RESEARCH PAPER Statistics Volume : 5 | Issue : 2 | Feb 2015 | ISSN - 2249-555X

OPTIMIZATIONOF DYNAMIC COLLABORATION STRUCTURES IN OBJECT-ORIENTED IT GOVERNANCE USING SWARM INTELLIGENCE

KEYWORDS IT Governance, Class Responsibility Assignment, Ant Colony Optimization, Coupling Cohesion

Dr.Carsten Mueller

Faculty of Informatics and Statistics, University of Economics

ABSTRACT *The main goal of Class Responsibility Assignment is to find the optimal assignment of responsibilities (presented in terms of methods and attributes) to classes with regards to dynamic aspects of coupling and cohesion. Ant Colony Optimization supported by an intelligent parameter recommender is successfully applied to solve this optimization problem with the focus on object-oriented IT Governance architecture.*

INTRODUCTION

Information Technology (IT) often entails large capital investments in organizations, while companies are faced with multiple shareholders that are demanding the creation of business value through these investments.IT Governance (ITG) is the degree to which the authority for making IT decisions is defined and shared among management, and the processes managers in both IT and business organizations apply in setting IT priorities and the allocation of IT resources (Luftman, 1996).ITG is deployed using a mixture of various structures, processes and relational mechanisms.

The granularity of an object-oriented ITG architecture and the used classes is important for its usability. If the classes are too large, then cohesion is weak. Consequently, by reusing them, too much functionality is obtained that is not needed at all. On the other hand, if the classes are too small then coupling will be strong and it takes too much effort to assemble an ITG architecture system from them.

In this research paper a solution will be developed to find the optimal assignment of responsibilities with regards to various aspects of coupling and cohesion with the focus on an object-oriented ITG architecture.

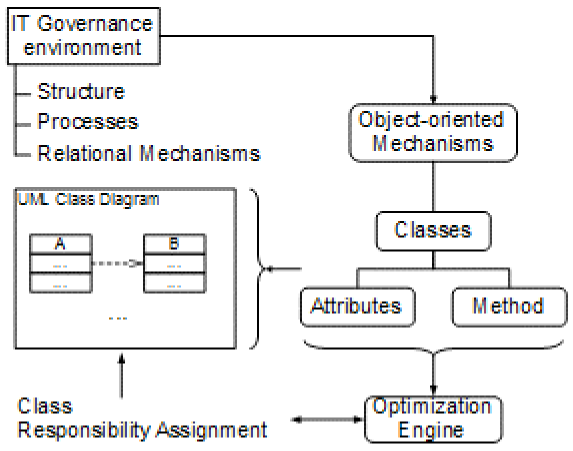


Figure 1: Object-oriented IT Governance

Assigning responsibilities to classes is a vital task in object-oriented analysis and design, and it directly affects the maintainability and reusability of software systems. The development process of object-oriented software involves several steps, in which each step has its own activities. Class Responsibility Assignment (CRA) is one of the most important and complex activities in the context of object-oriented analysis and design (OOAD).Its main goal is to find the optimal assignment of responsibilities (where responsibilities are presented in terms of methods and attributes) to classes with regards to various aspects of coupling and cohesion (Briand, Daly, &Wuest, 1998). It leads to a better maintainable and reusable model (Briand, Daly, &Wuest, 1998).

Several methods exist to recognize the responsibilities of a system (Larman, 2004) and assign them to classes (Bruegge&Dutoit, 2004). They depend on human judgment and decision-making.

Generally, the CRA problem is divided into two major sub problems: assigning responsibilities to classes and setting relationships between these classes. Recently, researches have proposed automated methods to solve these sub problems(Bowman, Briand, &Labiche, 2010; Glavas&Fertalj, 2011;Simons, Parmee, &Gwynllyw, 2010).

RELATED WORKS

Applying search space exploration and optimization to software engineering was proposed by Harman and Jones (Harman & Jones, 2001). In recent years, there has been a dramatic increase in work on Search-Based Software Engineering (SBSE).

SBSE is an approach to software engineering, in which search-based optimization algorithms are used to address problems in Software Engineering (Harman, Mansouri, & Zhang, 2009). The focus of most researches in this area is on software testing (Harman, Mansouri, & Zhang, 2009).

