Chapter IV

Data Analysis and Research Results

This chapter describes the process of data analysis and empirical findings through grounded theory study of Commercial Application development firm and e-Governance Application Development firm that implemented Scrum in their software development processes. Concepts suggested by the open coding of raw data and recurring themes identified from axial coding are also explained.

4.1 The Process of Data Analysis

In the first stage of data analysis, all the data produced by observations, e-mail surveys, documentations and interviews were examined and coded by focusing on the issues and challenges of Scrum identified using the empirical processes controls and the Scrum framework (roles, ceremonies, and artifact). In the first round of data analysis, an open coding technique [113] was used to identify possible concepts, along with their properties and dimensions. The open coding technique involves a form of content analysis where the data is read and categorized into concepts that are suggested by the data rather than imposed from the outside [114].

In the second round, the codes were reviewed, and the concepts were organized into recurring themes. These themes were used later as a basis for creating a set of stable and common categories. During the second round of data analysis, the documents were re-read, and analytic and self-reflective memos were created. The final stage of data analysis was completed through an axial coding [113] which depends on a synthetic technique of making connections between categories and subcategories to build a more comprehensive scheme. During this stage, all the codes were searched, sorted, grouped, and compared to the original data. This process continued until a final series of categories were identified, each having a high frequency of occurrence.
in the data. Figure 4.1 illustrates the process of the data analysis. In the following section, each of the concepts and categories found in the Commercial Application Development Firm and e-Governance Application development firm is presented and discussed.

Figure 4.1 : Data Analysis Process

4.2 Commercial Application development firm

Commercial Application development firm has about 30 software engineers in five development teams in the software division. Commercial Application Development Firm used a waterfall style traditional software development method for more than 20 years before they changed to the Scrum method. To test the success of Scrum method, all software engineers in the firm were invited to 2 days-long, 8 hours intensive Scrum training sessions, which were conducted by one of leading training experts in this field [115]. The firm then reorganized their development teams following the Scrum model. Each team consists of four to seven software engineers,
a Quality Assurance (QA) person, a Scrum master, and a Product Line Manager (PLM). These teams are specialized in the area of interface, jail, record, mobile, and architecture.

When Scrum was first introduced to the firm, most developers were reluctant to adopt a new method due to their previous experience with traditional development methods. However, they began to enjoy several unique features of Scrum as they exercised its various elements. In general, most elements of Scrum were successfully adopted and implemented with minor modifications dealing with meeting schedules and the open working environment. The following sections explain sixteen concepts that appeared in the data analysis process.

Table 4.1 lists the categories, concepts, and data related to Commercial Application development firm. As shown in the table, the factors of human resource management, structured development process, environmental, and information systems and technology were constructed. The concepts comprising these categories are discussed in the following section.

Table 4.1: Categories, concepts, and data related to Commercial Application development firm

<table>
<thead>
<tr>
<th>No.</th>
<th>Common Categories</th>
<th>Concepts</th>
<th>Data</th>
</tr>
</thead>
</table>
| 1   | Human Resource Management Factor | Team Management          | • Teams were reorganized without considering developers’ knowledge and skills  
• Self-managing team does not work well  
• Team needs a supervisor who can see a big picture  
• Small team size is more flexible and adaptable in defining and applying a variant of Scrum |
|     |                          | Collaboration             | • Developers and QA does not collaborate  
• Developers do not try to get to know new developers in remote site  
• Several teams simultaneously work on “look |
<table>
<thead>
<tr>
<th>2</th>
<th>Structured Development Process Factor</th>
<th>Scrum Framework</th>
</tr>
</thead>
</table>
| Training | • New employee training is one of the biggest problems  
• Because of the complex nature of the system, new employees need to spend a lot of time to be trained  
• Employee training problem gets worse when two development sites are involved |
| Lack of Accountability | • Tasks in a Sprint backlog do not get completed  
• Nobody takes responsibility on delayed tasks  
• Self-managing Scrum team provokes the lack of supervision  
• Project managers do not have any authority to control developers |
| Trust and Confidence | • Scrum master does not remove developer’s impediments  
• Trust and confidence are lost when people do not complete their work  
• It is important to know who you are talking to and what level of information they have |

and feel” design that should be implemented by a designated team  
• Tools are used to reduce collaboration problems
| Unit and Integration Testing | • A QA person needs to wait until developers finish their coding  
• Developers do not want to review the code that they finished during the previous Sprint  
• QA people want developers to set aside some time for bugs  
• QA people may not know what areas could be affected by the changes made by developers |
| Coding Standard | • Developers can understand other developer’s code better through coding standards  
• Too much coding standards may hamper developers performance |
| Documentation | • Detailed design documents were reduced significantly  
• (Example of documents: use cases, class diagram, sequencing diagram, activity diagram, communication diagram)  
• Quick implementation without creating a document can cause ripple effects that damage other parts of the project  
• No one takes time to think about inter-dependency  
• Bug rate is increased because of reduced documents which lead to the lack of standardization of features, field names, and error messages |
| Formal Code Review | • A web-based code review tool was created by a project manager  
• Reviewers checked if there were any side and ripple effects to other code Formal code review was a vital and critical process for high-quality applications |
| 3 Environmental Factor | • Customers cannot participate in Scrum meetings due to the large number of customers scattered in the U.S.  
• Project managers spend lots of hours to visit customers  
• The firm hosts a users’ conference where customers can vote for or against a policy and a direction of new application development  
• QA personnel cannot provide customers with quick turn around on bug fixes |
<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
</tr>
</thead>
</table>
| Working Environment | • Open working environment promotes communications, facilitates self-organization, and makes developers to get together easily  
• We are constantly distracted by the person we work with in a cubicle setting  
• Putting every team member in one cubicle area increases collaboration and teamwork |
| Interdependency among Modules | • QA department has found twice as many bugs since the firm switched to Scrum  
• As the size of application grows, the dependencies and interconnections among tasks increase  
• Developers have a tendency to complete a task in a quick and dirty way  
• Developers do not think of how code will be maintained and the code will be flexible enough for future needs |
| Social Facilitation, Social Loafing, Group Motivation, and Evaluation Apprehension | • Social facilitation effect and group motivational gains were observed as factors that reduced development times and lowered bug rates  
• Evaluation apprehension was observed when the company invited all of developers, QA personnel, and people in other to the Sprint review meeting |
| Communication System | • The firm put a lot of efforts to establish a good communication channel  
• Video conference equipment was used only for team meetings  
• Phone conferencing did not provide facial expression, gesture, and body language  
• A web demo tool together with a phone system worked very well  
• Some of tools including instant messaging, email, virtual private networking, and remote desktop were also used  
• Multimedia systems did not work as optimally as face-to-face conversation  
• If data line is down employees cannot do anything |
| Information and Knowledge Sharing System | • Well-structured information and knowledge sharing systems are needed between experienced developers and brand new developers  
• A web-based Wiki program is used to facilitate the information and knowledge |
4.2.1 Human Resource Management Factor

The concepts of team management, collaboration, training, lack of accountability, and trust and confidence constitute the human resource management factor. Each concept is discussed in turn.

4.2.1.2 Team Management.

In terms of development times and costs, Scrum was not very effective during the first two months after Commercial Application Development Firm switched to the Scrum method. They attributed this undesirable fact to the inappropriate composition of the development teams. The teams were reorganized without considering knowledge and skills of the developers. Therefore, some team members had to learn business logic (how the application works in a specific field) that they were not familiar with, and others needed to learn new development tools and programming languages. As a result, it took longer development times and more development costs.

A project manager stated: Teams cannot be thrown together. The team needs to be built. Without the right mix of team members Scrum development is much slower. You can still succeed with Scrum development. But without all members working
together, building the team, the interest, and the enjoyment, you are not going to have the success and the energy from the team that the Scum development can bring.

Some developers dislike a self-managing team though this is identified as one of the unique aspects of Scrum. They prefer to have a team lead who can keep things going in the right direction. They think the Scrum master might be a good candidate to do this job.

One developer added: I’m not sure that I like having the group of developers run the team. I would still prefer having a team lead type of person who can work with the team to help focus and keep things going in the right direction. I think the Scrum master should be doing this but I think this person needs to be more technical. I don’t think this person should take the team over however, I think they should just help focus the team on the right priorities.

Another developer also mentioned that, “Our teams aren’t very good at self-managing, we’re more used to someone being in charge.” A QA person revealed another possible problem with teams, developers given too much power to runs the team. He vented that “our problem was when team members didn’t interact well with other members of the team.” One developer expressed that a self-managing team was strange to him, and he wanted to have a supervisor who can see the big picture. He stated: When we are going to take vacation/sick leave, we report to our team members rather than a manager. I’m not sure if that’s good or bad, but it seems strange to me. It sometimes feels like we don’t have much supervision, which I personally like, but we seem to lack a person or group that is trying to see the big picture.
However, most of developers consider a self-managing and self-organizing team a great idea. One developer stated that, “The part of Scrum that has worked the best for me is the idea that the team decides how to do things based on the consensus of the team.” Another developer mentioned, “There seems to be better team work using the Scrum model. I like the idea that the team has more control over how the development is done and completed.” A project manager added, “Team work is very important! The most interesting and unique part of Scrum has been better team work.” The firm tried to keep the size of Scrum team as small as possible, believing that teams can be more flexible and adaptable in defining and applying an appropriate variant of Scrum. A vice president of software R&D department stated:

It seems the Scrum method provides better team work through the Scrum framework. We keep each team as small as possible as recommended by the Scrum method, and it appears that a small Scrum team is more efficient when implementing various components of Scrum.

4.2.1.3 Collaboration.

When Commercial Application Development Firm first adopted the Scrum method, five Scrum teams were organized and each team had a Quality Assurance (QA) person who was designated to test the code created by his/her own team. Recently, the company pulled the QA person from each Scrum team and created a QA personnel-only Scrum team so that a QA person can test the code generated by any Scrum teams. Because the QA Scrum team is located in one place, it sometimes causes collaboration problems between remote site developers and QA personnel. For example, when developers in a remote site pass the code over to QA personnel through the Concurrent Versions System (CVS), QA personnel may not know which part of code is affected by the changes that developers made. A lead engineer noted:
The development was kind of passed over to QA and they took care of testing. But the problem with that is QA may not know what particular area could be affected. So, they cannot test the area which might be broken by the changes made by developers.

