Efficient Customization of Software Applications of an Organization

Rajeev Kumar
Assistant Professor
Department of Business Administration
College of Business
Kutztown University, Kutztown, Pennsylvania 19530
USA

Abstract

Nowadays organizations perform a great deal of their work online that requires Information Systems. Although the availability of ready-to-use software applications has greatly reduced the acquisition timesof these systems, organizations still have to spend significant time and resources in their implementation and management. These applications are only offered as a part of generic software packages and organizations have to customize them according to their unique requirements. This requires gathering the requirements of the organization and then customizing the applications accordingly. The customization phase of software implementation can be challenging because the organization has to typically rely on a small team of programmer of its IT department and the process involves a great deal of learning. This paper shows that theexisting approaches of software development and project management are insufficient for the distinct challenges of this setting. The analyticalmodels of the paper provide insights into these challenges and show how this process can be streamlined. The paper also provides the complexity class of theproblems that corresponds to these challenges. A numerical analysis is conducted to demonstrate the effectiveness of the proposed approaches.

Keywords: Software customization, optimum assignment, computational complexity

1. Introduction

Nowadays organizations perform a great deal of their work online that requires Information Systems. For instance, academic institutions typically rely on Information Systems for the delivery of their course work and for maintenance of information about their student finances, human resources, student administration, etc. Desire2Learn (D2L 2013) is one of the software applications that is being utilized for the course delivery and PeopleSoft Campus Solutions suite (Oracle 2013) is one of the leading products for the maintenance of administrative information.

Although the availability of these ready-to-use software applications have greatly reduced the acquisition timesof these systems, organizations have to still spend a great amount of time and resources in their implementation and management. These applications are only offered as a part of generic software packages, and after their purchase, organizations still have to customize them according to their unique requirements (Pollock and Cornford 2004). In order to satisfythe distinct customization requirements of organizations, applications such as PeopleSoft are designed to allow the organization to configure its software with hundreds of options (components). This also allows the product to be mass marketed as essentially a generic product with sufficient customization potential to meet an endless variety of local needs of the organization.

In the customization phase, the organization typically relies on a small team of programmers of its IT department. Thisphase also requires a great deal of learning, as the local IT team has to depend on the external consultants and that requires consulting fees.Itmay take years for the customization phase to complete and the total costs can add up to millions of dollars (The Observatory 2003). The pricing of these applications (licensing fee)is very complex as it depends on numerous factors such as the componentspurchased, consulting fees, the size of the organization, the revenue earned by the organization, etc.For instance the license fee of the Reporting Tools module of PeopleSoft Campus Solutions could be $17500 or $35000 depending on the revenue of the organization and/or the number of its employees (Oracle 2013).

Two major issues were identified after observing the implementation of PeopleSoft at a local four year college.First, most of the activities related to the customization of the application only occurredafter a version of the generic product waspurchased by the management of the organization. That is, after the management acquired a generic product, only then the organization solicited the feedback of its users, such as faculty and students, about itscustomization. This was done through a central committee of the faculty members and the management. Second, there was no objective strategy for assigning the team members (programmers) to the customization tasks. This was done on anadhoc basis only after the tasks related to customization were known.Specifically, the existing approach does not consider the competencies of the programmers. This is important as some programmers may be more suitable and/or interestedincustomizingcertain components than the other.

The central aim of this research is to suggestmethods for the optimum execution of the customization phase of software implementation of an organization.In this context, given the requirements of the users and the programmers, the paper aims answers the following two specific questions:

1) Howcan the IT team prioritize its customization/developmentactivities?
2) How can the project manager optimally assign the team members to the differenttasks of customization?

