CHAPTER V
Discussion and Management Guidelines

This chapter discusses the discoveries from an in-depth agile software development study of Commercial application development firm and e-Governance application development firm utilizing the Scrum software methodology. The discussion covers the following topics: critical issues and challenges found by adopting the Scrum method and management guidelines for avoiding and overcoming obstacles when adopting Scrum method in developing e-Governance applications.

5.1 Issues and Challenges of Scrum
This section discusses issues and challenges identified in two firms by comparing concepts which emerged from the data. The issues and challenges discussed suggest lessons that Scrum practitioners can learn and provide a basis for management guidelines.

5.1.1 Human Resource Management
Human resource management comprises in-house people related issues. Table 5.1 presents issues identified from the two firms. These issues reveal the challenges of the Scrum method.

<table>
<thead>
<tr>
<th>Category</th>
<th>Commercial Application development firm Issues</th>
<th>e-Governance Application Development Firm Issues</th>
</tr>
</thead>
<tbody>
<tr>
<td>Human Resource Management</td>
<td>• Team Management</td>
<td>• Multiple responsibilities</td>
</tr>
<tr>
<td></td>
<td>• Collaboration</td>
<td>• Collaboration</td>
</tr>
<tr>
<td></td>
<td>• Training</td>
<td>• Training</td>
</tr>
<tr>
<td></td>
<td>• Lack of Accountability</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Trust and Confidence</td>
<td></td>
</tr>
</tbody>
</table>

Table 5.1
Human Resource Management Issues
5.1.1.1 Team Management.

When an organization decides to employ the Scrum method, the organization needs to reorganize their existing development teams into Scrum teams.

The Commercial Application Development Firm results indicate that when Scrum teams are created, they must be composed based on the knowledge and skills necessary for the projects in order to reduce the time to learn business logic, development tools, and programming languages. In addition, each Scrum team needs a team leader who can see the big picture and guide the team in the right direction, even though self-managing team are one of the unique aspects of Scrum.

The Scrum master might be a good candidate for this job and selected from a group of technical people, rather than a group of non-technical people like at Commercial Application Development Firm. Some developers like the idea that the team decides how to do things based on the consensus of the team and that the team has more control over how to complete the development project. When organizing Scrum teams, the organization needs to balance management between a pure self-managing team and a heavy leader-guided team. Another lesson from Commercial Application Development Firm is that small sized teams are more flexible and adaptable in defining and applying variants of Scrum. Interestingly, team management was not an identified issue at e-Governance Application Development Firm.

5.1.1.2 Collaboration.

The research identified several collaboration problems at Commercial Application Development Firm when the firm kept two geographically separated development teams. These problems could also apply to co-located teams. The first collaboration problem occurred between developers at one site and QA personnel at the other site; developers were checking code into CVS without telling QA personnel the potential
areas of code that might be affected by the changes made. To mitigate this problem, developers should take some responsibility in testing code, or collaborate with QA personnel by showing the potential areas of code that might be broken by their changes. The cause of the second collaboration problem was a divided team with members in two different development locations. When the team hired new members at one site, it was difficult for team members at the other site to work with them. This incident indicates that when new members are hired, assign them to a co-located Scrum team instead of a remote-site Scrum team. The third collaboration problem arose between Scrum teams due to the lack of a designated person responsible for checking the consistency of products across Scrum teams. The solution to this problem consists of appointing a Scrum master in each team and having them discuss any common issues across Scrum teams. The fourth collaboration problem concerned dividing and assigning tasks between two sites in the Sprint planning meeting, as well as how to track bugs reported by QA personnel and customers. The mitigation of this problem is to use commercial tools designed to address these issues, such as VersionOne (which offers good project management mechanisms) or JIRA (which provides excellent bug tracking mechanisms).

At e-Governance Application Development Firm, no major collaboration problems were noticed. On one occasion at e-Governance Application Development Firm, a situation developed between developers and a particular project manager. Due to insufficient human resources, one person had multiple responsibilities as product manager, product owner, and accounting manager. The first problem was that developers did not try to talk with the individual about issues that came up outside of the Scrum meeting because they knew he was too busy and overloaded with too many other tasks.

