TPM oriented, systematic review of Release Planning models

Samia NACIRI,

Abstract—Each year, organizations make considerable effort to be more efficient: save money, improve customer satisfaction, increase people morale, and so on. For software maintenance areas, efficiency requires control of the release planning process, and change request prioritization process for the operational software. Release planning is a crucial decision process that allows increasing the business value of the maintained system.

This paper aimsto present the main release planning problems, goals, and constraints in the context of third party application maintenance(TPM). For this purpose, itproposessystematic review principals that allow us to identify, analyze, and classify existing release planning models, then discuss the applicability of these models in TPM context, in regard to multiple research questions.

The release planning classification that will be proposed in this paper will help TPM researchers and practitioners to orient their release planning model proposal with the right goal, constraints and techniques.

Index Terms—Software Engineering, Software Maintenance, Third party application maintenance, TPM, Release Planning, Service Level Agreements, SLA.

1 INTRODUCTION

In software engineering industry, software maintenance phase is considered the longest phase of software life cycle(accounting for between 50% and 80% of the companies' IT budget).It is often under-dimensioned, carrying risks and generating hidden costs [1]. As a result, a new field called "Third Party Application Maintenance" or "**TPM**" has emerged, providingnew business opportunities and benefit to integrators and software engineering companies to work on subcontracting and outsourcing context.

During the last decade, TPM has known significant economic and scientific movements in order to improve the efficiency and performance of this engineering field, in both levelsprovider and client.

2.1 Research motivation

In our previous paper[2], we have presented TPM common problems and managerchallenges that make TPM management less proactive. On the one hand, TPM manager responds to change requests instead of anticipating user expectations in terms of defects correction and feature enhancement. On the other hand, TPM management is criticized for not considering project stakeholder priorities (end users, client representative, and others). Indeed, it focuses on commitment constraints for Service Level Agreements (SLAs) and required quality.

Accelerate defect delivery process and communicate to customer a clear way to decide which defect to include in the next release, are two main steps taken to improve TPM stakeholder satisfaction and to increase the economic value of the software under maintenance.

Despite this fact, this state of the art addresses one of the most well-known issues that was challenged by TPM managers, which is Release Planning. This paper aims to review the release planning literature in terms of existing models and approaches applied to TPM context. It proposes a classification framework of existing Release Planning models based on five TPM dimensions: goal, constraints, technique, decision variables and implementation. Thereafter, our work discusses results and deducts conclusion for the TPM release planning area.

In TPM, estimating CRs cost involves also multiple research problems. However, our work does not fit this research area. The current study is restricted to RP problem of CRs that have known costs.

2.2 Global definitions

TPM management has to build Release Planning (RP) process on lapped, confirmed and tooled models which produce quickly release schedules meeting customer expectations and not sensitive to priorities changes and environmental changes. In a challenged and competitive environment, TPM manager tackles daily the release planning tasks: schedules change requests to mini-releases called "corrective patches" in a very short time and with limited resources. This process is called release planning of the change request.

Change request management remains one of the main processes that were required to achieve quality in software maintenance process and maximize client satisfaction. It aims to organize the process of CR study, planning and delivery. Change request is an artifact that is used to track all stakeholder requests, including new features, enhancements, defects, and changes in requirement. It allows also recording stakeholders interactions and status evolution during the project life cycle [3]. From the TPM perspective, CRs are raised from the day-to-day use of the software product. It may be of urgent matter or not.

In software engineering literature, requirement selection and release planning problems are widely covered by researchers, but in Software maintenance area, they remain rarely dealt by researchers, especially in outsourcing context. Research paper [4]transcribed that software release planning (RP) activities

address assignment of change requests like correction or enhancement, to a sequence of consecutive releases of software products while respecting the most important constraints including technical, resources, budget and risks.

Strategic Release planning :

The act that allows scheduling CRs and planning their delivery in upcoming releases is considered as a **strategic** release planning activity and not operational release planning (named also as a Software Delivery Roadmap)[5]. Strategic RP aims to select and assign requirements to sequences of releases such that important technical and resource constraints are fulfilled. Once a strategic plan is generated, a decision is made on which CR should be developed inside which release. However, operational planning focuses on the development of features identified in a single software release.

From [6] point of view, software release planning is a complex activity that takes into account the perspectives of project stakeholders, integration problems, functional and non-functional requirements, existing technologies, anticipating customer needs and demands, concurrency and other goals. The interdependencies among change requests can be also considered.

Release planning approaches

Release Planning exists on two main approaches. In [7] paper, Saliu and Ruhe have categorized software RP into ad-hoc RP approach and science based RP approach (this last is called also systematic approach) as illustrated InFig. 1.

Between the two approaches exists the hybrid approach. The hybrid RP approach is considered the most effective, given that it enjoys the benefits of both categories. State of art in software maintenance RP indicates that, typically, the ad-hoc release planning is the most commonly used and is based on judgment and experience feedback of the maintenance team.

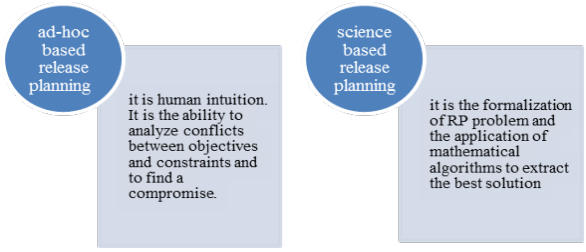

Fig.1: Release Planning approaches

Release planning is classified in the literature [8] as a "**Wicked problem**", it means a problem difficult to define and often, it has no clear and definitive solution.

TPM RP types

According to the standard classification of software maintenance made by ISO 14764 standard [9], there are four types of scheduled maintenance or change requests, distributed according to two dimensions: time and maintenance goal.

Table 1shows the four kinds of change requests requiring planning to software releases. They are: preventive, corrective, adaptive and perfective. Added to these four kinds the urgent corrective maintenance or urgent bug. Urgent bugs require flexible planning that accepts scope changes; it is also called "**non-plannable maintenance**" according to [10].

Grouping defects into larger projects or releases instead of delivering one by one may generate a cost reduction of up to 36%. This finding has been reported by Banker and Slaughter when investigating the benefits provided by the treatment of maintenance as software projects[8]. In order to support their

TABLE 1: SM CLASSIFICATION (ISO 14764)

	Correction	Enhancement /improvement
Proactive	Preventive	Perfective
Reactive	Corrective	Adaptive

claims, they have evaluated 129 software enhancement projects from a large financial organization. Their empirical evaluation has been conducted using a non-parametric method, called Data Envelopment Analysis (DEA), commonly used to measure productivity.

