STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LVIII**, Number 1, 2013

BUILDING AN AUTOMATED TASK DELEGATION ALGORITHM FOR PROJECT MANAGEMENT AND DEPLOYING IT AS SAAS

BOGDAN POP

Abstract. A large number of project management applications currently exist, some focusing on certain features or domains, while others are designed to work in multiple scenarios and domains. Task delegation and resource allocation is one of the crucial parts of project management. Errors in this aspect can result in significant loss of time, resources and deficient project results. No applications currently available on the market automate task delegation; therefore the entire management of the project relies on human experience and is subject to errors. This paper presents the characteristics of a modern, web based, software as a service project management application that focuses on a revolutionary approach to task delegation with the automation of the process. The goal is to reduce, if not eliminate, the necessity of the project manager and to minimize costs and maximize resource usage.

1. Introduction

Project management is the practice of organizing, planning and managing resources in order to achieve specific goals [9]. 27% of a manager's time and more than USD 100 billion is spent each year correcting or working with employees that are not suited for their tasks [11]. Currently, there are more than a few hundred project management software applications, each tackling different aspects of project management. This paper presents a concept application that automatically assigns without human intervention newly created tasks within a project, thus minimizing costs and maximizing resource usage to its fullest. The paper presents the database model, the base algorithm and a deployment scenario as software as a service of the aforementioned concept application. The model tries to provide an algorithm that would reduce, if not

Received by the editors: October 30, 2012.

2010 *Mathematics Subject Classification.* 68M14.

1998 *CR Categories and Descriptors.* C.2.4 Computer Systems Organization [Computer-Communication Networks]: Distributed Systems – *Distributed applications*.

Key words and phrases. nosql, project management, web, application, saas.

eliminate the role of a project manager as known today, allowing the usage of the project manager's knowledge for actual development.

This paper is structured as follows. Section 2 presents some of the most used and known project management applications and some of their main features. Section 3 describes the application concept, its base features and the information flow within those features. Section 4 presents some pros and cons for relational and non-relational databases with regard to current use case and presents why a NoSQL approach is suited for the proposed concept. The 5^{th} section of the paper presents the non-relational database model of the most important data structures used by the application and the algorithm used to automatically assign newly created tasks to users. The 6^{th} section of the paper presents the model that can be used to extend the base application and deploy it as Saas (Software as a Service). The 7^{th} and last section of the paper presents conclusions found as well as future extending possibilities.

2. Project Management Applications Classification

There are a number of factors that can be taken into account while making any classifications for project management applications. Applications can be classified based on the problem they are trying to solve: CRM (client relationship management), collaboration, SCM (supply chain management), bug tracking, issue management or time management. Others can be classified based on the platform or platforms they can run on, such as web, mobile, desktop, or hybrid applications. Some project management applications contain features for dealing with a bundle of the aforementioned problems. Some applications are designed for specific tasks and niches: managing personnel contacts, chatting or sharing scheduling tools. Project management applications can also be classified based on their licensing system, as follows: proprietary, either hosted on premises or not, open source, or SaaS.

2.1. **Project management applications by their licensing system.** Some established open source project management applications include *Trac*, *Launchpad*, *Redmine* or *MantisBT*. *Trac* uses a minimalistic approach to web-based software project management and it is an issue tracking system tailored towards software development projects. *Launchpad* is a collaboration platform that focuses on bug tracking and offers code hosting and reviews, mailing lists, answer tracking or frequently asked questions. It offers a timeline that shows all current and past project events in order, complemented by a roadmap feature that displays future milestones, therefore allowing project managers to easily grasp an overview of the project's progress. *Redmine* combines features from *Launchpad* and *Trac* but it also provides a time tracking feature that supports manual input at project and ticket level.

The most known proprietary project management applications are probably *JIRA*, *Kayako* or *Microsoft Project*. *Kayako*'s main focus is on customer support. It is widely used by web hosting companies of all sizes: resellers to end-point users up to multinational datacenter operating companies. *JIRA* is a project management application mainly used for bug and issue tracking while *Microsoft Project* is designed to assist a project manager in analyzing workloads, tracking progress, managing budgets, assigning tasks and resources.

The most popular applications are probably, due to their distribution channel, the ones offered as services. Some of these software as a service applications are *Basecamp*, *Mavenlink* or *Assembla*.

