A Systematic Review of Dynamic Reconfiguration of Software Projects

Daniel Antonio Callegari[1], Ricardo Melo Bastos[1]

[1] Faculdade de Informática – Pontifícia Universidade Católica do Rio Grande do Sul
Avenida Ipiranga, 6681 – 90619-900 – Porto Alegre – RS – Brasil

{daniel.callegari, bastos}@pucrs.br

Abstract. *Software companies often make use of project management knowledge as well as a set of development processes in order to build their solutions with quality and within scope, time and resource constraints. Software projects are very dynamic systems that demand recurrent adjustments of their plans even during execution due to their particular nature. Such adjustments can be viewed as reconfigurations on schedule, on resource allocation and on other project elements. A systematic review is a means of evaluating and interpreting all available research relevant to a particular area of interest. This paper presents the results from a systematic review of four subareas underlying the dynamic reconfiguration of software projects. A number of works were analyzed from the year of 2004 up to the present moment.*

1. Introduction

The development of a software product demands a unique effort that involves dealing with activities and resources among other elements to build the desired solution. Organizations frequently combine some project management methodology along with a specific software development process [Schwalbe 2002]. This combination translates the need to address both production and management activities for one or more projects that may share resources in an organization.

Because of their nature, software projects comprise very dynamic systems that require frequent adjustments of their plans, from the original planning to project kick-off, to execution and finally deployment. Nevertheless, the complexity and the number of projects a manager must simultaneously deal with increases every day. As the number of projects increase, the manager must deal with a greater number of variables [Schwalbe 2002] [Kerzner 2000].

This complex scenery can be more easily addressed if we provide means to help managers in making decisions that involve the resources, the activities and the flow of work items during the execution of one or more simultaneous projects. The area of dynamic reconfiguration of software projects deals with the events, the actions, the affected elements and the consequences of the adjustments in software projects during their execution.

We present the main results obtained from a systematic review of the area, along with comments and suggestions for future work. We first briefly present the area of interest (section 2) and then describe the adopted methodology (section 3). The collected information and discussion are presented in sections 4 and 5, respectively. Finally, section 6 presents suggestions for further investigation.

Study developed by the Research Group of the PDTI 01/2008, financed by Dell Computers of Brazil Ltd. with resources of Law 8.248/91.

2. Dynamic Reconfiguration of Software Projects

According to the PMBOK (2004), projects are temporary endeavors undertaken to create a unique product or service. In a simplified view, projects of any nature have two very distinct phases: the planning phase and the execution phase. During planning we collect all the necessary information for the execution of the project, including the activities to be performed, the necessary resources, the deadline and other restrictions. The execution phase comprehends the management, monitoring and control of the project as well as the development of the final product. Ideally, execution follows the plan with minor variations. Unfortunately, this is hardly ever the norm for software projects.

According to [Joslin & Poole 2005] and other reviewed work, purely "static" approaches do not solve the main problems in software development. The uncertainties underlying any software project must be constantly monitored and the plan must be adjusted accordingly during all the life cycle of the project. Current software development processes help addressing this question by means of incremental and/or cyclic refinements of both planning and product definition [Kruchten 2000] [Schwalbe 2000], but this is not sufficient. In order to help managers in dealing with all the variables and uncertainties of software projects, we must provide them increasing levels of support such as tools, guidelines, models and automated or semi-automated decision support systems. Desired solutions should address resource selection and allocation, scheduling of activities, single and multi-project support among other characteristics (details in section 4).

If we name the first planning of a project its "initial configuration" comprising not only the flow of activities, the project's priority, the involved resources, their characteristics, allocation and availability, as well as any other necessary information, then we can also define the term "project reconfiguration" as meaning any kind of adjustment performed during project execution in order to adapt the plan to the current state of all those variables. Thus, the term "dynamic reconfiguration of software projects" refers to all kinds of effort that act over a preexistent plan of activities and associated resources considering one or more simultaneous software projects. It is worth noting that this term is also being used by many other areas such as networking, autonomic computing and distributed systems [Coulouris et. al. 2002] [Horn 2001] (the actual definitions differ a little, but the main concept is fundamentally the same: semi or full automatic modifications that take place during execution).

Following the analyzed papers, and based on the problems they address, we have identified four main areas in the context of dynamic reconfiguration of software projects: (A) resource selection, (B) resource allocation, (C) scheduling of activities, and (D) the integration of the project's activities to the organizational workflows of the company. The details are presented in section 4.

3. Systematic Reviews in Software Engineering

Systematic reviews are very common research methodologies in areas such as medicine and social sciences. A systematic review aims to integrate empirical research in order to create generalizations. Some advantages are the elimination of common biases, the discovery of general principles and the identification of relative influences of the

different individual studies due to a more formal methodology than a simple literature review [Biolchini et. al. 2005]. The Software Engineering research area suffers from the lack of available evidence of the technologies we use. According to [Kitchenham 2004], most available evidence in the area is fragmented and limited, not properly integrated and often do not agree on standards for conducting experiments. This is the main motivation of our current work.

