IMPROVEMENT OF RISK IDENTIFICATION OF SOFTWARE PROJECTS USING CLUSTERING APPROACH

Ching-Pao Chang
Department of Information Engineering, Kun Shan University
No. 949, Da-Wan Rd., Yung-kang City, Tainan / Taiwan.
chingpao@ms28.hinet.net

ABSTRACT
Software risk item identification is an important activity of software risk management and project management. Along with the development of software technologies and integration, the complexity of identifying software risk items is increased and required significant efforts. Modeling techniques are commonly used to construct prediction models to facilitate risk items identification. The basic assumption of most modeling techniques is that the source domain is the same as the target domain. This study proposes an approach to reduce the impact of software process change while identifying software risk items, in which the clustering technique is applied on the data collected from past software projects to retrieve the knowledge of risk items. The information can be used to construct suitable prediction models for current software project to identify risk items. The advantage of the proposed approach is that the software risk models can be built at early stage of software project to facilitate the software risk mitigation planning.

KEY WORDS
Software Risk Management, clustering, Risk Items, Software Process

1. Introduction

Software Risk Management is a key factor to affect software project success, in which the software risk item identification is used to identify possible risk, while the software risk mitigation can be used to address identified software risk [1, 2, 3]. Software risks can be treated as possible events that may affect software products or software process [4]. McManus indicates that more than 65% software failures are caused by software risk management, such as the risks of resources and technical management [5]. The software risk management activities can be integrated into software development process to reduce software failure, such as the Spiral models [6]. The Capacity Maturity Models Integration (CMMI) defines the specific goals and practices of risk management, such as identifying and mitigating risk items [7]. To facilitate risk mitigation activities, the identified risk items are quantitatively management, such as the probability of occurring and the impacts to the software project [8]. The challenge of software risk items identification is that significant efforts are required to identify risk items from vast amount of data. The software risk taxonomy is a common way used to facilitate risk items selection, in which stakeholders evaluates the list of possible risk items. The list of risk items can be tailored to reduced selection efforts [9]. Another way to reduce the efforts of risk identification is to construct prediction models using the data collected from software process, in which the relationship between risk items and software process can be identified using statistical methods, such as the regression analysis [10]. The data mining techniques also can be used to build prediction models using software process data [11]. For example, the association rule mining can be used to predict defects remaining in software modules. The data used to build prediction models can be collected from work products and reported defects [12]. The obtained prediction models also can be used to predict possible error that may occur in subsequent activities of software process [13].

These modeling approaches are based on the assumption that the source domain is the same as the target domain. However, the obtained models may not be applied on subsequent software process due to software process change or different development stages. To address this problem, this study proposes a clustering based approach to identify software risk items, the Clustering Based - Software Risk Identification (CB-SRI). The CB-SRI applies the clustering techniques on software process data to build suitable prediction models for subsequent software activities. The main advantage of the proposed approach is that the clustering technique is applied on the source data and target data, in which the data used to build prediction model is selected based on the target domain (i.e. the data used to build prediction models and the target data are in the same cluster).

The remainder of this paper is organized as follows: Section 2 presents an overview of the related works of software risk management and modeling techniques. Section 3 describes the architecture and modeling process of the proposed approach. Section 4 presents analytical results and discussions, in which the proposed approach is applied on the data collected from past software projects to demonstrate how to build risk models. Conclusions are finally drawn in Section 5.
2. Related Works

2.1 Software Risk Management

A software risk can be treated as events that may affect project success, in which the impacts can be reduced by taking certain risk mitigation activities. The activities of software risk management include software risk identification and software risk mitigation [1]. Before taking mitigation activities, the risk items need to be identified and prioritized [7]. The Taxonomy-Based software risk identification provides a set of possible software risks that can be used to facilitate risk identification process, in which the risk items can be categorized. Carr et al. group the risk items by product engineering, development environment and program constraints. Each category contains elements, such as requirement, design and testing. Each element is described by several attributes. For example, the stability and completeness can be used to describe the requirement [14]. Barki et al. define software risks based on software process, such as the technological newness, project scope, team diversity and expertise [15]. Stevens and Timbrell list top-11 common software risks to facilitate risk item identification, such as lacks of commitments [16]. The identified software risk can be prioritized by impacts and probability to facilitate Risk-Reduction Planning [17].

