Function Point Analysis and Agile Methodology

By Dan Horvath

As new software tools, methods and technologies are employed, there is often a question about whether Function Point Analysis (FPA) will apply. Agile Methodology is another case where the question arises. This article demonstrates how to apply FPA to Agile software development.

History and Definitions

Software Metrics in general and FPA in particular, are closely linked with Project Management. Project Management, in turn, is intimately concerned with the software development methodology being used. Consequently, in order to produce accurate results when considering estimating techniques and productivity measurement, software metrics are often tied to the software development methodology.

According to Project Management lore, a project is a temporary endeavor, having a defined beginning and end, undertaken to meet unique, specific goals and objectives, generally for the purpose of causing required changes or adding value. The output of a project is usually considered to be a product of some kind. Traditional waterfall software development methodologies fit well with this definition of a project. The concept of a beginning and end are essential to both the methodologies and the management of the project. Once a project is defined, software metrics may be gathered and applied. Project metrics are meaningless without well-defined projects.

Rapid Application Development

Software development methodologies began to evolve in the 1990’s. Rapid Application Development (RAD) was developed and gained popularity during this time. RAD methodology uses minimal planning in favor of rapid prototyping. The "planning" of software developed using RAD is interleaved with writing the software itself. RAD approaches generally enable a project to be implemented in a shorter period of time.

There are several types of RAD methodologies. Some of these are listed below. Although most of the methodologies foster the reuse of software, distributed development and small team structure, many RAD practitioners recognize that there is no single “rapid” methodology that can provide an order of magnitude improvement over any other development methodology.

Application development using RAD generally favors smaller projects that can be developed in sizeable pieces. This has a considerable influence on the concept of a project, but Agile Methodology has an even greater impact.
Some of the “flavors” of RAD methodology include:

**Agile Software Development** Features extremely small release cycles by breaking software projects up into mini-increments. Real-time, face-to-face communication is featured. Documentation is often developed at the end of the project.

**Extreme Programming (XP)** Lowers project costs by emphasizing a spiral-in approach. Software developers work in pairs with relatively little up-front design.

**Joint Application Design/Development (JAD)** Closely related to RAD, the primary difference being that the crucial role of the customer is emphasized and they are actively involved in design activities.

**Scrum** Very closely related to Agile methodology, scrum development is reduced to a series of short iterations or sprints. Teams are self-organized.

Although this article will focus on applying FPA to Agile software development, the same principles can be applied to all of the above methodologies.

**Agile Software Development**

Although Agile development traces its roots as far back as 1957, in most circles, it’s considered the next evolutionary step of Rapid Application Development. Most of the Agile principles were developed, defined, and documented in the Agile Manifesto, published in 2001.

Here is the Agile Manifesto in its entirety:

“*We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:*

- **Individuals and interactions** over processes and tools
- **Working software** over comprehensive documentation
- **Customer collaboration** over contract negotiation
- **Responding to change** over following a plan

*That is, while there is value in the items on the right [side of the above list], we value the items on the left more.*
The Twelve principles behind the Agile Manifesto are stated as:

1. **Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.**
2. **Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.**
3. **Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.**
4. **Business people and developers must work together daily throughout the project.**
5. **Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.**
6. **The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.**
7. **Working software is the primary measure of progress.**
8. **Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.**
9. **Continuous attention to technical excellence and good design enhances agility.**
10. **Simplicity—the art of maximizing the amount of work not done—is essential.**
11. **The best architectures, requirements, and designs emerge from self-organizing teams.**
12. **At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.**

The result of this manifesto is a type of RAD methodology that favors small projects with team-oriented development groups and close customer contact and involvement. Project management of Agile projects consists mainly of monitoring and controlling progress towards goals.

Given this new direction, it is understandable why it might appear that there may be difficulties in applying traditional software measures for Agile projects. When a project was clearly defined up front with a beginning and an end, it was easy to identify points in the life cycle where data collection could occur and measures could be created. The same can be done with projects utilizing Agile methodologies, but the approach must be flexible.

**Agile Software Development “Projects”**

In order to gather and apply project related software metrics to projects developed using an Agile Methodology we must first define the concept of a project in this context.