From the foregoing paragraph, three categories of software releases may arise in the context of TPM:

-Corrective releases: these software releases include only change requests of "corrective" type,

-Evolving releases: these software releases concern only change requests such as adaptation or enhancement,

-Mixed releases: these software releases, deal all types of change requests made by the customer, corrective and/or evolving. This last category is considered as risks generator. In software engineering industry, it is not recommended to deliver enhancements and corrections or fixes in a single software release.

The remainder of the article is structured in four sections. Section 2, "TPM **RP common problems**" discusses TPM RP challenges, constraints and types of software maintenance inside outsourcing context. Section 3 presents the adopted **systematic review process** to explore RP literature papers. Section 4 draws the systematic review **results**, classifies founded RP models considering the classification dimensions and explores software RP model families involved in RP literature. Section 5 shows the RP models that can be applied tothe TPM area before summarizing our research findings and present our future work.

2. TPM RP COMMON PROBLEMS

In this section, we shall address common problems that are faced by third-party organizations which weredeeply described in our papers[2]and [11].These problems are categorized according to three concerns: software maintenance problems, outsourcing problems, and managerial challenges. Software maintenance researches that were interested in TPM release planning areas are not rich enough. They can be grouped into two categories. The first are associated to release planning, research, which are applicable to software development projects including evolution phase. The second specifically deals with software maintenance projects with or without outsourcing. Indeed, TPM RP is different from software development RP, it is a particular kind of software project that require special management techniques. Below is a non-exhaustive list of some of these characteristics according to[12] and [13]:

- Software change requests (CR) arrive in a random way and are classified by a set of customer priorities that

can change at any time,
- CR size and complexity require intervention of only one or two persons,
- The main aim of the maintenance team is the daily health of software in production (maintain the product in Operational Condition).

Beyond characteristics of Software Maintenance (SM), TPM requires also to manage outsourcing constraints, which are namely:
- The compliance with contractual obligations of the subcontractor, in particular commitments on service levels and quality
- The customer need of a periodic reporting and visibility in terms of troubleshooting
- The lack of maintainer visibility on business and technical change requests context
- The long time to get feedback related to distance and time zone difference

RP for TPM remains a challenging task that requires specific approaches, models, and practices to satisfy the restrictive and constrained environment of TPM projects.

In order to complete the literature overview of constraints that prevent managers to build an effective TPM release-plan, we have considered the empirical study that was led by [14] on five banking projects. This study was provided a comprehensive list of RP challenges that the manager confronted in RP was compiled and broken down to four constraints categories. These challenges can be summarized as follows:
- Difficulty to answer the question "what is the delivery date of the next release?" by a systemic way. Delivery date must be suitable to all project stakeholders while respecting resource, budget, and scope constraints;
- Embarrassment to create a cooperation and discipline climate between project team members, especially when customer requirements are neither clear nor accurate;
- Limitation of the allocated resources. Indeed, the respect of resources constraints (people, budget and time) presents one of software RP challenges;
- Availability of adequate tools, tracking systems, and project monitoring techniques which help managers in RP process;
- size and complexity of functional and / or technical components and its dependence with other external systems, which increase the RP process complexity.

Planning the next software release (one release) does not solve the TPM RP problem. Over time, TPM customer needs change as long as his surrounding environment changes. The RP process has to distribute customer CRs to multiple releases, and handle the priority change while maintaining a strong involvement of project stakeholders. The capitalization on previous releases undoubtedly may improve this process.

TPM RP constraints

RP constraints can be discussed over four categories: **business** constraints, **resource** constraints, **system** constraints and **monitoring** constraints.

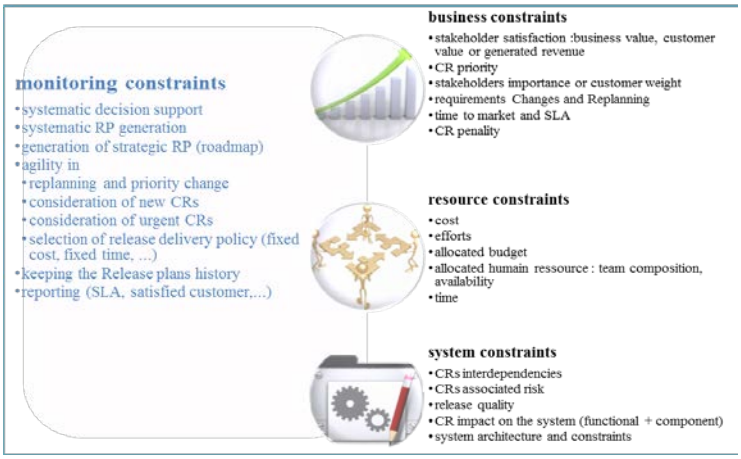

Fig.2 : TPM RP constraints and factors

Business constraints: it contains factors making change request with high visibility. We classify as business constraints all constraints helped to satisfy business stakeholders and evaluated by business stakeholders, as they know better the business context of each CR in terms of: market or business value and time to market (priority and emergency) required.
It may include also all factors satisfying software stakeholders or TPM subcontractor like maintainer revenue.
This category is called Soft factors, according to[2]. Soft Factors include those factors that are more difficult to estimate and provide exact numbers on, but may cause certain features or requirements being prioritized higher than others. Soft factors include Stakeholders Influence Factors, Value Factors, Risk Factors, and Resource Consumption Factors.

Resource constraints: it presents constraints for which a specified amount may be used during CR implementation in a release. This can then be matched against several constraints such as budget, effort, time, as well as resource constraints (materials, tools) including human resources.

System constraints: this category deals with constraints in the CR themselves and the ability to implement them in the maintained system. It includes also the quality required on the system, the system impacts and the generated risks when deciding about RP scope.

Our research separates resource constraints from system constraints. We have chosen for resource constrains external system factors like budget and resource availability. In contrast with[5] paper which combines resource and system categories in one category named hard constraints. This last includes factors that may restrict the order and the time when some features can be implemented. Hard constraints include Technical, Budget and Cost, Resource, Effort, and Time.

Monitoring constraints: this category includes decision support tool requirements, change management requirements, proactivity requirements, and agility in considering scope changes.
Fig. 2provides a summary view of constraints discussed in this

section.
Based on identified TPM Release planning constraints, we propose in the next section a classification framework to conduct the systematic review of existing RM models.