This problem might be solved if developers take on some parts of QA’s testing or show the QA personnel the potential areas that might be affected by the modification done. Another collaboration problem arose when new developers of the Scrum team were hired in one location. Developers in the other location did not try to get to know the new developers. One developer put it this way:

When new members of the team were hired at the other site, it was easier for team members here not to really put any effort into getting to know that person and learning how best to work with them. This problem was especially bad when the new members had some personality characteristics that made them a little annoying or perhaps difficult to develop a desire to want to work with them. This issue mainly resulted from the way the Scrum team is divided at two different sites. It would be better for the company to reorganize the Scrum team so that team members are located at the same place. One notable observation occurred at a daily Scrum of Scrums meeting with the firm’s vice president of software development. Duplicates works were found across Scrum teams due to a lack of collaboration among the different teams. It was also noted that Commercial Application Development Firm had a difficult time keeping the product output consistent across teams. For example, several teams may simultaneously work on the “look and feel” design that should be implemented by a designated team (e.g., an architecture team). This implies that there should be a person (possibly a Scrum master in each team) who is responsible for checking the consistency of products across Scrum teams. It was noted that the inconsistency and duplicates across different teams could negatively affect the efficient product management.
Other collaboration problems arose when developers and support staff discuss how to divide and assign tasks between the two sites in the Sprint planning meeting and how to track bugs reported by QA personnel and customers. To reduce the collaboration problems in this area, the firm has been using a web-based commercial tool called Version One (http://versionone.com), which has provided an excellent project management mechanism for both sites. Through Version One, developers at both sites can actually see how each project is divided, what projects are going on, the status of each project, who is working on each project, and when those projects are expected to be completed.

Recently, the company picked another commercial tool called JIRA to track information mainly on what kinds of bugs exist, who is working on each bug, and the status of each bug. JIRA actually replaced an old tool called MOM, which was developed internally and used to create information for support problems. However, using JIRA has some downsides. A project manager aired this issue:

But it (JIRA) also has drawbacks. We’ve now broken up for instances, the fixed description, the problem, various fields, and who’s worked on it. But in JIRA, you can’t search. If you are looking for some word in the problem somewhere, you have to search each and every individual field. So, if you think, for instance, some value is in there in fields other than the comments, summary, and description, you must put the value in each field and then search individually. And searching is not near as good as MOM had. For instance, you can only search on whole words. You have to have the beginning of word. You can’t search for the value within a word.

Despite the downsides, JIRA has been providing developers with a useful bug tracking mechanism for both sites. JIRA also has a special code, called the severity or priority code, which enables support people to automate the choice on what bugs to work on next. Version One and JIRA play a vital role in the company towards
4.2.1.4 Training.

New employee training issues emerged when information and knowledge-sharing issues and communication issues were brought up. A couple of project managers mentioned that one of the biggest problems that the company has been facing is new employee training. Because of the complex nature of the software program that the firm has been creating, new employees need to be trained for an extended period of time. A project manager noted that: due to the complexity of our application, new software engineers need to spend a good chunk of time to understand various aspects of the system. The typical employee training program usually lasts six months.

Another project manager added: “Right now, each Scrum team is in charge of training of new software engineers. This involves a selection of a mentor who leads and guides the new employee. The mentor spends a lot of his/her time with the new employee to provide the necessary knowledge and skills for projects. I think this is a big waste of time because the mentor in each Scrum team teaches some common parts of the system that can be taught together with other employees who belong to other Scrum teams.”

The project manager suggested, “It would be more time-saving if one mentor teaches all new employees together the common parts of the system, and then let each Scrum team teach its new employee team-specific parts.”

It seems the problem gets worse when an employee who has expertise in one field is at the remote site and the new employee needs to learn the new field from the remote employee. A developer declared, “We often times have a situation where a
new employee needs to know some parts of the system and the person who knows about them is at different location.” Another developer contributed, “We have some difficulties when we are involved with other team members, part of the team is here and part of the team is at the remote site. In particular, when we have a new member here and he/she needs to talk to other members at a different site.”

To address this problem, the company used multi-media, such as phone/video conferencing or web demos, but the training through these information technology has limitations. A developer pointed out, “We use a phone or a video set whenever a new employee needs to converse with an employee in remote site but it’s not efficient.”

Another developer explains it this way, “You have to have a face-to-face and one-on-one training in order to make it efficient. You can do the training through the phone or video conferencing, but it’s not efficient.”

4.2.1.5 Lack of Accountability.

One of the interesting aspects of the Scrum method is the ability to see the visibility of the progress of each project in every thirty or fifteen days, depending on the duration of the Sprint. In the Sprint planning meeting, team members divide items in the product backlog into a set of small and manageable tasks, which will be entered into Sprint backlog. Based on the estimated completion time for each task, the set of all tasks in Sprint backlog is determined in such a way that they can be completed within a month. However, the tasks in the Sprint backlog usually do not get completed as estimated, and tasks are consistently carried over to the next Sprint Meeting. Moreover, nobody takes responsibility for that. A project manager noted, “Some tasks just keep getting carried over to the next month. There does not seem to be any responsibility on the developer side to complete the task. If they do not, it is o.k. and they hope to do better the next month in estimating their project.”
Another unique aspect of the Scrum method is self-managed teams, where the team is managed by the team members without having too much supervision from outside of the team. As part of the self-managing team, team members in the daily Scrum meeting choose tasks that they will work on for the day and report what has been done since the last daily Scrum meeting. However, this setting seems to be not working properly due to the team member’s lack of accountability on the tasks they chose. This seems to be a source of delayed projects. A project manager noted, “In the daily Scrum, there seems to be some lack of ‘what did you do yesterday?’ accountability. And taking on specific tasks for that day to work on is usually too generic. I believe we have become too relaxed on what we accept and that encourages some projects to drag over time.” Another issue is that project managers do not have any authority to urge developers to work faster or harder. The managers do not take any accountability on delayed tasks because they do not have any control over that. The project manager contributed, “When we meet with the management team at the end of the month, no one seems to take the responsibility for those items not completed. The project managers do not have any control on getting people to work faster or harder, so it is a little frustrating.”

4.2.1.6 Trust and Confidence.

Trust and confidence is an issue noticed between developers and Scrum masters. A Scrum master is supposed to do administrative work, mainly by helping developers focus on their work, by providing what the developers need, or by removing developer’s roadblocks. It seems that developers badly want to have a Scrum master who does the job. However, one Scrum master was unable to get developers items or information they need and was unable to remove developer’s impediments. He also did not follow up with developers to explain why he was unable to help them. A developer mentioned, “We need somebody who is going to be able to get us what we need and somebody telling us that ‘You’re going to get your information back.’ And we weren’t getting any information back and things
weren’t followed up.” As developers do not get what they need from the Scrum master, they quickly lose their trust and confidence in the Scrum master. Another developer stated that, “I can’t trust people who are not doing the job that they are supposed to do or that they said they are going to do. I lose my confidence when things don’t get done.” Trust and confidence as an issue was also noticed between developers when developers worked together on the same project, and they did not see any progress on a module that was assigned to a developer. A developer mentioned that, “I usually work on a certain task with other developers, and sometimes I lose my trust if I don’t see any progress from a task assigned to other developers.”

The same issue arose if developers were divided into two development sites. For example, a Scrum team member asks a team member at the other site to do a certain task, but no progress on the task is visible. It seems developers at one site had a hard time establishing a feeling of trust in Scrum masters or developers at the other site when they do not complete tasks, and the lack of trust led to a reduction in confidence. As a developer stated, “People in one site didn’t trust people in the other site if they were not able to get the job done. I think that was one of major problems we had. That really hurts our confidence.” Another developer talks about the level of confidence and wants to know the level of information that other developer has. He noted that, “I need to be able to have a confidence level. I think it’s really important to know who you are talking to and what level of information they have.” He also mentioned, “I work better with people who I know better so I understand what they are really saying.”

**4.2.2 Structured Development Process Factor**

As shown in Table 4.1, the structured development process factor consists of the Scrum framework, unit and integration testing, coding standard, documentation, and formal code review. Each concept is presented in the following section.
4.2.2.1 Scrum Framework.

The majority of developers are in favor of a Scrum framework. One developer expressed that, “I like how the Scrum model defines a product backlog, helps us prioritize tasks, keep track of task assignments, and helps us monitor task progress.” Another developer stated that, “Scrum increases our communication in the team. There seems to be better teamwork using the Scrum model. Working in the Scrum team provides motivation, excitement, and interest.” A project manager also mentioned that, “I think the Scrum model gives us all a goal to strive for each month. This is motivating and helps us stay on task.”

In terms of daily Scrum meetings, a couple of teams had a standup meeting and the rest of teams had a seated meeting. However, the principal of short daily Scrum meeting time was well observed (less than 15 minutes) by all teams. Several developers claimed, “it was not necessary to have Scrum meeting everyday when there were no specific agenda to discuss”, but the majority of developers and QA personnel enjoyed this daily-based meeting. The daily Scrum meeting also seems to be a good time for team members to understand what other members were working on and identify obstacles to overcome. For example, if a developer says, “I am stuck with a certain problem,” other developers would typically respond with “I can help with that. Let’s get together after the meeting” or “I don’t know the solution for that but I can look at your problem after the meeting.” One developer noted that, “I like the daily status updates, especially on high priority tasks.” Another developer stated, “Through the daily Scrum meeting, team members are able to refine the goal for each Sprint and improve the quality of products.”

The firm appointed a Scrum master for each of the five development teams. Because the company believed that the main role of the Scrum master is to provide the administrative services for his or her team members, the company appointed nontechnical persons as Scrum masters. The non-technical Scrum masters sometimes
seemed to be a problem for developers. One developer stated that, “Not having the correct Scrum master was a problem. Team impediments were not taken care of and often needed to be repeated, slowing or even stopping progress.” Another developer specified, “Our Scrum masters don’t have enough technical backgrounds to understand the things that we’re dealing with. They sometimes had a hard time removing our roadblocks because of their lack of understanding of our technical stuff.”

The firm also held the Sprint review meeting, called a “products fair”. The products fair was unique in the sense that developers in all teams, QA personnel, and people in other department (sales, marketing, customer support) were strongly encouraged to attend. The firm expected that, given an opportunity to show their accomplishments during the Sprint, developers would diligently work to make sure that they are currently on the pre-determined development schedule with appropriate levels of intermediate deliverables of the products.

The Sprint planning meeting seemed to provide smaller and manageable tasks to developers. Several developers stated that, “I like the Sprint planning meeting where you have to breakdown the projects into smaller tasks. This makes it easy to manage big projects.” Also, the product backlog and burn down chart appeared to help developers to organize and prioritize the schedule, and track the progress during the Sprint. Several developers expressed similar opinions, like “I like having product backlog reviews to organize and prioritize the schedule each month. I also like a burn down chart and its ability to track our progress during the month.”