The second problem was that developers had to wait to get any questions answered.
The third problem was the developers had the notion that the overloaded manager was a bottleneck for the rest of team. A fourth problem was that developers had communication issues with him because he was rarely available to people. This situation might be beneficial to e-Governance Application Development Firm for a short time. In the long run, however, the firm will lose a lot more than it gains with a manager trying to fulfill multiple conflicting responsibilities.

5.1.1.3 Training.
One of the biggest problems Commercial Application Development Firm faced was in new employee training. Due to the complexity embedded in the system, new employees need to spend a lot of time becoming familiar with the system. Commercial Application Development Firm deemed training new employees separately in each Scrum team a big waste of time. A more efficient method is to appoint a mentor to train all new employees in the basic practices of the firm, with each team giving task specific training. The problem with new employee training gets worse when the employee who has expertise in an area is at a different site than the new employee. One observation noted that the use of multimedia may help the firm mitigate this problem, but is not a complete solution. The indication is that co-located Scrum team members should do employee training.

e-Governance Application Development Firm also had a training problem. However, the problem there is with the Scrum development method itself, rather than the existing systems. Some developers did not see the big picture and the benefits of Scrum. One developer had a misconception that he thought he should complete a full project within a single Sprint, instead of multiple Sprints. e-Governance Application Development Firm’s case indicates that it is important to have all new developers go through thorough, step-by-step, formal training and occasional follow-up training after that. Brown bag lunch training or the “lunch-
and-learn”-type training offered by the firm to all employees is an excellent way to make up for the lack of formal training and to get feedback from everyone.

5.1.1.4 Lack of Accountability.

Commercial Application Development Firm revealed that tasks in the Sprint backlog often did not get completed as estimated, and were consistently carried over to the next Sprint. Then, nobody took responsibility for the delayed tasks. The initial intention of a self managed team is to allow developers to run the team and have ownership of the projects; this seems to have been a big hole at Commercial Application Development Firm. If not operating correctly, self-managed teams promote a lack of supervision, which can lead to a lack of accountability and commensurate project delays. It appears that it would be helpful for the firm to confer the necessary authority to either project managers or Scrum masters to supervise developers.

In that way, project managers or Scrum masters may influence developers to work faster and harder, while allowing developers a certain degree of self-management. It is also a downside of distributed Scrum that developers do not take ownership of task completion, due to the lack of relevant control at both sites. Batra, Sin, and Tseng (2006) [120] suggested setting up a coordinator in one site and ambassador in the other site to ensure control. A lack of accountability issue was not found at e-Governance Application Development Firm.

5.1.1.5 Trust and Confidence:

Trust and confidence arose as an issue between a Scrum master and developers when the Scrum master was unable to get developers items or information they needed and was unable to remove developer’s impediments. The same issue arose between developers when developers worked together on the same project and they did not see any progress on a module assigned to another developer or group.
The trust issue got worse in the case of two development sites, causing confidence levels with other developers to degrade. The fifth principle of the agile manifesto (http://agilemanifesto.org/principles.html) says, “Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.” This implies developers should work in an environment that suits them and should have the support that they need, and at the same time they should give other team members trust to attain high confidence levels.

It takes a lot of time and effort to regain trust and confidence among developers. It is critical to prevent trust and confidence problems before they happen. To resolve a trust and confidence problem, the organization should foster collaboration and have project managers keep monitoring the situation. Also, when there are team members at different sites, it would be helpful if each member of the team at one site would take the time to get to know members at the other site. The trust and confidence was not an identified issue at e-Governance Application Development Firm.

5.1.2 Structured Development Process

The structured development process consists of systematic process related issues. Table 29 shows the structured development process issues identified in the research. They are discussed below.

5.1.2.1 Scrum Framework.

Most developers at Commercial Application Development Firm were in favor of the Scrum framework. They thought the Scrum model promoted communication and team work, and helped them keep track of task assignments and monitor task progress. They also thought that working in the Scrum team provided motivation, excitement, and interest. Some developers, however, did not like the frequent daily Scrum meeting. They felt that having the daily Scrum meeting was too much time
for not enough value, and that the various Scrum meetings took too much
developer’s time away from programming, even though the Scrum meetings helped
team members refine the goals for each Sprint and improved the quality of products.
Some developers and QA personnel also thought monthly Sprint planning meetings
took too much time and they want to streamline the planning session. As
Commercial Application Development Firm indicates, any organization should have
streamlined Scrum meetings and monitor whether or not various Scrum planning
sessions take too much time for not enough value.

