Evolution of an IS Capstone Class

Thom Luce
luce@ohio.edu
Analytics and Information Systems Department
Ohio University
Athens, Ohio

Abstract

This paper reviews the evolution of a senior level, live-client project development capstone class in the Analytics and Information Systems department of an AACSB accredited College of Business. The paper traces changes in methodologies and technologies leading to the current Scrum based approach, using ASP.NET Model-View-Controller, MVC, as the development platform. The paper discusses how Scrum is used in a class that only meets three times a week for 55 minutes each time and how the ASP.NET MVC approach to development fits nicely with the Scrum approach to project management.

Keywords: Pedagogy, Capstone Course, Systems Development, Scrum, ASP.Net MVC

1. INTRODUCTION AND EARLY COURSE EVOLUTION

This paper describes the evolution of the capstone project development class in an AACSB accredited College of Business. The story begins more than 30 years ago when the university was on the quarter system and the systems analysis and design portion of the MIS curriculum (then called Computer Systems in Business) was divided into two courses. The first course covered traditional systems analysis topics with an introduction to design issues while the second class continued the discussion of design along with implementation and a little testing. Since there were no easy to use software tools available at the time the project portion of the class was largely based on paper prototypes.

A desire to have hands-on projects that could easily be shared with clients led us to attempt web page development with HTML, but the required overhead made it very difficult to accomplish anything meaningful in less than a quarter. At the time there were no user controls available to speed development and, since HTML was designed to be stateless, there was no easy way to persist data from one session to the next.

In the mid-1990s Microsoft introduced ASP and the first set of ASP user controls. The controls made it much easier for students to develop functional user interfaces but there was still no solution for data in a stateless environment and no easy way to do authentication and authorization. A great deal of class time was spent teaching students how to manage state i.e. read all the data from the request page and write it back to the response page. Additionally, students had to learn some of the early database management tools like open database connectivity (ODBC) or Microsoft's OLE-DB to persist data from one session to another and to create and store user authentication data. Students also had to learn to use cookies and sessions to manage authentication and authorization issues.

In approximately the same time frame we were teaching Cobol (it was the late 1990s), Visual Basic, Java and Mantis, a fourth-generation language. Students complained that they learned tools (Cobol, Java, Mantis) and then never used them for anything else in the curriculum. We also realized that we were spending a lot of time teaching tools and interfaces and not as much time as we wanted on solving business problems and business systems development techniques.

By the end of the 20th century we phased out Cobol and Mantis and replaced Visual Basic and Java with C#. We also decided to use Microsoft's Visual Studio as a common interface in all technical courses so we could focus less on learning new developer interfaces and more on why we were using the tools – to solve business problems. The process was made even easier by the inclusion of enhanced user controls included in releases of ASP.NET, tools that included authentication and authorization that all but eliminated the student's need to spend their limited time trying to implement these functions. The next major change happened in 2008 when the State Board of Regents (2008) issued a report proposing a common academic calendar for all state schools. That report started the process of moving all courses in the department, college and university to semester long courses. In our College of Business it meant that all majors had to restructure and reduce the number of courses offered because 1) much of the curriculum was common to all majors and 2) there was a strong desire to facilitate double and triple majors in the College. For the MIS program the curriculum was reduced to six major courses (systems analysis and design, database, systems development/programming, enterprise systems, a capstone live project development course and a capstone concepts course). The change meant that there was more time to talk about systems analysis and design in the first course but less time to cover programming issues since two quarter length courses were reduced to one semester course. The end result was the need to cover more technical issues in the capstone project development course.

In addition to calendar year-based changes, the department also had an active advisory board that wanted to see more agile development in the curriculum along with more modern development techniques.

2. EVOLUTION OF THE CURRENT COURSE - SCRUM

The author was asked to look at ways to include agile methodology into the capstone development course (Luce, 2016). After much study it was decided to attempt to move the project management portion of the class from a traditional SDLC approach to Scrum because of the wide acceptance of Scrum in industry (Collabnet VersionOne, 2019) and the perceived ease with which course project management could be converted to Scrum

Scrum is an important approach to implementing the goals of the Agile Manifesto and the principles of Agile development derived from the manifesto (Beck, 2001). The Scrum process, scrum team members, scrum events and scrum artifacts are described in *The Scrum Guide* (Schwaber & Sutherland, 2018), Essential Scrum (Rubin, 2013) and numerous other publications.

