2014 47th Hawaii International Conference on System Science

Theme-based Product Release Planning: An Analytical Approach

Nishant Agarwal
BITS Pilani University, India
nishant21591@gmail.com

Reza Karimpour
University of Calgary, Canada
reza.karimpour@ucalgary.ca

Guenther Ruhe
University of Calgary, Canada
ruhe@ucalgary.ca

Abstract

Release planning is part of iterative software development and strongly impacts the success of a product by providing a roadmap for future releases. As such, it is of key importance for lean and agile organizations. Often features are highly dependent on each other and the value of a release is influenced by a set of bundled features constituting a theme.

This paper addresses the topic of theme-based release planning. Themes might be defined, manually, upfront or as the result of computer-based analysis. In this paper, we propose an analytical approach to detect themes from a given set of feature dependencies. On top of an existing release planning methodology called EVOLVE II, our approach applies clustering performed on a feature dependency graph. The release plans generated from such an approach are a balance between two goals: (i) considering the values of individual features, (ii) detecting and utilizing synergy effects between semantically related features.

As a proof-of-concept, we present a case study addressing the theme-based release planning for 50 features of a text processing system. The preliminary evaluation results show improved release plans with regards to accommodating themes.

Keywords: Release planning, theme-based product release planning, feature dependencies, clustering, case study.

1. Introduction

As software and other products are getting more and more complex and larger in size, incremental and iterative development is rapidly replacing monolithic product creation approaches [1]. Release plans have to consider various criteria like time-to-market, customer satisfaction and risk [2]. Systematic methods for product release planning are seen as an approach to maximize stakeholder satisfaction, generate higher revenues and achieve resource-efficient development [3].

One of the key aspects of lean product management is to focus on creating value, rather than blindly shipping features. Certain features would have higher value when they are released along with a specific set of features, leading to better release plans [4], [5]. Offering semantically related features as a bundle helps the product manager to deliver products that are focused on a particular "theme" per release.

A *theme* is meta-functionality of a product release, integrating a number of individual features under a joint umbrella. It can be thought of as an abstraction, i.e., a group of features that are inter-related to each other in a way that they depict a context and can be viewed as a single entity from a higher level.

Theme-based release planning aims at offering features in a particular release in consideration of their semantic cohesiveness. The release plans generated from such an approach are a balance between two goals: (i) considering the values of individual features, (ii) detecting and utilizing synergy effects between semantically related features.

The main contributions of the paper are:

- A clustering based approach for soliciting themes from existing feature dependencies (value, effort or usage related).
- Analytical approach for theme-based planning which is based on an existing release planning optimization method called EVOLVE II.
- Performing a case study for planning of 50 features and 81 feature dependencies and subsequent comparison of the results gained from application of theme-based release planning versus planning on isolated features.

Through our approach, we were able to generate theme-centered release plans of proven degree of optimality in terms of overall value and quality. Besides, the managers are given the flexibility to administer the release planning process.

The paper is structured into eight sections. Section 2 describes related work. Section 3 presents a short description of the formal approach to theme-based release planning. The methodology and implementation with integrated tool support are presented in Section 4. The case study design is described in Section 5, followed by the results and key findings presented in Section 6. Threats to validity are

978-1-4799-2504-9/14 $31.00 © 2014 IEEE
DOI 10.1109/HICSS.2014.582

discussed in Section 7. A summary and outlook for future research is provided in Section 8.

2. Related work

2.1. Release planning

One of the most prominent issues involved in incremental and iterative software development is to decide upon which features should be offered when and why. This decision is inherently complex and of high impact on overall business success. Various formal and informal methods for release planning exist. For an overview, we refer to [3]. In Scrum [6], releases are planned using a simple greedy algorithm by selecting the highest priority product backlog items into proposed releases. Schwaber [7] suggests that the product owner should use a pool of imaginary Ping-Pong balls to allocate relative business value to backlog items. Greening [8] discusses various challenges and application of Enterprise Scrum for larger project teams.

Carlshamre [9] observed that a large number of features might be non-isolated and depend on other features. An overview of the state-of-the art in feature dependencies is given in [10]. A solution accommodating advanced feature constraints and its empirical evaluation is presented in [11].

EVOLVE II [3] is an evolutionary release planning method that tries to generate optimized and diversified release plan alternatives. It is a complete framework encompassing every stage of the release planning process, from modeling to stakeholder input, plan optimization, and post-optimization tasks like what-if analysis. EVOLVE II has an associated decision support system called ReleasePlanner (RP) [12] which is designed to implement and support the planning process. A stakeholder-centric variation of it was evaluated in [13] for the planning of an agile tool called *Agilefant*. In this paper, we apply ReleasePlanner with the advanced capabilities of [11] to accommodate more complex feature dependencies. This allows us to determine optimized release plans in consideration of extracted themes.