3. TPM RP MODELS: SYSTEMATIC REVIEW PROCESS

As part of this research, we have conducted a literature review to identify and assess existing release planning models. We present in this sub-section our conducted systematic review of software engineering models dealing with the problem of release planning, particularly inside Third Party Application Maintenance context. This systematic review results helped us to deeply understand and categorize the most important selection factors and constraints-to be considered by TPM managers that were poorly covered in the literature.

2.3 Systematic review approach

Adopt a well-defined review methodology helps to obtain a credible result. Fig. 3 presents an out light method of our literature review that was founded on [15]paper methodology. It consists of three main stages which were enumerated as follows: **(1) problem formulation** through a list of research questions, **(2) research strategies** which define literature search terms, sources and selection criteria, **(3) RP models classification** which helps to classify RP models according to the dimensions and finally **(4) data analysis**.

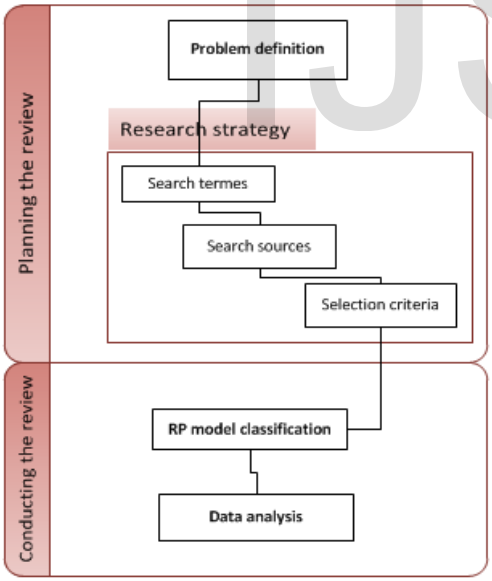

Fig.3 : Systematic RP review methodology

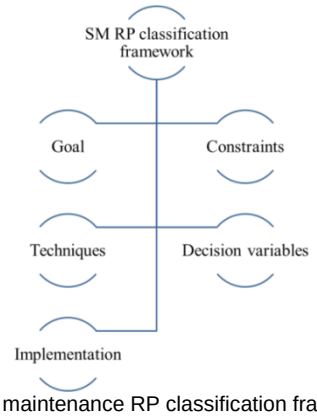

Fig.4 : Software maintenance RP classification framework

(1)Problem formulation

The first stage of our systematic review approach,as illustrated inFig. 3, was to define a review protocol that specifies research questions that be addressed. Our study concerns the four research questions (RQ) that are formulated below

- RQ1: what are the existing RP models that can be applied for evolving system?
- RQ2: which of TPM constraints(that were presented above in section 1.3) were be considered in these existing RP models?
- RQ3: which RP models were been applied or can be applied to TPM?
- RQ4: which resolution techniques are involved in these RP models?

(2)Research strategies

Search terms definition and literature sources identification are the first steps of research strategies stages. To define search terms, the identification of keywords from problem formulation was conducted. Search of synonyms and alternatives of major terms was necessary based on similar papers and books. The resulting search terms are: software, release, planning, plan, requirement, selection, planning, classification, change request, maintenance, grouping, schedule, and defect. After that, Boolean operators like "And" and "Or" to link them were performed during the research.
IEEE, ACM digital library, CiteSeerX, ScienceDirect, Springer through Google scholar motor were the electronic library sources used to extract papers for this research. The reference lists of all relevant papers were stored and organized using a free bibliographic tool named Zotero. Zotero helps to detect duplicate papers, renames properly founded bibliography, stores automatically and generates bibliography scripts. It also facilitates the navigation in paper through several search options.
More than **150 studies** and papers were obtained from the research stage. Critical check and examination were performed to stream-line these studies to the relevant ones.
Inclusion and exclusion criteria that were considered during this step are presented below in order.
-Inclusion criteria are: title of each study, bibliography, ab-

stract, studies in French and English language from peer reviewed journals, conference proceedings, workshops, symposiums and books. And finally, studies that were published from 1991 until December 2014.

-Exclusion criteria are: papers that arenot linked with the four questions of this study, duplicated papers, papers concerning product line and Commercial Off The Shelf (COTS) software and papers that considers new software development projects.

(3)Classification and data analysis

To better analyze collected data, we proposed in the last stage of this review a classification framework thatcategorizes existing RP models. Giving an objective function, set of constraints and decision variables; RP is about employing a technique or more to find the optimal combination of CR to deliver in the next releases. Based on RP literature and the four questions formulated in subsection "**Problem formulation**", a classification framework of RP studies was constructed based on the most considered dimensions. Fig. 4 shows the proposed classification framework for RP models in the software maintenance area based on the five dimensions below:

Goal dimension: the goal dimension presents the model expectations related to RP problematic. It can be a CR prioritization goal, CR grouping for the next release goal, CR scheduling goal or others. A model goal is formulated and presented with an objective function subject to maximize or minimize or multi-objectives functions. For example, the goal of the cost-value approach proposed in [16] is to prioritize requirements by selecting requirements with the highest ratio value/cost.

Decision variables dimension: in RP, this dimension presents the inclusion or the exclusion decision of a CR. Generally, decision variables are presented with the decision vector x =(x(1); x(2), …,x(n)) € (0, 1) which determines the CR that are to be satisfied in the next release. In this vector, x(i) is equal to "**1**" if CRi is selected and equal to "**0**" otherwise.

Constraints dimension: selection factors, constraints, input and criteria are all terms describing elements to consider when planning to release. The "Constraints" Dimension of RP classification framework classifies existing RP models based on inputs and factors considered when assigning CR to releases. We organize this dimension in tree sub dimensions as shown in Fig.5.

- **Business constraints**: each change request has an implicit or explicit impact in customer profit specifically and in business globally. To satisfy customers, it's necessary for the maintainer to increase business value generated by CR. In the other side, each CR for maintenance constitutes a source of revenue or penalty following contract engagement and so on. Business constraints include constraints related to customer satisfaction, service level agreement commitment and all aspects that may affect the business, revenue for both customer and maintainers.
- **System constraints**: these are constraints related to functional and technical characteristic of CR or software. It includes also dependencies between requirements.
- **Resource constraints**: resources are an essential part of the RP. They refer to any input to the software production process.