Table 5.2
Structured Development Process Issues

<table>
<thead>
<tr>
<th>Category</th>
<th>Commercial Application development firm Issues</th>
<th>e-Governance Application Development Firm Issues</th>
</tr>
</thead>
<tbody>
<tr>
<td>Structured Development</td>
<td>• Scrum Framework</td>
<td>• Scrum Framework</td>
</tr>
<tr>
<td>Process</td>
<td>• Unit and Integration Testing</td>
<td>• Unit and Integration Testing</td>
</tr>
<tr>
<td></td>
<td>• Coding Standard</td>
<td>• Coding Standard</td>
</tr>
<tr>
<td></td>
<td>• Documentation</td>
<td>• Documentation</td>
</tr>
<tr>
<td></td>
<td>• Formal Code Review</td>
<td>• Formal Code Review</td>
</tr>
<tr>
<td></td>
<td>• Project Estimation and Planning Poker</td>
<td>• Project Estimation and Planning Poker</td>
</tr>
<tr>
<td></td>
<td>• Use Cases</td>
<td></td>
</tr>
</tbody>
</table>

The Scrum method works quite well for developers at e-Governance Application
Development Firm and is a big improvement over the waterfall method. Developers
thought the Scrum method helped team members get involved in projects, be aware
of everyone’s progress, contain scope creep, and prevent projects from going too far
off course. Like Commercial Application Development Firm, some developers did
not like inefficient Sprint planning and review meetings. They felt that keeping daily
Scrum meetings to 15 minutes was difficult because people gab a little too long,
there is an excessive amount of material that needed discussed, and taking care of 2
or 3 projects at once.
Another problem was setting up the meeting time. It was difficult to get all developers together at one time without interrupting their work because of the flexible work schedule. The research noted that some project managers actually managed the Scrum team rather than let the team self-manage, and that developers did not spend enough time generating a detailed Sprint backlog.

5.1.2.2 Unit and Integration Testing.
When Commercial Application Development Firm first adopted the Scrum method, the firm placed a QA person on each Scrum team. This caused a problem because the QA person is always behind the Sprint schedule. This problem was resolved by creating a QA-only Scrum team that covered all code generated by other Scrum teams. This solution created another problem that when developers made a change and passed the code to the QA team, the QA personnel sometimes did not know the other areas that affected by the changes. To resolve this problem, the firm asked developers to do some portion of the testing themselves that the QA people usually cover. This may not be a good move economically for Commercial Application development firm because developers usually are paid more than QA personnel; this might be a sensitive issue that any organization needs to resolve wisely.

e-Governance Application Development Firm utilized a tool called “N-Unit” for unit testing, and each developer tested his/her own code. Interestingly, the firm also invited their clients to the test site and had them track, test, and enter bugs that they found. However, having developers test their own code had two issues. First, developers usually assume that their code always worked. Second, developers could not as thoroughly test their code as third-party testers. Another issue was related to large legacy code not designed for unit or integration testing. Developers had a difficult time testing the legacy code. The legacy code and the code working behind the curtain were a big challenge to the firm. The firm also needed to hire more
people who had a wide range of testing skills in software. Insufficient client budget also made it difficult to test everything covering one hundred-percent of the code.

It seems that the firm utilized the unit test well, but it did not cover all possible combinations of issues due the short client budget and lack of wide range of skilled QA personnel. In addition, the firm needed to rewrite the legacy code or find out an efficient way to make the legacy code unit-testable.

5.1.2.3 Coding Standard.
Commercial Application Development Firm utilized coding standards. They have very specific coding standards in many areas in order to have easily maintainable and expandable code. Most developers agree that having a formal coding standard enables them to understand other developer’s code, but some developers worry about putting too many coding standards on developer’s shoulders. Some developers actually think that heavy coercion to the standard may hamper their performance because they have to look at coding documents back and forth to see if their code conforms to the standard.

e-Governance Application Development Firm did not have a formal coding standard but had a verbal coding standard; developers felt that they were close enough to comment when a person who did not follow the norm. Developers had their own coding style, which was influenced by several commercial software packages, such as Microsoft Visual Studio and Borland. Most developers and project managers thought forcing a coding standard might hamper developer’s creativity and hinder performance because they had to relearn how to code in many places.