2.2. Theme-orientation

With regard to theme-based release planning, there have been a few attempts by researchers to group features into clusters, using different metrics and perform release planning using these clusters. Feature trees were used by Fricker and Schumacher et al. [14] to perform release planning by grouping features by constructing feature trees. They also model software evolution with introduction of new features through different graphs and visualizations. However, the entire process of construction of trees and decision making is mostly manual. Additionally, requirements in the software descriptions are not prioritized in their research. Even, some textual descriptions might lead to different interpretations resulting in improper groupings.

Agile and lean practices favor theme-based releases. Leffingwell, in [4], states that a release is characterized primarily by a release theme and a release date, followed by a list of prioritized features. He also mentions that in agile planning, a release plan constitutes a release theme, list of features, assumptions, dependencies and other components. It is mentioned that strategic product themes are realized by *epics* [5], and how epics are decomposed into specific features.

Finally, there have been studies on requirement clustering as well. For instance in [15], authors use clustering and visualization to facilitate the discovering of unknown requirement interdependencies.

3. Problem statement

In this section, a semi-formalized description of the theme-based release planning problem is given. The formulation is based on the definition of features and their interdependencies, the definition of resource constraints, and the description of the planning objective.

3.1. Features and their interdependencies

This paper uses the concept of a "feature" as the basic unit for release planning. Features are the characteristics of the product offered to the customer. According to the definition given by Wiegers [16], a product feature is defined *as a set of logically related requirements that provide a capability to the user and enable the satisfaction of business objectives*.

We assume a set of features $F = \{f(1), f(2), \ldots, f(N)\}$. The goal of theme-based release planning is to assign the features to a finite number K of release (or sub-release) options, such that the overall value of the releases is maximized, where the value is determined by both the value of the individual features and the contribution to a specific theme of the product.

Features are often dependent on one another. Many types of dependencies can exist between features. Some of the dependencies are described below:

Weak Precedence: Features A and B are in a *weak precedence* dependency if feature B cannot be offered in an earlier release than feature A. However, both the features can be offered in the same release.

Strict Precedence: Feature A and feature B are in *strict precedence* if feature A has to be offered in an earlier release than feature B. Both of them cannot be offered in the same release.

Coupling: Feature A *is coupled to* feature B signifies that both the features have to be offered in the same release creating a cyclic dependency between the two.

Synergy: If features A and B have a *synergy* dependency between them, then the value of both of these features increases by a fixed percentage (given by the user) in case they are offered together in a release.

NAND: If features A and B are connected together by a *NAND* dependency, then both of them cannot be offered together in a release.

Based on the nature of their dependency, we broadly classify feature dependencies into direct and indirect dependencies:

Direct Dependencies: They reflect a degree of similarity between features. Hence, features directly dependent are assumed to have a degree of commonality amongst them. Examples are coupling, weak precedence, and synergy.

Indirect Dependencies*:* They reflect a degree of dissimilarity between features linked together by such dependencies. Hence, features connected by such dependencies are assumed to differ from one another to some extent and their occurrence together in a release is not favorable. Examples are strict precedence, and NAND.

3.2. Resource constrains

Each feature consumes different types of resources for its implementation. Let us consider T different resource types being relevant to implement the features. Every feature f(i) requires an amount of human resources r(i,t). Every release option k has a certain amount of resource capacity of type t available. This capacity bound is denoted by Cap(k,t). Thus, the resource requirements for all features assigned to release k must satisfy the following constraints:

$$\sum_{i:\, x(i)=k} r(i,t) \leq Cap(k,t) \text{ for } k = 1 \ldots K \text{ and } t = 1..T$$

Besides human resources, features might consume financial resources as well. Human resources can also be expressed in monetary terms taking into account the financial effort to provide these resources.

In general, we have assumed linearity in the resource constraints, e.g. the financial effort for a set of features is defined as the sum of the financial efforts for all individual features. This is not necessarily fulfilled in all practical cases, but it typically represents a good compromise between meaningfulness of the model and the computational effort to solve it.

3.3. Objective function

Defining the objective of planning is critical for success of planning. Formulation of these objectives is difficult. As a simplification, objectives typically are formulated as an additive function defined on the set of features. For each individual feature, projections on its potential value, urgency, frequency of use, usefulness or risk of implementation are made. While often individual functions are pursued, it appears to be more realistic to consider a combination of them. We combine them linearly into a single objective function expressing the overall utility of a plan.