This paper presents an approach in the Section 3 that answers these two questions.In order to mitigate the inefficiencies of the existing method, the proposed approach solicits the feedback about the usability of different components before the decision to purchase the application is finalized. This leads toa more efficient scheduling of the tasks of customization,as the IT team,ahead of time, hasthe information about the components that are to be customized. The approach also solicits and then utilizes the feedback of the programmers about their competencies. This allowsan appropriate assignment of the programmers to the tasks. An additional benefit of the approach is that it can also be utilized to form teams (programmers working on the same task), which are based on the self-reposted (objective) data of the programmers about their competencies.Note that there may be different demands for different components in the organization. Some components may be demanded by a greater number of faculty members than the others. It is also possible that the requirements of some of the faculty members may not be satisfied within the time horizon of the problem. This may happen because they may demandcomponents that are not of interest to manyusers. Therefore, in the approach of this paper, those components are given priorities that arethe most sought-after in the organization.

The rest of the paper is organized as follows. The next section provides the literature that relates to this problem. Section 3 provides the analytical models which can be utilized by the IT team to manage the customization process, based on the requirements of its users and the programmers. Section 4 provides a numerical analysis that shows the working of the approach. Section 5 provides the conclusion of this paper and suggests ideas for the future research.

2. The Review of the Relevant Literature

The area of software development broadly relates to this research. The term software development refersto the process of writing and maintaining the source code (computer program)from the conception of the desired software through to the final manifestation of the software (McCarthy 1995). The customization issues discussed in the last section relates to this area as the broader issue there is of the developmentof a new software application. In the present setting,however, the software application does not require development from the foundation. A generic product already exists, which only needs to be customized. The research question of this paper is a special case of the broader setting of software development. Therefore, the concepts and methodologies of software development such as software development lifecycle and waterfall model (Royce, Winston 1970)are not specific enough for the current setting.

The problems discussed in the last section also relate to the project management area of Operations Management as the broader problem here is of the creation of an optimum schedulefor thecustomization process. The approaches of project management such as Critical Path Method (CPM) (Kelley 1961), however, are not directly applicable in this setting as they are relevant only for scheduling the tasks along the time horizon and are not applicable for finding a proper assignment of the tasks to the entities performing the tasks. In the current setting, the time requirement of a task is also not fixed and it depends on the programmer performing the task.

Note that in the current setting, different programmers can have different level of competencies (productivities)and interests for different tasks. But once the team members are assigned to the appropriate tasks, the approaches of project management can also be utilized forcreation of efficient schedules forthe customization process. Therefore, the project management approaches can complement the approach of this paper.The next section provides the formulations of the problems thatwere discussed in the last section.

3. Problem Definition and the Formulations

Soliciting the feedback from the users (accomplished by a centralized body)

Let, the available components (tasks) are indexed by j. The set **T** contains the indexes of the tasks: **T**= $\{1,\ldots,m\}$.The committee (a centralized body) provides this information along with the description of the components to the faculty members (users).In the first phase, thecommitteesolicits the feedback from the users on their preferences for the available components. To ensure that each user has equal influence on the customization process, the users are given equal voting right. That is, the sum of the weights (votes), for a user, across all the tasks must add up a constant number, which is the same for each faculty member. This constant could be 1, for instance. The faculty members can declare their preference for the components by breaking their vote across different components.Once the preferences of the users are obtained, the committee adds all the votes for each task. The committee willnow have the following information:

v_{j}:The numeric value corresponding to the importance of the task j according to the users' preferences $\forall j \in \mathbf{T}$.

Soliciting the data about the competencies of the programmersin the tasks (accomplished by the program manager or the team head)

The programmers are indexed by i. The set **P** contains the indexes of the programmers**P**= $\{1,\ldots,n\}$.
c_i: The total time available to programmer $i$in the customization phase$\forall i \in \mathbf{P}$.
Each programmer provides the estimates of the times she requiresto finisheach of the tasks.
t_{ij} : The estimated time that programmer $i$requires to complete task $j \forall i \in \mathbf{P}, j \in \mathbf{T}$

The issue of incentive compatibility

A relevant issue in this context is of incentive compatibility. That is, what is the incentive for the programmers to tell the truth about their competencies?An intuition for the truth telling property in the present setting is that if a programmer overestimates the time requirement for a task then she is under the risk of performing a task that she does not desire. Note that in the next phase the programmers will be assigned to the tasks based on their competencies. The programmers can be incentivized not to underestimate their time requirements by simply imposing a monetary penalty for the overtime. The theoretical issues related to incentive compatibility in this setting, although important, are outside the scope of the current research and are left for the future research.