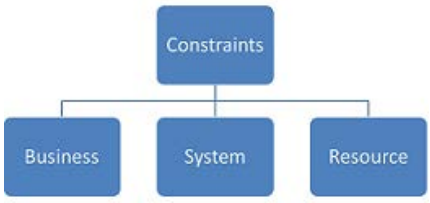

Fig.5 : Constraints dimension of classification framework

In software maintenance engineering, CR requires a budget to be managed and implemented. It uses also human and/or material resources (hard and soft). Usually, these resources relate to both budget and effort consumption, and there are bounds on the maximum capacities available for each resource type in each release cycle. Cost remains the most considered factor in RP models.

Techniques dimension: this dimension of the RP classification framework classifies existing RP model based on optimization techniques and algorithms used and considered to resolve release planning models.

The **implementationdimension** considers how managed output elements of RP models are and analyzes if the RP model is tooled or not.

Furthermore, it is interesting on how this output element can be generated, validated and managed. As an example, how many releases are considered when planning CR, impact of CR numbers in RP model, if the RP model is validated with real data (from software industry) and so on.

In next sections of this paper, we propose to dig further the identified RP models in two times:

- In first time, we will provide an analysis of generic RP models that were relevant to general software development projects;
- In second time, we will illustrate some specific RP models that were designed to support software maintenance.

2.4 General analysis

This section presents and discusses results of our systematic review of RP models.

In order to perform an exhaustive search for primary RP models, we have considered the major database libraries: IEEEXplore, ACM digital library, CiteSeerX, ScienceDirect, and Springer through Google scholar.

Papers distribution: Among 80 research papers related to RP problem, only 36 studies which discuss or propose RP models were selected (19 were published in journals, 15 papers appeared in conference proceeding and 2 as parts of book sections).

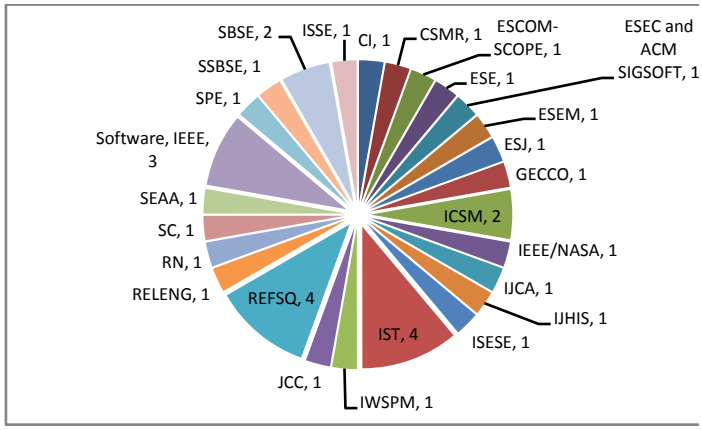

Fig.6 : paper distribution

Fig.6 shows publication sources of all selected RP papers and Fig.7 shows their distribution over the years.

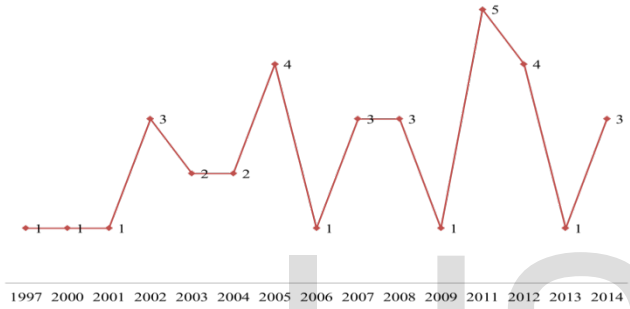

Fig.7 : paper distribution over years

Goal dimension overview: RP models aim to resolve CR selection problem which involved several selection factors and constraints. These constraints are used to define a mathematical objective function, subject to maximize or minimize. After this review, we note that customer satisfaction and cost minimization are the most relevant goals over the 36 papers according to Fig.8.

The respect of RP constraints appears as an important goal on all examined RP models but these constraints do not address the TPM contractual commitment in terms of SLA and penalties.

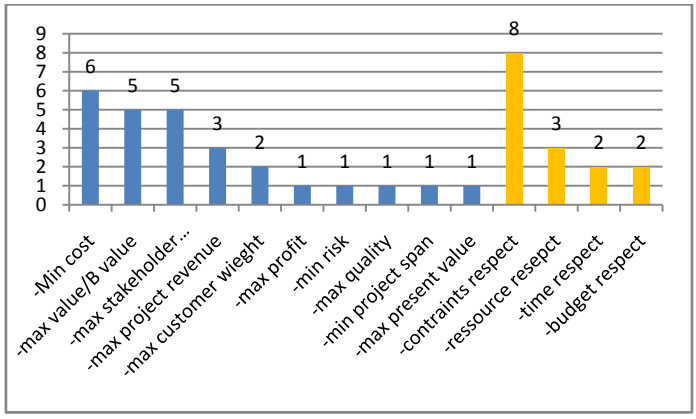

Fig.8 : papers over goals dimension

Most of RP models aim to primarily satisfy customers by maximizing the CR generated value and optimizing costs. Customer satisfaction goal is performed through maximizing values that affect directly the customer. These values are: the business value generated and assessed by the client for each CR, the number of customers having important weights, the customer profit, the quality of RP delivery, or the net present value. From the TMP point of view, the customer generated value that must consider the CR priority and the CR customer weight.

As opposed to the last point, some RP models (=3) were focused on the maintainer benefit that can be produced within an optimal RP, this is by the mean of maximizing the maintainer project revenue (against the consideration of the available resources in a given time period).

TPM RP Constraints dimension overview: 15 Software RP constraints were identified as part of this research. The most considered constraint is the cost of CR, thereafter, CR dependencies and CR priority from customer point of view. Fig.9 presents the distribution of the studied RP models over TPM RP constraints.

We observed clearly that SLA constraint was been managed by one RP model which was interested to plan resources for the only non-planned CRs and that the time constraint was been addressed by 3 RP models, as a resource that must be checked before each release.