Regardless of development methodology, a project is still defined as a temporary undertaking with a defined beginning and end and with the purpose of bringing about a change and/or producing a product. For Agile development, a project may consist of one or more “stories” or “sprints” that are required to produce this final result. A user story is a high-level definition of a requirement containing enough information so that the developers can produce an estimate of
the effort to implement it. The story may eventually become one of the main artifacts to use in function point counting. A sprint is an incremental piece of work used by the Scrum methodology (which is closely related to Agile), as well as by the Agile methodology itself. This article will refer to sprints and stories as sprints for simplicity, although either term may be used in most of the examples. What is included or excluded from a sprint (and the function point count) is determined by the project manager as well as the customer. The article will also generally refer to a project as a series of sprints, although it is recognized that one sprint or story may make up an entire project. Once the project is defined in this way, project related software metrics gathering may take place. Function point analysis may be performed at the completion of the Agile project or at any point during its development, just as it would for any project.

Agile Software Development and Function Point Analysis

Function Point Analysis provides the base measurement of several metrics. It may be performed for a sprint (functionality developed) or for an entire project (functionality delivered). For example, there may be requirements to add a new logical file, along with maintaining and viewing the data, and a new report. Counting functionality delivered, these functions would be considered together in one project FP count (this is also the approach used to count traditional Waterfall methodology projects). Considering “functionality developed” using Agile methodology, the logical file, view and maintenance functions may be added in one sprint and the report in another. If counts are performed for each sprint, it may be possible to add the results together to arrive at the total size, which is the same as the “functionality delivered” count.

The following examples are intended to demonstrate the use of FPA in Agile development.

Development or Enhancement Project Example Regardless of Methodology (Functionality Delivered)

<table>
<thead>
<tr>
<th>Function</th>
<th>Type and Complexity</th>
<th>Unadjusted Function Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>Human Resource File</td>
<td>ILF, Avg</td>
<td>10</td>
</tr>
<tr>
<td>Human Resource Maint – add</td>
<td>EI, Avg</td>
<td>4</td>
</tr>
<tr>
<td>Human Resource Maint – chg</td>
<td>EI, Avg</td>
<td>4</td>
</tr>
<tr>
<td>Human Resource Maint – del</td>
<td>EI, Low</td>
<td>3</td>
</tr>
<tr>
<td>Human Resource Report</td>
<td>EQ, Avg</td>
<td>4</td>
</tr>
<tr>
<td><strong>TOTAL</strong></td>
<td></td>
<td><strong>25</strong></td>
</tr>
</tbody>
</table>
The following demonstrates how function points would be generated by counting individual sprints (functionality developed):

**Example 1: Agile Methodology Count**

Sprint 1 Function Point Count - data and maintenance functions added

<table>
<thead>
<tr>
<th>Function</th>
<th>Type and Complexity</th>
<th>Unadjusted Function Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>Human Resource File</td>
<td>ILF, Avg</td>
<td>10</td>
</tr>
<tr>
<td>Human Resource Maint – add</td>
<td>EI, Avg</td>
<td>4</td>
</tr>
<tr>
<td>Human Resource Maint – chg</td>
<td>EI, Avg</td>
<td>4</td>
</tr>
<tr>
<td>Human Resource Maint – del</td>
<td>EI, Low</td>
<td>3</td>
</tr>
<tr>
<td><strong>TOTAL</strong></td>
<td></td>
<td><strong>21</strong></td>
</tr>
</tbody>
</table>

Sprint 2 – report function is added

<table>
<thead>
<tr>
<th>Function</th>
<th>Type and Complexity</th>
<th>Unadjusted Function Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>Human Resource Report</td>
<td>EQ, Avg</td>
<td>4</td>
</tr>
<tr>
<td><strong>TOTAL</strong></td>
<td></td>
<td><strong>4</strong></td>
</tr>
</tbody>
</table>

In this example, when function point counts for Sprint 1 and Sprint 2 are added together they will equal the total function points for the project (the delivered functionality).