There are also applications that combine proprietary free and paid solutions. Some have multiple components and only a couple of the central ones require a subscription or a one time fee, leaving the rest of the components free of charge. Two applications that fall within this category are *TeuxDeux* and *getFlow*. Other applications have a reduced lite free version that does not contain all the features of the full paid version. This licensing system is most common in smaller applications that have borrowed their licensing system from the modern application stores designed for mobile devices.

2.2. **Project management applications by their availability platform.** Some project management applications are designed for desktop usage and therefore must be installed on client premises. Some of these applications are *Microsoft Project*, *ConceptDraw Project* or *OmniPlan*. *Kayako* is available as a service on demand or as a downloadable, installable software.

The applications that have seen the largest growth [2] in the past years are the web based ones, due to their high availability, ease of use and lower costs. As they are easier to use, these applications allow for faster and more rapid reactions, which is a key aspect of a successful project management process as described by [5].

JIRA can be used either hosted or installed on the premises. Furthermore, it can be extended through a number of different proprietary applications and services, such as *Confluence*, *FishEye*, *Crucible*, *GreenHopper*, *Bonfire*, *Bonfire*. These extend *JIRA*'s purpose with online wikis, *Subversion*, *Git*, *Mercurial* and *CVS* integration, code reviewing, sprint planning and task tracking for agile teams. They also allow browser screen captures and sharing that facilitates easier tracking, annotating and testing activities.

Basecamp is a project management hybrid system tracking project discussions, files, and events from beginning to end in one place. *Basecamp* has the capability of displaying information about hundreds of projects in a single page including teams, clients, contractors, and vendors. The system can be used to follow along projects development in real-time or by later reviewing

what has happened. *Basecamp* design is simple and easy to use and is bundled with visual timelines, complex access permissions with an ultimate goal to bring responsibility and accountability to the projects managed through it and individuals working on them.

3. Application Concept

As presented in section 2, most of the project management applications require human intervention as they only assist in analyzing workloads, tracking progress, managing budgets, assigning tasks or resources. They do not offer any kind of automation with regards to task management and time management. However, the concept application presented in this paper focuses on an entirely different approach to project management. That is automating project management and eliminating the role of an actual project manager as much as possible.

In order to achieve the highest automation possible, with respect to task and time management, the proposed model has to store information regarding tasks, their types, the people that have completed them, the time required to complete them, personnel availability for future tasks and more. By storing this information one can programmatically determine the best match for a newly added task. By automating this process, the project manager is no longer required and can be part of the actual development team as opposed to only leading it. Moreover, programmatically assigning tasks results in fewer errors compared to those made by a human project manager. Therefore, the development costs and time required to complete projects is reduced to minimum.

The proposed application will show a maximum level of transparency towards clients, allowing access to all information in real time. In order to achieve high transparency and high availability of the entire system and data it withholds, a web application, a hybrid web and mobile system or an internet distributed system is proposed.

Out of the three options, the hybrid web and mobile system is likely the best option. First, it can be developed in two phases, initially with a full web application and secondly with a corresponding mobile application, as it has been the case with numerous successful project management solutions. Secondly, the internet distributed model is prone to bigger costs due to development of multiple applications and systems on different platforms which require different technologies, teams and are harder to maintain.

Apart from the development issues that would arise by the development of a distributed, multi-platform system, another issue could put costs even higher: keeping data consistent while converting it from one format to another.

This would be required given each of the composing platform's characteristics, communication protocols, storage devices and other constraints.

Developing automation of the aforementioned problems can be achieved using a number of evolutionary algorithms, neural networks or swarm intelligence algorithms. However, since the application is likely to store vast amount of information, such algorithms are likely to require a lot of computational time. Given the fact that time management is critical in project management, they may not be best suited for large project management applications.

A different approach, based on Gale-Shapely Courtship Algorithm [1], has been described. A task that has not been attached to a worker selects the worker with the highest preference. If the chosen worker prefers the task to their current task and can finish on time, he or she accepts the task. If the current task is replaced, then it becomes unattached and a callback is placed to attach it again to another worker. The process repeats itself until all tasks have been attached or rejected by the workers [7].

However, although the proposed algorithm has been theoretically proven, no case studies were performed on business groups and it has not been tested in practice. Proper feedback avenues are to be established with testing groups to utilize the mechanisms described in the paper [7].

The proposed model uses basic iterative algorithms to achieve automation by properly storying, sorting and indexing all data available on the project and having it easily accessible from distributed databases. The proposed concept application uses a non-relational database system (NoSQL) for storying the data and automating the process.

