programming systems and applications (OOPSLA 2007)

Supervisor: Malte Brunnlieb

JastAdd is a metacompiler tool, which provides advanced support for constructing modular an extensible compilers. Its extension mechanism is based on object-orientation and attributed grammars among other things. A Java 1.4 compiler has already been implemented using JastAdd. This paper describes the extension mechanisms used to extend the Java 1.4 compiler specification in JastAdd to support the Java 1.5 language specification.
For this topic it is recommended to have a good understanding of compiler construction, attributed grammars.

Schedule

- You have until February 26th to choose your favorite topics.
  Talk to supervisors to get more information about a topic.
  When you have decided send your top-3 to Peter Zeller.
- February 29th: Topics will be assigned
- Talk to your supervisor.
  Make a plan with intermediate steps.
- June 15th: Submit reviewable draft of your paper (8-14 pages)
- June 15 - July 1st: Review two papers from other students
- July 8th: Friday afternoon presentations
- July 15th: Friday afternoon presentations
- July 22nd: Submit final version of paper