Computational Intelligence in Software Engineering

www.cs.ubbcluj.ro/~istvanc/master

Hours per week:
Course 2 h/week
Lab 1h/week

Course outline:
1. Search-based Software Engineering
3. Program Comprehension
4. Refactoring
5. Defect Detection and Prediction
6. Software Testing
7. Software Visualization
8. Effort prediction and Cost estimation
9. Software Reuse
10. Design Patterns identification
11. Research reports presentation

Bibliography
Profesional competencies:

- Understanding CI concepts, methods and models
- Understanding the principles of Search Based Software Engineering.
- Learning to conduct incipient original research in Search Based Software Engineering.

Transversal competencies

- The ability to apply computational intelligence techniques in solving software engineering problems.
- Responsible execution of lab assignments, research and practical reports.
- Application of efficient and rigorous working rules.
- Manifest responsible attitudes toward the scientific and didactic fields.
- Respecting the professional and ethical principles.

Objectives

- To introduce the student a new field of Software Engineering- Search Based Software Engineering.
- To induce the necessity and importance of using computational intelligence techniques for solving software engineering problems.
- To present some important activities within software engineering and how are they solved using computational intelligence techniques.
Activity and grading

10%  Class attendance
30%  Theoretical research report (written and presented)
30%  Lab grade
30%  Final exam (written paper in exams session)

Extra session

In the extra session, all activities can be reconsidered, excepting the oral presentations (the maximum grade for a research report is 8).

Research report

Each student will have to prepare and present a theoretical research report on a search based software engineering topic based on some recent research papers: application of machine learning techniques for a software development related activities.

   a) a written paper of about 4 pages
   b) an oral presentation
      - an one-page outline of the presentation

The paper will contain theoretical considerations on the selected topic and advantages and disadvantages of the selected approach (here you can list your own opinions)

Deadline

The title of the theoretical report have to be sent to me by end of week 7.

The presentation will be scheduled for week 10,11,12.
The paper have to fulfill the requirements of a research paper:
   • suggestive title corresponding to the contents;
   • about 10 lines abstract;
   • introductory section, detailing the purpose of the paper;
   • a section integrating the topic of the paper in the general field;
   • a few main sections, according to your topic;
   • concluding remarks and further work section;
   • bibliography of five to 10 titles; the bibliography entries have to be written correctly and completely; all the bibliography items have to be cited in the text;
Lab Grade

Project 1: Download/install an existing open source machine learning framework and create a simple “hello world” application that uses a machine learning technique.
Sample Open source machine learning frameworks:
WEKA http://www.cs.waikato.ac.nz/ml/weka/
Orange http://www.ailab.si/orange/
https://wiki.python.org/moin/PythonForArtificialIntelligence
Numl http://numl.net/
others

Machine Learning techniques (pick one):
• Decision tree learning
• Association rule learning
• Artificial neural networks
• Self organizing maps
• Clustering
• Reinforcement learning
• Bayesian networks
• Support vector machines
• Genetic algorithms

Deadline week 4

Project 2: Apply a machine learning technique to a software engineering related problem

Deadline week 8
Computational Intelligence in Software Engineering

Will investigate the use of search techniques for solving several software engineering problems - search based software engineering.

Search based software engineering is a research and application domain that has many unresearched directions, and many already studied fields

Idea: solving software engineering problems using artificial intelligence techniques, particularly machine learning techniques.
Search Based Software Engineering

Reformulating software engineering problems as search problems.

Metaheuristic search techniques are used in solving different software engineering problems: testing, module clustering, cost estimation, requirements analysis, systems integration, software maintenance and evolution of legacy systems.

Harman and Jones (2001): Software engineering is ideal for the application of metaheuristic search techniques, such as genetic algorithms, simulated annealing and tabu search

Approaching software engineering problems as search problems allows solving these problems using computational intelligence and consequently benefits from the advantages offered by AI.

search-based techniques provide solution to the difficult problem of balancing competing (and sometimes inconsistent) constraints and may suggest ways of finding acceptable solutions in situations where perfect solutions are either theoretically impossible or practically infeasible.
Machine Learning

Machine learning is an important research direction in Artificial Intelligence.

Machine learning explores the construction and study of algorithms that can learn from data.

Machine learning is the study of systems models that, based on a set of data (training data), improve their performance by experiences and by learning some specific domain knowledge.

There are three basic machine learning strategies:

- supervised learning (the learning strategy that is supervised by a teacher)
- unsupervised learning (learning without supervision)
- reinforcement learning (learning by interaction with the environment).
Machine learning techniques

Decision tree learning
Decision tree learning uses a decision tree as a predictive model, which maps observations about an item to conclusions about the item's target value.

Association rule learning
Association rule learning is a method for discovering interesting relations between variables in large databases.