As discussed in [13], features are prioritized by stakeholders and resource constraints are applied before generating a release plan. The objective is the maximization of a function F(x) among all release plans x satisfying the technological and resource constraints, with added advantage for features which are part of a theme and occur together in a release. F(x) is composed of the weighted average priority vector defined for each feature f(n). For further details, see [3].

4. Theme-based release methodology

To keep focus on the main contribution of the paper, we concentrate on the new and additional aspects of the extraction and consideration of themes as part of the release planning process. On top of the established planning process of EVOLVE II, we describe three additional steps called graph transformation, clustering and theme-based plan generation. The three components are described below.

4.1. Graph transformation

4.1.1. Description. We consider the interdependencies between features as underlying criteria for clustering them into themes. As described earlier, we classify dependencies into two broad categories, i.e., direct and indirect. Direct dependencies reflect that there exists some similarity between the participating features. Similarly, indirect dependencies reflect that the participating features are dissimilar. As each

dependency has its own significance, we assign weights depending upon the importance of the dependency.

We construct a weighted graph G = (V,E), where V represents a set of the features of the graph and E is the set of edges representing the dependencies. If there is a constraint between two features, then an edge is added between them in E with positive (negative) weight for a direct (respectively indirect) dependency. We have used undirected edges because it is a generic approach that incorporates all types of dependencies, some of which are mentioned in the previous section. Directed edges could not have incorporated dependencies like synergy, NAND, XOR etc. Thus, an edge between two features depicts the presence of a constraint between them. The thickness of the edge represents the weight of dependency.

The process of graph transformation can be broken down into the following steps: First, select a constraint from a list of constraints. For that constraint, add features that appear in it as nodes, if they are not already present in the feature graph. Then an edge is added between every two features that are related to each other by a dependency in that constraint. The weight of an edge is determined by the type of dependency it visualizes. Weights can be pre-assigned to dependencies by the user. Finally, repeat the steps above for all constraints in the list.

4.1.2 Illustrative example. We consider a project with five features called A, B, C, D and E. Among them, the following three dependencies are defined:

i. A, B, C and D are mutually coupled
ii. A and C precede E
iii. A and C are mutually exclusive

To model the feature graph for this example, we start from the first dependency in which features A, B, C and D are involved. As seen in Figure 1(a), we create a graph in which each feature node is connected to all other nodes that participate in the dependency statement. The weight for each edge is set to one.

The second constraint links features A, C and E. In the model, A, C and E are connected (see Figure 1(b)). As A and C were already connected due to a previous constraint, we increase the weight of the edge connecting these two by one.

Finally, the third statement adds mutual exclusion constraint between A and C. Hence, to emphasize the negative effect of this constraint on previous constraints, we deduct one from the weight of the A-C edge (see Figure 1(c)).

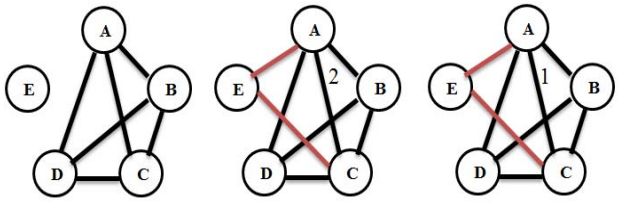

Figure 1. Graph transformation for sample dependencies

4.2. Clustering

Looking for themes from clusters of features, we applied the Chinese Whispers (CW) [17] algorithm. Form analyzing tool alternatives such as the one described in [18], we decided using CW, as it is simple, fast and interactive. The manager can alter the parameters in the algorithm to get different clusters.

CW is an efficient graph-clustering algorithm applicable to undirected, weighted graphs. It has a linear (in the number of edges) run-time complexity, which makes CW a very efficient algorithm. The output is a non-deterministic partitioning of the graph.

The algorithm performs in a bottom-up fashion: First, each node is assigned to a unique class. Then the nodes are processed for a specific number of iterations and inherit the strongest class in the local neighborhood. This is the class whose sum of edge weights to the current node is maximal.

In case of multiple strongest classes, one is chosen randomly. Regions of the same class stabilize during the iteration and grow until they reach the border of a stable region of another class. As the classes are updated immediately, a node can obtain classes from the neighborhood that were introduced there in the same iteration.

4.3. Creating theme-based planning alternatives

ReleasePlanner [12] is a planning tool that uses integer and constraint programming in conjunction with specialized heuristics to generate optimized release plans. It is designed to handle complex feature dependencies and to accommodate resource constraints. The tool takes a list of features, stakeholder priorities, resource constraints, and dependencies as input, and generates unique alternative release plans. Following the diversification principle

[3], it allows the manager choose the best plan for his purpose.