3.1 Systematic Review Details and Protocol

Systematic reviews begin with the definition of the research question. In our case: "*which automated or semi-automated approaches currently exist to select resources, (re)schedule activities and (re)allocate resources to activities in the context of multiple and simultaneous software projects?*". The next steps are the selection of sources, the definition of inclusion and exclusion criteria and the selection of the studies. When planning is done, we evaluate the protocol plan. If it is approved, information extraction begins (see Figure 1). The results are also evaluated before we perform the final analysis. Results packaging occurs during all the process in order to keep all relevant information and to document decisions taken by the researchers.

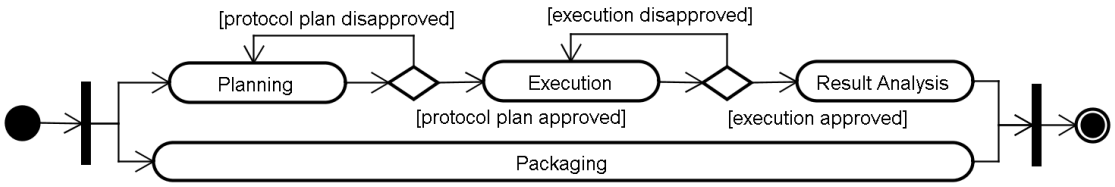

Figure 1. Systematic review process [Biolchini et. al. 2005]

We have selected the following sources for this study: Search engines of IEEE Xplore Digital Library, ACM Digital Library, Springer Link and Science Direct. Regarding the population and studies selection, all searches were made on January and February, 2008, on journals and conference proceedings for papers written in English, from the year of 2004 up to the present, in the areas of computer science, software engineering, project management, information systems, decision support systems, operations research and artificial intelligence. We chose this range of years in order to filter recent and state-of-the-art work in the area. Searches were made by inputting search strings in all of the aforementioned mechanisms.

We have experimented with several combinations and variations of the following keywords: *resource*, *selection*, *allocation*, *software project*, *management*, *dynamic*, *planning*, and *scheduling*. Validating the possibilities with other colleague researchers, we have selected the following string: "*(software project) AND (dynamic) AND (resource OR scheduling)*", when we noticed that the selected words embraced some of the other mentioned concepts. It is important to note that some of the search strings we tried have returned large amounts of papers (for example, 524, 377, and even larger numbers such as 12603). Following other systematic review studies in software engineering, we tried to stay in the range of 200 to 250 papers (the selected string returned 242 results). Just in terms of comparison, as a matter of fact, running the same query without the word "dynamic" returned 589 entries, which is far beyond the acceptable number and would not meet one of the main requirements of this research.

All 242 papers were analyzed by using the following workflow: (i) the search string is submitted to all mechanisms; then, for each paper: (ii) the title and abstract of the paper are read; (iii) if approved, the full text of each paper is then read for final approval; (iv) when in doubt due to lack of information in the abstract, a quick reading of the text is performed; (v) all the remaining papers were selected for full reading. This first resulted in a selection of 27 papers. Eight of them were later discarded due to lack of minimum expected relevance. The final selection included 19 papers (7.85% of the overall search engines results), which represents an adequate sample according to our references [Biolchini et. al. 2005] [Kitchenham 2004].

4. Results

The main criteria for the selection of the papers were to only accept papers that included a good level of description of the solution and sufficient information about the methods, algorithms, or strategies of any kind to solve the questions related to the four subareas of the research theme. More specifically, we were interested in the following items (see results in tables 1 and 2 – no particular order inferred):

- Resource selection: indicates whether the solution includes the choice of resources, no matter if treating them individually (individual characteristics) or uniformly (one does not differ from the other, only quantities matter);
- Resouce allocation: indicates if the solution supports resource allocation, whether or not among multiple and simultaneous projects;
- Scheduling of activities: indicates if it presents a solution to the sequencing of the activities;
- Multi-project support: does the solution support more than one single project?;
- Kind of solution: shows the "general class" of the solution (e.g. decision support, optimization, methodology, etc.);
- Method: the method used for the solution (e.g. bayesian networks, machine learning, dynamic programming, case based reasoning, etc.);
- Use of expert knowledge: is the solution based on expert judgment?;
- Use of simulation: does the solution involve some kind of computational simulation?;
- Dynamic solution: indicates whether the solution is static or dynamic (in the sense of "on demand", during project execution); synonyms such as "online", "on-the-fly" and "adaptive" were also accepted;
- Includes case study or experiment: was the solution scientifically evaluated?;
- Provides a tool: the solution includes a tool or a prototype?