Though most of people liked the daily Scrum meeting, some of developers expressed their worries. One developer stated: We meet each day as opposed to weekly or bi-weekly. It is good to have feedback, but sometimes it feels like too much time for not enough value. I don’t think it is necessary for that much reporting.
and it takes too much time away from programming. I think we could report 2 or 3 times a week and still have the productivity level we now have in our team.

Another developer also mentioned: “While it is important to deliver quality products in a timely manner, it is also important to not spend too much time in meetings that do not offer a good return on their investment. Meetings are an investment for any company and keeping a good balance requires input from all parties and honest evaluation of time spent.”

Several developers expressed their worries on the Sprint meeting. One of them thought the meeting occupies too much time. He mentioned that, “on the negative side, I think it’s a lot more overhead to have the Sprint meetings, which take all day at the end of each month.”

Some QA personnel also wanted to have a streamlined planning session. One QA person stated that, “We need to streamline the planning session as much as possible. Going to meetings repeatedly is not productive.” Finally, one developer didn’t like Scrum at all. He mentioned that, “Frankly, I don't like Scrum, I don’t think I'd do it given the choice. I would prefer to simply adopt some of the Scrum methods into current processes.”

4.2.2.2 Unit and Integration Testing.

When regular teams were reorganized into Scrum teams, a QA person was assigned to each Scrum team. However, the QA person in each Scrum team needs to wait until developers in the team finish their coding during the Sprint, which lasts usually for 30 days. This process causes a problem because the QA person in each team is always behind the Sprint schedule. A QA manager mentioned: I find that it’s really hard to test code if there’s no code to test. So, QA is always behind the rest of
the team. The programmers finish coding within the Sprint, but QA isn't finished within the Sprint. So, QA is added to the next Sprint.

Another problem with this setting is that when QA people work on the code that developers just finished in the previous Sprint, they often times need to talk with developers about the code and developers do not want to go back to the code that they already checked in. One QA person stated that, “It's like pulling teeth to try to convince the team that I'm going to find bugs with their code that they just threw together.” QA people want developers to spend some time to talk about bugs with them. Developers just want to keep writing code instead of revisiting the code created during the previous Sprint. Another QA person mentioned that, “Developers really should set aside some time to address the bugs in a timely manner so that I can continue with testing.”

To mitigate the problem, Commercial Application Development Firm pulled a QA person from each Scrum team and created a QA-only Scrum team. The QA-only Scrum team covers all code created by other Scrum teams. It seems this change has improved the efficiency in testing because QA people usually have some code to test, and they do not need to wait for developers in one Scrum team to finish their code. Associated with this setting, QA people were usually in charge of both unit and integration testing. The firm changed the testing process in such a way that developers took responsibility for unit testing, because QA people sometimes do not have the detail picture of the particular area where a developer worked.

An executive officer offered: QA people are asking for developers to do some testing. In the past, the development was kind of passed over to QA and they took care of testing. The problem with that is a QA person may not know what particular areas could be affected by the changes made by developers.
One developer also mentioned that, “If you know what could be affected, you may focus your testing just on those areas. But sometimes, QA people do not realize that there is another area also affected by some of the changes.”

Another developer shared his personal experience: One time I made some changes in UU command which updates a unit. It turned out that my change broke UC command which updates call and broke other commands. But QA people did not do any tests on those areas because they did not know they were related.

Due to the issues described above, Commercial Application Development Firm asked developers to do some portion of the testing that QA people usually cover, but this may not be a good move economically for Commercial Application Development Firm because developers usually get paid more than QA people.

4.2.2.3 Coding Standards.

Commercial Application Development Firm has utilized coding standards. They have very specific coding standards in many areas to facilitate easily maintainable and expandable code.

Commercial Application Development Firm’s document on C# coding standards describes comprehensive rules and conventions that developers should follow. Table 4.2 displays two coding standard examples associated with headers and naming conventions.

<table>
<thead>
<tr>
<th>Area</th>
<th>Sub Area</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Headers</td>
<td>File</td>
<td>The copyright notice and the CVS information are all that will appear at the top of a .cs file. These will account for the first 4 lines of each file.</td>
</tr>
<tr>
<td>Class/Delegate/Interface</td>
<td>Class headers will be in xml format to facilitate the automatic documentation feature. If the class description is longer than one line, add a remarks section.</td>
<td></td>
</tr>
<tr>
<td>--------------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>Method/Event Handler</td>
<td>Function headers will be in xml format to facilitate the automatic documentation feature. If there are any exceptions thrown in the function, they must be documented inside the exception tag in the function folder.</td>
<td></td>
</tr>
<tr>
<td>Properties</td>
<td>Class properties should be documented through ‘summary’ and ‘value’ tag. A remark tag may be used for more detailed explanations, but generally will not be found in properties.</td>
<td></td>
</tr>
<tr>
<td>Capitalization Rules</td>
<td>Class, Enum type, Event, Interface, and Method should use Pascal type. Parameter, Variables, and Protected should use camel type.</td>
<td></td>
</tr>
<tr>
<td>Abbreviations</td>
<td>Do not use abbreviations or contractions as parts of identifier names. Do not use acronyms that are generally accepted in the computing field.</td>
<td></td>
</tr>
<tr>
<td>Type Name Confusion</td>
<td>Different programming languages use different terms to identify the fundamental managed types. Class library designers must avoid using language-specific terminology.</td>
<td></td>
</tr>
</tbody>
</table>

By using the coding standard, developers agree that they can understand other developer’s code better without spending much time. One developer stated that, “I like a formal coding standard. It gives the same style and format, and this consistency helps me a lot in understanding other developer’s code.” However, one developer worried about putting too many coding standards on developer’s shoulders. He stated, “Having coding standards is good, but too much coercion to the standards may hamper our performance because we need to look at coding documents back and forth while we are coding to conform to standards.”
4.2.2.4 Documentation.

After the firm adopted the Scrum process, the amount of documentation was reduced significantly. For example, detailed design documents including class diagrams, sequencing diagrams, activity diagrams, communication diagrams, and use cases were reduced or eliminated. A developer mentioned that, “Before we went agile, we required that detail design documents were prepared before code was written. But we are not creating detail design documents anymore.” A QA manager also uttered that, “The detail design documents were routed to the QA manager, development manager, document manager, and sometimes to others.” She also explained how detailed design documents were utilized: As the QA Manager, I read though those documents with a red pen in hand. I verified that we were standardizing things. If I had questions that were not addressed in the design about how something should work given a certain scenario, then I made sure that those questions were answered and the design document updated. The documentation manager also reviewed the document to make sure that the error messages, field prompts, reports, etc. were grammatically correct and that everything was spelled correctly.

A certain amount of documentation seems to be very useful when developers work on a complex project, try to find and turn around fixes and problems, or need some ideas and questions for the project. One project manager said: When we were working on a complex project where we would make one change and have QA test, then work on the next change, test was very successful. Documentation was there and notes were taken as we went. We were able to find and turn around fixes and problems quickly using a document. Documentation, the input of QA, and others who were familiar with the project, gave us a wider range of ideas and questions for the project. Several subjects noticed that the reduced amount of documentation is a big headache. A project manager worries about quick implementations without creating detailed design documents. The quick implementations seem to cause
ripple effects that damage other parts of the project. She noted, “One drawback is that there are no detailed design documents. Whatever the idea is, it's just implemented. No one is taking the time to think about what effects it may have on others parts of the program.” Another problem is the increased number of bugs due to the lack of standardization of features, field names, and error messages. This kind of standardization would have been easily implemented if the documents had been created. A software tester stated, “There's no thought put into the standardization of features, field names, error messages, or anything. It appears that code is just thrown together. So, when I go to test it, it falls apart.” A QA manager confirmed this statement as she mentioned that, “I'm finding twice as many bugs since we went to agile as I did before we went agile. Once I have the basic bugs resolved, then I can really start testing it. And the system doesn't work the way that I anticipated.” The increase in bugs requires a lot of developer working hours to fix, and this is a major issue because the code should be re-written. The QA manager cited this example:

I go to the programmer and they didn't even address the issue because it wasn't part of the design and they don't know how it should work. I end up calling a meeting with part of the team to discuss design issues that should have been addressed prior to the project even being coded. And most of these are not little design issues that a couple lines of code can fix. These are major issues and major code re-writes. And then once it's done, I've got to start testing everything again.

Several developers agreed that design and documentation should be a necessary part of completing a project. Developers are not in favor of reducing the amount of documents. A developer explicitly expressed his opinion when he said, “I don’t like the fact that our design and documentation requirements have lost their focus in Scrum.”
4.2.2.5 Formal Code Review.

Commercial Application Development Firm has utilized formal code reviews since they started producing high-quality software applications. A web-based, formal-code review tool was created by a project manager in the firm and it has evolved to the current version through several revisions. When developers write or modify the code, they are required to go through a formal code review process before they check the code into the central code repository. Developers need to choose two code reviewers, and one of them is usually a senior software engineer or a developer who knows the area well. Once the code is delivered to the reviewers through the web portal, the reviewers provide feedback about if there is a side effect, a ripple effect to other code, or the code looks good. Developers and code reviewers sometimes have a meeting to discuss code changes that have a heavy impact on other modules. In this case, other senior developers or project managers are invited to the meeting.

All interviewees agreed that the formal code review was a necessary and essential step to construct robust applications. One developer declared, “I think the formal code review keeps us in the right track. We can find inaccurately written code in the initial development phase and fix mistakes overlooked by a developer.” Another developer commented, “The formal code review can keep our code from being exposed to vulnerabilities such as buffer overflows and memory leaks. This process makes our code more robust and secure.” A project manager expressed the following: We go through a very thorough code review. Often times, our code review requires a line by line review. We can avoid a lot of common mistakes by having two or more people examine the same code though it sometimes takes more effort and time. It has been proven that our code review process is effective at finding defects in the code under review. I think the formal code review is a vital and critical process in creating high-quality software applications.
4.2.3 Environment Factor

Four identified elements of the environmental factor include

1) customer involvement,
2) working environment,
3) interdependency among modules, and
4) the group of social facilitation, social loafing, group motivation, and evaluation appreciation.

They are discussed in the following section.

4.2.3.1 Customer Involvement.

One of characteristics of agile methods is a constant customer involvement in every aspect of the software development process. As part of the customer involvement, a customer representative is required to attend meetings. However, Commercial Application Development Firm could not invite customers to all of the meetings, such as the daily Scrum and the Sprint planning/review meetings, because there are so many customers scattered. Instead, product managers visit customers on site or talk to clients through WebEx to gather the project requirements, to show the progress of the products, and to obtain feedback from them. A project manager related: We spend almost a half of our work time in talking to our clients. Through the conversation with clients, we gather project specifications, show them the progress of our products, and receive their feedback. If possible, we use WebEx to show features of the products and some charts and graphs, otherwise, we visit them on site. Another project manager spends many of hours visiting agencies and comes back with project requirements. He then creates a to-do list and a graphical user interfaces for developers. He mentioned that I go out to meet our clients and conduct research on what we need in a product. I come back to company with lots of requirements that our clients want. I produce a list of things we need to create and talk with developers to develop the product for our clients. I
spent about 9 months last year doing research, going out to visit each agency. That is my job, just going out and gathering requirements, keeping track of equipments, putting together wire frames of what the screen should look like, what kind of data should be on it, and how it should be presented, and then take it back and review it.