However, RP is not inherently designed to manage themes while making release decisions. For this reason, we add additional synergy constraints in accordance to the clusters to support themes in the planning process. We formulate synergy constraints between all pairs of features in a cluster, with an increment factor for a pair. Defining clusters like this is advantageous because it favors bigger subsets of clusters to be formed. This aspect is integrated into the overall objective function serving as the goal of planning.

Table 1. Feature list

Feature ID	Feature content
1-10	New, Open, Close, Save, Save as, Search, Protect, Print Preview, Print File, Send To.
11-20	Set Properties, Exit, Undo, Redo, Cut, Copy, Paste, Paste Special, Go To, Find.
21-30	Replace, Select All, Default, Print Layout, Web layout, Zoom, Header/Footer, Page Numbers, Date/Time, Symbol.
31-40	Bookmark, Hyperlink, Font, Paragraph, Bullets/Numbering, Change Case, Background, Help, Search, Insert Table.
41-50	Delete Table, Format Table, Import Data, Sort, Check Spell, Check grammar, Speech, Mail Merge, Macro, Set Options.

5. Case study design and implementation

5.1. Context

For the purpose of evaluating our approach, we tested the methodology on a word processing product that incorporates 50 features and 81 feature interdependencies. Release plans are generated looking for three releases (iterations) ahead of time. The dataset is taken from a graduate course project [19]. It is available online at http://pages.cpsc.ucalgary.ca/~ruhe/publications.htm#CopyOfDataset. As the project is derived from the commonly known word processing tool MS Word, some intuitive dependencies were added. The set of features is summarized in Table 1.

5.2. Implementation

We perform the steps outlined in Section 4 to generate theme-based release plans for the given dataset.

5.2.1 Graph transformation. We use an open source tool Gephi [20] for graph visualization and application of CW graph clustering. Dependencies are mapped as edges, with suitable weights for different kinds of dependencies, according to the rule stated earlier.

For our study, we used three kinds of dependencies, namely weak precedence, coupling and synergy, with weights of 1, 2 and 3, respectively. The weights are decided based on user's perception of the impact of each type of dependency.

5.2.2. Clustering. We now apply the CW algorithm to detect themes from the feature graph. This step favors stakeholder involvement from the beginning. The user may (or may not) group the features into rough clusters beforehand, and the algorithm will refine them and use them as a basic solution.

As the algorithm is interactive, for different configurations of input parameters (number of iterations=15, minimum edge weight=0.0, class propagation type=top), it produces different clusters. Hence, the product manager can choose the ones that fit his purpose. Also note that the results of clustering algorithm are meant to be taken just as a suggestion. Managers can alter the clusters and feed them in a release planning engine. For a specific input, the algorithm generates different solutions in each iteration. Figure 2 shows eight clusters generated by CW.

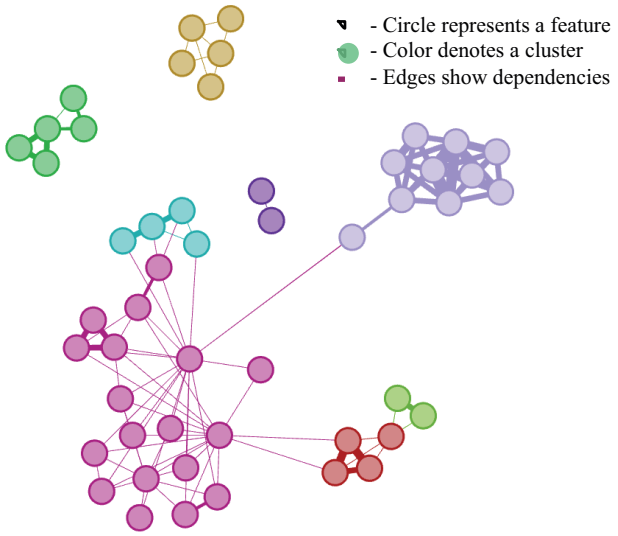

Figure 2. Clusters generated from CW

5.2.3. Generation of theme-based release plans. We use ReleasePlanner for finding optimized release plans, considering themes of products. We added a new dependency called *Synergy same release* into the tool and used it to add constraints between pairs of features in a cluster, as stated before. We then run the release planning optimizer to generate plans, which fulfill all

the added constraints as well. Alternative plans of different structure are generated, and the most suitable one is selected (see [3] for further details).

6. Case study results and key findings

The results of the case study are presented in Section 6.1. Three evaluation metrics are described in Section 6.2 and key findings of the case study are summarized in Section 6.3.