4. NoSQL vs SQL

This section presents the database model of the concept application presented in the paper and why a NoSQL approach is better in this scenario. A number of properties, such as atomicity, consistency, isolation, durability (A.C.I.D.) and at a number of guarantees, such as consistency, availability and partition tolerance will be taken into consideration. The three guarantees are known as the CAP or Brewer's theorem [3].

Key properties and guarantees that must be met by the database storage system that the application will use are atomicity, durability, delayed-T consistency, availability and partition tolerance.

In a highly available, partition tolerant system, results are always returned to clients. However, the information returned may be the latest value written in a node A_1, the latest value written in a node A_1 and read from a secondary node A_i $(i \neq 1)$, or a cached value that hasn't expired yet, read from any node A_i $(i = 2, ..., n)$ of the system [10].

Isolation is not a priority requirement since strong consistency cannot be met. No isolation allows two distinct processes to alter the same value in different nodes, which is a potential cause for incorrect information. However, after the set delay time has passed and no new updates are made to the data, the information it withholds becomes consistent based on timestamp ordering. This holds if transactions that started earlier but finished late have a higher priority than those transactions that started later but altered same data sectors sooner. As far as durability is considered, data are not lost, even in the eventuality of a hardware crash since most NoSQL systems are fault tolerant and decentralized.

Before choosing one database system or the other, the application requirements must be presented. First of all, the proposed application will run as software a service and may have a mobile component too. A software as a service application has a huge growth potential and may be required to store vast amount of information from multiple clients: tasks, projects, users, roles, permissions. Hypothetically 20.000.000 tasks, 250.000 users, 30.000 roles, 15.000.000 permissions would be stored if a project's median characteristics were 200 tasks, 10 users, 3 roles, and 150 permissions, with the service hosting 100.000 projects from 25.000 organizations.

Storing so much information on a single SQL server would be difficult, and with high hardware costs. Data could be partitioned across multiple databases and multiple datacenters: users in one server, tasks in other server etc. This would result in increased complexity of the algorithms that access the data. On the other hand, in a NoSQL system, data partitioning can be achieved automatically with little effort, and data accessing does not change as it is abstracted by the NoSQL database system.

SQL systems guarantee information consistency, not high availability as required by the proposed application. Obtaining high availability in SQL database systems is difficult and subject to many failures: communication between nodes or node failures, replication errors etc. With Apache Cassandra however, data is automatically replicated to multiple nodes for fault-tolerance. Replication across multiple data centers is also supported and crashed nodes can be replaced with no downtime. Furthermore, it is decentralized and has no single points of failure and no network bottlenecks. Having these features, Cassandra meets partition tolerance property, which guarantees high availability [8].

5. Database Model and Automation Algorithm

The base model of the application described in section 3 consists of 8 data structures, modeled as columns, column families and super columns. Being a

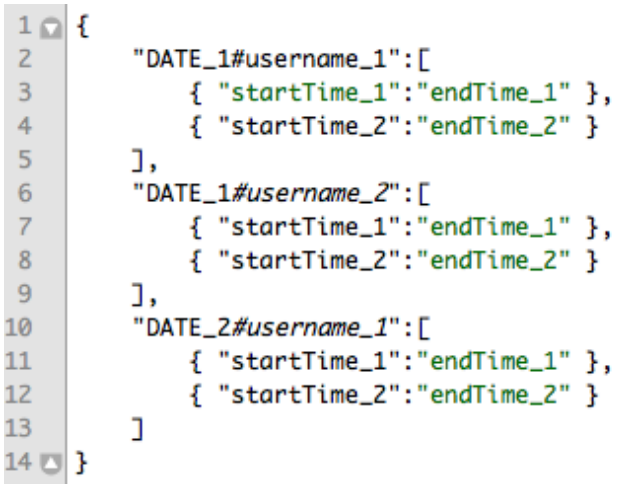
```
{
    "task name": {
        "keywords":
        [
            { "keyword_name_1": "optional_priority" },
            { "keyword_name_2": "optional_priority" }
        ]
    },
    "task details":[
        { "username": "username value" },
        { "project": "project name" },
        { "timeToComplete": "time" },
        { "completed": "date time" },
        { "deadline": "date time" },
        { "finishedOn": "date time" },
        { "assignedOn": "date time" }
    ]
}
```

FIGURE 1. JSON representation of the Tasks super column

web service the application requires a usernames structure. The users super column contains key value pairs for storying emails, passwords, encryption salts, usernames, contact information, a column family for permissions etc. The tasks super column contains string key value pairs for usernames, project names, time required to complete the tasks, whether the tasks are completed or not, date/time values for deadlines, assignation times or when the tasks were finished, as well as a column family that contains descriptive keywords for each of the tasks. Figure 1 displays the most basic representation of the tasks super column in JSON format.

