Software Engineering … Knowledge Engineering

This research provides in depth knowledge about legacy software systems and their architectures. In this respect I have done extensive research in both dynamic and static software system analysis by applying data mining and pattern matching techniques.

- ***Dynamic Analysis***. This research identifies the implementation of specific software functionality within a software system without any prior knowledge about the source code. The approach consists of applying specific sets of scenarios on an instrumented software system to extract execution traces. Next, sequential pattern mining algorithm and concept lattice analysis are applied to extract execution patterns and locate the target source code. We expanded this approach by applying it on service-oriented architecture (SOA) to measure the quality of web services in service selection and composition [j5, c37, c26, c23, c20, c18, c17, c16, c15, c14, c12, c27].

- ***Static Analysis***. This research addresses the design and development of an incremental software architecture recovery and evaluation environment using data mining techniques. The environment is interactive and provides: pattern-based architectural recovery using a query language and approximate graph pattern matching; optimization clustering; partitioning; and view-based architectural design evaluation. These techniques have been implemented within my Alborz toolkit [b1, ch1, j3, j1, c24, c10, c9, c8, c7, c6, c5, c4, c3, c2, c1].

Abstracts of Selected Publications

Dynamic Analysis of Software Systems

Dynamic Knowledge Extraction from Software Systems using Sequential Pattern Mining

Kamran Sartipi and Hossein Safyallah

International Journal of Software Engineering and Knowledge Engineering (IJSEKE)

World Scientific Publisher. vol 20(6), 2010, pages 761-782

(PDF)

This paper presents a novel technique for dynamic analysis of software systems to identify the implementation of certain software functionality known as software features. In the proposed approach, a specific feature is shared by a number of task scenarios that are applied on the software system to generate execution traces. The application of a sequential pattern mining technique on the generated execution traces allows us to extract execution patterns that reveal the specific feature functionality. In a further step, the extracted execution patterns are distributed over a concept lattice to separate feature-specific group of functions from commonly used group of functions. The use of lattice also allows for identifying a family of closely related features in the source code. Moreover, in this work we provide a set of metrics for evaluating the structural merits of the software system such as component cohesion and functional scattering. We have implemented a prototype toolkit and experimented with two case studies Xfig drawing tool and Pine email client with very promising results.

Identifying Distributed Features in SOA by Mining Dynamic Call Trees

Anis Yousefi and Kamran Sartipi

IEEE International Conference on Software Maintenance (ICSM'11). Williamsburg VA, USA Sept 25-30, 2011. Pages 73-82.

(PDF)

This paper proposes a new approach for identifying the implementation of web service features in a service oriented architecture (SOA) by mining dynamic call trees that are collected from distributed execution traces. The proposed approach addresses the complexities of SOA-based systems that arise from: features whose locations may change due to changing of input parameters; execution traces that are scattered throughout different service provider platforms; and trace files that contain interleaving of execution traces related to different concurrent service users. In this approach, we execute different groups of feature-specific scenarios and mine the resulting dynamic call trees to spot paths in the code of a service feature, which correspond to a specific user input and system state. This allows us to focus on a the implementation of a specific feature in a distributed SOA-based system for different maintenance tasks such as bug localization, structure evaluation, and performance analysis. We define a set of metrics to assess structural properties of a SOA-based system. The effectiveness and applicability of our approach is demonstrated through a case study consisting of two service-oriented banking systems.

An Amalgamated Dynamic and Static Architecture Reconstruction Framework to Control Component Interactions

Kamran Sartipi and Nima Dezhkam

IEEE Working Conference on Reverse Engineering (WCRE 2007), Vancouver, Canada Oct. 28-31, 2007, pages 259-268.

(PDF)

This paper proposes a novel approach that amalgamates dynamic and static views of a software system. The dynamic view is represented through profiling information that is extracted from executing a set of task scenarios that cover frequently used soft- ware features. The obtained profiling information is then embedded into a static view recovery process. We propose a pattern based structure recovery, as static view, that defines the high-level architecture of the software system using abstract components and interconnections that is defined using an architecture query language (AQL). In this con- text, both static and dynamic aspects of the software system are used to collect software entities into cohesive components whose dynamic interactions can be controlled. The whole recovery process is modeled as a Valued Constraint Satisfaction Problem (VCSP). A case study with promising results on the Xfig drawing tool has also been presented.

An Orchestrated Multi-view Software Architecture Reconstruction Environment

Kamran Sartipi and Nima Dezhkam and Hossein Safyallah

IEEE International Working Conference on Reverse Engineering (WCRE 2006). Benevento, Italy Oct 23-27, 2006, pages 61-70.