2.2 The Software Risk Modeling

The modeling techniques can be used to find the relationship among the risk items, software process, software products, organization and technology. For example, Ropponen applies the Analysis of Variance (ANOVA) to find the relationship between environment (such as the skills and technologies) and software risks. The information can be used to determine mitigation activities [18]. Han applies the Multivariate analysis of variance (MANOVA) on six dimensions of software risk based on literature reviews (i.e. requirement and satisfaction) to find the relationship of the variables [19]. Tüyüz applies Fuzzy Analytic Hierarchy Process (FAHP) on software process to identify possible software risk items. [20]. In addition to measure software process, the measurement of software products also provide useful information of software risks, such as the system architecture models [21]. The data collected from past projects can be used to construct prediction models to identify possible risk items at early stage, such as the risk of project bidding [22]. The identified information of software risk can further be used to simulate the impacts on software projects [23]. The challenge using collected data to construct prediction models is that the characteristics of the source domain may differ from the target domain due to process change, such as the skill of stakeholders, development tools or stages of software process.

2.3 The Clustering Technique

The clustering technique classifies the data into groups, in which the objective criterion function is defined to measure the similarity among objects [24, 25]. The similar objects fall into the same group, while distinct objects separate into different groups [26]. The clustering techniques have been successfully applies on many fields, such as the pattern recognition [27], psychiatry, biology, psychology, archaeology, geology, geography, marketing, image processing [28] and information retrieval [29]. Several approaches can be used to classify the data into clusters, such as the hierarchical approach creates a hierarchical decomposition of the given set of data, while the partition approach produces k partitions of the data, in which each partition represents a cluster. The density-based clustering is based on connectivity and density function, in which a cluster can be treated as a high dense region of points, and the clusters are separated by low-density regions. To make the models obtained from source domain (denoted as $D_S$) can be applied on the target domain (denoted as $D_T$), the data collected from source domain and target domain can be clustered to remove outliers, in which a domain comprises of data, feature and distribution, while a model comprise of label and function. The models constructed using the data collected from $D_S$ is denoted as $M_S$, while the models obtained using $D_T$ is denoted as $M_T$. The traditional prediction models is based on the assumption $D_S = D_T$ and $M_S = M_T$ [30]. However, the target domain may differ from the source domain (such as $M_S \neq M_T$) due to the process change. The transfer learning techniques can be used to address this problem.

The data mining approach can be used to construct prediction models using source data, while the adaptation process attempts to construct suitable models for target domain using small amount of target data and great mass of source data [31]. For example, the TrAdaBoost applies classification mining technique to build prediction models, in which each training data is assigned a weight initially, and the weight is adjusted according to the performance of the classifier [32]. The basic assumption of the TrAdaBoost is that the distribution and feature of source domain is the same as the target domain ($D_S = D_T$). However, the characteristics of data may be different (such as $D_S \neq D_T$), such as the data collected from different software stages. To address this problem, unsuitable data need removed, and only the data similar to the target data can be selected to build prediction models [33]. Xing et al. apply the clustering technique on target domain data based on the source domain data, in which the common features are identified to facilitate clustering process [34]. Wang et al. apply clustering techniques to construct virtual labels for unlabeled data, and utilize the labeled data (source domain and target domain) to reduce the dimensions of collected data. The iterative process is preformed to obtain suitable feature set [35, 36].
3. The System Architecture

The main purpose of the Clustering Based - Software Risk Identification (CB-SRI) proposed in this study is to construct software risk models for risk identification using the data collected from software projects. The components of the CB-SRI are depicted as Figure 1, including the Data Collection, Software Risk Representation (SRP), Risk Item Modeling and Risk Item Identification. A set of attributes related to software risks is defined in SRP. The Data collection collects the data of software projects and links the attributes of the collected data to SRP attributes. A set of possible software risk items is defined to facilitate risk item modeling process. The collected data then can be used to identify the relationship between software risks and software process, such as the re1, re2 and re3 shown in Figure 1. The obtained models can be used to identify software risk items for software projects.