6.1. Case study results

Figure 2 shows a visualization of the results of the clustering algorithm. Detailed results are shown in Table 2. The nodes represent features and edges are the dependencies. The weight of each dependency is set by the user. The edge also becomes thicker as the number of dependencies between a pair of features increases. Each color represents a cluster. Both release plans (with and without application of theme-centric planning) are 100% optimal in their feature assignment.

Table 2. Clustering results from CW

	Cluster	Features
C1	Cluster 1	Insert Table, Delete Table, Format Table, Import Data, Sort.
C2	Cluster 2	Undo, Redo, Cut, Copy, Paste, Paste Special, Go To, Find, Replace, Select All
C3	Cluster 3	New, Open, Close, Save, Save as, Search, Protect, Print Preview, Print File, Send To, Set Properties, Exit, Default, Print Layout, Web layout, Zoom, Header/Footer, Page Numbers
C4	Cluster 4	Date/Time, Symbol, Bookmark, Hyperlink
C5	Cluster 5	Font, Paragraph, Bullets/Numbering, Change Case, Background
C6	Cluster 6	Help, Search
C7	Cluster 7	Check Spell, Check grammar, Speech, Set Options.
C8	Cluster 8	Mail Merge, Macro

Figure 3 shows a visualization of the release plan generated with theme support. The figure clearly depicts the number of features assigned to a release, portion of clusters allotted to different releases and the dependencies between them. The release plans don't totally depend on the themes that were fed as input. The results are a tradeoff between the most optimal assignment and theme incorporation. The vertices denote the features in a release with colors denoting different clusters. The release plans proposed with and without clustering are depicted in Table 3.

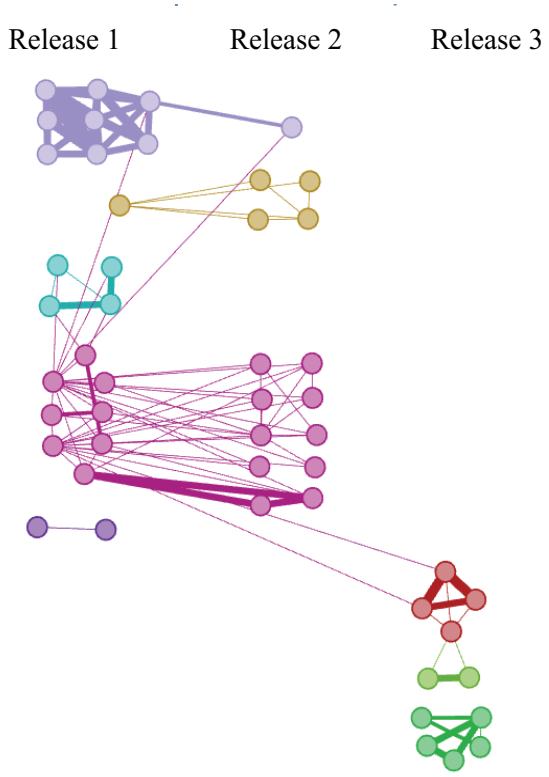

Figure 3. Release plans with theme focus

In Section 6.3, we discuss the key findings of this approach. The results clearly state that features in a cluster tend to stay together in a release. The methodology gives flexibility to the user to govern the clustering process. Also, the overall value of plans improved after applying our approach. The feature distribution improved because additional synergy was taken into account.

6.2. Evaluation metrics

We now empirically evaluate our results based on three metrics aimed to characterize the degree of theme cohesiveness.

6.2.1. Metric M1. Definition**:** For a given plan x, M1(j,k,x) describes the percentage of features of cluster j assigned to release k (related to the total number of features of cluster j).