5.1.2.4 Documentation.
After Commercial Application Development Firm started the Scrum method, many detail documents, such as class diagrams, sequence diagrams, activity diagrams,
communication diagrams, and use cases were significantly reduced, or disappeared. The lack of detailed design, as indicated at Commercial Application Development Firm, caused many problems in complex projects. One main area affected considerably was testing, because QA personnel depend heavily on documentation to find problems.

Another problem with a lack of detailed documentation was the tendency to write code without taking time to think about what effects the code may have on other parts of the application. These resulted in an increased number of bugs, which then required a lot of developer working hours to fix. This was a major issue and caused major code rewrites. It’s obvious from Commercial Application Development Firm that if any organizations deal with complex and large projects, they need to tailor the Scrum philosophy on reducing the amount of documentation.

e-Governance Application Development Firm also reduced the amount of documentations significantly. Developers tried to place more comments and explanations for any tricky logic in the code, along with explanations for any changes that they made, to compensate for the lack of documentation. However, it turned out that many developers had a hard time completing tasks without any documentation, especially developers who needed to work on parts of the system they had never worked on before and new developers who did not have much experience with e-Governance Application Development Firm’s projects. Further, those developers asked a lot of questions, which took much time away from developers who did understand the project. As e-Governance Application Development Firm indicates, no documents at all are a very dangerous idea that leads to many problems, including causing the agile method to be as slow as anything else.
In the agile methods, the code itself is regarded as all the documentation that developers need. However, it is apparent that zero documents are not always the right way for large-scale and complex projects, especially, in a distributed Scrum environment. The amount of documentation should be decided based on the context of the development environment, though Parnas (2006) [42] suggests a wordy document and Simon (2006) [121] suggests no more than a two page long document.

An additional problem was that only one main developer had extensive knowledge about the firm’s systems, rather than every developer on the Scrum team having shared skills and knowledge of the systems. If that main person leaves the firm for any reason, it would be a big problem because it may take several months to recover the knowledge lost. Keeping all team members equal by sharing skills and knowledge on the systems is not easy, and not feasible in reality.

5.1.2.5 Formal Code Review.

Commercial Application Development Firm has utilized a web-based formal code review, and developers think the formal code review is a vital and critical process in creating high quality software applications. Any issues and challenges were not identified at Commercial Application Development Firm.

Developers at e-Governance Application Development Firm did not have a formal code review, but they had an occasional informal code review. Not having a formal code review invoked some issues. First, developers did not pay extra attention to their code, because they believed no one would look at it again. If they believed that at some point somebody would go back and look at their code, they would have more accountability. Second, developers lost opportunities to improve the quality of their code and enhance their coding skills through feedback from other developers. Third, there was a high chance that developers wasted time and money by trying to re-invent the wheel from scratch because there are many code examples already
written, tested, and proved efficient by the Dot Net framework or other commercial builders. Most developers knew the benefits of having a formal code review, but they just never had the time to do it. e-Governance Application Development Firm needs to set a time aside for a formal code review and select an appropriate tool to facilitate the code review.

5.1.2.6 Project Estimation and Planning Poker.

This issue came up only at e-Governance Application Development Firm, though having an accurate project estimate was an important part of projects for both firms. Developers at e-Governance Application Development Firm had a hard time estimating the duration of a project, and the level of hardness increased when developers needed to deal with legacy code or when they did not have the experience required to finish a project. However, it seems they mitigated this issue by introducing a new project estimation method called “Planning Poker”. A lot of comments from developers revealed that the Planning Poker method provided developers with the opportunity to throw out their honest opinion without being biased or coerced by other developers. Also, Planning Poker helped developers have a chance to discuss estimation gaps between developers, and guided them to reaching better estimates. One noticeable benefit was when developers were able to break big tasks into the smallest measurable segments, they were easily able to make good estimates. The researcher noticed that every developer thought Planning Poker was very useful, effective, and produced reasonable estimates. However, though Planning Poker could help estimation, the bottom line was developers needed to be familiar with the technologies that they were going to use, the business logic, and the system itself. Otherwise, estimation will still be one of the most difficult parts of a project.
5.1.2.7 Use Cases.

This issue was identified only at e-Governance Application Development Firm. Developers at e-Governance Application Development Firm knew that they could understand the system better with use cases and that they had the best success when they had use cases. Though the firm has reduced the amount of specification documents a lot since the firm adopted Scrum, one Scrum team created a fair amount of use case documentation based on a list of items that team members needed to build.