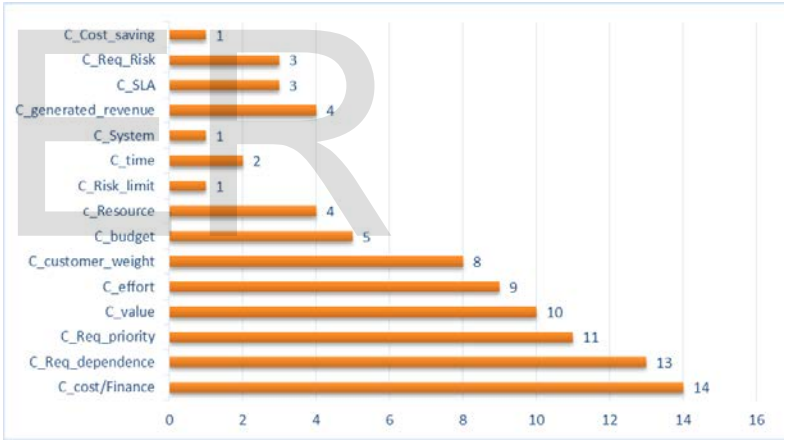

Fig.9 : Constraints dimension overview

C_ : prefix that signifies a constraint. Req_ :for requirement

RP Technique dimension overview: 14 RP techniques are identified over this study, there were used to resolve RP models. Fig.10 illustrates the distribution of those techniques over studied papers. We note that the most used techniques are:

- ILP or Integer Linear programming
- Genetic algorithm
- AHP or Analytic Hierarchy Process

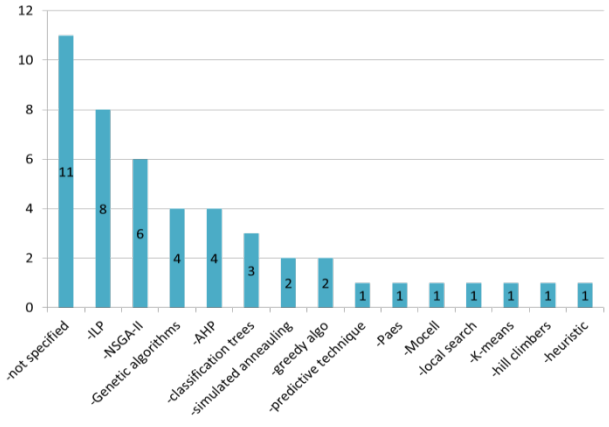

Fig.10: Techniques dimension overview

Decision variables overview: Generally, studied papers propose a RP formulation that uses the generic model of the binary knapsack problem. This last define decision variables as decision vector x =(x(1); x(2),…, x(n)) in (0, 1), x(i) is equal to ″1″ if CRi is selected and equal to ″0″ otherwise.

Implementation dimensions overview:3 tools were been identified as part of this systematic review. ReleasePlanner, the tool of Evolve models, is considered the most completed among the studied papers.

2.5 Detailed analysis

In this section, we propose to deepen the analysis of RP models identified by the systematic review in two stages: in a first time, an analysis of models relevant to general software development, in a second time, models designed for software maintenance.

Generic models

Among the generic RP models that were identified by the systematic review and were deeply studied, we propose to present the followings approaches: Hybrid intelligence Approach, IFM model, Optimization based techniques, COVAP and OVAC, and Provotype.

Hybrid Intelligence Approach : Ruhe and Greer suggested in their paper [17] a planning technique based on iterative and incremental architecture called ″EVOLVE″ which is mainly determined by genetic algorithms.

Considering priorities expressed by project stakeholders and estimated effort for each requirement, EVOLVE is proposed for incremental software development and addresses the following problems:

It proposes penalties if requirements are planned in reverse order of priority.

It considers that requirements are planned in several software increments,

It examines dependencies between requirements like precedence or coupling among the same software release,

This RP model generates alternative solutions and offers the choice to the manager to adopt the most suitable solution.

To address uncertainty issues associated with the implementation of each requirement, EVOLVE+, the enhanced version of EVOLVE, introduced the risk factor [18].

The risk factor is defined as any event that may negatively affect schedule, cost or quality of the project. This model (EVOLVE +) has been validated by academic case through the use of genetic algorithms to solve the problem and optimize proposed solutions. However, it is not validated by industry (complex or large projects).

To overcome the lack of human decision consideration when selecting the release plan, a new adaptation of EVOLVE model is presented in the paper [19] as a hybrid model. The purpose of the EVOLVE* model is not limited to intelligent calculations of planning generation, this model support the use of expert judgment in decision making. EVOLVE* is defined by an optimization problem based on linear programming and involves three phases:

Model variables: resource constraints, objective, and votes,

Explore alternatives generated by a tool,

Consolidate solutions by human decisions

EVOLVE* combines the automatic planning generation and human decision on RP. However, it produces a schedule of only two increments (software releases) and does not provide a roadmap to project manager for a specific period with several releases.

On an evolving software restricted area, the study carried by [20] proposes an extension of the EVOLVE* model, called S-EVOLVE*. This last is dedicated to the evolving software maintenance. RP based on S-EVOLVE focuses on the features of the software in maintenance when implementing the requirements. It adds to the initial problem of EVOLVE* the quantification of impact identified on software components.

Despite what precede, RP models derived from EVOLVE address RP problem based on the concept of decision variables listed in the Table 2.

Table 2 : EVOLVE family RP model

Dimension	EVOLVE RP model
Goal	Assignment of requirements to releases such that all technical and budget constraints are fulfilled
Business constraints	Requirement Business value, Importance degree of stakeholder, Requirement priority given by project stakeholder
System constraints	Dependence between requirements : Coupling and precedence, features of impacted system components (S-EVOLVE*)
Resources constraints	Effort, risk(introduced by Evolve+), time and budget
Techniques	AHP, ILP and Genetic algorithms

EVOLVE* is supported by a decision tool called ReleasePlanner (www.releaseplanner.com). ReleasePlanner is a tools suite, which provides a flexible and web-based support for RP. It is developed in the Laboratory for Software Engineering Decision Support (http://www.sengdecisionsupport.ucalgary.ca), University of Calgary, Canada.

IFM model (Denne and Cleland-Huang): Incremental Funding Method(IFM) is a data driven approach for RP that pro-

vides financial information on the software under development as proposed by [21]. IFM is designed to maximize income (NPV: Net Present Value) by delivering sets of features and adding value to the customer. The aim of this approach is to allow the subcontractor, particularly outsourcing subcontractor, to respect the allocated budget and to minimize development cost as low as possible to maintain a reasonable profit margin,thus, by ensuring that the delivery time was optimized to deliver value to the customer.

IFM decomposes software into smaller functional units called MMF (Minimum Marketable Function) that can be delivered quickly and can provide a business value to the customer. MMF is measured in terms of tangible and intangible factors which include the income generated by the functionality, cost saved, competitor differentiation, brand, customer loyalty, dependence between MMF, and time adequate to deliver based on the calculated NPV of each period.