Sources marked with an asterisk (*) have atypical characteristics: [Harman 2007], [Shepperd 2007] and [Wernick & Hall 2007], for instance, deal with one or more issues in a broader view, without directly presenting a solution to the mentioned problems (yet they are useful in this research); [Trainer et. al. 2005], by its turn, is used as a supplementary work over [Souza et. al. 2007], bringing contributions to their original approach. Items marked with "-/N.A" are not applicable.

5. Discussion

By analyzing tables 1 and 2 we can derive some finding based on a purely quantitative analysis (individual analysis of the four areas is presented afterward):

- Firstly, five from the selected papers deal with resource selection (but not exclusively) – about one third of the works underline that resources should be treated individually due to their unique characteristics. Treating resources homogeneously, though, is typical for solutions that make use of simulation;
- Nine of the analyzed work support resource allocation. Alternatives cover the use of Bayesian Belief Networks and System Dynamics, among others, and are often associated with decision support approaches. The remaining papers do not even mention this issue;
- Activity scheduling is directly addressed by eight papers. A distinctive work is [Jalote et. al. 2004], an approach that seem to eliminate the need of a complete planning of activities;
- Only five of the analyzed papers deal with a multiproject scenario;
- It is worth noting that a few more than the half of them indicate: (i) being based on expert knowledge, (ii) were elaborated through case studies or experiments, and (iii) have included some kind of tool (not necessarily the same papers on each case);
- Regarding the "dynamic" solutions, some authors use alternative terms like "online", "on-the-fly" and even "adaptive" for basically the same idea. In this systematic review, 10 papers explicitly address this issue. According to [Harman 2007], techniques such as Particle Swarm Optimization and Ant Colony Optimization seem promising in this context. Kabbaj et. al. (2007) is the only one to support this idea at the "meta" level (e.g. at process level, instead of project level), through the use of Process-Centered Software Engineering Environments, or PSEEs, and a set of 30 "rules". This kind of solution is often associated with workflow systems, in which the effect of a failure or unhandled exception reflects an action that breaks the consistency relationships of the process. In these cases we need to operate outside the PSEE, change the underlying model, or adapt the PSEE to tolerate some deviation;
- As a final remark, there is some prominence of optimization techniques (dynamic programming, local search and simulated annealing, for instance), decision support systems, and many apply Bayesian Networks, which are basically cause-effects graphs associated with tables of probabilities, that belong to the group of "decision systems".

Despite that last remark, Harman (2007) states that deterministic optimization algorithms are often inapplicable in real world software engineering problems, because the problems have objectives that cannot be characterized by a set of linear equations.

The use of simulation aims to maximize (or minimize) some fitness function, created according to the specific goals of the solution, for instance: minimization of cost, minimization of time, or a combination of factors. According to [Joslin & Poole 2005], simulation offers the possibility for representing the complexity that is necessary

for realistic reasoning about a project, including the inherent uncertainty. A common example is the use of Monte Carlo techniques.

Instead of simulation, some proposals, such as [Fan & Yu 2004], bet on the use of Bayesian Belief Networks (BBNs) due to their capacity of modeling uncertainty and providing probabilistic estimates when solving problems. They state that the main difference of the two approaches relies exactly over this characteristic: while simulation typically provides deterministic results, BBNs may yield a set of probabilistic results at each run, allowing the manager to preview situations even based on incomplete information.

As we can see from the above examples, many researchers do not agree on the techniques when dealing with uncertainty, for instance. The combination of methods of many of the proposals (see last column of Table 1) seems to indicate alternative efforts to solve the problem by merging solutions.

More detailed conclusions for each of the four areas can be found below.

5.1 Area A – Resource Selection

Resource selection deals with choosing which resources to work on which projects based on a set of criteria such as individual capacities, cost, sharing level and availability. The simplest approaches do not distinguish among resources, treating them only as quantities (ignoring individual characteristics); this is the case with most approaches that use simulation. In the other hand, ability levels, startup penalties and individual or role characteristics are used by the counterpart solutions such as [Fan & Yu 2004], [Fenton et. al. 2004] and [Joslin & Poole 2005] – "employees are not interchangeable". The startup penalty refers to the famous Brooks' Law [Brooks 1995] that adding more people to a late project only makes it even later. Another interesting finding (pointed out in [Joslin & Poole 2005]) is that "resources are even less interchangeable after work on a task has begun. [Thus…] reassigning someone from one task to another, or adding an additional resource to a task after substantial progress has already been made on it, will usually be inefficient in the short term".

Among the analyzed papers, there is an expressive number of solutions that make use of simulation. One of the distinctive works is [Padberg 2004] in which diverse team abilities are modeled via different probability distributions for the time necessary to complete the activities. Following that, works that explore team work, such as [Peslak 2006], gain attention when suggesting that "team processes, or at least their perception, are independent of personality. Team processes can be viewed to function well regardless of the makeup of the team."