SRP can be defined, such as $A' = \{a'_1, a'_2, \ldots, a'_n\}$ (the $a'_i$ denotes the timeline). The intermediate attributes also can be used to link measures with different units. For example, assume that $a_{ik} \in A_i$ denotes the number of function points and $a_{jk} \in A_j$ denotes the number of screens, in which both $a_{ik}$ and $a_{jk}$ represent the size of a work product (such as the requirement size). Two links, $lnk_1(a_{ik} \to a'_i)$ and $lnk_2(a_{jk} \to a'_j)$, are constructed, and both $a'_i$ and $a'_j$ link to $A_{pi} \in A^P$ (an attribute used to represent the size of work product).

![Figure 1. The Architecture of the CB-SRI](image)

3.1 The Data Collection

The Data Collection component collects the data of software projects that can be used to identify software risk, such as the data of software process, work product and stakeholder. As shown in Figure 2, let $D = \{D_1, \ldots, D_n\}$ denotes the data collected from different projects, and $A = \{A_1, \ldots, A_n\}$ denotes the attributes of $D$, in which $A_i$ is the attribute of $D_i$. Assume that the $D_T \in D$ denotes the data collected from current project (target domain), and $A_T$ denotes the attributes of $D_T$. Each $D_i$ comprises attributes and data entries, such as $D_i = (A_i, d_i)$, in which $a_i = \{a_{i1}, \ldots, a_{im}\}$ and $d_i = \{d_{i1}, \ldots, d_{in}\}$ represent the attributes and data entries respectively. The main purpose of the CB-SRI is to construct suitable risk models for $D_T$ using the data $D_S$ and $D_T$. To facilitate modeling process, a set of attributes (denoted as SRP) need to be defined and linked to the collected data $D$. The SRP contain the attributes of risk (denoted as $A^R$), software process (denoted as $A^P$), work product (denoted as $A^W$) and stakeholder (denoted as $A^S$). To facilitate the mapping process, a set of intermediate attributes between $D$ and $D_T$ are defined in $A'$ which is $\text{cluster} = \text{pair}$. The attributes $\text{cluster}$ and $\text{pair}$ are used to link $A'$ to $D_T$.

![Figure 2. The Software Risk Attribute](image)

3.2 The Risk Item Modeling

The Risk Item Modeling component constructs the software risk knowledge using the collected data $D$. The knowledge indicates the relationship between software risk ($A^R$) and software process ($A^P_1$, $A^P_2$ and $A^P_3$). The knowledge can be used to predict possible risks based on the status of current project. The modeling process comprises two main steps, the Data Pre-processing and Analysis. The data pre-processing identifies transactions that may cause software risks from collected data according to the analysis engine. A simple way to identify transactions is based on the events of risk. For example, the data $d_{33}$ shown in Figure 6 indicates that a possible risk $r_{31}$ with $\alpha_{i(k+1)} = u_{kji}$ occurs at time $t_{k3}$, in which the value of attribute $a_{ik}$ is $v_{ik3}$. The data items that may cause the risk include $d_{11}$, $d_{22}$ and $d_{33}$. These data items can be selected as a transaction (denoted as $tr_{ij}$) of the dataset $D$. The same procedure can be used to identify the
transaction \( tr_{5j} \) in which the number of selected data items to form a transaction can be predefined as \( \delta = 3 \). In this case, three data items before a risk \( r_i \) are selected as a transaction, such as \( tr_{i1} = \{ d_{i1}, d_{i2}, d_{i3}; r_{i1} \} \) and \( tr_{i2} = \{ d_{i3}, d_{i4}, d_{i5}; r_{i2} \} \) are identified. The attributes of data sets may be different, and the attributes of data items within a data set may also be different. In Figure 3, the attributes of the collected data set \( D_i \) (\( A^{P_i}, A^{P_j}, A^{R} \)) differ from the attributes of another data set \( D_j \) (\( A^{P_i}, A^{P_j} \)). The attributes of \( d_{i1} \) and \( d_{i2} \) are also different.