We note that IFM is adapted for software in development phase and adequate for enhancive maintenance rather than the corrective ones, because it is based on decomposition of the target system to small units that is difficult to realize in TPM context with limited budget and resources. Indeed, IFM addresses the RP problem based on decision variables, constraints and techniques listed in the Table 7 mentioned below.

Table 3 : IFM RP model

Dimension	IFM RP model
Goal	Maximize the Net Present value
Business constraints	Requirement business value, generated revenue
System constraints	Dependence between requirements
Resources constraints	Cost and saving cost, right delivery time, risk
Techniques	Heuristic

Optimization-Based Techniques: Bagnall optimization model [22] is one of RP techniques that is based on optimization techniques and aims to resolve the Next Release Problem (NRP). This model assigns weights to customers according to their importance degree in the business. It aims also to identify customer's subset to satisfy within the allocated budget. This technique focuses on customer satisfaction and does not address all TPM problems that were mentioned previously. **Erreur ! Source du renvoi introuvable.**presents NRP decision variables, constraints and used techniques.

Table 4 : NRP RP model

Dimension	NRP RP model
Goal	Find subset of important customer whose CR tobe satisfied within their budget
Business constraints	Customer Weight
System constraints	Dependence between requirements
Resources constraints	Estimated cost and budget
Techniques	IPL, greedy algorithm, local search

COVAP (or Cost-Value Approach for Prioritizing Requirements) is a RP model introduced by Karlsson and Ryan in paper [16]. It is a requirements prioritization approach that recognizes to arrange requirements according to two dimensions: customer value and implementation cost. These lasts are calculated by comparing requirements pairs according to AHP (Analytic Hierarchy Process) technique. COVAP aims to provide maximum customer value with minimal cost.

AHP technique is difficult to apply when requirements number becomes important. For this raison, Jung has proposed in paper[23] an extension of the COVAP model that uses optimization techniques which were allowed overcoming AHP technique limitations, this extension is called OVAC(Optimizing Value and Cost in Requirements Analysis). A summary of COVAP/OVAC used decision variables, constraints and techniques is given in Table 9.

Table 5 : COVAC/OVAC RP model

Dimension	COVAP/OVAC RP model
Goal	Prioritizing requiremeents by selection the CR higher ratio calue/cost
Business constraints	Customer value
Resources constraints	Cost
Techniques	AHP

Provotype model: Provotype is a market-driven RP tool developed by Carlshamre[24]. Assuming that requirements cannot be partially selected in a software release, Carlshamre was implemented the RP problem on principle of a binary bag "**knapsack problem**". This toolis based on selection factors: value, estimated resources, and interdependencies and aims to produce a number of release-plan suggestions. However, it must be improved to manage several design shortcomings, among them we note.

Value of release that is hard to define,

Criteria(value, cost) that cannot defined in advance,

Judgments that are typically subjective when comparing requirements

Table 10 presents a summary of provotype decision variables, constraints and techniques

Table 6 : Provotype RP model

Dimension	provotype RP model
Goal	Selecting an optimal subset of CR where maximizing profit contribution and respect resources constraints
Business constraints	Customer value
System constraints	Dependence between requirements
Resources constraints	Estimated resources, budget
Techniques	AHP

TPM Specific models

Software release planning models that were applied to software maintenance, and were examined thereafter are: EMFEM, MANTEMA, New approach for software RP, and PASM.

EMFEM model: EMFEM (Estimation-based Management Framework for Enhancive Maintenance) is a framework that was proposed in paper [25], for high level management. It

allows producing a maintenance release-plan of requirements. This framework needs a periodic capture of requirement implementation costs in order to examine the balance between the needed and the available effort along the project progresses, as mentioned in Table 11.
Although this framework focuses on releases monitoring through a continuous verification of the "resource capacity" constraint, it does not provide a solution to maintenance problems such as SLA, customer satisfaction and so on.

MANTEMA model: Schedulable maintenance, which includes non-urgent requests, was addressed by several studies. Additionally, to prevent economic loss in software maintenance context, MANTEMA model has presented a new approach that proposes to estimate the needed quantity of human resources to handle the non-schedulable Maintenance. This last concerns the unexpected and urgent corrective customer requests[10].
Software maintenance releases planning process involves estimating the required resourcesto deal with customer change requests.This operation is relatively simple and clears as reported in the paper [10], however, determining the needed resources for urgent and unexpected defects is complicated and often omitted.MANTEMAmodel is a based on predictive techniques and economic parameters that prevent corrective change requests arrival. It produces a (future) distribution of change requests that allows allocating the required resources for a given period.

Table 7: MANTEMA RP model

Dimension	MANTEMA RP model
Goal	Estimate resources to be devoted to the non-plannable maintenance
Business constraints	Economic model of the maintenance project and SLA
Techniques	Predictive technique

New Approach to the Software Release Planning: The paper [26]has proposed a multi-objective approach to software RP problem that has two goals: maximizing customer satisfaction and minimizing project risks. The resolution of this approach through metaheuristic techniques was produced a high quality solutions in comparison with statistical algorithms or human decisions ones.
The main advantage of this approach is the consideration of risk factor which is rarely treated by other RP models. However, this model did not manage directly SLA and time constraints and their authors have considered that requirements priorities is the more effective way to deal with these constraints.

PASM: According to [5], PASM (Process for Arranging Software Maintenance Requests) is a lightweight process that advocates regrouping maintenance CRs into projects for the following reasons:
to facilitate planning and control actions;
to benefit from software engineering best practices (such as requirements specification, design, testing, etc.).
to take advantage of economies of scale that can occur when small tasks are grouped into large projects.
PASM proposes to provide a periodic maintenance policy that is based on three main steps below:
registration : this first step collects received CRs for a time period,
-grouping : in this step, collected CRs (by the previous step) will be grouped in software releases in regards with two factors: project size factor which is must be compatible with the maintainer capacity and functional similarities factor that may exist between CRs.
-treatment : this final step concerns the construction and delivery of software releases that was defined by the previous step.
The PASM process puts ahead the benefit of grouping CRs in projects of software releases, which helps to save the allocated resources (costs). However, this process does not consider either customer's business factors to evaluate the customer satisfaction, or risks associated with grouped CRs.