Scrum starts with a partial list of requirements known as the project backlog items. The work is broken down into a series of timeboxes called Sprints.   Each Sprint starts with a planning session where the development team refines the project backlog, prioritizes backlog items, estimates the relative difficulty of each item using relative units sometimes called story points and then moves a group of items to another list called the Sprint backlog which the team then works on during the Sprint. The goal of each Sprint is a potentially deliverable product. Each Sprint ends with a Review of the work with any item that doesn't meet an agreed upon definition of "done" being returned to the project backlog.   Each Sprint also ends with a Retrospective where the process is reviewed and modified as needed.

Initial attempts to use Scrum in the classroom were problematic for a number of reasons including 1) students didn't know anything about Scrum and had to learn it before they could use it and 2) we had too many projects, all with clients who were unfamiliar with the approach and weren't able to provide a project owner (one of the required roles in Scrum development) or a workable initial set of project backlog items for consideration by the student teams. Also, while we broke the projects into a series of two or three week sprints, we didn't have a good way to do end-of-sprint reviews or retrospectives and, because the teams only formally met three times a week for a short time, there was no apparent way to do meaningful Daily Scrum stand-up meetings (another required part of the Scrum process).

After two frustrating years attempting to make Scrum work in the course, one of our advisory board members, a former student, said that his company, a midsized consulting firm, would like to work with the department in any way that would be helpful. After some fruitful discussions we decided that they would provide a project for us to use – something they wanted for internal use but couldn't spare the resources to develop. We agreed to have all student teams working on the same project in a friendly competition to see which team product could produce the best solution. The firm agreed to come down to our

campus and launch the project and provide an initial set of project backlog items for the project. They also agreed to try to answer questions from the student teams, to review the work when possible and to come back to campus for final presentations.

This approach worked reasonably well for a year but was still missing important pieces of the Scrum process such as having a product owner, doing a daily Scrum and having a meaningful retrospective at the end of each Sprint. However, our client found benefit in the engagement with us and agreed to make more resources available the next year. They recruited more associates, many were either our former students or graduates of other programs in our university, and had them serve as product owners (POs) for one or two teams each. Each PO was asked to communicate with their teams on a regular basis by whatever remote channel worked best for all of them, to review the team's work at the end of each Sprint and to let the instructor know how the teams were working, both technically and professionally.

3. SCRUM IN THE CLASSROOM TODAY

Students still come to the class knowing little or nothing about Scrum. We spend three or four weeks using readings, videos, quizzes and a few exercises to teach the basic concepts. At the conclusion of the training our client comes to class to present the project charge and provide an initial set of product backlog items. When we first started this collaborative effort the initial set of backlog items provided by the client were fairly specific but in later cycles of the course they became more generalized in an attempt to promote creativity in the solutions.

Each team is assigned a product owner (PO) and a Trello task board containing the initial backlog items. The initial team assignment includes making contact with the product owner and establishing a communication protocol. Teams then work with their PO to understand the initial product backlog, groom the backlog items, attempt to assign priorities and difficulty levels to the items and finally to select items for the initial project Sprint. The deliverable for this task is the updated Trello board.

The next several classes are project work days. In the spirit of true Scrum teams, the student teams self-select roles and start working on the Sprint backlog items. Daily Scrums involve the team talking over the traditional points of this meeting, i.e. what did we do yesterday (since the

last class), what are we going to do today and what is inhibiting our progress. Since there are as many as nine teams in the classroom and it is impossible for the Scrum Master (instructor) to attend very many of the Daily Scrum meetings, the teams submit a summarized version of their discussion on the three main questions along with a list of who was in attendance at the meeting. A small number of points that cannot be made up any other way are awarded each day for attending the meeting.

At the end of the Sprint the team publishes the current state of their project, a potentially shippable version of the project, and makes it available to the instructor. Teams meet virtually with their PO to review the work done and post a written summary of the review session, a screen shot of their Trello board at the end of the Sprint and a retrospective discussion for the instructor.

Sprints are primarily graded on following the process and making progress on the Sprint backlog items. Incomplete Sprint backlog items are returned to the project backlog for consideration in the next Sprint cycle. Finally, and very un-Scrum like, team members do a peer evaluation of their team members for the time covered by the Sprint.