Three issues were identified related to creating use cases. First, some developers were not well prepared to write use cases because they were unfamiliar with a system. Second, clients did not have a clear and precise idea what they really wanted to have in their system. Third, clients did not know what use cases are or how to use them.

The first issue is an in-house issue and the other two are client-related issues. The in-house issue can be resolved through a well-organized employee training program. The client-related issues were resolved by having developers come up with some specifications and having clients review it. It would be better if clients know what they really want to have in their system and understand use cases. If clients have a notion of use cases and have a clear idea of their system, developers can communicate with clients better and create better use cases, which can lead to successful projects.

5.1.3 Environment

Table 5.3 shows issues belong to the environmental factor. The issues and challenges related to the environmental factor are discussed below.
### 5.1.3.1 Customer Involvement.

Due to the large number of customers scattered across the United States, Commercial Application Development Firm needed to come up with a different solution to incorporate customer feedback. One way that the firm employed was to send out product line managers to the customer and have them collect project requirements. The project line managers also utilized WebEx to show features of the products and some charts and graphs to reduce the number of onsite visits. Another way the firm employed to get customer feedback was to host a user conference once a year. At the conference, the firm demonstrated new policies and directions of product development. The customers then voted for or against the policies and new development direction.

One issue associated with the customer involvement at Commercial Application Development Firm was that QA people had a difficult time providing the customers with quick bug fixes. This resulted from the Scrum method principles, which required the QA people to focus more on the code generated during the Sprint process than on responding to customer problems.

Though each project at e-Governance Application Development Firm was only for one customer, developers at e-Governance Application Development Firm had

<table>
<thead>
<tr>
<th>Category</th>
<th>Commercial Application Development Firm Issues</th>
<th>e-Governance Application Development Firm Issues</th>
</tr>
</thead>
<tbody>
<tr>
<td>Environment</td>
<td>• Customer Involvement • Working Environment  • Interdependency among Modules • Social Loafing</td>
<td>• Customer Involvement • Working Environment • Common Tools and Problems between Teams • Government Projects and the Scrum Method • Social Loafing</td>
</tr>
</tbody>
</table>
difficulties getting customers involved in the decision making process. The customers did not willingly participate in the process because they were busy and had other things to do. Poor customer involvement in projects caused problems for the firm because developers needed to create specifics without conversing with clients. Often times it took a lot of hours to figure out what exactly customers really wanted to include in their system, because they did not know what they want in their future system. This was a big roadblock for developers as they went through the development process.

Though e-Governance Application Development Firm did not have enough customer involvement, most final products were accepted by their customers with minor changes requests. e-Governance Application Development Firm’s case indicates that if developers get together more often with their customers, organizations can deliver a software product to customers sooner with better features and functions. Organizations may also reduce maintenance fees by delivering a more correct product that customers want.

5.1.3.2 Working Environment.
There were mixed feelings among developers about the open-space working environment. Some developers liked a cubicle setting because they thought it increased the number of communications between team members and fostered collaboration and the teamwork. However, most developers did not like the open-space working environment because they could not concentrate on their work while their coworkers talked to one another. It was apparent that most developers liked having their own office rather than a cubicle in order to be productive.

Many developers at e-Governance Application Development Firm liked the open-space working environment because it provided easy access to other developers and it fostered communication. Though some developers enjoyed the open-space
working environment, other developers did not like it and thought it brought some downsides and problems. First, developers were easily distracted when their co-worker’s talked to other co-workers or when they had a phone conversation with someone. Second, developers were less productive when they could not concentrate because of a lot of background noise. To cancel out the noise, developers utilized headphones, which they put on to drown everything else out. Though this helped most developers, some could not focus on their works just because they have the headphones on.

5.1.3.3 Interdependency among Modules.
At Commercial Application Development Firm, as the size and complexity of the project grew, the dependencies and interconnections among tasks in the application increased. However, developers were not able to fully consider all the dependencies and interconnections among modules because of their narrow-focused planning and design in each Sprint planning meeting. The developers also had a tendency to do things in a quick and dirty way without thinking whether the code would be flexible enough for future needs. As Commercial Application Development Firm indicates, any organizations should support and encourage developers to spend more time on considering the dependencies and interconnections among modules.