3.5 RESULTS AND DISCUSSIONS

Considering RP models that were presented and discussed in previous sections and based on common TPM problems, our systematic review was allowed us to understand advantages and limitations of existing RP models and moreover,to reveal the following observations which have been illustrated in Fig. 11.

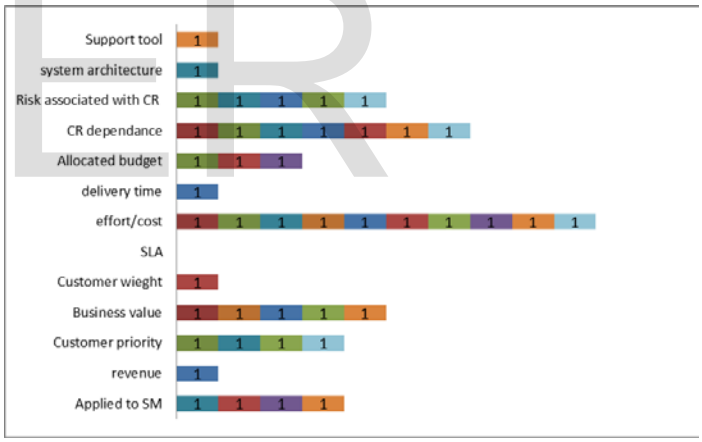

Fig.11 : constraints coverage of the RP models

We note that existing RP models do not address all constraints of TPM RP problems and none of studied RP models was interested with corrective maintenance in outsourcing mode.
Additionaly, Most of the studied RP models propose an approach which takes into account development costs (100% of studied models) and dependencies that may exist between requirements. However, SLA commitment constraints were not been addressed. In addition, composition of maintenance team, and system architecture were also few addressed by researchers.
Studied RP models were often limited to one release (next release) or two software releases. In TPM context, manager and customers need a full visibility through a strategic RP approach.
Risk factor that is associated with each requirement implementation was been evaluated differently in each RP model,

while some models evaluate it by penalties, others assume manage it using other constraints such as the dependency that may exist between requirements or customer priority.
As a decision support tool, only ReleasePlanner tool for Evolve RP models is considered as a valid tool that can be used in TPM industry.
Studied RP models neither manage directly SLA constraints nor take it into consideration by minimizing penalties that can be generated if the maintainer does not respect contractual agreements.

We conclude from what precede that no one of the studied RP models wereaddressedfully constraints and problems of TPM projects. Indeed, Evolve* RP models can be evolved and adapted to manage TPM specificity.

4 Conclusion

In light of tThis study was conducted a literature overview of TPM RP common problems to identify the most considered constraints when planning TPM releases which was helped us as part ofto identify and assess TPM existing release planning models from the TPM perspective. In ,tthis paper, we have presented a systematic review approach and a classification framework of existingsoftware engineering models dealing with the problem of software release planning, particularly inside from the Third Party Application Maintenance context-perspective. It was helped usto identify and categorize the most important selection factors and constraints to consider by TPM managers, poorly covered in the literature.
Our systematic review revealed that the most RP models target goals aim maximizing customer value (which signifies stakeholder satisfaction) and minimizing cost, while considering the highest constraints: cost and change request priorities. Moreover, Integer Linear Programming remains was identified as the most used technique for resolving release planning problemsmodels. We have noted through this paper study that the existing RP models do not cover all TPM RP problems, especially SLAs constraints.
These findings will show the need to assist us to propose a RP models appropriate to TPM context, allowing to meet TPM manager's needs with the aim to design a RP decision support system, helping to produce effective schedules of software maintenance without advanced knowledge (technical or functional) on change requests, while respecting project constraints.

References

[1] R. Canning, «The Maintenance Iceberg,» *EDP Analyzer,* 1972.

[2] S. Naciri et M. A. Janati Idrissi, «Third-Party Application Maintenance Management,» *International Journal of Computer Applications,* oct 2014.

[3] M. N. A. Rahman et S. Suhailan, «Managing Software Change Request Process: Temporal Data Approach,» *International Journal of Computer Science and Security (IJCSS),* 2009.

[4] G. Ruhe et M. O. Saliu, «The art and science of software release planning,» *IEEE Software,* 2005.

[5] M. Svahnberg, T. Gorschek, R. Feldt, R. Torkar, S. B. Saleem et M. U. Shafique, «A systematic review on strategic release planning models,» *Information and Software Technology,* 2010.

[6] P. Bhawnani et G. Ruhe, «ReleasePlanner-Planning new releases for software maintenance and evolution,» *In Industrial Proceedings of the 21st IEEE International Conference on Software Maintenance,* 2005.

[7] S. Keele, «Guidelines for performing systematic literature reviews in software engineering,» Technical report, EBSE Technical Report EBSE, 2007.

[8] A. Seyed Danesh, R. Ahmad, M. R. Saybani et A. Tahir, «Companies Approaches in Software Release Planning -- Based on Multiple Case Studies,» *Journal of Software,* 2012.

[9] «International Standard of Software Engineering, Software Life Cycle Processes and Maintenance,» *ISO/IEC 14764:2006 (E) IEEE Std 14764-2006 Revision of IEEE Std 1219-1998),* 2006.

[10] M. Polo, M. Piattini et F. Ruiz, «Using a qualitative research method for building a software maintenance methodology,» *Software: Practice and Experience,* 2002.

[11] S. Naciri, M. A. Janati Idrissi et N. Kerzazi, «A strategic release planning model from TPM point of view,» *IEEE,* pp. 1 - 9, 2015.

[12] A. Abran et H. Nguyenkim, «Measurement of the Maintenance Process from a Demand-based Perspective,» *Journal of Software Maintenance: Research and Practice,* 1993.

[13] A. April, «Studying Supply and Demand of Software Maintenance and Evolution Services,» *In Quality of Information and Communications Technology (QUATIC), 2010 Seventh International Conference,* 2010.

[14] A. S. Danesh et R. Ahmad, «Software Release Planning Challenges in Software Development: An Empirical Study,» *African Journal of Business Management,* 2012.

[15] Galaup et Nurcan, «Evolution de la maturité du processus de maintenance du logiciel dans une organisationen mode projet,» *Université Paris1 Panthéon-Sorbonne, Institut d'administration des entreprise de Paris,* 2009.

[16] J. Karlsson et K. Ryan, «A Cost-Value Approach for Prioritizing Requirements,» *IEEE Software,* 1997.

[17] D. Greer et G. Ruhe, «Software Release Planning: An Evolutionary and Iterative Approach,» *Information and Software Technology,* 2004.