To assign the programmers to the tasks, the program manager can utilize the Mixed Integer Program (MIP) model, **Q,** given below. Thismodel also provides a method for the team formation (programmers working on the same task) based on the self-reported competency dataof the programmers, rather than an ad hoc approach tothe team formation.

Decision Variables

x_{ij} :The fraction of task$j$to be completed by programmer

y_j =1, If task $j$isto be completed
0 otherwise

$$\mathbf{Q}: \max \sum_{\mathrm{j} \in \mathbf{T}} v_j y_j$$

Constraints

$$\sum_{j\in\mathbf{T}} t_{ij}\, x_{ij} \le \mathrm{c}_i \forall i \in \mathbf{P}$$
$$\sum_{i\in\mathbf{P}} \mathrm{x}_{ij} \ge y_j \forall j \in \mathbf{T}$$
$$\sum_{i\in\mathbf{P}} \mathrm{x}_{ij} \le 1 \qquad \forall j \in \mathbf{T}$$
$$0 \le x_{ij} \le 1, y_j \in \{0,1\} \qquad \forall i \in \mathbf{P}, j \in \mathbf{T}$$

The above program assigns the programmers to the tasks such that the total value of the tasks performed is maximized. That is, the most important components for the organization are customized within the problem horizon. Note that some “unpopular” tasks may not be selected/completed by the programmers. The first set of constraints ensures that, only the tasks that can be performed by a programmer within the time available to her, must assigned to her. The second and the third set of constraints together ensure that if a task if selected then it must be completed by some of the programmers. Note that if$y_j = 1$, then$\sum_{i\in\mathbf{P}} \mathrm{x}_{ij} = 1$.

Observation 1: *In the MIP,* **Q**, *the number of integer decision variables and integer constraints are of the order of* $|T|$.

Below is the formulation of a version of the problem where one task can only be assigned to one programmer:

Decision Variablesof the Integer Program version of the problem

$x_{ij} = 1$, if programmer*i* is assigned to task*j*
0, otherwise.

Objective Function

$$\mathbf{C}: \max \sum_{\mathrm{i}\in\mathbf{P}} \sum_{\mathrm{j}\in\mathbf{T}} v_j x_{ij}$$

Constraints

$$\sum_{j\in\mathbf{T}} t_{ij}\, x_{ij} \le \mathrm{c}_i \forall i \in \mathbf{P}$$
$$\sum_{i\in\mathbf{P}} \mathrm{x}_{ij} \le 1 \qquad \forall j \in \mathbf{T}$$
$$x_{ij} \in \{0,1\} \forall i \in \mathbf{P}, j \in \mathbf{T}$$

Note that the second set of constraints in the above formulation ensures that each task can only be assigned up to one programmer. On the other hand, a programmer can perform several tasks.

Theorem 1:*The abovementioned problem* (**C**)*is an NP hard problem.*

Proof: Knapsack problem (keller et. al. 2004) reduces to the above problem. To see that, consider the case with only one programmer. That eliminates the bottom set of constraints and the problem becomes a knapsack problem.

Theorem 2: *The above problem, in fact, is a special case of generalized assignment problem.*
Proof: Observe that this is the case where the value of completing the task is independent of who completes the task.