Commercial Application Development Firm also hosts users’ conference once a year to get feedback from the customers. At this conference, customers vote for or against a new direction of application development. A project manager articulated that, “We’ve been hosting users’ conference for several years, and it has been very successful. One of the important events in this conference is that customers can vote for or against a policy and a direction of new application development.”

One interesting notice associated with the customer involvement is that QA people have a hard time providing customers with a quick turn around on bug fixes because of Scrum settings. A QA manager stated that, “Going to agile has been a hard adjustment for most of our customers. We’re unable to give them the quick turn around on bug fixes like they used to get. We can’t give them a quick turn around on bug fixes because we can’t interrupt the Sprint.” She continued to explain it by saying that, “Before we adopted Scrum, we had been able to fix customer’s bugs quickly. But now, we focus more on the code created during the Sprint and place less priority on customer’s bugs because we don’t want to interrupt the Sprint process.”

4.2.3.2 Working Environment.

An open working environment is recommended by the Scrum method because it promotes communications, facilitates self-organization, and helps developers get together easily. Before Commercial Application Development Firm started the Scrum process, most developers had their own office. After adopting the Scrum process, all the developers were reorganized into cubicles in such a way that
all the team members are placed inside the same cubicle area. Most developers do not like this new work environment and consider the cubicle settings less efficient because they cannot concentrate on their work while their coworkers talk to other coworkers. One developer complained: We share cubicles with a coworker. I think this is a bad thing, because we are constantly distracted by the person we work with. Cubicle-partners are often having conversations with other coworkers about their tasks, and it is hard not to get drawn into their conversations, even if you don’t have anything pertinent to add to it. Overall, I feel it is distracting and generally less efficient. Another developer stated, “Don’t pair developers up in a cubicle with someone else. Place them near team members, but give them enough space that they can concentrate when necessary.” A QA person also mentioned, “I was more productive when I had my own office. I was not distracted by my coworker’s phone conversation or other noise going around me.” However, some developers think that cubicle settings increase the amount of communication between team members. One developer mentioned, “I like the open space working environment. It promotes our communication. I can easily grab one of my team members and discuss issues and problems.” Another developer stated, “I feel I can talk to our team members easily and ask questions quickly.” One developer proclaimed, “I like the idea of placing every team member inside one cubicle area because I think it fosters collaboration and teamwork.”

4.2.3.3 Interdependency among Modules.

In the first stage of the projects, the Scrum method was not very effective in terms of bug rate. During the first three months, for example, the QA department found almost twice as many bugs after the firm switched to the Scrum method. A QA manager confirms, “I’m finding twice as many bugs since we went to agile as I did before we went agile.” They attributed the finding to the complexity of the project. As the size and complexity of the application grew, the dependencies and interconnections among tasks in the application increased. However, the developers
were not able to fully consider all the dependencies and interconnections among modules because of their narrow-minded planning and design in each Sprint planning meeting. A developer explained: Team members seem not to take enough time to think about interdependencies and interconnections between modules. After several Sprint cycles, we realized that we didn’t think about what effects this module may have on other part of the project or how this module may be interrelated with other tasks in the product backlog.

It was also noticed that when developers had to complete a set of tasks, determined in each Sprint planning meeting, they had a tendency to complete tasks in a quick and dirty way rather than to think of how the code will be maintained, and how each task in the current Sprint planning meeting can be flexible for future changes. A developer stated, “We have a tendency to do things in a quick and dirty way rather than to think into the future. (How will we maintain the code, and will it be flexible enough for future needs?)” Another developer noted, “Sometimes things are rushed into the Sprint meeting and the team doesn’t take the time to add such tasks into the Sprint backlog.” Further, the developers often found that their monthly work schedules for producing intermediate deliverables were optimistically estimated, failing to finishing all the tasks in the Sprint backlog. A developer mentioned that, “When we plan things in our monthly Sprint planning session, we underestimate how long things will take, and that leads to the failure of finishing all the tasks in Sprint backlog.” A project manager also stated that, “Some tasks just keep getting carried over to the next Sprint backlog. They hope to do better the next month in estimating their project.” Social Facilitation, Social loafing, Group Motivation, and Evaluation Apprehension. This section is quite different from the other sections because it was developed through observations based on social facilitation, social loafing, and group motivation. Social facilitation indicates that in the presence of other people, performance is focused on a simple task, whereas performance on a difficult task is hampered [100]. Social loafing is the tendency to take advantage of
others’ efforts when working in groups, while group motivational gain is obtained when people increase their effort to help co-workers whose performance is poor.

Social facilitation effect and group motivational gains were observed as factors that reduced development times and lowered bug rate. In particular, evaluation apprehension [117][118] was observed when the firm invited all developers, QA personnel, and people in other departments (sales, marketing, customer support) to the Sprint review meeting. In the Sprint review meeting, all developers had a chance to show what they had done in an interdepartmental environment and they were concerned about how they were evaluated by these people. Group motivational gains were also obtained as a social compensation effect [119] when a team member helped other team members who were not familiar with new development tools or programming languages. At the same time, Commercial Application Development Firm did not have ways to evaluate individual performance, leaving room for individuals seeking to free-ride.

4.2.4 Information Systems and Technology Factor

The information systems and technology factor is comprised of communication system, information and knowledge sharing system, and bug tracking system and management tool. These concepts are discussed in the following section.

4.2.4.1 Communication System.

It seems Commercial Application Development Firm realized and understood the important role of communication in the software development process because they have put a lot of efforts into establishing good communication channels. As mentioned in the third chapter, Commercial Application Development Firm had two development sites for about a year and developers were divided into two geographic locations. With the two development sites, they set up devices to have video conference capabilities between the two sites. However, due to the time and
use constraint, the video conference was not always available for each individual software developer. Rather, the video conference was usually used for team meetings between sites once or twice a month. Because of the unavailability of video conferencing, phone conferencing was used a lot. It seems that phone conferencing has problems as well. A lead engineer stated: When we have the daily Scrum meeting in the morning we did it in a couple of different ways. One was video conferencing and that was actually quite good. We also did phone conferences, which were not as good as video conferencing, mostly because when somebody made a comment, you couldn’t see their face, which gives the background. Of course, the Scrum master couldn’t do anything about that. That was counter-productive. If you have two groups video conferencing, that is much better. You can see face-to-face and if somebody is upset, you know about it. You can’t hear people making faces over the phone. That did happen. Other developers also mentioned in the survey that video conferencing helped them to see the other person’s facial expressions, gestures, and body language. One developer stated that, “it’s still hard to know whether they understand what I am explaining to them”. Though video conferencing is much better than phone conferencing, there are still limitations to using video conferencing. The daily Scrum meeting and the daily Scrum of Scrums meeting have mostly been done through phone conferencing. The monthly Sprint planning and Review meeting have been done through video conferencing. Other than video and phone conferencing, software developers have been using a tool called web demo (http://www.beamyourscreen.com) to show people in remote sites things on someone’s screen. This tool, together with a phone system, seems to work very well, as a developer mentioned, Actually we used web-demo which worked out very well for us when we are going to do code reviews and we used that web demo with phone conferencing. And it was like being in a media-room. Actually, it was much better than being in a media-room in a lot of cases because we were able to scroll to where we want to look at. It was very efficient. The last multimedia that the company has been using are an Instant Message (IM) and an email
system. All of the developers mentioned that they were using the instant message and email system much more than before they were divided into two groups. It seems the instant message system is used more effectively than the email system for a short and quick question and answer. The instant message system was also used more by new employees. Several developers agreed on the following statement: Yea, a lot more when you are working with a new person. You are using it (IM tool) all the time because he’s asking questions back and forth over an IM session. You are using it a lot with a new person when, if he would have been here, he would have come in the office to show to him instead email him back and forth, or you would have been looking at it together. You could have used web demos to do that but in most cases it is short questions. We didn’t feel like we had to do something like that. One developer, who used to use VPN mentioned, “Sometimes, the VPN is too slow, which causes frustration.” It was obvious that people did not use VPN once they found it was not fast enough. Though the video/phone system, web-demo system, and instant message/email system help software developers reduce the geographic distance between two sites, it seems those multimedia systems have not been working as optimally as face-to-face conversation. Several developers mentioned that, “It would so much easier to be able to work with somebody right here in my office.” An executive officer also mentioned, “Well, in some ways it would be better to be down there at times.” He went on to explain that it hasn’t been that big of a disadvantage but once in a while, we just get two or three guys together, if there’s an issue and somebody wants to talk about options or kind of brainstorm on the idea. For me to join that group from this remote site is more difficult.

The company seemed to suffer a lot when a communication line was down because of their heavy dependence on it. A lead engineer mentioned that, “We had a few problems last month with phone systems not working very well. When the phone systems were down it was a big deal, plus when phone systems were down the data line went down too.” Because the telephone systems and data systems share the
same physical line, employees could not do anything, as a project manager stated: For example, you’re working on a summarizer or something like that, all of sudden, the screen is closed down. And if you are in the middle of some debugger and the data line is gone, you will lose everything and you have to start it from the beginning. That’s why one of our programmers went over home. She’s working out of her home now.

4.2.4.2 Information and Knowledge Sharing System.
When the main part of Commercial Application Development Firm moved to a different location, the firm went through major changes. One of them was a change in staff. The vice president of the software development division related that, “There was a change when we moved to a different location with new staff, and I think we lost about 15% of our staff, including software developers.” To fill the empty positions, Commercial Application Development Firm hired several new software developers. These newly hired people created more bugs; a project manager said, “There are a lot more bugs now because people are not always knowledgeable about the software that they are working on” and “often they change some code, something like a radio log which is very complex piece of code and slight changes to that have a lot of impacts in lots of areas. It takes a time to be familiar with our complex systems.”