Agile methodology introduces additional opportunities for information such as sizing of sprints. Let’s say that for the example above, an additional data element is added to the logical file, the view, and the maintenance functions in the second sprint. In this case, the FP count after the first sprint would be the same as Sprint 1 above, but for the second sprint all of the functions (the file, view, maintenance, and the report) would be counted as being changed.
Example 2: Agile Methodology Count

Sprint 1 Function Point Count – data and maintenance functions added

<table>
<thead>
<tr>
<th>Function</th>
<th>Type and Complexity</th>
<th>Unadjusted Function Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>Human Resource File</td>
<td>ILF, Avg</td>
<td>10</td>
</tr>
<tr>
<td>Human Resource Maint – add</td>
<td>EI, Avg</td>
<td>4</td>
</tr>
<tr>
<td>Human Resource Maint – chg</td>
<td>EI, Avg</td>
<td>4</td>
</tr>
<tr>
<td>Human Resource Maint – del</td>
<td>EI, Low</td>
<td>3</td>
</tr>
<tr>
<td><strong>TOTAL</strong></td>
<td></td>
<td><strong>21</strong></td>
</tr>
</tbody>
</table>

Sprint 2 – data and maintenance functions changed; report function is added

<table>
<thead>
<tr>
<th>Function</th>
<th>Type and Complexity</th>
<th>Unadjusted Function Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>Human Resource File</td>
<td>ILF, Avg</td>
<td>10</td>
</tr>
<tr>
<td>Human Resource Maint – add</td>
<td>EI, Avg</td>
<td>4</td>
</tr>
<tr>
<td>Human Resource Maint – chg</td>
<td>EI, Avg</td>
<td>4</td>
</tr>
<tr>
<td>Human Resource Maint – del</td>
<td>EI, Low</td>
<td>3</td>
</tr>
<tr>
<td>Human Resource Report</td>
<td>EQ, Avg</td>
<td>4</td>
</tr>
<tr>
<td><strong>TOTAL</strong></td>
<td></td>
<td><strong>25</strong></td>
</tr>
</tbody>
</table>

In Example 2 above, the Waterfall version would not be the same as the sum of the Agile sprint counts, since some of the functions would be counted twice as part of sizing the sprint.

When the function point counts for the two sprints in Example 2 are added together, the sum is greater than the total amount delivered for the project, which is incorrect. To obtain the function point size of the project it is always necessary to analyze what is ultimately delivered to the end user at the conclusion of the effort.

Whether to count by sprint or by project (developed or delivered) depends on the goals of the metrics program. Counting for each sprint provides additional measurement capability since it is now possible to determine the cost of changing functionality delivered in an earlier sprint. This is the difference between the sum of the sprints and the single version of software delivered. Even though the individual sprints cannot be added together to derive the final project count, a final count can be performed for the entire project at the time of delivery. This is similar to the counting that would be done for a traditional Waterfall project. In this instance the sprint
counts would provide the input for establishing the final project count. It is also possible to use the sprint FP counts to arrive at a count of the delivered functionality by ensuring that no function is counted more than once for the delivered function point count.

**Agile Software Development and Software Metrics**

Any measurement program is an attempt to answer basic questions such as:

- How productive are we?
- How good is our software quality?
- How much effort is required?
- How much should it cost?
- How long should it take?
- How many resources should be assigned?
- How is a project impacted due to changes in scope?

As noted, Function Point Analysis is used to provide the base measurement of several metrics such as productivity, quality, estimation and change of scope. These, in turn, may be used to answer the types of questions noted above. Some of these major metrics will be examined here.

**Productivity**

Productivity measurement involves function point analysis along with determining the amount of effort expended. A typical calculation for productivity is Function Points Delivered per Effort hour (FP/Hr). Using Waterfall methodology, the function points delivered and the effort hours would typically be measured at the final implementation phase of the project. Using Agile methodology, such measurements may also be taken at the implementation phase or at the end of a specific sprint. In the former case, the productivity rate reflects the delivered functionality (as it is for a Waterfall project) and, in the latter case, the productivity measurement reflects the function points developed for that sprint.

There may be an impact to productivity when functions are enhanced in more than one sprint for a given project. However, the trade-off is that the projects developed using Agile methodology may be able to deliver at least some functionality at an earlier point in time which in turn can increase customer satisfaction.
**Quality**

Quality may be defined as the number of delivered defects per function point (defects/FP). Related measurements include defect removal efficiency (ratio of the defects resolved before delivery to the total number of defects found) and defect cost (the cost of correcting the defects). Quality metrics are often evaluated at the same project phase as productivity metrics. That is, during the implementation phase of projects using Waterfall methodology and either at the end of a story or sprint and/or at the time of delivery for projects using Agile methodology.

Example 2 outlined above shows some functions counted more than once for the same project since one Agile sprint creates functions, and the next one changes them. The effort for changing the functions in the second sprint may be planned or unplanned. If unplanned, it may be due to new/changed requirements or defects. Care must be taken to ensure that the Function Point Count as well as the related Quality and Productivity metrics accurately reflect the measurement objectives.