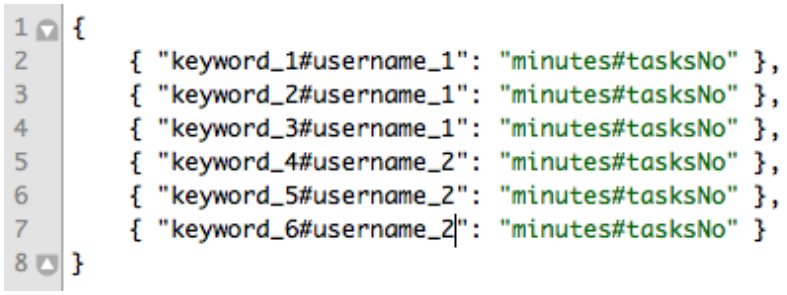
A user availability column family is also required. It has key value pairs as follows: keys consisted of a string containing the calendar date and a username while the adjacent value holds a column with key name as start time and value as end time. Figure 2 displays the JSON representation of the user availability column family. Project information can be modeled as either a column family or a super column, depending on the information stored. Timelogs are used to store tasks IDs or names, the users working on those tasks and the hours worked on that specific task [10].

The keywordTimeToComplete column family stores a median value of the time required to complete a task by each user. The key name is composed of



```
{
    "DATE_1#username_1":[
        { "startTime_1":"endTime_1" },
        { "startTime_2":"endTime_2" }
    ],
    "DATE_1#username_2":[
        { "startTime_1":"endTime_1" },
        { "startTime_2":"endTime_2" }
    ],
    "DATE_2#username_1":[
        { "startTime_1":"endTime_1" },
        { "startTime_2":"endTime_2" }
    ]
}
```


FIGURE 2. JSON representation of the userAvailability column family



```
{
    { "keyword_1#username_1": "minutes#tasksNo" },
    { "keyword_2#username_1": "minutes#tasksNo" },
    { "keyword_3#username_1": "minutes#tasksNo" },
    { "keyword_4#username_2": "minutes#tasksNo" },
    { "keyword_5#username_2": "minutes#tasksNo" },
    { "keyword_6#username_2": "minutes#tasksNo" }
}
```


FIGURE 3. JSON representation of the keywordTimeToComplete column family

the keyword and the username while the value is composed of the work hours required in minutes and the number of similar tasks the user has worked on. Figure 3 displays this column family in JSON format. This column family together with the following two column families, which are presented in figures 4 and 5, are the basis of the task assignment automation algorithm.

The taskAssign column family stores the following information in the column key name: they keyword, its score and the number of tasks this keyword has been assigned to. The value of the column holds the username that has the certain score for the given keyword. The userScores column family is similar to the previously presented one but it is used to store, track and modify individual user's scores. The key names are composed of a combination of a keyword and a username, while the value holds the score and the number of