![Figure 3. Identifying transactions using collected data](image)

Forth, apply the clustering technique on the obtained transactions, the \( TR_i \) and \( TR_j \), to find suitable transactions to construct risk models. The selection of \( tr_{kj} \) is based on similarity measure of \( tr_{ik} \) and the transactions of \( TR_i \) and \( TR_j \), such as \( C_{TR_i} = \{ tr_{ik}, tr_{il}, ..., tr_{lj}, ... \} \) (for some \( tr_{ij} \in TR_i \) and \( tr_{lj} \in TR_j \)). For example, let \( dis(x, y) \) denotes the distance between \( x \) and \( y \) based on similarity measure. The \( tr_{ik}, tr_{il} \) and \( tr_{lj} \) are in the same cluster \( C_{TR_i} \), if \( dis(tr_{ik}, tr_{il}) < \sigma_i \) and \( dis(tr_{lj}, tr_{lj}) < \sigma_j \) (\( \sigma_i \) is a predefined threshold). The transactions of \( C_{TR_i} \) can be treated as the transactions with similar characteristics, and can be used to construct suitable risk models.

![Figure 4. Transactions of the target domain](image)

Fifth, apply the clustering technique on the \( C_{TR_i} \) based on \( A^{R} \) attributes to obtain \( C_R = \{ CR_1, ..., CR_{ie} \} \), in which the transactions with similar \( A^{R} \) attributes are clustered into the same cluster. The transactions clustered into the same cluster indicate that the same risk item may occur in these transactions. Each cluster \( CR_i \) of \( C_R \) indicates a possible risk, and a project with such transactions may cause the risk. The transactions with no \( A^{R} \) attributes are also clustered into the same cluster. A project with such target transaction can be treated as safe state.

### 3.3 The Risk Item Identification

The **Risk Item Identification** identifies possible risk of the current project based on the project status. The project status can be expressed by a set of measures or attributes. The collected data items can be treated as the project status at certain point. A sequence of the collected data items constitutes a transaction and a series of project states. The target transaction \( TR_{R} \) obtained in previous subsection can be treated as the current project status. For example, as shown in Figure 5, the data item \( d_{1t} \) denotes the project status at time \( t_1 \) in which the obtained target transaction is \( tr_{1k} \in TR_R \) (for \( \delta_t = 4 \)). The \( tr_{1k} \) can be used to identify risk at time \( t_1 \) by measuring the distance between \( tr_{1k} \) and the transaction of risk models. When the \( tr_{1k} \) is clustered into a cluster \( CR_i \in C_R \), certain software risk item...
may occur at (or after) time $t_k$, such as the time $t_i$ and $t_k$ shown in Figure 5. The CB-SRI identifies possible software risk items based on the current project status, but the time of the risk occurrence due to different timeline of the collected data. To locate the time interval of possible software risk items, such as the time $t_i$ and $t_k$, the time interval of the collected data need to be calibrated.

![Risk identification process](image)

Figure 5. The risk identification process

The similarity measurement of transactions is based on the data items contained in the transactions, such that the two data items contained in two different transaction respectively are treated as same data items, if the distance between these two data items is smaller than the predefined threshold (such as $\sigma_\delta$). Assume that the distance of two data items $d_{i1}$ and $d_{i2}$ contained in transactions $t_1$ and $t_2$ respectively is smaller than $\sigma_\delta$, such as $dis(d_{i1}, d_{i2}) < \sigma_\delta$, the next two similar data items of $t_1$ and $t_2$ (such as $d_{i2}$ and $d_{i3}$) are compared. The comparison is performed until the last data items of the transactions.

4. Results and Discussions

To illustrate how the CB-SRI approach can be used to identify risk items for software projects, the CB-SRI is applied on two simple software projects, in which one software project (denoted as $P_S$) is treated as source domain data $D_S$ (the past project), while another project $P_T$ is treated as target domain data $D_T$ (the current project). The prediction results are compared to the actual records of $P_T$ to show the efficiency of the proposed approach. The $P_S$ and $P_T$ are research projects, in which the $P_S$ is started at 6/1/2009 and ended at 12/31/2009, while the $P_T$ is started at 3/1/2010 and ended at 6/30/2010. The attributes of the project $P_S$ are shown in Table 1, in which the number of uncompleted tasks (denoted as $ong_u_task$), the estimated efforts (denoted as $ong_u_est$) and actual used efforts (denoted as $ong_u_used$) of uncompleted tasks are both defined in $P_S$ and $P_T$. The risk item defined in $P_S$ is the cost overrun (denoted as $cost_overrun$), in which the value indicates the severity of the identified risk items, such as 0 denotes no impact, while 1 denotes great impact. In addition to the attributes defined in $P_S$, the attributes $defect_int$ defined in $P_T$ denotes the number of defects detected between two milestones, while the $req_chg$ denotes the number of requirement changes between two progress reviews. The data of $P_S$ and $P_T$ are collected based on the progress review date, denoted as $pr_review$ (biweekly).