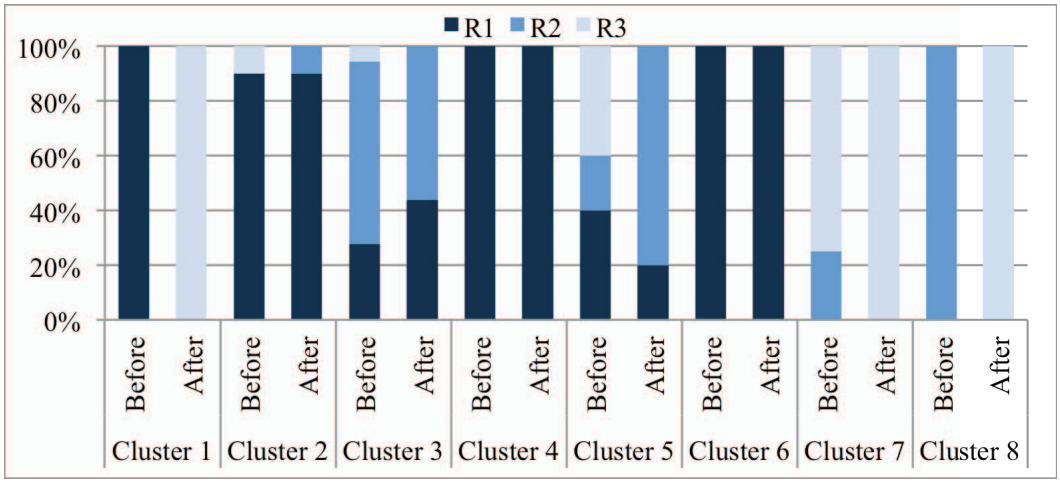

Figure 4. M1 values of plans before and after clustering

Figure 4 shows the results of the evaluation in terms of M1 applied on release plans generated with and without application of the theme focus.

6.2.2. Metric M2. Definition**:** For a given plan x, M2(j,k,x) describes the percentage of features of cluster j assigned to release k (related to the total number of features of release k).

Figure 5 shows the results of the evaluation in terms of M2 applied on release plans generated with and without application of the theme focus.

6.2.3. Metric M3. Definition**:** For a given plan x, M3(j,k,x) describes number of interdependencies amongst the features released in release 'k'. In other words, it is the number of edges in the feature graph, between the features released in a release 'k'.

Table 4 shows the results of the metric applied on release plans, before and after applying our approach.

6.3. Key findings

This section describes the key findings based on the case study and its detailed analysis.

6.3.1 Features in a cluster tend to stay together. We added synergy constraints for features in a cluster, if they occur together. Hence, in the final release plan, features in a cluster tend to be released together. The RP tries to find a tradeoff solution between the optimum feature assignment, and an assignment that puts all features of a cluster together in one release.

Table 3. Release plans with and without the application of theme based approach

Release	Feature set without theme based approach	Feature set with theme based approach
1	Insert Table, Delete Table, Table Format, Sort, Import Data, Help, Search, Font, Bullets/Numbering, New File, Open File, Close File, Page Numbers, Header/Footer, Undo a Task, Select All, Cut, Copy, Paste, Paste Special, Go To, Find, Replace, Date/Time, Symbol, Bookmark, Hyperlink	Help, Search, Font, New File, Open File, Close File, Save File, Save as Different File Format, Page Numbers, Print Layout, Header/Footer, Undo a Task, Select All, Cut, Copy, Paste, Paste Special, Go To, Find, Replace, Date/Time, Symbol, Bookmark, Hyperlink
2	Change Case, Send To, Set Properties, Exit, Save File, Save as Different File Format, Search File, Protect File, Print Preview, Print File, Default, Print Layout, Web layout, Mail Merge, Macro, Set Options	Paragraph, Bullets/Numbering, Change Case, Background, Send To, Set Properties, Exit, Search File, Protect File, Print Preview, Print File, Default, Web layout, Zoom, Redo a Task
3	Paragraph, Background, Zoom, Check Spell, Check Grammar, Speech, Redo a Task	Insert Table, Delete Table, Table Format, Sort, Import Data, Mail Merge, Macro, Check Spell, Check Grammar, Speech, Set Options

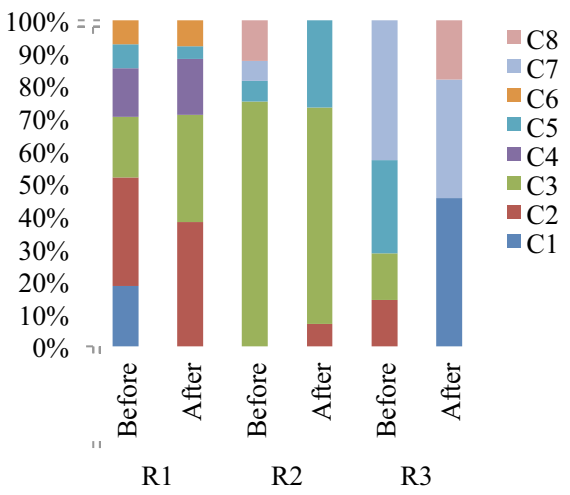

Figure 5. Application of metric M2 before and after clustering (with and without theme focus)

Table 4. Release coherency values M3

	Without clustering	With clustering
R1	57	60
R2	19	12
R3	6	21
Total	82	93

Considering metric M1, if we compare the release plans of both the approaches, C3 appears to be spread across all three releases, if the theme-based methodology was not applied. After applying the methodology, C3 is spread to just two releases. Also, in the conventional approach, cluster C5 is spread across all three releases, while with the new approach, a major portion of C5 is allocated as theme in Release 2.