Because of the complexity of Commercial Application Development Firm’s software system, it is important to have well-structured information- and knowledge-sharing systems between experienced software developers and brand new software developers. In particular, it is critical to have such a knowledge sharing system if new software developers in one location need some of the expertise of software developers in another location. One developer stated, “It is a little difficult to share information and explain ideas over those large distances.” Another way to get around this problem might be placing new developers with senior developers
within the same Scrum team in one location. But this solution is not always feasible for Commercial Application Development Firm because of the complexity of software systems and the situation that developers often need expertise from other developers in a different Scrum team and in a different location. So, to facilitate the knowledge and information-sharing, the firm has been using a web-based Wiki program that enables developers to add and edit items that might be critical to other developers. For example, the section called “gotcha” in the company’s Wiki includes the most frequent mistakes that developers can make in many different parts of the company’s software system. Due to easy access to the Wiki program, the information stored on the Wiki database mitigated the problems between the two sites.

4.2.4.3 Bug Tracking System and Management Tool.
Commercial Application Development Firm has been using a bug tracking system called MOM, which was developed by the firm on the UNIX platform. The MOM bug tracking system is shared by many groups, including support people at the help desk, developers, and QA personnel. A brief description about a bug and the step-by-step procedures to duplicate the bug are entered into the system. One developer mentioned that we use our own bug tracking system called MOM. QA personnel and support people usually describe the nature of a bug, the place where it is found, and the procedures to duplicate the bug if possible. Once the description is entered into the system, it can be viewed by developers, QA personnel, support people, and managers. Based on the level of severity, each bug is labeled with a number between zero and three. The number zero represents very urgent and the highest level of severity, whereas the number three represents the lowest level of severity. A person who is assigned to a zero level bug is required to stop all the current tasks and work full-time on the bug until it is resolved. One developer mentioned: All bugs are entered into MOM and are assigned to an appropriate developer or group of developers. A zero level bug is uncommon but when it happens it is very critical
to solve the problem as soon as possible using all possible resources because the zero
level bugs could stop some parts of or the entire operation. As mentioned earlier, the
firm adopted a commercial bug tracking system called JIRA
(http://www.atlassian.com/software/jira/). A vice president in the software
development department stated: Well, recently we moved away from the MOM
system. We used to create support problems and then create a duplicate for
development in the MOM support system. We have now moved to JIRA. I think in
MOM you didn’t have fields for certain information. Everything was pretty much
put into a narrative environment or a report. JIRA is used to record defects found in
alpha and beta testing and in other areas. A manager stated that, “JIRA only logs
defects that we found in alpha and beta testing and also defects in standard code
that a customer told us about. It’s just a tracking tool for the development division
for defects in products.” It seems JIRA has filter functions that MOM does not
provide. The manager mentioned that, “JIRA has a lot of information that used to be
found in MOM. But it has important filter functions which we couldn’t do in MOM.
It has some advantages so that I don’t want to go back to MOM.” Another
advantage of using JIRA seems to be that it provides priority codes that rank the
level of the severity of bugs. This function automates the selection of a next bug to
be tackled by listing the bugs in sequential order based on the priority. The VP in
software development stated: Well, one of benefits of JIRA is that it has severity
priority codes. It was a part of BETA test, and there’s a number of other fields that
down the road we will be using to divide the information. We are working a way to
try to determine as efficiently as possible what bugs to work on next. JIRA can still
automate the choice versus having a person just look at and say what you need to fix
or not. We try to use that information to do that. One of disadvantages of using JIRA
appears to be that it does not offer a decent search function on the bug information
already typed into the system. This is mainly caused by the JIRA interface, which
requires a user to enter the narrative bug information into separate text boxes.
In order to further improve the effectiveness of project management, Commercial Application Development Firm used two different managing tools: Microsoft Excel and VersionOne. Microsoft Excel was used first, but was replaced by VersionOne because of Excel’s limited capability for efficient project management. VersionOne, as mentioned earlier, is a web-based commercial management tool that provided various functions, such as simplifying project planning and management, enhancing business and project adaptability, improving project visibility, and increasing project predictability and confidence. Some developers expressed difficulties in using three different bug tracking and management tools. One developer complained that . . . it’s hard because we are using MOM for part of our tasks, we are using JIRA for another portion, and we are using VersionOne to track the development cycle and Scrum cycle. So, it makes difficult that way. We can’t use one product for everything.

4.3 e-GOVERNANCE APPLICATION DEVELOPMENT FIRM

The e-Governance Application Development Firm had five Scrum teams, and each team consisted of three or four developers, a project owner, and a Scrum master. Due to the small size of the firm, only one quality assurance person, and one database administrator were designated to provide services for all Scrum teams. Every Scrum team worked on a new project except for a team that worked on ongoing maintenance. Table 4.3 summarizes the common categories, concepts, and data related to e-Governance Application Development Firm. As shown in the table, the factors of human resource management, structured development process, environmental, and information systems and technology were constructed for e-Governance Application Development Firm. The concepts comprising these categories are discussed in the following section.
4.3.1 Human Resource Management Factor

The concepts of training, collaboration, and multiple responsibilities constitute human resource management factor. Each concept is discussed in turn.

4.3.1.1 Training.

It seems that e-Governance Application Development Firm needs to provide developers with more formal step by step training. Most developers feel that they did not have enough training on the Scrum development method. Due to the lack of training, some of developers do not see the big picture and the benefits of Scrum as a whole. One developer mentioned, “I don’t think our whole team buys-in to Scrum development in general because we are not getting a whole picture of how Scrum is working and how it benefits us.” Another developer confessed that he had a misconception on the relationship between a project and a Sprint. He thought he should complete a full project within a single Sprint. In reality, a project is usually completed through multiple Sprints.

<table>
<thead>
<tr>
<th>No.</th>
<th>Common Categories</th>
<th>Concepts</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Human Resource Management Factor</td>
<td>Training</td>
<td>• Developers need more formal step-by-step training</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Some of developers do not see the benefits of Scrum</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Company provides brown back lunch training and a “luncheon learn” meeting</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Collaboration</td>
<td>• Scrum provides good collaboration mechanism between developers</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Collaboration between developers and product managers should be improved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Scrum helps developers to be aware of other developers’ tasks.</td>
</tr>
<tr>
<td>Structured Development Processing Factor</td>
<td>Scrum Framework</td>
<td>Formal Code Review</td>
<td>Unit and Integration Testing</td>
</tr>
<tr>
<td>-----------------------------------------</td>
<td>----------------</td>
<td>-------------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td>Multiple Responsibilities</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• One person is responsible for many tasks in different field</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Product manager is a bottleneck in the development process</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Developers have a communication problem with a product owner</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Scrum ceremonies force developers to be on the same page</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Daily Scrum meeting, Sprint planning meeting, and Sprint review meetings are sometimes inefficient</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Setting up the meeting time is difficult</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Company employs informal code review</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• The formal code review can impose accountability to developers</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Company uses a N-unit testing</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Testing self-created code is not efficient</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Legacy code is not suitable for unit or integration testing</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Wide range of testing skills are needed</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>The coding standard can provide easy maintainable code</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Company has a verbal coding standard</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• The coding standard can hamper developer’s creativity and reduce the efficiency</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Less documentation</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• More comments on the code</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Hard to complete the system without having any documents</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Equally shared skills and knowledge among team members</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Environmental Factor</td>
<td>Customer Involvement</td>
<td>Government Project and Scrum Method</td>
</tr>
<tr>
<td>---</td>
<td>----------------------</td>
<td>---------------------</td>
<td>-----------------------------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
|  | Project Estimation and Planning Poker | • Developers have difficulties to estimate legacy code related project  
  • Planning poker reduces estimation difficulties  
  • Breaking big tasks into smaller ones helps developer have good estimation |                                   |
|  | Use Cases | • Developers realize the importance of creating use cases  
  • Use cases help developers understand the system they are going to build  
  • Users do not know what user cases are |                                   |
|  |          |                      | • Government project requires a big planning and a big design up front  
  • Developers work with many unfamiliar jargons and acronyms  
  • Company needs a new hybrid development method |
When developers first began to work on Scrum projects, their project manager demonstrated the concepts of Scrum projects and how they related to the existing systems. After that, developers were to learn the Scrum concepts and working environments all by themselves. One developer commented, “We are usually taught briefly by a project manager how the system works the first time and then kind of go from there, I guess.” Another developer stated, “I did not have formal training. I looked up Scrum in the Wiki to get an idea what the working environment looked like.” Many developers want to have a continuous formal set of training. One developer noted that, “It would be helpful if we have a class that trains us, more

| 4 Information Systems and Technology Factor | Social Loafing | • Everyone is fairly motivated and quick to point out if anyone is not doing his share  
• Individual hard work is not recognized  
• Accurate measurement of individual performance is needed |
| Communication | • Daily scrum meetings improve communication within a team  
• Lack of communication between teams  
• Work is being duplicated  
• Lack of communication with customers |
| Bug Tracking System | • A web-based bug tracking system is helpful in prioritizing bugs and keeping track of bugs.  
• Mantis, a free bug tracking system works well except for some cases  
• Developers want customers to report their bugs in Mantis |
| Version Control Systems | • Open source revision control system called Subversion is a main system  
• Subversion provides a decent tagging and branching method  
• Subversion rebuilds the system automatically if code changes |
than just an overview or just a kind of tour around.” To mitigate problems caused by the lack of a formal training program, the company recently began hosting regular brown-bag lunch meetings to provide more training. A Scrum Master affirmed that, “Recently, we have been hosting more formal lessons on how things should work through a brown-bag lunch. I have been attending BB lunch regularly. It’s really helpful. Other than that, it’s really been just a brief overview.” In addition to the BB lunch meeting, the company offers a “lunch-and-learn” program. An operations director explained, “We do have pretty frequent lunch-and-learns where the company buys us lunch and then we have someone like a operation director go through and explain how Scrum works.” One advantage of having the lunch-and-learn program is that everyone including developers, Scrum Masters, Product Owners, Project Managers, and other staff members can attend the meeting and provide feedback. The operations director said, “The whole company can attend, not just some people, so that everyone can provide feedback. It also helps people to be reminded how to do tasks as well, because sometimes through the day-to-day we forget certain aspects of Scrum, so it’s helpful to be reminded of them and remember to do it right.”

4.3.1.2 Collaboration.

It appears that the Scrum method has provided the company with a good collaboration mechanism. Most developers feel that they have good collaboration between developers within the same team. One developer mentioned that, “I think it’s really good. I have often seen a team member go to another to ask about coding practices and methodology. I have seen that several times so I think that’s really good.” Another developer stated: Whenever I have a problem or issue to discuss with other developers, I am very comfortable talking to any of my team members. I think Scrum practices promote collaboration between team members. For example, throughout the daily Scrum meeting, we can identify what roadblocks we have to complete tasks specified in a Sprint backlog and what items need to be done
together. Further, it seems the Scrum method has helped developers be aware of other developers’ tasks and be interested in other developers’ success. This atmosphere seems to make developers self-motivated. One developer stated: It seems everyone is involved and they are interested in everyone’s success. Everyone is very self motivated. So, if they don’t have something to do, they go in and look at our Sprint backlog and choose the next thing. So, it’s more reach out and get than assigned to us. However, some developers think the collaboration needs to be improved between developers and the product manager. One developer stated that, “There has been great collaboration between developers, but there has not been enough collaboration between developers and the product manager. He has been busy with many tasks and has not been able to allocate enough time to discuss problems and issues with developers.” The collaboration issue between developers and a particular product manager is elaborated in the multiple responsibilities section.