The issue of interdependency among modules was not identified at e-Governance Application Development Firm because e-Governance Application Development Firm’s projects were relatively small and less complex compared to Commercial Application Development Firm’s projects.

5.1.3.4 Common Tools and Problems between Teams.
Though e-Governance Application Development Firm did not show any signs of interdependencies among modules due to the firm’s small size of projects, the firm did have issues with common tools and problems between teams. It appeared that
one Scrum team’s members could spend many hours finding the right tools or technologies suitable for their project without knowing that other Scrum teams already employed similar tools or technologies. This is a big waste of precious developer’s time if two Scrum teams can utilize the same or similar tools or technologies.

The research also noted that each Scrum team had similar problems, which might be resolved using similar solutions. Teams spent time resolving similar problems, each in their own way, which is another waste doing duplicate work if the same solution can be applied to both problems. The firm should appoint a person to inform teams if there are similar technologies that other teams already took advantage of and whether there are similar problems that other teams faced and resolved successfully.

5.1.3.5 Government Project and the Scrum Method.
This issue was identified only at e-Governance Application Development Firm because the firm has been dealing with many government projects. A government project usually requires heavy documentation, big planning, and big design up front. This does not conform to the philosophy of the Scrum method. It is a big challenge to complete government projects with the Scrum method because the government itself is not agile, and the nature of government is bureaucratic. An additional issue is that developers have to learn the jargon and acronyms used in the descriptions of government projects, adding unproductive time to complete a government project.

5.1.3.6 Social Loafing.
The social loafing issue was identified at Commercial Application Development Firm because the firm did not have ways to evaluate individual performance. In addition, social facilitation, group motivational gain, and evaluation apprehension were also found. Social facilitation and group motivational gain contributed to
reduced development time and to lower bug rates. When the firm invited all developers, QA personnel, and people in other departments to the Sprint review meeting, developer evaluation apprehension was observed.

Social loafing was generally not an issue at e-Governance Application Development Firm, though it was present to a certain extent in some projects. Though e-Governance Application Development Firm has been very fair in recognizing developers’ hard work, some developers did not think there was much individual recognition, rather they thought there was more team recognition.

If the firm does not provide a way to accurately measure an individual’s performance, and the performance is measured only by the unit or team, a social loafing issue within the team might be raised. It is import to eliminate social loafing by making each individual’s contribution verifiable [122] and to offset its possible negative impact on development time and cost. It might be a good idea to invite all developers, QA personnel, and staff in other departments to the Sprint review meeting, where developers present what they implemented. In this way, developers might gain an evaluation appreciation of how others view their work.

5.1.4 Information Systems and Technology

Table 5.4 shows issues associated with information systems and technology. Those issues and challenges are discussed below.

Table 5.4
Information Systems and Technology Issues

<table>
<thead>
<tr>
<th>Category</th>
<th>Commercial Application development firm Issues</th>
<th>e-Governance Application Development Firm Issues</th>
</tr>
</thead>
</table>
| Information Systems and Technology | • Communication System  
• Information and Knowledge Sharing System  
• Bug Tracking System and Management Tool | • Communication  
• Bug Tracking System  
• Version Control Systems |
5.1.4.1 Communication System.

Ineffective communication is the root of most failures in software products [26][121]. Commercial Application Development Firm utilized a lot of communication devices to set up good communication channels when they kept two geographically different development sites. Some of the multimedia tools utilized by the firm included video conference systems, phone conference systems, a web-demo system, an instant message system, and an email system. A virtual private network system was also utilized at the beginning, but it was discarded because it was not fast enough. Of the different multimedia tools used, the video conference system was most effective because people could see facial expressions, gestures, and body language as issues were discussed. However, none of the multimedia systems worked as optimally as face-to-face conversation. Beck et al. (2001)[123] stated that the most efficient and effective method of conveying information to and within a development team is face-to-face conversation. A contributing problem was unstable communication and data lines. When the phone systems were down, the data line went down with it because they shared the same physical line.

Overall, the Scrum method provided an excellent communication mechanism at e-Governance Application Development Firm. Developers believed that the Scrum method improved communications considerably between team members. Each team in the firm was fairly separated and generally there was not much communication between teams. The lack of communication between teams caused problems as explained in the issues of common tools and problems section. This problem might be easily resolved by holding a daily Scrum of Scrums meeting and having Scrum masters of each Scrum team communicate with each other.