Considering metric M3, we conclude that, as a tendency, there has been an increase in the number of interdependencies between features packaged into a release. With the exception of Release 2, the interdependencies increase for releases 1 and 3. Hence, the features in a release are more inter-related if our approach is applied.

Releases are now more theme-specific. Our goal was to have a release plan where each release reflects a theme of a product. Considering the M2 metric, for each release, there has been an improvement in the coverage of clusters. In the old approach, Release 1, 2 and 3 had portions from 6, 4 and 4 different clusters, respectively. After applying the theme based approach, the number reduced to 5, 3, and 3, respectively. Also, the distribution of clusters has improved in our approach, leading to more theme-centric release plans.

There is a huge scope of variations in results. To keep up with the agile and lean methodologies, our approach allows scope of incremental development by involving the product manager to vary different aspects to generate different release plans. The release plans are not only dependent on the way the clusters are fed into the software. The plans can be altered by changing the number of releases, resource capacities per release, increment factor in synergy constraint, etc. Hence, the project manager can alter these and can get desired changes in the results.

6.3.2. Product manager governing the clustering process. As stated earlier, the product manager can govern the clustering process. Various input parameters in the algorithm allow the user to get different clusters for different configurations. Also, weak groups can be pre-specified so that the algorithm refines them into better clusters. Hence, the product manager can play an important role in deciding the final clusters. It can serve as a strategy to tap hidden business value. The proposed method tends to release dependent features together, thus promoting specific and theme- based products, rather than releasing features from different domains. Hence, specificity of a release can be tapped as a unique marketing strategy and can be used to get the hidden business value, which otherwise might have been ignored.

6.3.3. Improved release plans. Adding additional synergy value to a group of features makes the release planning optimizer understand that it is beneficial to release them together. For example, in the earlier release plan, Save was not released together with open or new. There is no value of a word processing tool, if we are not able to save our work. With the implementation of additional synergy, the RP understands that if save is release along with Open, New, etc., it will add more value to the release. Similarly, Formatting options like Paragraphing, Bullets etc., are offered together in the theme-base release plans.

Current implementation leads to constraint explosion. Our current implementation of this approach is based on the synergy relationship that may lead to constraint explosion for bigger datasets and is not scalable.

7. Threats to validity

Although the results for the current dataset look convincing and the evaluation metrics suggest the release plans are theme-based, yet there are some threats to validity of our approach.

Firstly, dependencies cannot be the sole criteria for clustering features into groups. There are many other approaches like, textual comparison of feature names, textual comparison of feature description, counting references between codes of features etc. that can symbolize themes equally well.

Secondly, we can modify the clustering algorithm more, so that it gives more freedom to the user to govern the way clusters are formed. There can be functionalities like the user could manually redistribute features amongst clusters after the algorithm groups them. In addition, the PBI size could be included in the clustering process. Instead of treating features, just as mere nodes, the algorithm should consider them as features with value, and apply clustering accordingly. Also, iterative value addition to subsequent releases should also be taken into consideration.

Thirdly, using undirected edges for the feature graph also has a drawback. Though it depicts that the participating features are connected in some way, it does not clearly show which feature is dependent on the other, in case of 'precedence' or 'requires' dependency. However, it is used as a generic approach so that it can accommodate several other types of dependencies mentioned in the paper.

Lastly, the synergy constraint should be modified to solve the problem of constraint explosion. Instead of putting a constraint for every pair of feature, it should be designed in a way such that it favors maximum features from a cluster to be allocated to one release. Furthermore, additional evaluation metrics should be designed to judge whether the approach actually gives theme based release plans. Additional metrics to evaluate if a given cluster symbolizes a theme or not should also be proposed.

8. Summary and future research

We have presented an explorative study on theme based product release planning. We have discussed its meaning, advantages and a methodology to implement it:

- Theme-based release planning helps the project manager to tap hidden business opportunities by releasing highly interdependent features together.
- The approach favors delivering specific theme-oriented products in each release, as opposed to releasing generic products.
- The method allows stakeholder involvement in the early stages of release plan development, in terms of clustering features, feature prioritization, thus promoting agile methodology.
- A tool support and detailed evaluation supports our methodology of theme based release planning.

Future work is targeted on mitigating existing threats to validity and improvement of the methodology. We will focus on modifying the clustering algorithm to allow more stakeholder participation and create alternative clusters that symbolize the idea of theme. The next milestone is to modify the synergy constraint, so that the number of constraints does not increase too much with the addition of new features. Most importantly, we are planning for intensive real-world evaluation of the proposed theme-centric planning approach. Additionally, integration with issue management systems like JIRA may be considered. The latest version of ReleasePlanner supports importing from and exporting to JIRA.