(PDF)

This paper proposes an orchestrated set of techniques and a multi-view toolkit to reconstruct three views of a software system such as design, behavior, and structure. Scenarios are central in generating design and behavior views. The design view is reconstructed by transforming a number of scenarios into design diagrams using a novel scenario schema and generating an object base of actors and actions and their dependencies. The behavior view is represented by different sets of functions that implement different features of the software system corresponding to a set of feature-specific scenarios that are derived from the design view. Finally, the structure view is reconstructed using modules and interconnections that are resulted by growing the core functions related to the software features that are extracted during the behavior recovery. This orchestrated view reconstruction technique provides a more accurate and comprehensive means for reverse engineering of a software system than a single view reconstruction approach. As case studies we applied the proposed multi-view approach on two systems, Xfig drawing tool and Pine email system.

Dynamic Analysis and Design Pattern Detection in Java Programs

Lei Hu and Kamran Sartipi

International Conference on Software Engineering and Knowledge Engineering SEKE'2008
San Francisco Bay, USA. July 1-3, 2008, pages 842-846

(PDF)

Identifying design patterns within an existing software system can support understandability and reuse of the system's core functionality. In this context, incorporating behavioral features into the design pattern recovery would enhance the scalability of the process. The main advantage of the new approach in this paper over the existing approaches is incorporating dynamic analysis and feature localization in source code. This allows us to perform a goal-driven design pattern detection and focus ourselves on patterns that implement specific software functionality, as opposed to conducting a general pattern detection which is susceptible to high complexity problem. Using a new pattern description language and a matching process we identify the instances of these patterns within the obtained classes and interactions. We use a two-phase matching process: i) an approximate matching of class attributes generates a list of candidate patterns; and ii) a structural matching of classes identifies exact matched patterns. One target application domain can be software product line, which emphasizes on reusing core software artifacts to construct reference architecture for several similar products. Finally, we present the result of a case study.

=.=

Static Analysis of Software Systems

Software Architecture Recovery based on Pattern Matching Techniques

Kamran Sartipi

Book: LAMBERT Academic Publishing. 244 pages. ISBN: 978-3-8433-5697-8, 2010.

(Amazon Site)

On Modeling Software Architecture Recovery as Graph Matching

Kamran Sartipi and Kostas Kontogiannis

IEEE International Conference on Software Maintenance (ICSM 2003)
Amsterdam, The Netherlands, Sep 22-26, 2003, pages 224-234.

(PDF)

This paper presents a graph-matching model for the software architecture recovery problem. Because of their expressiveness, the graphs have been widely used for representing both the software system and its high-level view, known as the conceptual architecture. Modeling the recovery process as graph matching is an attempt to identify a sub-optimal transformation from a pattern graph, representing the high-level view of the system, onto a subgraph of the software system graph. A successful match yields a restructured system that conforms to the given pattern graph. A failed match indicates the points where the system violates specific constraints. The pattern graph generation and the incrementality of the recovery process are the important issues to be addressed. The approach is evaluated through case studies using a prototype toolkit that implements the proposed interactive recovery environment.

A User-assisted Approach to Component Clustering

Kamran Sartipi and Kostas Kontogiannis

Journal of Software Maintenance: Research and Practice (JSME), John Wiley Publishers

July/August 2003, vol 15, issue 4, pages 265-295

(PDF)

This paper presents a user assisted clustering technique for software architecture recovery based on a proximity measure that we call component association. The component association measure is computed on the shared properties among groups of highly related system entities. In this approach, the software system is modelled as an attributed relational graph with the software constructs (entities) represented as nodes and data/control dependencies represented as edges. The application of data mining techniques on the system graph allows generating a component graph where the edges are labeled by the association strength values among the components. An interactive partitioning technique and environment is used to partition a system into cohesive subsystems where the graph visualization aids and cluster quality evaluation metrics are applied to assess and fine tune the partition by the user.

A Software Evaluation Model Using Component Association Views

Kamran Sartipi

IEEE International Workshop on Program Comprehension (IWPC 2001)

Toronto, Canada, May 12-14, 2001, pages 259-268

(PDF)

In this paper, we introduce a view-based architectural design evaluation model that allows quantitatively evaluating and categorizing the design of a software system. The model is based on the notion of component association, which is a generalization of coupling and cohesion metrics. The component association is defined as a measure of the overall dependency among high-level system components such as files, modules, or subsystems with regard to a collection of criteria. The associations are discovered by applying data mining techniques on a database of data and control flow dependencies extracted from the software sys- tem. The proposed association-views and modularity metrics allow the user to evaluate the design quality of a soft- ware system.