The class following the Sprint review is a planning session to start the next Sprint. As before, team members meet with their POs to groom the backlog, re-prioritize backlog items and select the work for the next Sprint. The process continues for three or four Sprints until the end of the semester when the client and typically one or two POs come to campus for final presentations.

Scrum doesn't use traditional planning tools like Gantt Charts of Critical Paths but instead uses a series of timeboxed Sprints. Each Sprint is followed by another Sprint until a releasable product is completed. Progress is typically measured by Velocity, the average number of story points completed per Sprint and Burns Down Charts (Agile Alliance, 2019) showing the completion of backlog items over time.

Instructors interested in investigating Scrum might want to start with the Agile Manifesto (Beck, 2001), *Essential Scrum* (Rubin, 2013) and the professional organizations dedicated to Scrum: Scrum.org and ScrumAlliance.org.

4. PROJECT DEVELOPMENT TOOLS IN THE CURRENT COURSE

Because of changes made in the prerequisite classes, students come to the capstone development class knowing SQL, C# and ASP.Net web forms and have developed web applications using the Visual Studio development environment. These are the tools that were used when the class was first converted to use Scrum. When we first started working with the consulting firm they offered to provide a project template for the teams to use as a starting point for their project development.

When our instruction staff looked at the template we were completely confused. There weren't any web forms in the template and there were lots of other files and folders not normally found in a web forms project. The consultants had given us a ASP.Net MVC project template which neither the instructors nor the students had ever used. Since the semester had already begun and there was no time to learn a completely new approach to web development, the client agreed to go with web forms for another year. During that year we learned about ASP.Net MVC and were ready to launch the revised course the next school year.

5. ASP.NET MVC IN THE CLASSROOM

The current version (Luce, 2017) of the course uses the latest non-Core version of ASP.NET MVC (Microsoft, 10/3/2018) for the development platform. The Model-View-Controller (Tutorialspoint) design pattern, MVC, implements a separation of concerns with program logic in one module (the controller), data in another module (the model) and user-interface in a third (the view). MVC applications use Routing tables (TutorialsTeacher) to establish public paths to applications. A typical path is based on a controller, a method in a controller and possibly optional items such as an ID. For example, the default page of an MVC application is normally the Index method of the Home controller (technically, the Index method returns the View as an ActionResult (TutorialsTeacher) which is sent back to the user). This could be done by entering the full path: **servername.com\home\index**, or just the controller: **servername\home**, or even just the server: **servername\.**

Controllers are implemented as C# class files while views use the Razor view engine (C# Corner, 2018) that combines HTML and C# code and are formatted using Bootstrap (Bootstrap) styles. The model portion of the MVC application uses Entity Framework (Microsoft) to map classes in the application to tables in the database. We initially tried the "database first" approach (Microsoft, 1/14/2019) (entities are created from an existing database) but quickly settled on "code first" approach (Microsoft, 1/21/2019) (entities are defined in classes and a context class that relates them to each other) because it gave students a better understanding of what their data looked like and made database change management easier. Data validation (Tutorialspoint) is accomplished through the use of data annotations.

6. PEDAGOGY FOR USING ASP.NET MVC IN THE CAPSTONE CLASS

As mentioned previously, students come into the capstone class having taken an introduction to systems analysis and design, a database class and a C# programming class. Both the database class and the programming class use Visual Studio Community Edition (Microsoft) and developed pages using web forms. Students have no prior exposure to the MVC design pattern or developing ASP.NET MVC web sites.

Since our major has a very limited number of classes with only one programming class, the programming class needs to focus on fundamentals and doesn't have time to introduce students to ASP.NET MVC. Because of this we developed and use a series of three individual learning activities in the capstone class. We also created instructional documents outlining the required steps to accompany the learning activities. The result of this approach is that students learn new techniques and procedures just-in-time as they use them, much as they will do on-the-job as working developers and analysts.

The first learning activity involves using a standard template to create a default MVC web site for an application of the student's choice. Default web sites have a home page, an about page and a contact page. The home page uses Bootstrap's three column format and is full of links to Microsoft technologies while the contact page has fictitious Microsoft contact information and the about page has a very generic "about us" statement. The learning activities are designed to introduce the students to much of the basic functionality they will need in the client project that follows. The student's first assignment is to 1) modify the home page to display information about their application and eliminate the links to Microsoft pages. They are free to keep the three-column format or use Bootstrap to modify it to something else. 2) Modify the contact page to be

their contact information, or, if they are concerned about privacy, some other fictional, non-Microsoft information and 3) the about page to be about them or whatever organization they have chosen. They are also required to change the navigation bar, which by default contains a link to "ApplicationName", and the copyright information. The navigation bar and the copyright information are both in a shared layout file (similar to a Master page in the web forms world). Finally, they are required to publish the web site to the web college server and submit a link to the published site to show that the assignment is complete.