From this perspective, it is clear that the maturity of our knowledge on software project management demands treating resources as individuals. Personal abilities and soft skills play an important role in such human centered systems [Verma 1996]. Thus, although quantitative simulation of resources can help understanding some aspects in resource selection and allocation (as pointed out in this review), more realistic solutions will only be possible if we consider individual characteristics.

For the present analysis we expected more research to (i) find out which cases demand working with individually characterized resources and which cases we can treat

them as a homogeneous group; and (ii) to find out what are the most relevant criteria distinguishing among resources in terms of selection.

While dealing with individual characteristics may lead to a more realistic solution, we must also be aware of the scalability, specifically in terms of combinatorial explosion. It is worth noting that only one of the analyzed papers suggests grouping resources on roles and assigning corresponding skills to each role [Fan & Yu 2004]. This represents and interesting intermediate solution.

Table 1. First part of the results from the systematic review

Source	Resource selection	Resource allocation	Scheduling of activities	Multi-project	Kind of solution	Method
[Fan & Yu 2004]	Yes – role skills	Yes – some support	No	Yes	Decision support	Bayesian networks + Fuzzy Functions
[Fenton et. al. 2004]	Yes – individual skills	Yes	No	No	Decision support	Bayesian networks
[Harman 2007]*	-	-	-	-	Optimization	A review on "Search Based Software Engineering"
[Jalote et. al. 2004]	No	Yes	Eliminates the need	No	Methodology	Time boxing, activity pipelining and parallelism
[Joslin & Poole 2005]	Yes – individual skills	Yes	Yes – two kinds of activities	No	Decision support	Search on strategy space
[Kabbaj et. al. 2007]	No	No	Unusual	No	*Workflow*	Detects workflow exceptions
[Lee & Miller 2004]	No	Yes	Yes	Yes	Decision support	Critical Chain + System Dynamics + Sceneries
[Melo & Sanches 2008]	No	Yes - some support	Yes	No	Decision support	Bayesian networks
[Padberg 2004]	No	Yes	Yes	No	Optimization	Machine learning + Simulation + Markov Decision Process
[Shepperd 2007]*	-	-	-	-	-	-
[Yiftachel et. al. 2006]	Yes – individual skills	Yes	Unusual	No	Optimization	Dynamic programming
[Lee & Lee 2005]	No	No	Yes	No	Semi-automated	Case-based reasoning
[Padberg 2005]	No	No	Yes	No	Optimization	Machine learning + Simulation + Markov Decision Process
[Peslak 2006]	Yes	No	No	No	Study	Empiric Sw. Eng.
[Sentas et. al. 2007]	No	No	Yes	Yes	Methodology	Statistical
[Souza et. al. 2007]	No	Yes – component based	Yes	No	Decision support	Dependency graphs
[Trainer et. al. 2005]*	-	-	-	-	-	-
[Vähäniitty 2005]	No	No	No	Yes	Methodology	Conceptual framework
[Wernick & Hall 2007]*	-	-	-	-	-	-

Table 2. Second part of the results from the systematic review

Source	Expert knowledge	Simulation	Dynamic solution	Case study / experiment	Provides a tool
[Fan & Yu 2004]	Yes	No	Yes	Yes	Yes
[Fenton et. al. 2004]	Yes	No	No	Yes	Yes
[Harman 2007]*	-	-	-	-	No
[Jalote et. al. 2004]	No	No	Yes	Yes	Yes
[Joslin & Poole 2005]	No	Yes	Yes	No	No
[Kabbaj et. al. 2007]	No	No	Yes	No	No
[Lee & Miller 2004]	No	Yes	(probably) Yes	No	Yes
[Melo & Sanches 2008]	Yes	No	Yes	No	Yes
[Padberg 2004]	Yes	Yes	Yes	No	Yes
[Shepperd 2007]*	-	-	-	-	-
[Yiftachel et. al. 2006]	Not clear	No	Yes	Yes	No
[Lee & Lee 2005]	Yes	No	No	Yes	Yes
[Padberg 2005]	Yes	Yes	Yes	No	Yes
[Peslak 2006]	N.A.	N.A.	N.A.	Yes	N.A.
[Sentas et. al. 2007]	Yes	No	Yes	Yes	No
[Souza et. al. 2007]	No	No	No	Yes	Yes
[Trainer et. al. 2005]*	-	-	-	-	-
[Vähäniitty 2005]	Yes	No	Yes	Yes	No
[Wernick & Hall 2007]*	-	-	-	-	-

5.2 Area B – Resource Allocation

Resource allocation is highly associated with resource selection and scheduling of activities. When removing resources from an activity, for instance, we expect some impact on the execution time of that activity which, in turn, may affect other dependent activities. As another important issue, some researchers suggest that multitasking of resources is a bad idea and they mathematically show the consequences of it, especially when considering multiple and simultaneous projects [Lee & Miller 2004]. Joslin and Poole (2005), by their turn, state that multitasking can work with specific types of activities but only with certain resources (based on individual characteristics).