Artificial neural networks
An artificial neural network (ANN) learning algorithm, usually called "neural network" (NN), is a learning algorithm that is inspired by the structure and functional aspects of biological neural networks. Computations are structured in terms of an interconnected group of artificial neurons, processing information using a connectionist approach to computation. Modern neural networks are non-linear statistical data modeling tools. They are usually used to model complex relationships between inputs and outputs, to find patterns in data, or to capture the statistical structure in an unknown joint probability distribution between observed variables.

Support vector machines
Support vector machines (SVMs) are a set of related supervised learning methods used for classification and regression. Given a set of training examples, each marked as belonging to one of two categories, an SVM training algorithm builds a model that predicts whether a new example falls into one category or the other.
Machine learning techniques

**Clustering (Unsupervised classification)**
Cluster analysis is the assignment of a set of observations into subsets (called clusters) so that observations within the same cluster are similar according to some predesignated criterion or criteria, while observations drawn from different clusters are dissimilar. Different clustering techniques make different assumptions on the structure of the data, often defined by some similarity metric and evaluated for example by internal compactness (similarity between members of the same cluster) and separation between different clusters. Other methods are based on estimated density and graph connectivity. Clustering is a method of unsupervised learning, and a common technique for statistical data analysis.

**Reinforcement learning**
Reinforcement learning is concerned with how an agent ought to take actions in an environment so as to maximize some notion of long-term reward. Reinforcement learning algorithms attempt to find a policy that maps states of the world to the actions the agent ought to take in those states. Reinforcement learning differs from the supervised learning problem in that correct input/output pairs are never presented, nor sub-optimal actions explicitly corrected.
Software Engineering

Design, development, and maintenance of software

Application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software

Software development life cycle:
1. Requirement gathering and analysis
2. Design
3. Implementation or coding
4. Testing
5. Deployment
6. Maintenance

Common software development methodologies:
- waterfall
- prototyping
- iterative and incremental development
- feature driven development
- spiral development
- rapid application development
- extreme programming
- agile methodology
- ...
Software Engineering subdisciplines

**Requirements engineering**: The elicitation, analysis, specification, and validation of requirements for software.

**Software design**: The process of defining the architecture, components, interfaces, and other characteristics of a system or component. It is also defined as the result of that process.

**Software construction**: The detailed creation of working, meaningful software through a combination of coding, verification, unit testing, integration testing, and debugging.

**Software testing**: An empirical, technical investigation conducted to provide stakeholders with information about the quality of the product or service under test.

**Software maintenance**: The totality of activities required to provide cost-effective support to software.

**Software configuration management**: The identification of the configuration of a system at distinct points in time for the purpose of systematically controlling changes to the configuration, and maintaining the integrity and traceability of the configuration throughout the system life cycle.

**Software engineering management**: The application of management activities—planning, coordinating, measuring, monitoring, controlling, and reporting—to ensure that the development and maintenance of software is systematic, disciplined, and quantified.

**Software engineering process**: The definition, implementation, assessment, measurement, management, change, and improvement of the software life cycle process itself.

**Software engineering tools and methods**: The computer-based tools that are intended to assist the software life cycle processes (see Computer-aided software engineering) and the methods which impose structure on the software engineering activity with the goal of making the activity systematic and ultimately more likely to be successful.

**Software quality management**: The degree to which a set of inherent characteristics fulfills requirements.
Software engineering activities

Requirements analysis
  – cost/time estimation
  – traceability

Design
  • decompose system into subsystems
  • components, classes, state machines

Coding
  • Implement new functionalities
  • Repair bugs
  • Refactoring
  • Source code repository
  • Continuous integration

Testing
  • create test cases/unit tests
  • integration testing
  • system testing

Deployment
  • configuration management
  • application monitoring

Maintenance
  • program comprehension
  • refactoring/improving overall design
  • fix bugs/create test cases
  • concept location
  • software visualisation
Data available to use

Any artifact produced during the development of a software system.

Design documents
  UML Diagrams- class diagrams, sequence diagrams, use case diagrams etc.
  Functional/architectural/user interface design documents

Source code
  instructions, methods, classes
  comments
  modules, package structure
  test functions, test cases
  source code files (size, creation date/time, modification date/time)

Project management data
  tasks, dependencies, allocation
  task estimated/actual time

Source code repository
  comments entered for checkins
  commit/update date/time, persin, location, team member
  modified/moved/removed files

Configuration/deployment
  config files, build files (ant, maven,
Data we can produce/collect

Software metrics
  method/class/package level metrics

Runtime data
  stack traces
  code coverage
  memory allocation/usage
  error conditions

Project management data
  project velocity

User data
  survev
  user behavior (clicks, selects, navigation)
Computational Intelligence in Software Engineering

We present some existing work in the search-based software engineering literature.

Existing approaches we are focusing:
- Refactoring
- Defect Detection and Prediction
- Object-oriented transformation
- Behavioral adaptation of software systems
- Software Testing
- Software Visualization
- Effort prediction and Cost estimation
- Software Reuse
- Design Patterns identification