Table 1

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$pr_review$</td>
<td>The date of progress review</td>
</tr>
<tr>
<td>$pr_id$</td>
<td>The progress review id</td>
</tr>
<tr>
<td>$ong_u_task$</td>
<td>The number of on-going uncompleted tasks.</td>
</tr>
<tr>
<td>$ong_u_est$</td>
<td>The estimated efforts (hours) of on-going uncompleted tasks</td>
</tr>
<tr>
<td>$ong_u_used$</td>
<td>The used efforts (hours) of on-going uncompleted tasks</td>
</tr>
<tr>
<td>$cost_overrun$</td>
<td>Indicate the severity of identified risk item (between 0 and 1)</td>
</tr>
</tbody>
</table>

Table 2

<table>
<thead>
<tr>
<th>$pr_review$</th>
<th>$pr_id$</th>
<th>$ong_u_task$</th>
<th>$ong_u_est$</th>
<th>$ong_u_used$</th>
<th>$cost_overrun$</th>
</tr>
</thead>
<tbody>
<tr>
<td>06/01/09</td>
<td>A1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.0</td>
</tr>
<tr>
<td>06/15/09</td>
<td>A2</td>
<td>1</td>
<td>28</td>
<td>26</td>
<td>0.1</td>
</tr>
<tr>
<td>06/29/09</td>
<td>A3</td>
<td>2</td>
<td>56</td>
<td>60</td>
<td>0.6</td>
</tr>
<tr>
<td>07/13/09</td>
<td>A4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.0</td>
</tr>
<tr>
<td>07/27/09</td>
<td>A5</td>
<td>2</td>
<td>40</td>
<td>44</td>
<td>0.2</td>
</tr>
<tr>
<td>08/10/09</td>
<td>A6</td>
<td>3</td>
<td>76</td>
<td>84</td>
<td>0.3</td>
</tr>
<tr>
<td>08/24/09</td>
<td>A7</td>
<td>4</td>
<td>92</td>
<td>104</td>
<td>0.7</td>
</tr>
<tr>
<td>09/07/09</td>
<td>A8</td>
<td>2</td>
<td>60</td>
<td>72</td>
<td>0.3</td>
</tr>
<tr>
<td>09/21/09</td>
<td>A9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.0</td>
</tr>
<tr>
<td>10/05/09</td>
<td>A10</td>
<td>1</td>
<td>16</td>
<td>18</td>
<td>0.1</td>
</tr>
<tr>
<td>10/19/09</td>
<td>A11</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.0</td>
</tr>
<tr>
<td>11/02/09</td>
<td>A12</td>
<td>2</td>
<td>28</td>
<td>26</td>
<td>0.0</td>
</tr>
<tr>
<td>11/16/09</td>
<td>A13</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.0</td>
</tr>
<tr>
<td>11/30/09</td>
<td>A14</td>
<td>1</td>
<td>14</td>
<td>16</td>
<td>0.2</td>
</tr>
<tr>
<td>12/14/09</td>
<td>A15</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.0</td>
</tr>
<tr>
<td>12/28/09</td>
<td>A16</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.0</td>
</tr>
</tbody>
</table>

The data collected from $P_S$ are shown in Table 2, in which the values of $cost_overrun$ (indicates the impacts to the cost) are evaluated by expert in progress review and ranking from 0 to 1. For example, the cost impact at A7 is evaluated as 0.7 due to the large number of uncompleted tasks and used efforts. The data collected from $P_T$ are shown in Table 3. In addition to the common attributes ($ong_u_task$ and $ong_u_est$), the $defect_int$ and $req_chg$ represent the number of detected defects and the number of requirement changes respectively. The $D_T$ does not contain the information of used efforts of uncompleted tasks, such as the $ong_u_used$. The risk items of cost overrun are detected at progress review B3, B8 and B9.

The CB-SRI approach is applied on the source data $D_S = \{A1, ..., A16\}$ and some target data, such as $D_T =$
\{B1, \ldots, B6\} to construct risk models, such as to find the clusters with the risk of cost overrun. The obtained models can be used to identify possible risk items for subsequent software activities, such as \(B7, B8\) and \(B9\).