Another communication issue was with customers. More than anything else, this was the biggest communication issue the firm had. As described in the issues of customer involvement section, customers tended not to communicate with
developers and not to give a lot of feedback because they usually had other pressing work to do.

5.1.4.2 Information and Knowledge Sharing System.
At Commercial Application Development Firm, newly hired developers created a lot of bugs because they were not knowledgeable about the software that they were working on. Some parts of the application were very sensitive to changes, so a slight modification on these parts had a lot of impact in other areas. It is important to have well-structured information and knowledge-sharing systems between experienced software developers and brand new software developers. In particular, it was very critical to have a knowledge sharing system if new software developers in one location need some of the expertise of software developers in another location. To resolve this problem, the firm utilized a web-based Wiki program which enables developers to add and edit items that might be critical to other developers. Another issue was assigning newly hired developers to Scrum teams. Though the firm maintained two geographically different development sites, each Scrum team’s members should be co-located in only one site. In that way, new developers can have face-to-face communication with other team members.

5.1.4.3 Bug Tracking System and Management Tool.
Commercial Application Development Firm developed a bug tracking system called MOM, which ran on a UNIX platform. Later, the firm adopted a commercial bug tracking system called JIRA because of its additional functionalities, such as filter functions and severity priority code functions. The firm also utilized a commercial web-based management tool called VersionOne to provide useful functionalities, such as simplifying project planning and management, enhancing business and project adaptability, improving project visibility, and increasing project predictability and confidence. The problem associated with these tools was that the
firm could not use one product for both bug tracking and management of a Scrum cycle. Developers had a difficult time using all three products at the same time.

e-Governance Application Development Firm utilized a free, web-based bug tracking system called “Mantis,” which worked very well for bug tracking and auditing purposes. However, developers thought they could use Mantis more effectively by integrating features, tasks, and burn-down charts into Mantis. Mantis sometimes also sent out false email notifications when a new bug was entered in the system. Customers associated with two Scrum teams entered bugs into Mantis, which was very helpful for developers because every bug could be managed by one tool. Customers associated with other Scrum teams reported their bugs through an email system. This sometimes caused a problem because developers needed to deal with two systems to track bugs, and they did not follow up on customer’s emails. Another problem was vague customer bug reports, which developers had a difficult time understanding. This took a lot of developer time because they had to take extra steps deciphering the customer report. Developers usually asked customers to put in a more detailed description, sent an email, or called directly to get more feedback on the report.

5.1.4.4 Version Control Systems.
e-Governance Application Development Firm utilized a version control system called “Subversion,” a well-known version control system in the open source community. It has been a great utility for the firm because of its superior tagging and branching capability and a functionality that checks periodically to see if there are any changes made in code and then rebuilds whole system if any changes are detected in the repository. In addition to the automatic rebuild capability, it also makes sure every piece of code is still working properly. Some developers think they can use it better by creating more branches and expanding its use. Other than a
human error problem that checked some code into the wrong branch, the firm picked a good revision control system.

5.2 Management Guidelines
This section provides management guidelines to help organizations that are already utilizing Scrum or planning to implement Scrum in the future. The guidelines explained here also help organizations avoid stumbling blocks in their Scrum implementation.

The issues and challenges identified and discussed in the previous section provide the basis of the guidelines. The first set of guidelines is for co-located Scrum teams, and the second ones for geographically scattered Scrum teams.

5.2.1 Guidelines for Co-located Scrum Teams
1. When a new Scrum team is organized, managers must consider whether each team member’s knowledge and skills are pertinent to the project the team members are going to work on.

2. Each Scrum team needs a team leader who can show team members the big picture and guide them in the right direction. The Scrum master might be a good candidate for the team leader. The team leader should not manage the team, but should instead coach the team. The Scrum master should be a technical person able to easily remove technology-related developer impediments.

3. When developers check code into a code repository, they should inform QA personnel, in addition to the direct code that the developers modified or added, about any other code sections that need to be tested as a result of the modifications.
4. The project manager should not be a bottleneck to Scrum teams due to his/her multiple responsibilities in other areas that are not directly related to the team project.

5. Formal step-by-step new employee training should be a requirement, and managers should monitor if the training is efficient.