In the following the most important studies in the field of search-based software design are presented. Recently, researches have used meta-heuristic optimization algorithms to automate the object-oriented software design. Bowman et al. (Bowman, Briand, &Labiche, 2010) and Engelbrecht(Engelbrecht, 2007) used multi-objective genetic algorithm (MOGA) to solve the CRA problem. Their study used the coupling and cohesion metrics as fitness function and the Strength Pareto Approach (SPEA2) as MOGA algorithm (Zitzler, Laumanns, & Thiele, 2001). Bowman et al. (Bowman, Briand, &Labiche, 2010) also com-

Volume : 5 | Issue : 2 | Feb 2015 | ISSN - 2249-555X

pared MOGA to other search algorithms, such as Random Search (RA), Hill Climbing (HC), and a simple Genetic Algorithm (GA), and concluded that a more complex algorithm is needed to solve the CRA problem. Glavas and Fertalj(Glavas&Fertalj, 2011) used four different heuristic and metaheuristic optimization algorithms, i.e., GA, HC, Simulated Annealing (SA), and Particle Swarm Optimization (PSO) to solve the CRA problem. They used Responsibilities Dependency Graph (RDG) as input of optimization algorithms and the coupling and cohesion metrics as their fitness function.

APPROACH

Applying search space exploration and optimization to software engineering was proposed by Harman and Jones (Harman & Jones, 2001). In recent years, there has been a dramatic increase in work on Search-Based Software Engineering (SBSE).

A. Ant Colony Optimization

In this research a dynamic model based on Ant Colony Optimization (ACO) is proposed to solve the CRA problem. The proposed model has four main phases. In the first phase, the responsibilities (attributes and methods) and their dependencies are collected. In the second phase, features are normalized based on dependencies between responsibilities. In the third phase, ACO techniques are used to determine responsibilities. Finally, in the fourth phase, the relationships among classes are set and an optimized class diagram is generated.

ACO is a recently proposed metaheuristic approach for solving hard combinatorial optimization problems (Dorigo, Birattari, &Stutzle, 2006). The inspiring source of ACO is the pheromone trail laying and following behavior of real ants which use pheromones as a communication medium. In analogy to the biological example, ACO is based on the indirect communication of a colony of simple agents, called (artificial) ants, mediated by (artificial) pheromone trails.

The pheromone trails in ACO serve as distributed, numerical information which the ants use to probabilistically construct solutions to the problem being solved and which the ants adapt during the algorithm's execution to reflect their search experience.

Artificial ants used in ACO are stochastic solution construction procedures that probabilistically build a solution by iteratively adding solution components to partial solutions. These procedures take into account (i) heuristic information on the problem instance being solved, if available and (ii) (artificial) pheromone trails which change dynamically at runtime to reflect the agents' acquired search experience (Dorigo, Bonabeau, &Theraulaz, 2000).

A stochastic component in ACO allows the ants to build a wide variety of different solutions and hence explore a much larger number of solutions than greedy heuristics. At the same time the use of heuristic information guides the ants target-oriented towards the most promising solutions.

A colony of ants concurrently and asynchronously moves through adjacent states of the problem by building paths on the graph. They move by applying a stochastic local decision policy that makes use of pheromone trails and heuristic information.

By moving, ants incrementally build solutions to the optimization problem. Once an ant has built a solution, or while the solution is being built, the ant evaluates the (partial) solution and deposits pheromone trails on the components or connections it used. This pheromone information will affect the search of the future ants.

Besides ants' activity, an ACO algorithm includes two more procedures: pheromone trail evaporation and daemon actions (this last component being optional). Pheromone evaporation is the process by means of which the pheromone trail intensity on the components decreases over time. From a practical point of view, pheromone evaporation is needed to avoid a too rapid convergence of the algorithm towards a suboptimal region. It implements a useful form of forgetting, favoring the exploration of new areas of the search space. Daemon actions are used to implement centralized actions which cannot be performed by single ants.

Algorithm 1 Ant Colony Optimization

```
while termination conditions not met do

AntBasedSolutionConstruction()

PheromoneUpdate()

DaemonActions() {optional}

end while
```

In this research article artificial ants are regarded as probabilistic constructive heuristics that assemble solutions as sequences of solution components. The finite set of solution componentsis derived from CRA optimization problem.

Respective probabilities - also called transition probabilities - are defined as follows [11]:

$$p(c_i|s) = \frac{[\tau_i]^{\alpha} \cdot [\omega(c_i)]^{\beta}}{\Sigma_{c_j \in N(s)} [\tau_j]^{\alpha} \cdot [\omega(c_j)]^{\beta}}, \forall c_i \in N(s) \quad (1)$$

where is an optional weighting function depending on thecurrent sequence, assigns at each construction step a heuristicvalueto each feasible solution component.The values that are given by the weighting function are calledthe heuristic information.