### Table 3
The data collected from the project \(P_T\)

<table>
<thead>
<tr>
<th>Milestone</th>
<th>pr_id</th>
<th>(\text{ong}_u_\text{task})</th>
<th>(\text{ong}_u_\text{est})</th>
<th>(\text{defect}_\text{int})</th>
<th>(\text{req}_\text{chg})</th>
<th>(\text{cost}_\text{error})</th>
</tr>
</thead>
<tbody>
<tr>
<td>03/01/10</td>
<td>B1</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>03/15/10</td>
<td>B2</td>
<td>1</td>
<td>18</td>
<td>4</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>03/31/10</td>
<td>B3</td>
<td>2</td>
<td>38</td>
<td>8</td>
<td>2</td>
<td>V</td>
</tr>
<tr>
<td>04/15/10</td>
<td>B4</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>04/30/10</td>
<td>B5</td>
<td>1</td>
<td>16</td>
<td>2</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>05/15/10</td>
<td>B6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>05/31/10</td>
<td>B7</td>
<td>1</td>
<td>12</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>06/15/10</td>
<td>B8</td>
<td>2</td>
<td>30</td>
<td>4</td>
<td>0</td>
<td>V</td>
</tr>
<tr>
<td>06/30/10</td>
<td>B9</td>
<td>4</td>
<td>46</td>
<td>6</td>
<td>0</td>
<td>V</td>
</tr>
</tbody>
</table>

By setting the length of transaction to one, the transactions can be identified \(TR_S = \{A1, \ldots, A16\}\) and \(TR_T = \{B1, \ldots, B6\}\). The set of suitable transactions \(C_{TR}\) can be obtained by applying the clustering on common attributes of \(TR_S\) and \(TR_T\) (the \(\text{ong}_u\_\text{task}\) and \(\text{ong}_u\_\text{est}\)).

![Figure 6. The clusters of \(C_{TR}\)](image)

Figure 6 shows the obtained clusters \(C_{TR}\), in which the cluster \(C^{\text{TR}}_3\) and \(C^{\text{TR}}_4\) contain transactions with risk items (\(A3\) and \(A7\)), and can be treated as the cluster with risk items. The same results can be obtained by applying clustering on \(D_S\) (using all attributes), in which the transaction \(A3, A5, A8, A6, A7\) are clustered in the same cluster (the cluster with risk items). The obtained clusters can be used to identify possible risk items of target transactions. As shown in Figure 7, the transaction \(B8\) and \(B9\) are clustered into \(C^{\text{TR}}_3\), the cluster containing cost overrun risk items, and can be treated as the target transactions containing possible risk items. The target transaction \(B7\) is clustered into the cluster \(C^{\text{TR}}_1\), the cluster with no cost overrun risk items. Figure 8 shows the results that applying clustering on all attributes of target transactions (the attributes \(\text{ong}_u\_\text{task}, \text{ong}_u\_\text{est}, \text{defect}\_\text{int}\) and \(\text{req}\_\text{chg}\)). The transactions with possible risks are clustered into the same cluster (such as \(B3, B8\) and \(B9\)), while the transaction with no risk items are clustered into another cluster (such as \(B7\)).

![Figure 7. The cluster contains target transactions](image)

The results show that the transactions with possible risk items can be clustered into the same clusters. These clusters can be used to identify risk items. However, the data collected from different software projects may be different, and require further data pre-processing to select proper attributes for clustering. The proposed approach can be improved to obtain knowledge of software risks from different types of past projects.

### 5. Conclusion

This study proposes an approach, the Clustering Based Software Risk Identification (CB-SRI), to identify possible software risk items. The CB-SRI applies the clustering techniques on the data collected from past projects and current project to build suitable prediction models for subsequent software activities. The main advantage of the proposed approach is that the clustering technique is applied on the source data and target data, and the data used to build prediction models are selected according to the target data (i.e. the data used to build prediction models and the target data are in the same cluster). The proposed approach can be used to identify possible software risk items at early stage of software project to facilitate the software risk mitigation planning.

### Acknowledgement(s)

This work is supported by the National Science Council of Taiwan, R.O.C., under grant NSC 99-2221-E-168-033.
References