The second learning activity provides hands-on experience with Entity Framework. Students are asked to pick a simple many-to-many relationship, other than Order:Product which was demonstrated in class, and create entity classes for at least three entities along with a context class to link them together. They then create controllers and views for each of the entities and link the new controllers to their home page and navigation bar. As a note, ASP.NET MVC provides easy scaffolding to create a controller and views supporting the full set of CRUD functionality – create, list (read), update and delete. Students are then asked to run the application and add at least four or five records for each entity. Next, students are asked to modify one or more of the models and implement data migration to automatically update the database. Finally, they need to recreate any controllers and views that use the revised model and run the application again.

The final learning activity requires students to become familiar with data annotations, creating dropdown lists and basic Bootstrap formatting. By default, ASP.NET MVC creates all data views using the model's field names but this can be changed through the use of data annotations. For example, a field called **lname** would normally be displayed as "lname" but a data annotation could cause it to be displayed as "Customer last name" instead.

Entity Framework assumes that the primary key of an entity is called **ID** or the name of the class followed by ID, e.g. **customerID**, but this can be changed with the **[Key]** annotation. Some foreign keys are implied by the data but specific foreign keys can also be specified with a data annotation. Data annotations are used to specify required fields, minimum and maximum field lengths, data types and required formats.

When ASP.NET MVC generates a view on the many side of a one-to-many relationship it typically creates a dropdown list to allow the user to select the record on the one side of the relationship. If the primary key of that entity is an integer or GUID, ASP.NET automatically creates a dropdown using data from the next field in the entity on the assumption that no one would reasonably know what the key is. For example, if the record on the one side of the relationship had an integer ID followed by first and last names, the dropdown would list the first name as the display text and have the ID as the value. Since displaying only a single name isn't a useful practice, students are taught how to modify the dropdown list to display the whole name and are required to do that as part of this assignment.

When ASP.NET MVC creates views it creates simple page headers like Index, Edit and Create. While these indicate what the view shows they don't say what entity is involved. Editing these headings to include the entity is another part of this assignment.

Finally, students are asked to add Bootstrap styling to their applications. They are asked to include some of the table formatting options (stripping, hover, boarder, etc.), at least some button styling, several examples of text formatting using wells, alerts, text and background coloring, etc. They are also asked to include at least one dropdown menu on the home page or navigation bar and at least a few of the icons available in Bootstrap 3.

No textbook or trade book is used because none has been found that covers the topics needed for this class. As an alternative to a book, a series of documents have been produced to go along with the three learning assignments and numerous other topics that typically come up during the execution of the client project. Topics included in these documents include:

- Creating your first MVC Application
- Introduction to Views
- Creating data models
- Understanding MCVC Controllers and Scaffolding
- Views with Entity Framework
- Modifying your data model to allow access to names and not just keys
- Linking Microsoft identity Users to application specific user data and restricting access Views
- (and others)

In addition to the documents and learning activities, students are given a set of links to web sites helpful to developers (stackoverflow.com, scrum sites previously listed, Microsoft tutorial and reference sites, w3schools.org, other tutorial sites, etc.) and are encouraged to better use Google to search for information.

7. THE FINAL COURSE STRUCTURE

In its current configuration the course is divided into two distinct parts. The first four to five weeks involves training in Scrum and MVC. Scrum training involves readings, on-line videos, short classroom exercises and daily quizzes. MVC training involves in-class demonstrations, readings and three learning activities.

The remainder of the semester is devoted to the team-based, live-client project. The client comes to class to launch the project and provides an initial set of project backlog items. The project launch is followed by a series of four or five two-week Sprints. Each Sprint starts with a planning session and is followed by four development sessions. The Sprint ends with a review and retrospective session held in conjunction with each team's product owner. At the end of the semester the client returns to campus for final project presentations by each team.

8. COURSE DESIGN STRENGTHS AND WEAKNESSES

At the end of the semester students are given the opportunity to provide feedback on various parts of the course, separate from the university's standard course and instructor evaluation. The following quotes list some of the student feedback received during the most recent course offering.