6. New employees should be given enough time to understand both the existing systems and the Scrum method before they get into a project.

7. Brown bag lunch training or “Lunch-and-learn” type training should be implemented often to refresh developers on Scrum and to exchange information.

8. A self-managing Scrum team still needs a supervisor who has authority to get developers to work faster and harder.

9. Project managers should monitor if there are any trust and confidence issues among developers, and between Scrum masters and developers.

10. The duration and rules of the daily Scrum meeting should be strictly observed; the duration of other Scrum meetings should be dynamically adjusted based on the agenda for efficiency.

11. Project managers should foster collaboration between developers and QA personnel. Developers should be able to do a unit test of other developer’s code and work closely with QA people on integration testing.
12. Organizations should educate developers that every piece of code should be testable and designed for ease of testing.

13. Formal coding standards increase readability and understanding of other developer’s code; too many coding standards hamper developer performance.

14. Lack of documentation is a source of problems, especially for large-scale and complex projects. The philosophy of the Scrum method which reduces documentation significantly should be tailored. Organizations need to determine how much documentation is adequate for their projects.

15. Along with documentation, organizations need to promote each Scrum team member having an equal amount of skill and knowledge relative to the project they are working on.

16. Formal code review is a vital and critical process for quality applications. Organizations should establish an efficient way to do formal code reviews.

17. Planning poker is a very easy, useful, and efficient way to evaluate projects. Developers should break a big project into the smallest possible tasks to get better estimates on those tasks.

18. Use cases are important specifications that elevate a developer’s understanding of the project that they are working on; both developers and clients need to be educated in how to write good use cases.

19. Customer involvement is very critical for the success of a project. Organizations should invite customers to participate in the decision making
process and find out a good way to include them in the various Scrum meetings.

20. If any organizations have a large number of scattered customers, they should consider the use of an annual or semi-annual user conference to explain their new products, collect user feedback, and have them vote for or against the organization’s new product direction.

21. Open-space working environments promote teamwork and communication, but organizations should come up with methods to help developers deal with environmental distractions.

22. For large-scale and complex projects, organizations should encourage and support developers spending sufficient time thinking about dependencies and interconnections between modules.

23. If any projects require heavy documentation, big planning, and/or big design up front, the Scrum method might not work well unless combined with another method, such as a Unified Process.

24. Organizations should provide a fair way to measure individual performance to prevent social loafing being an issue.

25. A product fair, which invites people in other departments to a Sprint review meeting and have developers present their works, is an excellent way to boost developer performance.

26. Bug tracking systems and project management tools should be combined into one piece of software to improve efficiency.
27. If possible, invite customers to a test site and have them test code and enter their bugs into the organization’s bug tracking system, rather than letting them email bugs to developers.

28. Good version control systems should be established and utilized to maintain various branches of each product.

29. A person should be selected in each team to work as a liaison between Scrum teams to discuss any common issues and common tools that can be shared among Scrum teams. The Scrum master of each team might be a good candidate, and daily Scrum of Scrums meetings need to be utilized for this purpose.

5.2.2 Guidelines for Geographically Distributed Scrum Teams

1. When developers check code into a code repository, they should inform QA personnel at the other site, in addition to the direct code that the developers modified or added, about any other code sections that need to be tested as a result of the modifications.

2. Scrum team members should be co-located at one site, rather than having a team comprised of team members from two different sites.

3. There should be a person to work as the liaison between sites. This person should facilitate collaboration when dividing and assigning tasks between sites. The project manager at each site might be a good candidate.

4. There should be good bug tracking tools and project management tools, accessible and shared by multiple sites.
5. New employee training should be performed through members at the same
development site.

6. Project managers at each site should pay extra attention to trust and
confidence issues between developers at different sites.

7. Trust and confidence issues are reduced if each Scrum team member at one
site takes time to get to know members at the other site.

8. A video conference system between sites works best for communication
between sites, but does not work as optimally as face-to-face conversation.

9. Tools like a web-based wiki program should be utilized to share information
and knowledge between developers at different sites.

5.3 Conclusion

This chapter presented above discussed the discoveries from an in-depth agile
software development study of two firms utilizing the Scrum software
methodology. The discussion also covered critical issues and challenges found by
adopting the Scrum method in both of the study firms and management guidelines
for avoiding and overcoming obstacles when adopting the Scrum method.