**Estimation**

Estimation may use any of several methods. One of the most rigorous and accurate employs an estimate of the number of function points to be delivered and the expected productivity, effort, cost, schedule and staffing. For both Waterfall and Agile projects, these estimation metrics would typically be determined prior to the start of project development. For Agile projects, however, this means that objectives are determined at the beginning of a story or sprint, rather than at the beginning of all development. The reason is that too little is known about the requirements until each story/sprint begins.

Estimating Agile projects at a complete project level may still be done using rough order of magnitude (ROM) techniques. These techniques are used when little else is known about the detailed requirements. If the approximate number of internal logical files is all that is known up front, a ROM count can be done using only that information. The ROM count can be refined as more information, such as maintenance or reporting requirements, becomes known.

For example, an enhancement project has requirements that five logical files are to be added. If little else is known, assumptions can be applied (such as average complexity for most functions) to develop a ROM count as illustrated here:

- 5 average ILF’s: 50 UFP
- 10 average EI’s (add and change functions for each): 40 UFP
- 5 low EI’s (delete function for each): 15 UFP
- 5 average EO’s (one report related to each ILF): 25 UFP
- **Total= 130 UFP**
With this ROM function point count we can now estimate effort using productivity measurements from past projects. Effort may then be used to determine resource requirements and expected quality, etc. The estimate may also be used to break the project up into manageable sprints, depending on organizational standards and guidelines.

‘Time boxing’ may be an additional consideration for Agile projects and can be used in different situations. In one case where the requirements for the sprints, or the sprints themselves are not initially well defined, the project may be broken into time segments, and each of those may be estimated. It should be noted here that time boxing may be used for other metrics as well. If historical data is available (such as the function point counts of previous projects), the organization will be able to determine the number of FPs that can be delivered in a certain period of time so that Time Boxes can be planned. Alternatively, when the requirements are well defined and a good function point count exists, time boxing can be used to determine how much functionality will be delivered within each sprint.

**Change of Scope**

Project Scope is defined by the Project Management Institute as “The work that needs to be accomplished to deliver a product, service, or result with the specified features and functions.” Changes to project scope result in changes to the work that had originally been defined and these changes may result in changes to project cost or quality. The amount of change can be expressed in function point difference, or in effort to make the change(s). Change of scope may be measured at the time the changes occur, at specific points within the project development, or at the completion of the project. Since scope changes may occur more frequently during Agile projects, it may be better to consider scope changes at a higher level for the entire set of software delivered rather than at project intervals. It is best to define up front what constitutes a scope change. This way it can be distinguished from changes to functions that are inherent in Agile projects.

As in Example 2 above, if a function is added in one sprint and changed in a later one, this may be considered a change in scope and it may be measured if each sprint is counted. Once again, Agile development provides the opportunity to gather this additional information in an accurate and timely manner.

**Conclusion**

Although the function point counting process itself is the same for the various development methodologies, there are some additional considerations when counting projects developed using Agile methodology. The sprint artifacts may be different than the documents used when counting projects using other development methodologies. In order to be properly utilized for the counting process, these artifacts must describe in detail the functionality added, changed or deleted by the project. Another consideration is that if multiple sprints are considered part of
the same project, it’s possible that some application functions may be affected more than one time.

As noted, Function Point Analysis may also be performed at various stages during the Agile development effort. These cases would be different than the function point counts performed at various stages of development using Waterfall methodologies. In the case of Waterfall development there is a defined point in the project to perform the count, such as after the analysis or functional design phases. Counts done during the course of Agile development would measure an intermediate work product at the completion of one or more stories or sprints. This may be for the purpose of assisting the customer or project manager in monitoring and/or controlling the project, or measuring the functionality of a sprint as part of the overall project.

It is important to be able to determine the impact of using any new development methodology. By analyzing the data to determine the impact of quality, productivity, schedule, and cost, software development organizations may be better able to choose the most appropriate methodology.

In conclusion, the development methods utilized for projects need not be an impediment to measuring the project. As demonstrated in the examples presented above, Function Point Analysis (FPA) can be used for measuring Agile projects. Several considerations need to be addressed depending on how the project is defined, the goals and objectives of the organization, and supporting measures. Using Function Point Analysis and the other software metrics discussed in this article provide new opportunities to apply metrics in additional and beneficial ways.

References