Only two of the analyzed work [Souza et. al. 2007] [Padberg 2004] suggest using information from technical dependencies among software components to influence resource allocation. The idea is to allocate the same set of resources to groups of interrelated tasks, which are determined based on the artifacts they relate to.

The main findings in this area are: (i) most of the analyzed work offer different levels of decision support (as opposed to automated solutions); thus investigating the limits between decision support and automation in this context is a future research possibility; (ii) Brooks' Law needs more investigation when considering a project's characteristics such as size and available time; (iii) there is no consensus on the

"granularity" of the activities, i. e. breaking down an activity to smaller steps may affect the way resources are allocated; (iv) some papers indicated the necessity of distinguishing among types of activities (prioritary or optional, for instance), but none of them explored the three kinds of activities we have proposed in [Callegari & Bastos 2007] and [Rosito et. al. 2008]: managerial activities, productive activities and supporting managerial activities, which are related to the area D of this research; and (v) different types of software processes also demand different approaches (e.g. "agile" vs. "traditional") [Anderson 2003] [Lee & Miller 2004].

5.3 Area C – Scheduling of Activities

The scheduling of activities is the most affected area in terms of uncertainties and external events [Jalote et. al. 2004] [Joslin & Poole 2005]. The idea of "contingency plans" (as in [Joslin & Poole 2005]) seems very promising because it fights great part of the subjectivity regarding decisions made by managers every day. But the solution is very simplified as they only simulate one type of resource (developers).

The proposal in [Joslin & Poole 2005] is a search in a space of planning strategies and the use of simulation to evaluate the strategies. As the authors point out, "a strategy is not a static set of commitments, but rather a representation of high-level considerations that can be used to guide dynamic decision-making during simulated or actual execution". The problem here is the large amount of possible situations, leading to the problem of combinatorial explosion.

Jalote et. al (2004) brings an interesting approach based on "timeboxing" (a process model for iterative software development), which involves pipelining concepts, where development is done in a series of fixed duration time boxes. Each time box is divided into a sequence of fixed duration stages, with a dedicated team for each stage. They claim that "the turnaround time for each release is reduced substantially, without increasing the effort requirement" and that this technique "eliminates scheduling". This clearly makes it incompatible with all other "common" approaches that follow a chain of activities and dependencies among them. Nevertheless, they acknowledge that timeboxed projects with large teams are hard to manage.

One of the analyzed works adopts Case Based Reasoning (see Table 1) to assist the construction of a software project network of activities, claiming that managers already do this naturally [Lee & Lee 2005]. The solution is based on 17 independent factors and a base of 30 cases with information for each one of the factors. One obvious characteristic of these kinds of solutions is that we need to store a significant amount of historical data in order to compute similarities to the case we are facing at the present. In addition, the simple automatic generation of a project schedule will not satisfy the various project managers, they admit, because "even for the same project, the project network may be designed differently depending upon the development methodology adopted, a project manager's style, and preferred way of displaying the network". Besides, although the idea of applying CBR seems interesting, this particular proposal works only for the first configuration of a project (in other words, it will not work as a dynamic solution).

As mentioned before, under a single software project several correlations of activities can be found in terms of the elements they handle (artifacts or software

components, for instance). As a result, we can expect that allocating the same resources to each such groups of interrelated activities may reduce the so called startup penalty.

Also, Padberg (2004) claims that the teams in software projects do not work independently: "the progress of one task not only depends on the productivity of the corresponding team, but also on the progress of the other tasks. This kind of feedback is typical for software projects and is a key driver for the uncertainty about the future progress of a software project". Thus, when re-planning the schedule of the activities, their dependencies must also be adjusted by means of the dependency of the associated artifacts.

Therefore, the following aspects needs more development: (i) identifying the types of feedback (as in [Padberg 2004]) among different types of activities in order to evaluate the impact of a change to other related activities; (ii) identifying which events fire subsequent re-planning needs to one or more software projects; (iii) determining an effective manner of coding predefined actions, in response to the fired events; and (iv) distinguishing the different types of activities, as mentioned earlier.

5.4 Area D – Integration of Workflows and Project's Activities

In [Callegari & Bastos 2007] we have presented a model that integrates productive and managerial activities in software projects. Companies often have a set of related activities (workflows) that may be shared among two or more simultaneous projects. Those activities are called supporting managerial activities and they do not belong to any specific project; rather, they can be "instantiated" by the projects when they need. Some examples are workflows for hiring new people or for setting up some required environment for work.

The closest idea in this respect in our analysis is presented in [Lee & Miller 2004]. Here the authors show some strategies that the manager may take according to defined "policies". Another relevant work [Joslin & Poole 2005] just mentions this fact as an example. Thus, it becomes clear that more effort is needed in this direction of research.

5.5 Integrating the subareas

Although the four subareas have been presented in individual sections, in practice it is very hard to treat them as independent problems. Figure 2 represents an integrated view of the problem of dynamic reconfiguration of software projects.