With regard to the validity of this work, one other future direction can be to test the entire approach on a live project possibly by integrating to project's backlog dataset, generating the plans and receiving feedback from development team.

Acknowledgments

This research was supported by the Natural Sciences and Engineering Research Council of Canada, NSERC Discovery Grant 250343-12. Authors would like to thank Fazlul Chowdhury for providing access to the project data used for the case study. Thanks also to Amanpreet Singh for useful discussions in the course of the project and anonymous reviewers for their constructive feedback.

References

[1] C. Larman and V. R. Basili, "Iterative and incremental developments: a brief history," *Computer*, vol. 36, no. 6, Jun. 2003, pp. 47–56.

[2] M. Svahnberg, T. Gorschek, R. Feldt, R. Torkar, S. B. Saleem, and M. U. Shafique, "A systematic review on strategic release planning models," *Inf. Softw. Technol.*, vol. 52, no. 3, 2010, pp. 237–248.

[3] G. Ruhe, *Product Release Planning: Methods, Tools, and Applications*. Auerbach Publications, 2010.

[4] D. Leffingwell, *Scaling software agility: best practices for large enterprises*. Addison-Wesley Professional, 2007.

[5] D. Leffingwell, *Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise*, 1st ed. Addison-Wesley Professional, 2011.

[6] K. Schwaber and M. Beedle, *Agile software development with Scrum*. Upper Saddle River: Prentice Hall, 2002.

[7] K. Schwaber, *Agile project management with Scrum*. Microsoft Press, 2004.

[8] D. R. Greening, "Enterprise Scrum: Scaling Scrum to the Executive Level," in *2010 43rd Hawaii International Conference on System Sciences (HICSS)*, 2010, pp. 1–10.

[9] P. Carlshamre, K. Sandahl, M. Lindvall, B. Regnell, and J. Natt och Dag, "An industrial survey of requirements interdependencies in software product release planning," in *Requirements Engineering, 2001. Proceedings. Fifth IEEE International Symposium on*, 2001, pp. 84–91.

[10] Å. G. Dahlstedt and A. Persson, "Requirements Interdependencies: State of the Art and Future Challenges," in *Engineering and Managing Software Requirements*, A. Aurum and C. Wohlin, Eds. Springer Berlin Heidelberg, 2005, pp. 95–116.

[11] M. Przepiora, R. Karimpour, and G. Ruhe, "A hybrid release planning method and its empirical justification," in *ACM-IEEE international symposium on Empirical software engineering and measurement*, Lund, Sweden, 2012, pp. 115–118.

[12] ReleasePlanner [Online]. Available: www.releaseplanner.com. [Last access: Sep-2013].

[13] V. Heikkilä, A. Jadallah, K. Rautiainen, and G. Ruhe, "Rigorous Support for Flexible Planning of Product Releases - A Stakeholder-Centric Approach and Its Initial Evaluation," in *2010 43rd Hawaii International Conference on System Sciences (HICSS)*, 2010, pp. 1–10.

[14] S. Fricker and S. Schumacher, "Release Planning with Feature Trees: Industrial Case," in *Requirements Engineering: Foundation for Software Quality*, vol. 7195, B. Regnell and D. Damian, Eds. Springer Berlin / Heidelberg, 2012, pp. 288–305.

[15] S. Reddivari, Z. Chen, and N. Niu, "ReCVisu: A tool for clustering-based visual exploration of requirements," in *Requirements Engineering Conference (RE), 2012 20th IEEE International*, 2012, pp. 327–328.

[16] K. E. Wiegers, *Software requirements*. Microsoft press, 2003.

[17] C. Biemann, "Chinese whispers: an efficient graph clustering algorithm and its application to natural language processing problems," in *Proceedings of the First Workshop on Graph Based Methods for Natural Language Processing*, Stroudsburg, PA, USA, 2006, pp. 73–80.

[18] U. Brandes, M. Gaertler, and D. Wagner, "Experiments on Graph Clustering Algorithms," in *Algorithms - ESA 2003*, G. D. Battista and U. Zwick, Eds. Springer Berlin Heidelberg, 2003, pp. 568–579.

[19] F. A. Chowdhury, "Full Course Project - SENG 652," University of Calgary, Dep. of Computer Science, 29 pages.

[20] M. Bastian, S. Heymann, and M. Jacomy, "Gephi: An open source software for exploring and manipulating networks," in *International AAAI Conference on Weblogs and Social Media*, 2009, vol. 2.