- The daily scrum did an excellent job of keeping our team on track. It's easy to look to other courses as being priority one when projects are involved. The daily scrum helped ensure we were aware of our project and what needed to be done.
- I'm not exactly sure who the ScrumMaster was in relation to this project
- The frequent meetings with our product owner allowed our team to better understand what needed to be done from a client perspective.
- It was difficult to get negative feedback from our project owner. He would always praise us during our meetings but that was not always reflected in our project.

- The short Sprints allowed our team to better target goals and set sub-accomplishments that reinforced the notion that we were making progress.
- It was hard to simulate scrum in the short class time
- we initially only had 2 people programming which didn't work well so we had everyone try to understand the programming aspects
- R&R's (review and retrospective) were helpful for the team to see what they've done and what they could do better next Sprint.
- The four features of scrum development that worked best for our team were, the daily scrums helping us stay on track, the R&Rs that helped us to plan what we needed to get done, having the Sprint planning days to figure what we did and didn't get done, and lastly the ability to get real time feedback from our product owner.
- I did not realize how heavily involved this course was with an actual client with a meaningful project. I really liked the hands-on nature of this course and it made showing up to class every day a little easier.
- I think prioritizing tasks was one of the most useful things for us. It helped us manage our time.
- It was difficult to set up times with our PO at times. Balancing 4 college students' schedules and a working professional's schedule is not easy. Because it was difficult to get in contact with our PO, we often found ourselves starting on the next Sprint without feedback from the prior Sprint and just assuming the work was satisfactory.
- Our daily Scrums allowed our team to consistently meet and set expectations

9. LESSONS LEARNED

Based on student comments and observations throughout the semester we have learned a number of important lessons about implementing Scrum in the live project based classroom.

1. It takes a lot of time. Having all teams working on the same project helps but students have a lot of questions and need rapid turnaround on assignments.
2. Scrum is difficult to do when you have short classes. We had to allow teams to do their own daily stand up meetings and just provide a written summary.

3. If you are working with an external client or have external product owners helping with the project, it is critical to have a single contact point to help coordinate product owners and deal with any issues that arise.
4. Dedicated product owners are essential but hard to control when you are working with volunteers from outside the university. In addition to the single contact point mentioned above, we have annual meetings to disuses ways to better coordinate and make the feedback better.
5. Establishing regular communication channels and schedules between student teams and their product owner is vital
6. Scrum is a person-to-person approach and doesn't work well with absentee project team members. We have tried different approaches to encourage participation including assigning a small number of points for daily attendance and participating in the daily standup meetings, and peer evolutions that affect individual grades on team deliverables.
7. Students can learn new technical and managerial skills on the fly but it is difficult when they are graduating at the end of the semester and already have jobs.
8. Five or six people cannot work on project code at the same time, especially not in the MVC environment. Some type of source code management is required but we have yet to be successful making this work.
9. Interesting projects that are relevant to the student team members are important. Our projects have largely involved recognizing individuals for their contributions to the organization – something students find relevant. The projects can also have some social network integration and gamifications aspects (leader boards, awards, etc.) and find those aspects interesting.
10. Grading can be difficult. We grade the planning and the daily scrums along with the review and retrospective. Since it isn't a major problem in Scrum if all backlog items aren't complete, they just go back on the backlog, watching how much is done each Sprint is important. Perhaps more important is watching how many backlog items, and which ones, are moved to the Sprint backlog during each planning session. This also feeds back into the first point on timeliness of

feedback to students since you don't want them wasting time on items that don't logically belong in the current Sprint.

10. SUMMARY AND CONCLUSIONS

As with courses at other schools, the capstone systems development project class described here has undergone continuous change and improvement over the years driven by changes in technology, changes in accepted business practices and input from our advisory board. The current version of the course, using a live client, Scrum and ASP.NET MVC tools gives students hands-on experience with agile methods and with modern development tools.

The use of Scrum and short Sprints forces students to prioritize tasks that need to be completed and then to focus on a limited set of those tasks at any one time. The implementation of Daily Scrums, while not ideal, allows teams to concentrate on what they have accomplished and what still needs to be done. Reviews and retrospectives at the end of each Sprint allow team members to accurately see how well they completed the work laid out in the Sprint backlog and to look for ways to improve their processes while there is still time for improvements to help. Having product owners who work for the client gives the students first-hand experience working with clients and developing professional communication skills.