By means of the previous analysis it becomes evident that the main pillars for a solution are the resources, the activities and the projects of a given software company.

Determining the events that may disturb one or more projects sharing resources can help in coding a set of corresponding actions to be taken appropriately. Some of these actions may be fully automated (if desired), such as rescheduling, while other actions may need human intervention (some options are presented to the manager for the problems that the system cannot handle).

Regarding the resources, an important decision is needed between solutions that handle resources individually (a heterogeneous pool) or as a group of leveled persons (a homogeneous pool of resources). While the latter may be more computationally cost effective, when we handle resources' individualities we gain in accuracy. Another idea

that is current under development by the authors is the concept of affinity between resources and activities, in order to reduce startup penalties.

In addition, when considering a multi-project scenario we must deal with different projects' characteristics such as size, type, priority and base process model. While some authors are emphatic in this respect (for instance, [Fan & Yu 2004] [Lee & Miller 2004]) claiming it is more realistic, there is no strong evidence the heuristics they present work well on diverse scenarios. As an example, the solution proposed by Lee and Miller (2004) first plans each project on a common basis and then adjusts the individual projects based on different priorities and requirements. But their work is mostly based on [Turner & Payne 1997], which is not specific for software projects. Fan & Yu (2004), by their turn, make a clear distinction of different sources of factors that may affect projects: some are related to the organization as a whole, and some are directly related to one or more specific projects (a link to area D is clear here).

Besides, in order to build a model that integrates all the aspects identified in this review, one should also consider the distinct types of activities (e.g. prioritary; mandatory or optional; managerial or productive) [Lee & Miller 2004] [Joslin & Poole 2005] and their possible groupings, as in [Souza et. al. 2007] and [Padberg 2004].

Walking towards the development of a more complete model for the dynamic reconfiguration of software projects is one of the goals of the authors. This systematic review allowed us to achieve an overall perspective of the most recent work on the area.