The exponentsandare positiveparameter values that determine the relation between pheromoneinformation and heuristic information.

The behavior of ACO is significantly determined by the parameters and.

- $\alpha > \beta$: choice of path is more influenced by the knowledge of the ants,
- $\alpha = \beta$: choice of path is equally influenced by both pheromone and heuristic,
- $\alpha = \beta$: ants ignore their knowledge and choose the path only regarding the heuristic.

The pheromone update rule consists of two parts.

First, a pheromone evaporation, which uniformly decreas-

Volume : 5 | Issue : 2 | Feb 2015 | ISSN - 2249-555X

es all the pheromone values, is performed. Second, one or more solutions from the current and/or from earlier iterations are used to increase the values of pheromone trail parameters on solution components that are part of these solutions (Dorigo, Birattari, &Stutzle, 2006):

$$\tau_i \leftarrow (1-\rho)\cdot\tau_i + \rho\cdot \sum_{\{s\in s_{update}|c_i\in s\}} w_s \cdot F(s) \quad (2)$$

$$;i = 1,\cdots,n$$

S_{update} denotes the set of solutions that are used for the update. is the pheromone evaporation rate and is the objective function.

B. Architecture

The implementation of this approach follows the Object-oriented Architectural Style and 3-Tier Architectural Style.

The responsibilities for the application are divided into individual reusable and self-sufficient objects, each containing the data and the behavior relevant to the object. The system is viewed as a series of cooperating objects. These objects are discrete, independent, loosely coupled and they communicate through interfaces by calling methods and sending and receiving messages. In addition, the implementation is characterized by the functional decomposition into the layers presentation, logic and persistence. HyperSQL Database (HSQLDB) is a relational database system and used as repository for diagrams. HSQLDB runs entirely in memory using dedicated fast memory structures for the applied ACO.

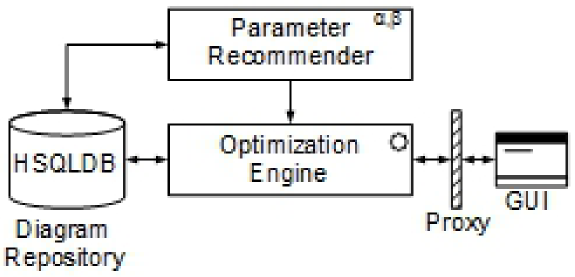


Figure 2: Architecture

Objective of the parameter recommender is the intelligent determination of best suitable values of the ACO parameters and concerning solution quality regarding the data instance stored in the HSQLDB.

C. Cohesion

The cohesion for the class diagram is the arithmetic mean cohesion of the classesin a diagram.

$$coh(D) = \frac{1}{|D|}\cdot\sum_{i=0}^{|D|}\left(\frac{1}{|D_{iM}|\cdot|D_{iA}|}\cdot\sum_{j=0}^{|D_{iM}|}\left|D_{iMA_j}\right|\right) \quad (3)$$

where

$\lvert D\rvert$	:number of classes in class diagram
$\lvert D_{iM}\rvert$	: number of methods for class
$\lvert D_{iA}\rvert$	: number of attributes for class
$\lvert D_{iMA_j}\rvert$	: number of referenced attributes

by the methodfor class

D. Coupling

The coupling () between two classesandis the ratio ofthe existing method relationships and the number of all possiblerelations. Two classesandwith a one-way relation in thecase of only class, using the method(s) of class, havethe same rate of coupling, disregarding the order of the classes.Formally, the function that calculates the coupling between twoclasses and is commutative.

$$cpl(D_i, D_j) = \frac{\sum_{k=0}^{|D_{iMM}|}\left|D_{iMM_k}\cap D_{jM}\right|}{|D_{iM}|\cdot|D_{jM}|} \quad (4)$$

For determining the coupling for a whole class diagram, the arithmetic mean of the coupling between all possible pairs of classes within the diagram is calculated.

$$cpl(D) = \frac{1}{|D|^2-|D|}\cdot\sum_{i=0}^{|D|}\sum_{j=0,j\neq i}^{|D|}\frac{\sum_{k=0}^{|D_{iMM}|}\left|D_{iMM_k}\cap D_{jM}\right|}{|D_{iM}|\cdot|D_{jM}|} \quad (5)$$

Since Formula 4 is commutative Formula 5is reduced

$$cpl(D) = \frac{1}{\binom{|D|}{2}}\cdot\sum_{i=0}^{|D|-1}\sum_{j=i+1}^{|D|}\frac{\sum_{k=0}^{|D_{iMM}|}\left|D_{iMM_k}\cap D_{jM}\right|}{|D_{iM}|\cdot|D_{jM}|} \quad (6)$$

E. Objective and ACO algorithm

High cohesion and low coupling of classes are important indicators for maintainable and future-oriented software architecture. From the research perspective in IT Governance and the view on software architecture it is not commonly preferable to focus just on maximizing cohesion and minimizing coupling.