ASP.NET MVC works well with agile development and Scrum. Because MVC practices separation of concerns it is easy to update the data model as needed and to independently work on different parts of the project at the same time. The Entity Framework code-first approach and the extensive scaffolding available when creating controllers and views allows students to focus on the client's business needs and not get bogged down in technical details. Also, the full-featured templates provided in Visual Studio make the standing Sprint goal of creating a potentially shippable product, however limited, much easier to accomplish than is possible with traditional web-form development.

11. REFERENCES

Agile Alliance (2019), "Burndown Chart", accessed from https://www.agilealliance.org/glossary/burndown-chart/#q=~(infinite~false~filters~(postType~(~'page~'post~'aa_book~'aa_event_sessio

n~'aa_experience_report~'aa_glossary~'aa_r
esearch_paper~'aa_video)~tags~(~'burndow
n*20chart))~searchTerm~'~sort~false~sort
Direction~'asc~page~1)

Bootstrap (n.d.), "Introduction", accessed from
https://getbootstrap.com/docs/4.3/getting-
started/introduction/.

C# Corner (4/26/2018), "Razor View Engine in
ASP.NET MVC", accessed from https://www.c-
sharpcorner.com/article/learn-about-razor-
view-engine/.

Collabnet VersionOne (5/7/2019), "13th Annual
State of Agile Report", accessed from
https://www.stateofagile.com/#ufh-c-
473508-state-of-agile-report.

Luce, T. (2016)"Adding Agility to the MIS
Capstone", *Issues in Information Systems*,
17(1), pp 119-127, 2016.

Luce, T. (2017), "Adding MVC to the Capstone
MIS Systems Development Course", *Issues in
Information Systems*, 18(10, pp119-127.

Beck, K., Beedle, M., van Bennekum, A.
Cockburn, A., et.al. Manifesto for Agile
Software Development" accessed from
http://agilemanifesto.org.

Microsoft (1/14/2019), "Tutorial: Get started with
EF Database First using MVC 5", accessed from
https://docs.microsoft.com/en-
us/aspnet/mvc/overview/getting-
started/database-first-development/setting-
up-database.

Microsoft (1/21/2019), "Tutorial: Get Started
with Entity Framework 6 Code First using MVC
5", accessed from
https://docs.microsoft.com/en-
us/aspnet/mvc/overview/getting-
started/getting-started-with-ef-using-
mvc/creating-an-entity-framework-data-
model-for-an-asp-net-mvc-application.

Microsoft (10/3/2018), "Getting Started with
ASP.NET MVC 5", accessed from
https://docs.microsoft.com/en-

us/aspnet/mvc/overview/getting-
started/introduction/getting-started.

Microsoft (n.d.), "Entity Framework
Documentation", accessed from
https://docs.microsoft.com/en-us/ef/.

Microsoft (n.d.), "Visual Studio Community",
accessed from
https://visualstudio.microsoft.com/vs/commu
nity/.

Rubin, Kenneth S., *Essential Scrum*, Pearson
Education, Inc, 2013.

Schwaber, K. and Sutherland, J., *The Scrum
Guide*, accessed from
https://www.scrumguides.org/scrum-
guide.html

Stack Overflow (n.d.), "For developers, by
developers", accessed from
https://stackoverflow.com/.

The Ohio Board of Regents (3/31/2008), *Report
on the Condition of Higher Education in Ohio:
Meeting the State's Future Needs*, accessed
from
https://www.ohiohighered.org/sites/ohiohigh
ered.org/files/uploads/board/condition-
report/ConditionReport_2008.pdf.

Tutorialspoint (n.d.), "ASP/.NET MVC – Data
Annotations", accessed from
https://www.tutorialspoint.com/asp.net_mvc
/asp.net_mvc_data_annotations.htm.

Tutorialspoint (n.d.), "MVC Framework –
Introduction", accessed from
https://www.tutorialspoint.com/mvc_framew
ork/mvc_framework_introduction.htm.

TutorialsTeacher (n.d.), "Action Method",
accessed from
https://www.tutorialsteacher.com/mvc/action
-method-in-mvc.

TutorialsTeacher (n.d.), "Routing in MVC",
accessed from
https://www.tutorialsteacher.com/mvc/routin
g-in-mvc.