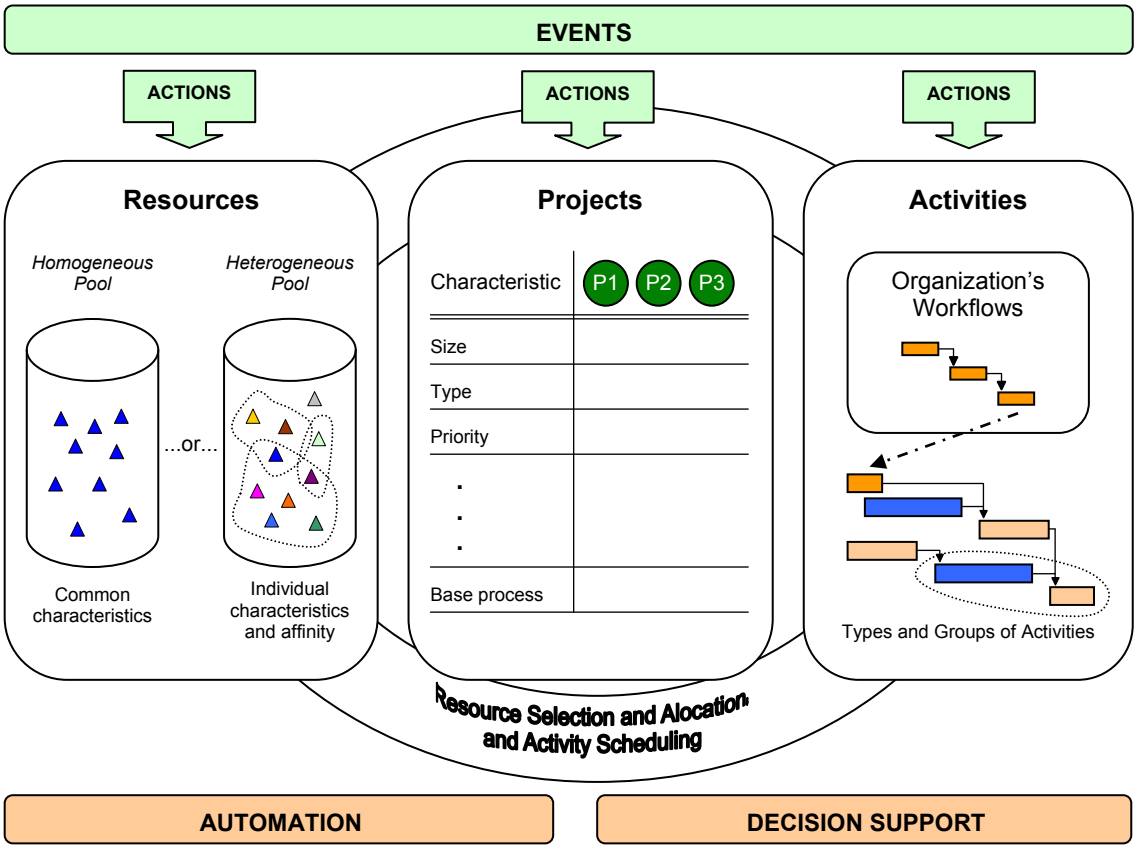

Figure 2. A view of the integration of the four areas

6. Conclusion

This paper presented some results from a systematic review of dynamic reconfiguration of software projects. As a general finding, we can say that this multifaceted problem is far from being solved by the current approaches. Even the more recent works do not present a solution that addresses all the problems at the same time. The trend is for approaches that deal with at most two of the identified problems simultaneously. Integrating the solutions is thus a clear area for future work. Dealing with resource multitasking was found to be one of the most complex problems to which we do not have a satisfactory solution yet. While resource allocation has many acceptable solutions in areas such as manufacturing, this problem in software engineering remains unsolved. Yet, some approaches that differentiate among the resources based on individual characteristics are starting to gain focus. Many solutions make use of expert knowledge to help in decisions, but we need to advance in identifying and modeling the relevant variables in this subarea.

It is important to note that none of the reviewed works present a solution to the problem of allocating the same people (or teams) to related activities in order to reduce the startup time on new activities, although the problem has been indentified by some of them. There is also a concern with the combinatorial explosion of possible states due to the large number of interrelated variables in solutions based on Bayesian networks or simulation, for instance. The integration of the scheduling solutions with an organization's common activities (workflows) is also a subarea of research that demands further investigation.

6.1 Limitations of this research

According to [Biolchini et. al. 2005], applying systematic reviews in software engineering is much more difficult than in other areas, greatly due to the lack of rigor and conscience in reporting the results of the primary studies in software engineering. It is also hard to make comparisons when we do not have quantitative data; and the lack of standardization on the form of presenting the results is also a difficulty.

In this particular study, the main extraction was performed by the first author, while all the process of choosing and refining the search string, the set of inclusion and exclusion criteria, and the evaluation of the extracted information were discussed among three other colleagues. This certainly represents some bias. Nevertheless, this methodology enables us to summarize current evidence on the literature to some extent, identify any gaps by comparing different approaches, and to provide a wider view of the field of research, while pointing out opportunities for future work.

7. References

Anderson, David. (2003) Agile Management for Software Engineering. Prentice Hall, Pearson Education.

Biolchini, J., Mian, P.G., Natali, A.C.C., Travassos, G.H. (2005): Systematic review in software engineering. Technical report, Systems Engineering and Computer Science Department, Rio de Janeiro.

Brooks, F. (1995) The Mythical Man-Month: Essays on Software Engineering. Addison-Wesley, 2nd Ed.

Callegari, D.; Bastos, R. (2007) Project Management and Software Development Processes: Integrating RUP and PMBOK. ICSEM – International Conference on Systems Engineering and Modeling. Haifa, Israel.

Coulouris, G., Dollimore, J., Kinberg. (2002) Distributed systems: concepts and design. 3a. edição. Addison-Wesley, 2001. Tanenbaum, A. S. Distributed systems: principles and paradigms. Prentice Hall.

Fan, Chin-Feng; Yu, Yuan-Chang. (2004) BBN-based software project risk management. Journal of Systems and Software, Volume 73, Issue 2, Applications of statistics in software engineering, pp. 193-203.

Fenton, Norman; Marsh, William; Neil, Martin; Cates, Patrick; Forey, Simon; Tailor, Manesh. (2004) Making Resource Decisions for Software Projects. In Proceedings of the 26th international Conference on Software Engineering (May 23 - 28). IEEE Computer Society, Washington, DC, 397-406.

Harman, Mark. (2007) The Current State and Future of Search Based Software Engineering. In 2007 Future of Software Engineering (May 23 - 25). International Conference on Software Engineering. IEEE Computer Society, Washington, DC, 342-357.

Horn, P. (2008) Autonomic computing: IBM perspective on the state of information technology, IBM T.J.Watson Labs, NY, 15th October 2001. AGENDA 2001, Scottsdale. Available at http://www.research.ibm.com/autonomic/.

Jalote, Pankaj; Palit, Aveejeet; Kurien, Priya; Peethamber, V.T. (2004) Timeboxing: a process model for iterative software development. Journal of Systems and SoftwareVolume 70, Issues 1-2, February, pp. 117-127.

Joslin, David; Poole, William. (2005) Agent-Based Simulation For Software Project Planning. In Proceedings of the 37th Conference on Winter Simulation (Orlando, Florida, December 04 - 07). Winter Simulation Conference. Winter Simulation Conference, 1059-1066.

Kabbaj, Mohammed; Lbath, Redouane; Coulette, Bernard. (2007). A Deviation-tolerant Approach to Software Process Evolution. In Ninth international Workshop on Principles of Software Evolution: in Conjunction with the 6th ESEC/FSE Joint Meeting (Dubrovnik, Croatia, September 03 - 04. IWPSE '07. ACM, New York, NY, 75-78.