Based on the current situation the software architect specifies dynamically the values for coupling and cohesion. Two variables and are used to represent the target cohesion and target coupling.

The aim of ACO is to minimize the difference between the actual coupling and the user-specified coupling, as well as the difference between the actual cohesion and the user-specified cohesion.

Based on Formulas 3 and 4 the Formula 7 is stated as

$$dev(D) = |tcoh - coh(D)| + |tcpl - cpl|D|| \quad (7)$$

The co-domains of both Formulas3 and 4are bound to theclosed interval, just as and .

With the user-specified settingand(see Figure2) the special case for minimization of coupling and maximization of cohesion as previously described is obtained.

The objective of the proposed ACO algorithm regarding CRA is denoted as

min(dev(D)) (8)

Algorithm2Ant Colony Optimization for CRA

```
initialize empty class diagramwith an empty class
while assign. attributes and/or methods not empty do
for all existing classes indo
```

Volume : 5 | Issue : 2 | Feb 2015 | ISSN - 2249-555X

```
calculate prob. and store in roulette wheel
end for
add probabilities to a new class to roulette wheel
end while
roulette wheel selection and extend

calculate fitness of the diagram using Formula 7

update (pheromone) the edges in
```

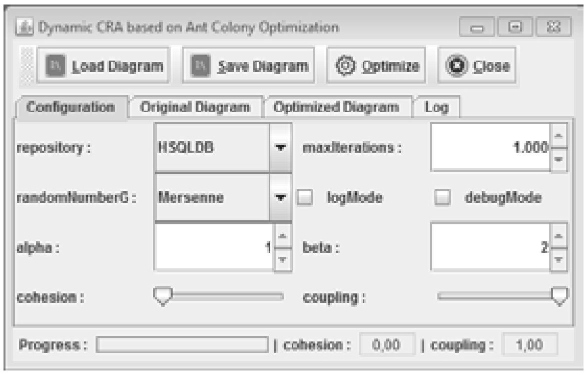
To compute the certainty of an assignment the quality of theactual class diagramis compared to the fitness of the samediagram with the addition already performed on it, named.

$$\omega(D, D^*) = \frac{dev(D) - dev(D^*) + 2}{4} \qquad (9)$$

The probability is then computed as:

$$\rho = \left(\tau_{eD_i}\right)^{\alpha} \cdot \omega(D, D^*)^{\beta} \qquad (10)$$

whereis the pheromone on the edge between an elementand a class. The pheromone value is increased by the resultof Formula7if the specific edge is used in the diagram.



repository : HSQLDB maxIterations : 1.000
randomNumberG : Mersenne logMode debugMode
alpha : 1 beta : 2
cohesion : coupling :
Progress : | cohesion : 0,00 | coupling : 1,00

Figure 3: Dynamic CRA - Graphical User Interface

PRACTICAL EXAMPLE

In this practical example three classes Strategy () Audit () and Change are given.is the-th attribute andis the-th method.Table I presentsthe structure of the classes regarding attributes and methods.

TABLEI
STRUCTURE OF THE CLASSES

name	A	B	C
attributes	a_1 a_2	a_1 a_2	a_1 a_2
methods	m_1 m_2	m_1 m_2 m_3	m_1 m_2 m_3

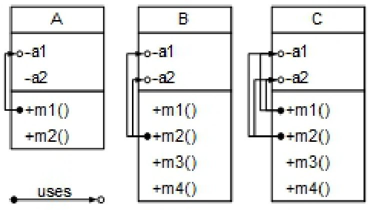


Figure 4: Method-Attribute Relationship
TABLEII
METHOD-ATTRIBUTE RELATIONSHIP

name	A	B	C
x uses $(\rightarrow)$ y	$m_1 \rightarrow (a_1)$ $m_2 \rightarrow ()$	$m_1 \rightarrow ()$ $m_2 \rightarrow (a_1, a_2)$ $m_3 \rightarrow ()$ $m_4 \rightarrow ()$	$m_1 \rightarrow (a_1, a_2)$ $m_2 \rightarrow (a_1, a_2)$ $m_3 \rightarrow ()$ $m_4 \rightarrow ()$