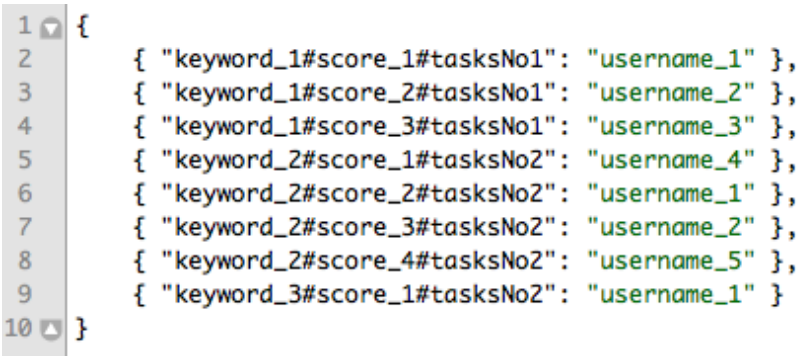


FIGURE 4. JSON representation of the taskAssign column family

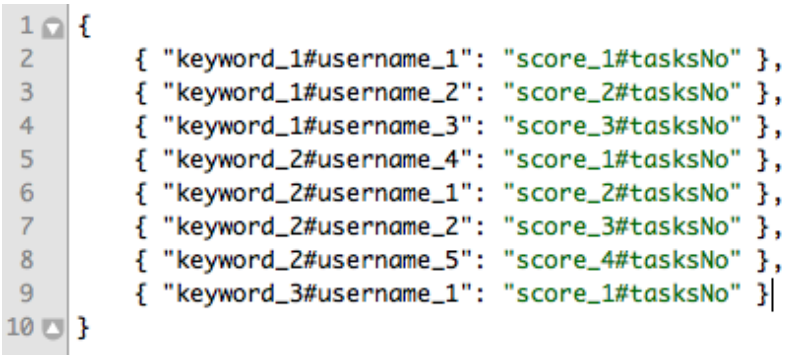


FIGURE 5. JSON representation of the userScores column family

tasks containing the given keyword the user has completed in the past. Whenever one user completes a task, these two columns are updated so that they show the average score per keyword for each user. If two users have the same score, the one that has a lower number of tasks is considered better at the task, as the score was achieved with fewer steps, thus faster [10].

As key names are alphabetically ordered and stored in Cassandra, by using the above model, especially for the keywordTimeToComplete, taskAssign, userScores column families, fast and sorted access is guaranteed towards the information in the database. This means no A.I algorithms are required to pull and assign the best user suited for newly added tasks. Moreover, having the information sorted within the database, the time required to perform the assign operation is low, as searching is fast. This in turn makes the application highly responsive.

```
def function addNewTask(task)

  foundUsers = null
  iteration = 1
  taskAssigned = false

  while (taskAssigned == false && iteration < 10) do

    resetOverallScore(foundUsers)

    for keyword in task.keywords do

      foundUsers.push( getNUsersWithBestScoreFrom_taskAssign(keyword,Start = 10*iteration - 9, End = 10 * iteration) )

      for user in foundUsers
        userScore = GetScoreForKeywordFrom_userScores(keyword,user)
        user.overallScore += userScore
      end

    end

    foundUsers.sort_by { |overallScore, atr1, .. , atrn| overallScore }

    count = 0
    while (taskAssigned == false and count < foundUser.size) do
      if foundUser[count].isAvailable?
        addTaskToUser(foundUser[count],task)
        taskAssigned = true
        return true
      end
    end

    iteration = iteration + 1
  end

  return false

end
```

FIGURE 6. Task Assign Algorithm

When a new task is added, in order to assign it to the best suited user, the algorithm will loop through the taskAssign column family for each keyword, starting from top to bottom, from the best score to the lowest possible. It will compute the overall score for each user and if the user score for a given keyword is null it will count as 0. The user with the best score is assigned the task if and only if the cross checking of the userAvailability column results in a valid response, that is the given user has enough available time to complete the task before the deadline. If the user is not able to complete the task on time, the next user with the best score lower than the previous user's score is selected. If no user can complete the task on time, a message is displayed to the client that is trying to add the new task and two options are provided. The first option is to extend the deadline, which triggers a resume of the above algorithm. The second one is to notify the project manager of the issue and solely in this case human task assignment and decision is required [10].

```
{
    { "project_1#keyword_1#username_1": "minutes#tasksNo" },
    { "project_1#keyword_1#username_2": "minutes#tasksNo" },
    { "project_1#keyword_2#username_1": "minutes#tasksNo" },
    { "project_1#keyword_2#username_2": "minutes#tasksNo" },
    { "project_1#keyword_3#username_1": "minutes#tasksNo" },
    { "project_1#keyword_3#username_2": "minutes#tasksNo" },
    { "project_2#keyword_1#username_1": "minutes#tasksNo" },
    { "project_2#keyword_1#username_2": "minutes#tasksNo" }
}
```

FIGURE 7. JSON representation of the keywordTimeToComplete column family optimized for SaaS

6. Deploying the Application as a SaaS

The model described in section 5 has a few limitations that block the possibility of deploying the application as software as a service. This is due to the fact that all data structures are global and contain no information regarding the projects they are part of, except the task description data structure. For example, information in the keywordTimeToComplete column family (Figure 3) is stored and indexed based on the keyword name and username tuple.

As hypothetically issued in section 4, with 100.000 projects one could have more than 100.000 duplicate keyword names and plenty duplicate usernames. Therefore, searching the keywordTimeToComplete column family for one project would require going through other projects as well. This increases computation time and has a detrimental effect on the entire user experience.