Kerzner, H. (2000) Applied project management: best practices on implementation, John Wiley & Sons.

Kitchenham, B. (2004): Procedures for performing systematic reviews. Technical report Software Engineering Group, Department of Computer Science, Keele University.

Kruchten, P. (2000) The Rational Unified Process: An Introduction, Addison-Wesley, 2nd edition.

Lee, Bengee; Miller, James (2004) Multi-Project Management in Software Engineering Using Simulation Modelling. In Software Quality Journal, Volume 12, Number 1, pp. 59-82, March.

Lee, Jae Kyu; Lee, Nobok. (2005) Least modification principle for case-based reasoning: a software project planning experience. In Expert Systems with Applications, Volume 30, Issue 2, February, pp. 190-202.

Melo, Ana C.V. de; Sanchez, Adilson J. (2008) Software maintenance project delays prediction using Bayesian Networks. Expert Systems with ApplicationsVolume 34, Issue 2, February, Pages 908-919.

Padberg, Frank. (2004) Linking Software Process Modeling with Markov Decision Theory. In Proceedings of the 28th Annual International Computer Software and Applications Conference, 2004. COMPSAC 2004., vol.2, pp. 152-155, 28-30 Sept.

Padberg, Frank. (2005) On the Potential of Process Simulation in Software Project Schedule Optimization. Computer Software and Applications Conference, 2005. COMPSAC 2005. 29th Annual International, vol.2, pp. 127-130, 26-28 July.

Peslak, Alan R. (2006) The Impact of Personality on Information Technology Team Projects. In Proceedings of the 2006 ACM SIGMIS CPR Conference on Computer Personnel Research: Forty Four Years of Computer Personnel Research: Achievements, Challenges &Amp; the Future (Claremont, California, USA, April 13 – 15). SIGMIS CPR '06. ACM, New York, NY, 273-279.

PMBOK. (2004) – Project Management Body of Knowledge. Project Management Institute. Available at http://www.pmi.org.

Rosito, M.; Callegari, D.; Bastos, R. (2008) Gerência de Projetos e Processos de Desenvolvimento de Software: uma proposta de integração. SBSI – IV Simpósio Brasileiro de Sistemas de Informação, Rio de Janeiro.

Schwalbe, K. (2002) Information Technology Project Management. 2nd Ed., Thomson Learning, Canada.

Sentas, Panagiotis; Angelis, Lefteris; Stamelos, Ioannis. (2007) A statistical framework for analyzing the duration of software projects. In Empirical Software Engineering, Journal, Springer.

Shepperd, Martin. (2007) Software project economics: a roadmap. In 2007 Future of Software Engineering (May 23 - 25). International Conference on Software Engineering. IEEE Computer Society, Washington, DC, 304-315.

Souza, Cleidson R. B. de; Quirk, Stephen; Trainer, Erik; Redmiles, David F. (2007) Supporting Collaborative Software Development through the Visualization of Socio-Technical Dependencies. In Proceedings of the 2007 international ACM Conference on Supporting Group Work (Sanibel Island, Florida, USA, November 04 - 07). GROUP '07. ACM, New York, NY, 147-156.

Trainer, Erik; Quirk, Stephen; Souza, Cleidson de; Redmiles, David. (2005) Bridging the Gap between Technical and Social Dependencies with Ariadne. In Proceedings of the 2005 OOPSLA Workshop on Eclipse Technology Exchange (San Diego, California, October 16 - 17). eclipse '05. ACM, New York, NY, 26-30.

Turner, J.R. and Payne, J.H. 1997. The problem of projects of differing size and skill mix. Journal of the Project Management Association 3(1): 14–17.

Vähäniitty, Jarno. (2005) A Tentative Framework for Connecting Long-Term Business and Product Planning with Iterative & Incremental Software Product Development.

In Proceedings of the Seventh international Workshop on Economics-Driven Software Engineering Research (St. Louis, Missouri, May 15 - 15). K. Sullivan, Ed. EDSER '05. ACM, New York, NY, 1-4.

Verma, V. (1996) Human Resource Skills for the Project Manager: The Human Aspects of Project Management. Project Management Institute.

Wernick, Paul; Hall, Tracy. (2007) Getting the Best out of Software Process Simulation and Empirical Research in Software Engineering. In Proceedings of the Second international Workshop on Realising Evidence-Based Software Engineering (May 20 - 26). International Conference on Software Engineering. IEEE Computer Society, Washington, DC, 3.

Yiftachel, Peleg; Peled, Dan; Hadar, Irit; Goldwasser, Dan. (2006) Resource Allocation among Development Phases: An Economic Approach. In Proceedings of the 2006 international Workshop on Economics Driven Software Engineering Research (Shanghai, China, May 27 - 27). EDSER '06. ACM, New York, NY, 43-48.