4.3.1.3 Multiple Responsibilities.

The Scrum method does not recommend that developers be involved in multiple projects at the same time. At e-Governance Application Development Firm, all developers work on a single project. The one person with multiple responsibilities acts as the product manager, product owner, and accounting manager. Due to his multiple responsibilities, he is too busy and overloaded. Developers have a tendency not to talk with him enough outside of Scrum meetings or ask him to deal with other issues when they come up. One developer stated, “He tends to be a bottleneck for the rest of team members. We’ve been waiting on him giving us feedback. Sometimes it tends to be a little bit of lag in getting any questions answered.” Another developer mentioned: You know we have a product owner, but he is overworked and he has too many other responsibilities so he can’t dedicate himself to the product and can’t get all kinds of tasks done. It hinders our progress, and he doesn’t have enough time to find what exactly we have to do. So, we need to get
someone else to help with that. Developers have also a communication problem with him. One developer articulated: The only sort of communication problem we have is with the product owner. He is overloaded with too many other tasks throughout the company, so he is rarely available, and we have a lack of communication with him. So, that hurt us, but when he is available, the communication is fine. It seems the company needs to release some of this person’s responsibilities so he can concentrate on a single project and make himself available to other developers for more decent communications.

4.3.2 Structured Development Processing Factor

As shown in Table 4.2, structured development process factor consists of Scrum framework, formal code review, unit and integration testing, coding standard, documentation, project estimation and planning poker, and use cases. Each concept is presented in the following section.

4.3.2.1 Scrum Framework.

It seems that Scrum has been well adopted and has brought big improvement to e-Governance Application Development Firm. One developer stated, “Scrum agile has been working quite well for us on all of our projects. It has been a huge improvement over the waterfall method.” Another developer mentioned, “Scrum really helps team members to get involved in the project.” One project manager also noted, “The Scrum method helps team members to be aware of everyone’s progress, and in Scrum, nobody can fall behind without people taking notice.” Another project manager mentioned, “The Scrum framework prevents a scope creep because tasks can only be dropped from a Sprint, and Scrum also prevents the project from going too far off course if client’s requirements are not accurate.” Scrum ceremonies including the Daily Scrum Meeting, the Sprint Planning Meeting, and the Sprint Review Meeting, seemed to help developers focus on producing quality applications. Most developers testified that the Scrum ceremonies have been very
useful and very productive. One developer mentioned that “The 15-minute standup Daily Scrum meeting has allowed us to be on the same page because we can talk to each other and everybody knows what other members are working on.” Another developer expressed a similar opinion on the Daily Scrum meeting. He mentioned: This is a great way to make sure everyone is on the same page regarding work accomplished and work to be done. This allows quick assessment of those people who are already behind from the beginning and forces the team to make an adjustment and compensate. A project manager noted that, “The daily Scrum meeting is valuable and essential. We try to include clients in these meetings as much as possible.” Regarding the Sprint planning meeting, most developers considered it as an imperative and critical meeting. One developer noted: Time spent in elaboration during the Sprint planning meeting significantly helps estimating and scope planning because we are able to determine the number of employee working hours for the next 30 days and how much work we will be able to accomplish. Additionally, we have 3 or more developers estimate each development task which promotes in-depth planning discussions when estimates vary widely. This creates a more accurately defined Sprint backlog. One developer regarded the Sprint review meeting as a bridge meeting which leaded people into the next Sprint. He stated that, “The Sprint review meeting is important for transitioning into the next Sprint. Depending on the project, this may be a good time to plan a short (around 5 days) Sprint to develop post-launch patches that may be critical for the client.” Another developer also mentioned that, “This is very advantageous as long as all team members, including the client, are open-minded enough to have a candid discussion. It allows all team members to discuss the items that did not work well and what can be done to address them in future Sprints.” Regarding a product backlog and Sprint backlog, one developer stated: The master product backlog is invaluable in long-term planning as well as quickly ascertaining the next tasks. With our clients, we hold a master product backlog reorganizing meeting every few months to make sure the priorities are still specified accurately.
The Sprint backlog is the key to knowing what everyone is working on, how much time is left, and what tasks are left to do. Keeping the Sprint backlog in priority order and working on tasks in that order assures that the most important stuff is done first. Another developer also mentioned: Product backlog is important, especially for a project that is going to span multiple Sprints. It will help keep track of remaining tasks, help you prioritize them, and prevent them from falling through the cracks between Sprints. Be sure to update it as new feature requests come in. Having a product backlog makes Sprint planning much easier. Sprint backlog is the output of your Sprint planning meeting, and an essential tool for the daily Scrum.

However, some developers talked about inefficient Sprint planning and review meetings. One developer argued that, “Some of our Sprint meetings are so simple and it seems to be a waste of time spending a whole day just for planning and review. I think it needs to be adjusted based on the complexity of the project that we are working on.” One senior developer noted that, “Keeping daily Scrum meetings to 15 minutes was difficult. Some of this is caused by just gabbing a little too long, but valid reasons for taking too long are the amount of material we needed to discuss, and also because we hold Scrums for 2 or 3 projects at once.” Another developer mentioned: Our daily standup Scrum meetings sometimes go on a little longer just because everybody is talking about what they did last night. I think there probably is some good advice on trying to keep your daily standup meetings consistent and short so that people are not distracted and they can go back to work quickly as most people would rather work productively than waste time. Another issue is related to setting up the meeting time. Due to the flexible work schedule among developers, it is difficult to get all developers together at one time without interrupting their work. One developer stated: I think the hard things for us in Scrum is when to do it because some of us get in at 7:30 am and some of us at 9:30 am. So, as a team, we just have the Scrum as soon as everyone gets in. That’s usually at ten or eleven. The problem is that those who get in early are interrupted from
their work because they’ve been working very well for two or three hours. They are in the grove or zone so being interrupted is frustrating. We talked about doing it at the end of the day but that also has a problem because some people come in at 6:30 am and leave at 3:30 pm, and some people come in at 9:30 am and leave at 6:30 pm. It makes it hard for our team to get together all at one time.

Some of developers feel that Scrum might not be appropriate for large-scale projects because it is not easy to make a large team be agile. One developer stated: Agile is difficult with really large projects. It’s hard for large teams to be agile. It is necessary to split the large project up into teams of 8 with common goals. Let team of 8 Scrum self-manage. Project Managers can facilitate communication, and check status between teams. Another developer made a suggestion about tasks on the Sprint backlog. He suggested that tasks on the backlog should be updated dynamically and that tasks taking more than a day should be divided into small sub-tasks. He suggested: When updating the Sprint backlog every day, if the tasks you are working on don’t match the granularity of the tasks on the backlog, update the tasks on the backlog and redistribute the hours among the tasks. This is to prevent treating a group of tasks as a single bucket of hours that you are working against, which makes it more difficult to see exactly where the progress is. If a task is more than a day of work, try to break it up into sub-tasks. A developer noticed that some of Scrum team members were not flexible and did not spend enough time to create a detailed Sprint backlog. He stated, “I would like to suggest that people participating in Scrum be flexible. Rigidity seems to be the bane of Agile. Also, when we generate a detailed Sprint backlog we need to spend as much time as possible.” One last comment from a developer showed that some managers tried to manage the Scrum teams instead of coaching them. He asserted, “Managers – avoid managing! Let the team self-manage. Managers/Project managers should act as coach only – it doesn’t work if the project manager is running the Scrum like a status meeting. Project manager – step back and let the team run the Scrum meeting.” Formal Code Review.
Developers have not set aside a time for a formal code review, but they have been having informal code reviews. Due to the informal code reviews, tools that facilitate formal code review have not been developed and utilized. In addition, the informal code reviews has not been employed frequently. One developer stated that, “We don’t incorporate proper tools in our code review.” Another developer mentioned that, “We have an informal code review occasionally. We don’t do it very often and probably should do it more. We talk about it a lot but just never have the time to do it.” It seems developers are aware of the benefits of having a formal code review though they do not use it. One developer stated that, “Other people can offer feedback on how a developer is doing and how the particular code can fit into the rest of a project.” Another developer also mentioned that, “Through a formal code review, other people in the team can look at the assumptions and choices that I made and say this is a good work for that situation or not good for the situation.” Another benefit of having the formal code review is to impose accountability on developers because they know their code will be reviewed by other people. One developer stated: I think it’s really useful. It gives you accountability because you know at some point somebody else will go back and look at your code. In the short term, developers may not pay extra attention to their code because a lot of developers just take an assignment and write the code with the belief that no one will look at it again. Formal code reviews would solve this problem. Most developers agreed that visiting the code retrospectively is important in terms of improving the quality of code and enhancing developer’s coding skills.

The Scrum master mentioned a problem that might be caused by the lack of formal code reviews. He stated that, “There is a chance that a developer writes code which is already built in, for example, in the Dot Net framework or ASP Dot Net framework.” If a developer tries to write code which is already written, tested, and proved to be efficient by a third party, it would be a big waste of time and money. It appears that developers are good at looking at other developer’s code and giving
good feedback. It seems it is just a matter of setting a time aside for a formal code review and selecting an appropriate tool to facilitate the code review.

4.3.2.2 Unit and Integration Testing.
The company has been using a unit testing tool called “N-Unit,” which provides a unit-testing framework for all Dot Net languages. Each developer tests their own code, but they try not to because they think testing their own code is not effective. One developer mentioned that, “We test our own code but we try not to because it’s not effective to test one’s own code, and developers usually have an assumption that their code always works.” When developers test their own code they also consider an integration test. One developers stated, “Everybody tests their own stuff just to make sure it works and also making sure it works with the big picture and everything else works with it.” It appears that several quality assurance personnel can help other developers in testing regardless of team boundaries. A quality assurance person stated that, “We have a few people who do help out with testing. They are technically on the team but they are just available for everybody to help testing.” In addition to internal testing done by in-house people, clients are also involved in testing by visiting a test site and conducting a test. A project manager stated that, “Generally, when the application is getting ready for clients, the clients get into a test site and test. They do track and test, and enter bugs that they find.” It appears that the firm has a large legacy code base and the legacy code was not designed for unit or integration testing. This is a big challenge for the firm. A project manager stated that, “We’ve got all legacy code and it wasn’t written really for test cases.