[18] G. Ruhe et D. Greer, «Quantitative Studies in Software Release Planning under Risk and Resource Constraints,» *Empirical Software Engineering,* 2003.

[19] R. G. et A. Ngo, «Hybrid Intelligence in Software Release Planning,» *International Journal of Hybrid Intelligent Systems,* 2004.

[20] O. Saliu et G. Ruhe, «Software release planning for evolving systems,» *Innovations in Systems and Software Engineering,* 2005.

[21] M. Denne et J. Cleland-Huang, «The Incremental Funding Method: Data-Driven Software Development,» *Software, IEEE,* 2004.

[22] A. J. Bagnall, V. J. Rayward-Smith et I. M. Whittley, «The next Release Problem,» *Information and Software Technology,* 2001.

[23] H. W. Jung, «Optimizing value and cost in requirements analysis,» *IEEE software,* 1998.

[24] P. Carlshamre, «Release Planning in Market-Driven Software Product Development: Provoking an Understanding,» *Requirements Engineering,* 2002.

[25] D. Penny, «An Estimation-Based Management Framework for Enhancive Maintenance in Commercial Software Products,» *Software Maintenance,*

2002.

[26] F. Colares, J. Souza, R. Carmo, C. PÃdua et G. R. Mateus, «A new approach to the software release planning,» chez *SBES'09. XXIII Brazilian Symposium of Software Engineering*, 2009.

[27] G. Ruhe et A. Ngo, «Hybrid Intelligence in Software Release Planning,» *International Journal of Hybrid Intelligent Systems,* 2004.

[28] G. A. Junio, M. N. Malta, H. A. Mossri, H. T. Marques-Neto et M. T. Valente, «On the benefits of planning and grouping software maintenance requests,» *Software Maintenance and Reengineering (CSMR), 15th European Conference,* pp. 55-64, 2011.

[29] R. D. Banker et S. A. Slaughter, «A field study of scale economies in software maintenance,» *Management Science,* vol. 43, n° %112, 1997.

[30] M. Polo, M. Piattini et F. Ruiz, «Planning the non-plannable maintenance,» pp. 49-57, 2000.

[31] G. A. Di Lucca, M. Di Penta et S. Gradara, «An Approach to Classify Software Maintenance Requests,» *Proceedings of International Conference on Software Maintenance,* 2002.

[32] O. Saliu et G. Ruhe, «Supporting Software Release Planning Decisions for Evolving Systems,» *29th Annual IEEE/NASA In Software Engineering Workshop,* 2005.

[33] P. Baker, M. Harman, K. Steinhofel et A. Skaliotis, «Search based approaches to component selection and prioritization for the next release problem,» 2006.

[34] C. Li, V. D. A. J. M., B. Sjaak et D. Guido, «Integrated Requirement Selection and Scheduling for the Release Planning of a Software Product,» *In Requirements Engineering: Foundation for Software Quality, Springer,* 2007.

[35] M. O. Saliu et G. Ruhe, «Bi-Objective Release Planning for Evolving Software Systems,» *Proceedings of the the 6th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering,* 2007.

[36] Y. Zhang, M. Harman et S. A. Mansouri, «The Multi-Objective next Release Problem,» *In Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation,* 2007.

[37] G. Ruhe, «A systematic approach for solving the wicked problem of software release planning,» *Soft Computing,* 2008.

[38] A. Szoke, «A Proposed Method for Release Planning from Use Case-Based Requirements Specification,» *34th Euromicro Conference on Software Engineering and Advanced Applications, SEAA'08.* , 2008.

[39] J. J. Durillo, Y. Y. Zhang, E. Alba et A. J. Nebro, «A Study of the Multi-Objective next Release Problem,» *1st International Symposium on Search Based Software Engineering,* 2009.

[40] G. F. Fabrício, P. C. Daniel et T. S. Jerffeson, « "Software Next Release Planning Approach through Exact Optimization,» *International Journal of Computer Applications* , 2011.

[41] B. Regnell et K. Kuchcinski, «Exploring software product management decision problems with constraint solving-opportunities for prioritization and release planning,» *Fifth International Workshop In Software Product Management (IWSPM),* 2011 .

[42] G. van Valkenhoef, T. Tervonen, B. de Brock et D. Postmus, «Quantitative release planning in extreme programming,» *Information and software technology,* 2011.

[43] X. Cai, O. Wei et Z. Huang, «Evolutionary approaches for multi-objective next release problem,» *Computing and Informatics,* 2012.

[44] S. Fricker et S. Schumacher, «Release planning with feature trees: Industrial case,» *In International Working Conference on Requirements Engineering: Foundation for Software Quality, Springer Berlin Heidelberg,* 2012.

[45] S. L. Gupta, R. Soni, A. Jolly et A. Rana, «OPTIMIZED APPROACH TO SOFTWARE RELEASE PLANNING WITH VOLATILE REQUIREMENT,» *European Scientific Journal,* 2012.

[46] M. Przepiora, R. Karimpour et G. Ruhe, «A hybrid release planning method and its empirical justification.,» *In Proceedings of the ACM-IEEE international symposium on Empirical software engineering and measurement, ACM,* 2012.

[47] B. E. Isong et O. O. Ekabua, «Towards Release Planning Generic Model: Market-Driven Software Development Perspective,» *Journal of Communication and Computer,* 2013.

[48] J. Ho, S. Shahnewaz et G. Ruhe, «A prototype tool supporting when-to-release decisions in iterative development,» *In 2nd International Workshop on Release Engineering (RELENG),* 2014.

[49] M. R. Karim et G. Ruhe, «Bi-objective genetic search for release planning in support of themes,» *In International Symposium on Search Based Software Engineering. Springer International Publishing.,* 2014.

[50] Y. Zhang, M. Harman, G. Ochoa, G. Ruhe et S. Brinkkemper, «An empirical Study of meta-and hyper-heuristic search for multi-objective release planning,» *RN,* 2014.

[51] M. Van den Akker, S. Brinkkemper, G. Diepen et J. Versendaal, «Software Product Release Planning through Optimization and What-If Analysis,» *Information and Software Technology,* 2008.

[52] M. van den Akker, S. Brinkkemper, G. Van Diepen et J. Versendaal., «Flexible Release Planning Using Integer Linear Programming,» *In Proceedings of the 11th International Workshop on Requirements Engineering for Software Quality (REFSQ'05),* 2005.