To compensate and to further optimize data indexing, storage and accessing, limiting the amount of time required to retrieve needed information and therefore improving efficiency and user experience, one could easily prefix the database access keys with a project identifier as shown in Figure 7. This way searching would be kept through one project alone, separate from the rest of the projects.

Similar prefix would also be added to taskAssign, userScores and any data structure that does not have a global scope. Furthermore, given the automatic partitioning and distribution model of Apache Cassandra, this model would also allow local, close to client storage for each project. For example, the entire SaaS application would be distributed across 3 continents. Clients from US would access data from US servers; clients from EU would access servers from EU and clients from Australia would access data from servers located there.

7. CONCLUSIONS

The proposed concept is viable as it reduces the role of the project menager. The few cases when a project manager's input would be required within the application are few. One case would hold when deadlines for newly created tasks are really short and the client is unwilling to extend them. A solution for this issue would be to introduce a new user (developer) to the project and have the task assigned to the new person. This however requires human interaction within and outside of the application. Another case when the project manager would be required would be when the algorithm cannot find a single user that can complete the task (noted by $task_x$), as nobody in the system has completed a similar task before.

Future developments may include solving the above issues and enhancements of the current model, as follows: assign tasks based on user's skill set as defined in their profile, not just by their performance, mobile apps integration, automatic time tracking based on proximity sensors or based on straightforward interfaces within the mobile applications, taking user preference into consideration when automatically assigning the tasks, or even employee satisfaction / happiness while doing a specific task based on reports collected from coworkers. This is important because studies have shown that misinterpreted user preference and lack of employee satisfaction inhibits productivity [6].

The presented model limits the accessibility of the software to projects that don't require hardware tools, or that require hardware tools that are by their own definition always available. A huge improvement for the algorithm would be a component that tracks availability, cost and logistics for hardware equipments required by individuals working on the project.

Lastly, the model proposed is eventually consistent and highly available, which means that some information may not be available to all users at a given point in time due to external factors. The algorithm is therefore prone to errors with regards to task assignation when such events occur. This is due to the fact that not all the information would be available to make the proper, exact estimations. Future work should include comprehensive tests that would yield the severity of the algorithm's deviation from its best result when such events do happen.

References

[1] D. Gale, *The two-sided matching problem. Origin, development and current issues*, International Game Theory Review, Vol 3, Nos. 2 & 3, p. 237-252, 2001

[2] Gartner, *Worldwide Software-as-a-Service Revenue to Reach $14.5 Billion in 2012*, Gartner Newsroom, March, 2012

[3] S. Gilbert, N. Lynch, *Brewers Conjecture and the Feasibility of Consistent, Available, Partition-Tolerant Web Services*, MIT, 2002

[4] M. Herlihy, J. Wing, *Linearizability: A Correctness Condition for Concurrent Objects*, ACM Transactions on Programming Languages and Systems, Vol 12, No 3, 1990
[5] L. Ireland, *Future Trends in Project Management*, PrezSez 08-2008, 2008
[6] J. M. Ivancevich, *High and low task simulation jobs: a causal analysis of performance-satisfaction relationships*, Academy of Management Journal, Vol 22, No 2, p 206-222, 1979
[7] B. Lagesse, *A game-theoretical model for task assignment in project management*, IEEE International Conference on Management of Innovation and Technology, Singapore, 2006
[8] A. Lakshman, P. Malik, *Cassandra - A Decentralized Structured Storage System*, The 3rd ACM SIGOPS International Workshop on Large Scale Distributed Systems and Middleware, Big Sky Resort MT, LADIS, 2009
[9] S. Nokes, I. Major, et al., *The definitive guide to Project Management: Every executives fast-track to delivering on time and on budget*, Prentice Hall, 2004
[10] B. Pop, *NoSQL Model Design for Automating Project Management*, IEEE International Conference on Automation, Quality and Testing, Robotics (AQTR 2012), Cluj-Napoca, 2012
[11] J. Skabelun, *Are non-performers killing your bottom line?*, Credit Union Executive, Vol 31, No 13, 2005

Department of Computer Science, Babeş-Bolyai University, 1 M. Kogălniceanu St., 400084 Cluj-Napoca, Romania
E-mail address: popb@cs.ubbcluj.ro, bogdan.pop@webraptor.eu