It would be nice if we can figure out someway of going back and kind of cleaning some of those things up. In terms of quality assurance, that’s really about biggest challenge right now.” The firm also has code, called “code behind,” which is difficult for developers to test because that code works behind the curtain and there
is no efficient way to test the code. Developers wish to have a wider range of testers. One developer stated: Probably, we would get managed better if we had more people who have a wider range of software testing skills. Right now, usually, one person is testing, sometimes someone who is technical and sometimes someone who is nontechnical. They usually catch different types of bugs. Another developer uttered that developers could not test everything because clients did not have enough budget to cover one hundred percent testing. He also explained the process of a unit and product owner level testing. Our developers create their own unique tests. But we don’t have one hundred percent coverage because our client doesn’t have the budget to make us test everything. We do create a unit test, then we have our product owner-level testing, then developers test their own code as well, and then from there we put everything in to MANTIS, which is our bug tracking system.

4.3.2.3 Coding Standard.
Several coding standards for different computer programming languages exist. For example, Sun Microsystems provides the standard conventions for the programming language and Microsoft proposes coding guidelines for the C++ and C# programming languages. In addition to Sun Microsystems and Microsoft, other organizations such as GNU Free Software Foundation and CERT at Carnegie Mellon University also suggest standard coding styles and guidelines. The main benefits of having coding standards is that developers can provide easily maintainable code, and that developers can decipher other people’s code without difficulties. It seems developers are aware of the main benefit of following the coding standard even though they do not have a formal coding standard. One developer stated, “There is no formal coding standard; probably a coding standard could be of good benefit to us.” Another developer mentioned that, “There is no written coding standard, but we are close enough to propose to a person who is not following a norm or our verbal coding standard.” Some of the developers employ coding standards provided by the tools they are using or modify suggested guidelines to suit their environment.
One developer stated that, “The coding standard is dictated pretty much by MS Visual Studio.” Another developer mentioned that, “I took the Borland coding standard and changed it and updated it for what I thought it would be best for us.” All interviewed developers think that they are doing fine without having a coding standard. One developer stated, “Most people have their coding styles, which are decent enough you can follow them pretty well. So, it’s not a big deal.” On the contrary, one project manager thinks that forcing the coding standard can hamper developer’s creativity and reduce their efficiency. He stated that, “There are definitely cons to different styles of code being everywhere. People are creative where they can work the most efficiently. The problem with enforcing a coding standard is that it limits creativity.” Another project manager mentioned that, “I think it is not worth to it force coding standards that hinder our performance if a lot of people have to relearn how to code in a lot of places.

4.3.2.4 Documentation.

The Scrum method, like other agile software development methods, significantly reduces the amount of documentation. In fact, the agile methods claim that the code itself should be a document. That is why developers who are accustomed to agile methods place more comments in the code. Several developers mentioned they placed more explanations in any tricky piece of code and for any changes that they made. However, many developers agree that without having any documents, it is very difficult to complete tasks for those developers who are working on parts of the system they never worked on before and also for new developers who do not have much experience with the project. For both cases, developers who do not understand the project ask a lot of questions, which takes time away from developers who do understand the project. One developer mentioned that, “When I first got here, of course, I was overwhelmed. It would have been nice to have some documents that explain why certain things were done in a particular way and what they were.” One more developer mentioned that, “Agile methods do not use
specification documents. I think that might be a weakness in agile methods. The agile methods allow you to go much quicker as long as whoever is specifying has a very good idea of what clients want. If this is not the case, the agile methods are just as slow as anything else because you are going to have to get clarification.” Another developer also raised the issue of the lack of documentation. He stated “Right now, we have one guy who is the main guy. He knows all of the systems and I think, personally, that might be a mistake. Not because he is not good at it, but because it just makes one gigantic point of failure if he is hit by a bus or if he leaves for another company.” It would take several months for the firm to recover the knowledge that one main developer has. The idea behind reducing documents in the agile methods is to keep every team member equal by sharing skills and knowledge on the systems. In that way, if one person leaves, there is still a lot of shared knowledge that has gone around among other team members, so it is not a big deal. However, in reality at e-Governance Application Development Firm, this is not feasible.

4.3.2.5 Project Estimation and Planning Poker.
The developers at e-Governance Application Development Firm had a hard time estimating how long a particular project should take. The level of project estimation difficulty increases when developers need to deal with legacy code or when developers do not have the experience required to finish a project. One developer said, “It’s hard to get estimation on legacy code.” Another developer stated, “We are just doing the first project and the accuracy of estimation is hard to determine because we just don’t have much experiences on this kind of project.” To resolve or reduce the difficulties, developers were introduced to a new project estimation method called “Planning Poker.” In Planning Poker, all developers are required to pick a card, each of which has an estimation number to express their estimate of a particular task. If there is a big gap among developers’ estimation, developers discuss the task and try to narrow down the gap. One developer explained the procedure for Planning Poker as: For each task, we discuss for a minute or two and
call the vote from the planning porker card. Everyone holds up their estimates and then if they are pretty close then we just take an average of them and put them as an estimate. If they are very far apart, if someone says two hours and someone says 20 hours then we pause and re-discuss it to find out why the estimates are so different. As soon as everyone has a thorough understanding of the problem, then we go again until we get closer. Another developer commented: I guess the idea is to kind of narrow down the numbers. In the past we just made an educated guess, which is O.K. All the people on the product made a guess. There are usually three guesses, which result in three estimates on a task. If they are too different, then we talk about the task and figure out why we have different estimates. In Planning Porker, I guess there are ranges of numbers. I think 1, 2, 3, 5, 8, and then 13, 20, 40, and 100. So, when someone chooses 40 and someone else chooses 8, then we need to discuss about the task. If we are not sure about the task, then we talk about it and do something on the whiteboard to lay it out, and discuss what could be done. Developers think that there are many merits to using Planning Poker. One developer stated, “You can get the opportunity to say why you think it’s going to take so long, why you think it’s going to take not nearly as long, and then you can actually work out some of the issues.” Another developer mentioned: I think the advantage is that we can have a lot of discussion. That’s an advantage. A few times I come through…because I don’t know the whole systems as well as they do. I give some number and they give some other number. I think it eventually gets close together, but at least a project manager can tell me an overview about the things that are involved and make sure we are on the same page. A project manager also stated: It is kind of nice to just have some sort of anonymous way we can throw cards up and discuss it rather than writing it down and be coerced into thinking by something or by whatever everyone else thinking. So it is nice…it helps us to have each developer have their own opinion. It is noticed that breaking big tasks into smaller ones helps developers have better estimates. One developer commented: I think one of the things that really help is that everyone here seems to be conditioned to breaking
their tasks down into the smallest possible measurable tasks and then to say, this field needs to be changed to accept this variable, or something like that. Very small tasks...you could say that yeah that task by itself will take me an hour or two. And so it’s a lot easier to make the good estimates that way if you break it down into a small element. Another developer also expressed the importance of breaking tasks into small one. He said, “We already have our tasks from our product backlog. We break them up into small ones first in the Sprint backlog so that we can have more accurate estimates.” It is noticed that that every developer thinks Planning Poker is very useful, effective, and pretty reasonable as well because it gives developers a chance to express their honest estimation. However, experience, pre-knowledge, and skills are noted to be important factors in good estimation. One developer stated, “I think Planning Poker provides a more accurate estimation. That’s part of it. But I really think it’s just matter of experience for good estimation.” He also mentioned that, “I think it’s a good thing to have Planning Poker, even if it brings big differences in estimation numbers the first time around.” Another developer reiterated: It works well in the middle to the end of a project because now you have a very good idea of what you’re working with and the technology you’re working with. But initially, it got to be horrendous unless you’re working with technology that you’re extremely familiar with. Also, you know the technology is only half the battle. It’s the logic and the system itself that you have to figure out and sometimes it takes a lot more time then you think. This developer argued that, though Planning Poker can help your estimation, the bottom line is you need to be familiar with the technologies that you are going to use, the business logic, and the system itself. Otherwise the estimation will be horrendous. Use Cases. It is noticed that all developers agree that use cases can help them better understand what needs to be built and that they have the most success when they have use cases. One developer mentioned that, “In our Sprint planning session we did a fair amount of documentation in terms of creating use cases. When we had use cases we had the best success.” He continued to explain what his team did. He stated, “We first
received a list of items that we needed to build and then we wrote out specifics, maybe a couple of paragraphs for each item.” Another developer added, “I think it really helps us figure out the system that we are going to build.” However, there are several problems in creating use cases. A developer who was not familiar with the system had a hard time creating use cases. He stated, “I felt I was a little unprepared to write use cases for a web site that I was not really familiar with.” Another problem is that clients usually do not know the systems that they want to have and what use cases are. One developer stated, “They don’t know really what use cases are. So, basically you need to come up with something you think makes sense and write it down and then go back to review it. The idea is that it’s really difficult to come up with specifics for it.” Though clients do not know about use cases, it seems that they are comfortable with revising them. A project manager stated, “It’s a lot easier for them to revise something we’ve written.” Based on this description, it would be better if clients knew what they really want to have in their system and understand use cases. If this were the situation, developers could communicate better with clients and would create better use cases, which then leads to success of the project.

4.3.3 Environmental Factor
The environmental factors including customer involvement, working environment, common tools and problems between teams, government project and Scrum method, and social loafing were identified and are discussed in the following section.

4.3.3.1 Customer Involvement.
Customer involvement in the software development process is very critical to the success of the project. The agile methods state that the customer should be part of the development process from analysis and design to implementation and maintenance. However, at e-Governance Application Development Firm,
developers have difficulty working with customers on the projects. A project manager commented that, “Customers are not involved in the decision making process until it is all done.” He added, “We don’t get as much customer involvement as we want. Our customers are busy and they have other things to do than to talk to programmers all day.” One developer complained that, “We request our customers to talk to us every day and at a minimum once a week, but they are not very involved. We end up with talking with them maybe twice per Sprint.” Another developer stated, “Our customers did not give us specification documents. We basically had an hour-long meeting to make a specification. So it was vague when we started it. It was up to us to make specifics and estimations. I think the biggest roadblock in our development process was in the customer involvement. Though we did not have enough customer involvement, our customers accepted most parta of the system that we created and asked us for minor changes. But I think it would be much better if we get together more often with our customers.” It appears that, most of the time, customers do not know what they really want in their future system and it becomes a roadblock for customers to get involved in the project development process. Said one developer, “Customers think they have a clear idea but they do not. For example, the customer wants to track people’s credit. To them, that’s clear and precise. But to us, we need to know who the people are, what the credits are, when they expire, how long we track them, what rewards are earned for many credits.” Due to unclear customer requirements, developers have a hard time figuring out what exactly the customer wants to include in their system. One Scrum Master mentioned, “We need to get out a lot of information from unclear statements, which takes more time, which causes us to get involved less because it takes too much time. But we don’t have any other way to do it because we don’t have information.”
4.3.3.2 Working Environment.