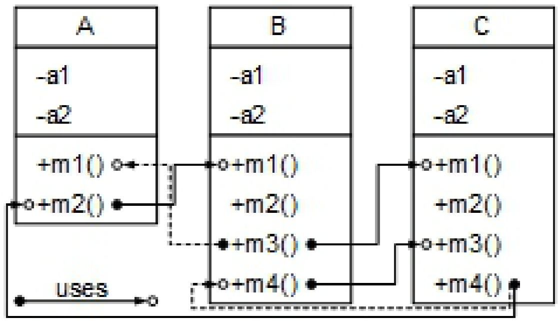


Figure 5: Method-Method Relationship

A. Cohesion

TABLEIII
COHESION: $|D_{i_M}|$ AND $|D_{A_M}|$

name	A	B	C
$\lvert D_{i_M}\rvert$	$\lvert D_{0_M}\rvert = 2$	$\lvert D_{1_M}\rvert = 4$	$\lvert D_{2_M}\rvert = 4$
$\lvert D_{i_A}\rvert$	$\lvert D_{0_A}\rvert = 2$	$\lvert D_{1_A}\rvert = 2$	$\lvert D_{2_A}\rvert = 2$

Cohesion – Class:

$$coh(D_0) = \frac{1}{|D_{0M}||D_{0A}|} \cdot \sum_{j=1}^{|D_{0M}|} |D_{0MA_j}| \quad (11)$$

$$D_{0MA} = \{ ([illegible]) \cdot ([illegible]) \} \quad (12)$$

$$\sum_{j=1}^{|D_{0M}|} |D_{0MA_j}| = |D_{0MA_1}| + |D_{0MA_2}| = 1 + 0 = 1 \quad (13)$$

$$coh(D_0) = \frac{1}{2 \cdot 2} \cdot 1 = \frac{1}{4} \quad (14)$$

Cohesion – Class B (D_1):

$$coh(D_1) = \frac{1}{|D_{1M}||D_{1A}|} \cdot \sum_{j=1}^{|D_{1M}|} |D_{1MA_j}| \quad (15)$$

$$D_{1MA} = \{ ([illegible]) \cdot ([illegible]) \cdot ([illegible]) \cdot ([illegible]) \} \quad (16)$$

$$\sum_{j=1}^{|D_{1M}|} |D_{1MA_j}|$$

Volume : 5 | Issue : 2 | Feb 2015 | ISSN - 2249-555X

$$\sum_{j=0}^{|D_{2M}|} \left|D_{2MA_j}\right| =$$

$$\left|D_{2MA_0}\right| + \left|D_{2MA_1}\right| + \left|D_{2MA_2}\right| + \left|D_{2MA_3}\right| =$$

$$2 + 2 + 0 + 0 = 4 (21)$$

$$coh(D_2) = \frac{1}{4 \cdot 2} \cdot 4 = \frac{1}{2} (22)$$

$$coh = \frac{1}{|D|} \cdot \sum_{i=0}^{|D|} \left(\frac{1}{|D_{iM}| \cdot |D_{iA}|} \cdot \sum_{j=0}^{|D_{iM}|} \left|D_{iMA_j}\right| \right)$$

$$= \frac{1}{3} \cdot \left(\frac{1}{4} + \frac{1}{4} + \frac{1}{2}\right) = \frac{1}{3} (23)$$

B. Coupling

Coupling $A \leftrightarrow B$

$$cpl(A,B) = \frac{\sum_{k=0}^{|D_{0MM}|} \left|D_{0MM_k} \cap D_{1M}\right|}{|D_{0M}| \cdot |D_{1M}|}$$

$$= \frac{|\{B:m_4\} \cap D_{1M}| + |\{B:m_1, C:m_4\} \cap D_{1M}|}{2 \cdot 4} = \frac{1+1}{8} = \frac{1}{4} (24)$$

$$\text{where } D_{1M} = \{B:m_1, B:m_2, B:m_3, B:m_4\}$$

Coupling $A \leftrightarrow C$

$$cpl(A,C) = \frac{\sum_{k=0}^{|D_{0MM}|} \left|D_{0MM_k} \cap D_{2M}\right|}{|D_{0M}| \cdot |D_{2M}|}$$