Observation 2: *A polynomial time (α+1) approximation scheme for generalized assignment problem* (Cohen et. al. 2006) *can also be utilized to solve the above problem.*

4. Computational Results

The previous section presented the formulations of the problems that can handle all the requirements of the programmers and the users. One of the formulations was a mixed integer program (MIP) and the other was an integer program (IP), which wasalso NP hard. This section presents computational experiments to test the time required to solve different sizes of the problems (**Q**) and (**C**). Table 1and 2belowshow the times required to solve different instances of the problems on Solver software (Solver 2013).The software was installedon a computer with the following specifications: Intel (R) Core ™ i3 CPU M 330 @2.13GHz, Installed Memory (RAM): 4.00GB.

Table 1 below summarizes the times taken to solve different instances of problem Q. These instances had up to 30 components (tasks) and 6 programmers. All these instances took less than 2 seconds to solve.

Table 1: Time taken to solve different sizes of problem Q

|T| /|P|	**3**	**6**
10	< 1 second	< 1 second
15	< 1 second	<1 second
20	< 1 seconds	<1.5 second
25	<2 second	<2 second
30	<2 second	<2 second

Table 2 below provides the computation results for different instances of problem **C**. For comparison purposes, the instances had the same sizes as Q. As we would expect, the C versions of the problem required significantlygreater times to solve than Q versions, as they hadgreater number of integer variables and constraints. Moreover, the problem C is also in the NP hard class.

Table 2: The time taken to solve different sizes of problem C

|T| /|P|	**3**	**6**
10	< 1 seconds	< 1 second
15	< 6 seconds	16 seconds
20	< 18 seconds	30 seconds
25	28 seconds	50 seconds
30	40 seconds	1.5 minutes

As discussed in the previous section, a polynomial time algorithm is availablefor the problem C. The team manager can also utilize that to efficiently solve larger sizes of this problem. Another approach to solve larger sizes of the problem is to break the problem into smaller sub problems. The team manager can first select a subset of tasks that are to be completed first and the programmers canprovide their competencies for only thosetasks. The optimization problem can then be solvedby onlyconsidering those tasks. This approach can be repeated till no further task is left.

5. Conclusions and the Future Research

Ready-to-use software applications are only offered as a part of generic software packages and organizations have to still customize them according their ownunique requirements. Customization of these applications can be a time consuming task as the organization typically have to rely on a small team of its local programmers to accomplish the task. This paper provided an approach that helps the organization to efficiently customize these applications. The limitations of the existing approaches of software development and project managementin these contexts are presented.

The analytical models of this paper also provide a wayof effective team formation. The complexity class associated with the IP version of the problem is presented. Thenumerical study of the paper demonstrated the effectiveness of the proposed approach.This paper identifies two specific areas where this research can be further developed. First, even after assigning the programmers to the customization task, the project manager can use an approach from project management to find an efficient schedule for the tasks ofcustomization. Here, issues such as: which tasks can be simultaneously accomplished, the required sequencing of the tasks, etc., will also needs to be considered. Second, the issue of how to design proper incentives and penalties for the programmers so that they report their true competencies is of importance in the setting of this paper. These issueswill be further considered in detail in the future work.

References

D2L 2013: www.desire2learn.com/

Jim McCarthy (1995): Dynamics of Software Development.

Keller H., Pferschy, and D. Pisinger.(2004). Knapsack Problems. Springer

Kelley, James. (1961). Critical Path Planning and Scheduling: Mathematical Basis. Operations Research, Vol. 9, No. 3, May–June

Oracle 2013: http://www.oracle.com/us/corporate/pricing/peoplesoft-price-list-070612.pdf

Pollock, N and Cornford, J. (2004).Customising Industry Standard Computer Systems for Universities: ERP systems and the University as a "Unique" organisation' Information Technology and People, 17(1): 31-52.

Reuven Cohen, LiranKatzir, and Danny Raz. (2006). An Efficient Approximation for the Generalized Assignment Problem, Information Processing Letters, Vol. 100, Issue 4, pp. 162–166, November

Royce, Winston. (1970). Managing the Development of Large Software Systems, Proceedings of IEEE WESCON 26 (August): 1–9

Solver 2013: www.solver.com

The Observatory 2003: www.obhe.ac.uk/documents/download?id=594