Most agile methods, including Scrum, recommend removing the cubicles and setting up collocated team space because cubicles promote isolation, and the Scrum process relies heavily on high-bandwidth, face-to-face communication, and networking. Open space is considered better than the cubicles and private offices in the Scrum process. Many developers like the idea of an open-space working environment. One developer mentioned, “I feel like I am little closer to other developers in open space. It’s really nice to be able to look across the room and talk to somebody else in the team and ask questions quickly. I don’t feel like I am shouting over the cubicle wall to get to them.” Another developer stated that, “Open space is good because everyone is easily accessible. I like it because I think it fosters communication. It’s very easy to say hey, I need some help, information, or come, look at this. Everyone is just kind of open, and it seems to work very well.” Though some developers enjoyed the open-space-working environment, other developers did not like the open space, and they mentioned downsides and some problems. One developer stated that, “the open areas are very nice to communication but it does hurt when you try to concentrate because there are a lot of distractions. For example, when co-workers are having a conversation with somebody or having a phone conversation, it’s very distracting.” Another developer groused that, “I am less productive because a lot of noises are going all around. Without having cubicle walls or private offices, the distractions are pretty high which is hard to work with.” A team lead stated, “You know the best working environment is an office. In your private office, you can do things your way, and focus on things without being distracted by other noises.” To cancel out the noises, most developers use headphones. The director of operations noted that, “Everybody has headphones and they can just put those on and listen to something. That pretty much drowns everything else out.” However, several developers complained that, “We developers, are usually working while listening to music. We all have nice
headphones. Everything is going under that. But if I need to focus on something, that’s really difficult just because I have headphones on.”

4.3.3.3 Common Technologies and Problems between Teams.
One observation is that one team’s members could spend a lot of hours finding the right tools or technologies suitable for their project without knowing that the team next door is already using tools or technologies that could be used for their project. There are many similar technologies can be utilized between teams. One developer mentioned, “A lot of things are similar between teams who are working on different projects, there are even some versions of technologies that we use.” He continues by saying that “One is driven by the other so I think it makes sense that we can solve each other’s problems using the same tools in many cases.” Another developer voiced this thought, “I think that the technologies we use between teams are similar enough that if we do our level best, we are able to cooperate.” It is also noticed that some problems that separate teams have are similar. One developer noticed, “I think when problems come up, even though we may not know the languages used in another team’s project, a lot of problems are the same.” One of the teams sometimes uses a quite different setup in a different environment, as reported by a project manager, “They are doing things quite differently.” He also added, “I think they will eventually move to some of the technologies we are using. Then we will be able to cooperate more. But now it does make sense that we use a different environment.”

It is noticed that having a person who can play the role of a liaison between teams is important because of the similar technologies used in multiple teams and the similar problems arising with the teams. The person should be able to inform a team whether there are similar technologies that other teams already take advantage of and whether there are similar problems that other teams faced and resolved.
4.3.3.4 Government Project and Scrum Method.

Government projects usually require heavy documentation, big planning, and big design up front. Due to the bureaucratic nature of the government, a government project seems not to conform to the principles of the Scrum method. Developers consider that working on a government project is challenging because government itself is not agile. One developer expressed: You know here we are really struggling with a government project because they think they are doing agile. But government tends to be very bureaucratic because it tends to be a lot of red-tape and tends to have many layers of accountability inside their organizational structure. It takes a long time for decisions to get made. They are very not agile just by default. And so, trying to get them to function that way can be a challenge.

Another issue in dealing with a government project is that there are many jargons and acronyms used in the descriptions of the government project. Developers have to learn all these unfamiliar terms. They think the government project is complicated. One developer stated, “Of course, it’s affiliated with government and anything with affiliated with government has a tendency to be complex and then there’s a lot of acronyms and there’s a lot of vocabulary that I know nothing about.” Conflicts between the bureaucratic nature of a government and the principles of Scrum method seem to create a big hurdle in fulfilling the government project.

4.3.3.5 Social Loafing.

As explained in Commercial Application Development Firm’s social loafing section, social loafing is an observed fact that people are less productive and less motivated when they work in a group than when they work alone. Also, social loafing is the tendency to take advantage of other’s effort when working in groups. It is noticed that most developers feel there has not been much social loafing in their projects. One developer stated, “I don’t really see that happening now so much. There are three of us on our team. I think everybody does their fair share of work.” Another
developer held that social loafing is, “...really not in our team. Everyone is fairly motivated and quick to point out if anyone is not doing his or her share.” However, one developer thinks there has been social loafing in his project. He held that, “In our particular project, that’s been true to a certain extent.” Another developer stated that, “I think that’s true sometimes, but I have never been involved in that kind of situation in my project. I haven’t got a credit that I think I don’t deserve.” He also thinks the firm has been very fair in recognizing developers’ hard work. He felt that, “They are very good about it. Actually, they are very good at giving credit toward developers who worked hard on a project.”

Overall, most developers think social loafing has not been a problem, but some developers expressed an interesting issue. One of them stated: I don’t think there is so much individual recognition, whereas there is team recognition. So for example, in an e-Governance Application Development Firm team, when we get the project done, then we recognize the team, rather than ‘Oh, John did the most, Amy did the second.’ Most stuff like that. As a team, we are all respected as doing an equal amount of work. If you are not part of it, you will get fire.

4.3.4 Information Systems and Technology Factor
The information systems and technology factor is comprised of these concepts:

- communication
- bug tracking system
- version control systems

These concepts are discussed in the following section.

4.3.4.1 Communication.
The Scrum process recognizes the important role of communications in the software development process and provides an excellent means of communication. All interviewees agree that the Daily Scrum Meetings improve communications
between team members. However, each team in the firm is fairly separated and generally there is not much communication between teams. The lack of communication between teams could cause problems, such as duplicated work. This problem can be solved, or at least mitigated, if the firm holds daily Scrum of Scrums meetings so that the Scrum masters from each Scrum team can make sure no work is being duplicated. Good within-team and between-team communication can be accomplished through the framework of Scrum, but communication with the customer can be problematic. Several developers observed that, “the biggest area of communication issues that we have is with the customer more than anything else, because they tend to not give us a lot of feedback.” Part of the reason that the customer does not provide feedback is that, in most cases, they have other daily jobs to take care of in addition to the work with developers. This is related to the customer involvement issue, which is explained in the customer involvement section.

**4.3.4.2 Bug Tracking System.**

All developers agree on that any web-based bug tracking system can assist them in the task of prioritizing bugs and keeping track of bugs. The company has been using a bug tracking system called “Mantis,” which is a free web based bug tracking system. Most developers think Mantis has been working well for the bug tracking and auditing. One developer mentioned that, “It does a good job as far as bug tracking and auditing.” Another developer states that, “I have been pleased with Mantis. We use it very effectively, and it’s very customizable.” Mantis also has a search function so that developers can do a key word search. One developer observed that, “Mantis has a good search engine. You can use any subsidiary field, which is a track list, and it can categorize very well.” However, some developers think they could use Mantis more effectively by integrating features, tasks, and burn-down charts into Mantis. One developer stated, “We only put bugs into Mantis instead of features and tasks. I want to incorporate burn-down charts into Mantis.”
A couple of developers also think there is a bug in the software. One developer noted, “There are couples of weird things it does occasionally, but it’s nothing catastrophic.” The other developer uttered, “Mantis is supposed to send an email whenever a new bug is entered in and sometime it sends false notification. It seems to be a bug in the software.” One interesting thing found in the interviews is that developers want their clients to report bugs into Mantis. A project manager affirmed that, “It would be nice if our clients could get into Mantis and report bugs there.” In fact, a project manager stated that, “Basically, two teams had their clients enter bugs into Mantis. That can be helpful for them.” However, the client in one of the projects does not like using Mantis as a bug report system, so they use an email system instead. This seems to be a problem for developers, because one noted, “they [the client] email the bugs to somebody and often we don’t follow up on them.” Another issue is that clients’ bug reports are often vague and difficult to understand. One developer mentioned, “Usually, their reports are a little vague and hard to understand. They say, ‘I did this and enter the information and then save and then the page went blank.’ Sometimes, you don’t know exactly what page they were in the website and sometimes you don’t know which information didn’t get saved.” To solve the issue, developers ask clients for more feedback on their reports. One developer stated that, “We can ask our client to put more feedback directly in Mantis and hope our client sees it or sends an email, or we can give them a call if we have any questions on the feedback. It seems that Mantis works well as a bug tracking system for the firm, though it has a little glitch. However, it seems that more effort is needed to integrate features, tasks, and burn down charts into Mantis.

4.3.4.3 Version Control System.

The company has been using an open-source revision control system called “Subversion”. All developers commented that Subversion would be a great tool in any environment where a version control system is needed. One developer stated, “We use Subversion and it works very well for us.” Another developer added, “I’ve
been really impressed with Subversion.” Another developer claims that Subversion, “. . . works very well for small teams and for large teams as well.” A project manager who used another revision control system thinks Subversion provides various functions and is superior in many ways. He claimed that, “It has great utility, great tagging and branching. I can only think of pros for it. It has been great for us and you know especially compared to SourceSafe. It’s superior in every way.” Another developer claimed that Subversion makes revision control easy, especially when creating and utilizing a branch. He stated, “I came from using the Concurrent Versions System. Subversion just makes a lot of things easier. We are using Subversion and it has an advantage when you want to create a branch with some project. That’s the biggest advantage of using it.”

In addition to the advantages mentioned above, Subversion periodically checks if there are any changes made. If there are any changes made in the repository, it rebuilds the system. One developer stated, “We use Subversion. It just polls every five minutes I think and whenever there’s change, it checks out the code and rebuilds it, and make sure everything is still working.”

Some developers think they can use it better by creating more branches. One developer commented, “We need to use it better and do more branching.” Another developer talked about a recent mistake, “Some code got checked in recently into the main branch and got deployed. It caused a problem because it was in a half-ready state. It should have been checked into a branch.” Other than the problem caused by human error, it seems that Subversion is a well-chosen revision control system for the firm, and the firm just needs to expand its use.
4.4 Conclusion

The results chapter presented above describes the process of data analysis and empirical findings through grounded theory study of commercial application development firm & e-Governance application development firm that implemented Scrum in their software development processes. Finally, concepts suggested by the open coding of raw data and recurring themes identified from axial coding were investigated.