$$cpl(A,C) = \frac{|\{B:m_3\} \cap D_{2M}| + |\{B:m_1, C:m_1\} \cap D_{2M}|}{2 \cdot 4} = \frac{1}{8} (25)$$

$$\text{where } D_{2M} = \{C:m_1, C:m_2, C:m_3, C:m_4\}$$

Coupling $B \leftrightarrow C$

$$cpl(B,C) = \frac{\sum_{k=0}^{|D_{1MM}|} \left|D_{1MM_k} \cap D_{2M}\right|}{|D_{1M}| \cdot |D_{2M}|}$$

$$cpl(B,C) = \frac{|\{B:m_3\} \cap D_{2M}| + |\{B:m_1, C:m_4\} \cap D_{2M}|}{4 \cdot 4} = \frac{1}{16}$$

(36)

$$\text{where} D_{2M} = \{C:m_1, C:m_2, C:m_3, C:m_4\}$$

$$cpl(D) = \frac{1}{|D|} \cdot \left(\frac{1}{4} + \frac{1}{8} + \frac{1}{16}\right) = \frac{1}{3} \cdot \frac{9}{16} = 0.1875 (37)$$

$$cpl(B,C) = \frac{\sum_{k=0}^{|D_{1MM}|} \left|D_{1MM_k} \cap D_{2M}\right|}{|D_{1M}| \cdot |D_{2M}|}$$

$$cpl(B,C) = \frac{|\{B:m_3\} \cap D_{2M}| + |\{B:m_1, C:m_4\} \cap D_{2M}|}{4 \cdot 4} = \frac{1}{16}$$

(36)

$$\text{where} D_{2M} = \{C:m_1, C:m_2, C:m_3, C:m_4\}$$

$$cpl(D) = \frac{1}{|D|} \cdot \left(\frac{1}{4} + \frac{1}{8} + \frac{1}{16}\right) = \frac{1}{3} \cdot \frac{9}{16} = 0.1875 (37)$$

CONCLUSIONS

In this research paper a flexible optimization architecture for solving the CRA in the context of ITG Architecture was successfully developed based on ACO. Additionally a parameter recommender is used to optimize the behavior of ACO regarding different class diagrams stored in the memory-based HSQLDB.

Future steps are developing a framework with additional metrics and further research in parameter settings and heuristic dependencies.

Updates are found on http://www.itg-research.net.

Volume : 5 | Issue : 2 | Feb 2015 | ISSN - 2249-555X

REFERENCE Bowman, M., Briand, L., &Labiche, Y. (2010). Solving the class responsibility assignment problem in object-oriented analysis with multi-objective Genetic Algorithms, IEEE Transactions on Software Engineering, 36(6), 817–837. | Briand, L., Daly, J., &Wuest, J. (1998). A unified framework for cohesion measurement in object-oriented systems, Empirical Software Engineering, 3(1), 65–117. | Bruegge, B., &Dutoit, A. (2004). Object-Oriented Software Engineering using UML, Patterns and Java. Prentice Hall. | Dorigo, M., Birattari, M., &Stutzle, T. (2006). Ant Colony Optimization, Computational Intelligence Magazine, 1, 28–39. | Dorigo, M., Bonabeau, E., &Theraulaz, G. (2000). Ant algorithms and stigmergy, Future Generation Computing Systems, 16(8), 851–871. | Engelbrecht, A. (2007). Computational Intelligence: An Introduction. John Willey. | Glavas, G., &Fertalj, K. (2011). Metaheuristic approach to class responsibility assignment problem, Information Technology Interfaces, 591-596. | Luftman, J. (1996). Competing in the Information Age. Oxford University Press. | Harman, M., & Jones, B. (2001). Search-based software engineering, Information and Software Technology, 43(14), 833–839. | Harman, M., Mansouri, S., & Zhang, Y. (2009). Search based software engineering: A comprehensive analysis and review of trends, techniques and applications, Technical Report 09-03, London. | Larman, C. (2004). Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development. Prentice Hall. | Simons, C., Parmee, I., &Gwynllyw, R. (2010). Interactive, evolutionary search in upstream object-oriented class design, IEEE Transactions on Software Engineering, 36(6), 798–816. | Zitzler, E., Laumanns, M., & Thiele, L. (2001). SPEA2: Improving the strength pareto, Evolutionary Algorithm, 103, 95-100. |