Wed 3:30-5:00 pm - Rose Ballroom B

now happens-before later --- Static Schedule Analysis of Fine-grained Parallelism with Explicit Happens-before Relationships

- Christoph M. Angerer, ETH Zurich, Switzerland
- Thomas R. Gross, ETH Zurich, Switzerland

Current compilers are still largely ignorant of the scheduling of parallel tasks at runtime. Without this information, however, they have difficulties optimizing and verifying concurrent programs.

In this paper, we present a programming model where the program contains explicit scheduling constraints in the form of happens-before relationships between scheduled tasks. This model allows for flexible and fine-grained ad hoc parallelism while still enabling us to statically extract an abstraction of the runtime schedule. The result of this schedule analysis can answer the question as to whether two tasks execute in sequence, exclusively, or in parallel with each other.

Collaborative Model Merging

- Maximilian Koegel, Technical University Munich, Chair for Applied Software Engineering, Germany
- Helmut Naughton, Technical University Munich, Chair for Applied Software Engineering, Germany
- Jonas Helming, Technical University Munich, Chair for Applied Software Engineering, Germany
- Markus Herrmannsdoerfer, Technical University Munich, Chair for Applied Software Engineering, Germany

Models are important artifacts in the software development life-cycle and are often the result of a collaborative activity of multiple developers. When multiple developers modify the same model, conflicts can occur and need to be resolved by merging. Existing approaches for model merging require developers to solve all conflicts before commit. The later a developer commits the more the probability for even more conflicts increases. This forces the developers to solve every conflict as soon as possible and without consulting the other developer. However we claim that developers often need to discuss their choice of conflict resolution with another developer in case of a complex conflicts, since a conflict also expresses differences in opinion about the model. In this paper we propose to allow developers to postpone a decision of a modeling conflict. We present an approach to make conflicts part of the model and represent them as first-level entities based on issue modeling from the field of Rationale Management. This facilitates the possibility for collaborative conflict resolution and merging. Furthermore it allows for a complete batch merge instead of interactive merging, where all conflicts are added to the model and then solved later. To substantiate our claim that developers favor to discuss complex conflicts we have also conducted a case study.

A Recommender for Conflict Resolution Support in Optimistic Model Versioning

- Petra Brosch, Business Informatics Group, Vienna University of Technology, Austria, Austria
- Martina Seidl, Business Informatics Group, Vienna University of Technology, Austria, Austria
- Gerti Kappel, Business Informatics Group, Vienna University of Technology, Austria, Austria

The usage of optimistic version control systems comes along with cumbersome and time-consuming conflict resolution in the case that the modifications of two developers are contradicting. For code as well as for any other artifact the resolution support moves hardly beyond the choices "keep mine", "keep theirs", "take all changes", or "abandon all changes".

To ease the conflict resolution in the context of model versioning, we propose a recommender system which suggests automatically executable resolution patterns to the developer

responsible for the conflict resolution. The lookup algorithm is based on a similarity-aware graph matching approach incorporating information from the metamodel of the used modeling language. This allows not only the retrieval of recommendations exactly matching the given conflict situation, but also the identification of similar conflict situations whose resolution patterns are adaptable to the current conflict.

Emergent Feature Modularization

- Màrcio Ribeiro , Federal University of Pernambuco, Brazil
- Humberto Pacheco, Federal University of Pernambuco, Brazil
- Leopoldo Teixeira, Federal University of Pernambuco, Brazil
- Paulo Borba, Federal University of Pernambuco, Brazil

Virtual Separation of Concerns was introduced as a way to reduce drawbacks of implementing product line variability with preprocessors. Developers can focus in certain features and hide others of no interest. However, these features eventually share elements between them, which might break feature modularization, since modifications in a feature result in problems for another. We present the concept of emergent feature modularization, which aims to establish contracts between features, to prevent the developer from breaking other features, when performing a maintenance task. These interfaces are product-line-aware, in the sense that it only takes into account valid feature combinations. We also present an initial prototype of a tool that implements the concept.

Thu 3:30-5:00 pm - Rose Ballroom B

Sonifying Performance Data to Facilitate Tuning of Complex Systems

- Cody Henthorne, Raytheon BBN Technologies, United States
- Eli Tilevich, Virginia Tech, United States

In the modern computing landscape, the challenge of tuning software systems is exacerbated by the necessity to accommodate multiple divergent execution environments and stakeholders. Achieving optimal performance requires a different configuration for every combination of hardware setups and business requirements. In addition, the state of the art in system tuning can involve complex statistical models, which require deep expertise not commonly possessed by the average software developer. This paper presents a novel approach to tuning complex software systems by leveraging sound to convey performance information during execution. We conducted a scientific survey to determine which sound characteristics (e.g., loudness, panning, pitch, tempo, etc.) are most accurate to express information to the average programmer. As determined by the survey, the characteristics that scored the highest across all the participants were used to create a proof-of-concept demonstration. The demonstration showed that a programmer who is not an expert in either software tuning or enterprise computing can configure the parameters of a real world enterprise application server, so that its resulting performance surpasses that exhibited under the standard configuration. Our results indicate that sound-based tuning approaches can provide valuable solutions to the challenges of configuring complex computer systems.

Inferring Arbitrary Distributions for Data and Computation

- Soham S. Chakraborty, TCS, TRDDC, India
- V.Krishna Nandivada, IBM IRL, India

In the era of mult-core systems, one of the key requirements of achieving better utilization of multiple available cores is that of parallelization of code across multiple distributed nodes; this involves (re)distribution of both data and computation. Such a transformation can be a fairly tedious activity considering the possible dependencies (data, control) and interference between different segments of the code. Further, to keep the data accesses local, computation distribution requires appropriate data distribution and vice versa. And this inter-dependence between distribution of data and computation makes the problem challenging. Another important challenge in this context is that the desired distribution may not be one among the well-known distributions (such as blocked, cyclic etc), and thus reasoning about it can be nontrivial. We present a refactoring framework that can help an application developer to incrementally distribute programs in the context of distributed memory multi-core systems. Given a loop and an array accessed therein, the goal of our framework is to distribute the array based on a specified distribution for the loop (or vice versa) such that the number of remote accessed are reduced. Our framework goes beyond the well-known distributions, and can handle any arbitrary distributions. In our initial investigation, we have used our transformations on varied parallel benchmark programs and have been able to show its applicability along the expected lines.

Ficticious: MicroLanguages for Interactive Fiction

- James D Palmer, Northern Arizona University, United States

In this paper we provide an experience report where language oriented programming approaches are applied to complex game design. Ficticious is a G-expression based pidgin of several microlanguages designed for describing complex narrative worlds that exist within interactive fiction. Ficticious code execution is realized through a series of G-expression language transformations that transform Ficticious elements into the general programming language Ginger and in turn is translated into calls against the underlying machine. In this paper we explore Ficticious's unique object model and demonstrate how dynamic language transformations can be a powerful tool for implementing separation of concerns, rich text markup, complex virtual world design and character interaction.

Harnessing Emergence for Manycore Programming: Early Experience Integrating Ensembles, Adverbs, and Object-based Inheritance

- David Ungar, IBM Research, United States
- Sam S. Adams, IBM Research, United States

We believe that embracing nondeterminism and harnessing emergence have great potential to simplify the task of programming manycore processors. To that end, we have designed and implemented Ly, pronounced "Lee", a new parallel programming language built around two new concepts:

1. ensembles which provide for parallel execution and replace all collections; and,
2. iterators, and adverbs, which modify the parallel behavior of messages sent to ensembles.

The broad issues around programming in this fashion still need investigation, but, after our initial Ly programming experience, we have identified some specific issues that must be addressed in integrating these concepts into an object-based language, including empty ensembles, partial message understanding, non-local returns from ensemble members, and unintended ensembles.