Estimating Software Projects

By

Bill Meacham

Bill Meacham – bmeacham@bmeacham.com – http://www.bmeacham.com
Revised: April, 2003

Contents

Figures

Revision History

Date	Person	Revision
April, 2003	Bill Meacham	Prepared for PMI Austin

Introduction

This class describes a method of estimating used by the author in numerous engagements as a software development consultant. The assumed context is that of a consultant doing work for hire for a client. The concepts should be applicable to staff developers doing work for clients internal to their organization as well.

The author has used this method successfully on projects ranging in size from 150 to 10,000 person-hours (one person for a month up to five people for a year). The principles are applicable to even larger projects, but the level of detail would have to be increased.

Objectives

At the conclusion of this class the participant should be able to demonstrate an understanding of:

- How to estimate the size of a software development project
- How to estimate the effort required to develop a software system

Overview

The ability to estimate accurately the size, effort and duration of a software development project is crucial to project success. Accurate estimates enable you to do several things:

- Bid the job with some assurance that you will make money
- Satisfy the client
- Keep your sanity as you execute the project

In order to provide accurate estimates you need to know several things:

- Deliverables: what you will deliver to the client. This includes the component elements of the system you will build as well as other deliverables such as technical documents. You need to have a list of web pages, back-end processing, reports, etc. that will make up the final system. The component elements must fulfill the requirements.
- What phases and stages of activity you will go through in order to build the system. The phases and stages make up the lifecycle model.
- How much effort it typically takes to build each element. For small projects this can be the total effort for all stages for each element. For

more complex projects this should be the effort it takes *by phase* for each element.

Requirements and the lifecycle model are covered in depth in other classes in this series. What follows briefly recapitulates that material.

Software Requirements

Deliverables must fulfill the requirements, or the system will not be what the user has asked for. There are two types of requirements:

- Functional requirements, sometimes called *customer requirements*. These are the things that the customer wants the product or system to do, the benefits the customer wants from the product or system.

- Software requirements, sometimes called *derived requirements*. These are the things the product or system logically has to do in order to fulfill the customer requirements.

Once the functional requirements are documented and well understood, we can create a Software Requirements Specification (an outcome of the Requirements Analysis stage in the lifecycle model pictured below on page 7.) In a structured programming environment the Software Requirements Spec typically includes such things as a data flow diagram and processing specification, entity relationship diagram and data dictionary, a process flow chart, etc. In object-oriented development it may include use case diagrams and specifications, class diagrams, activity diagrams, sequence or collaboration diagrams, etc. These components tell us what the components of the system will be.

Entity-Relationship Diagram

For instance, following is an example of part of an Entity-Relationship diagram (ERD):

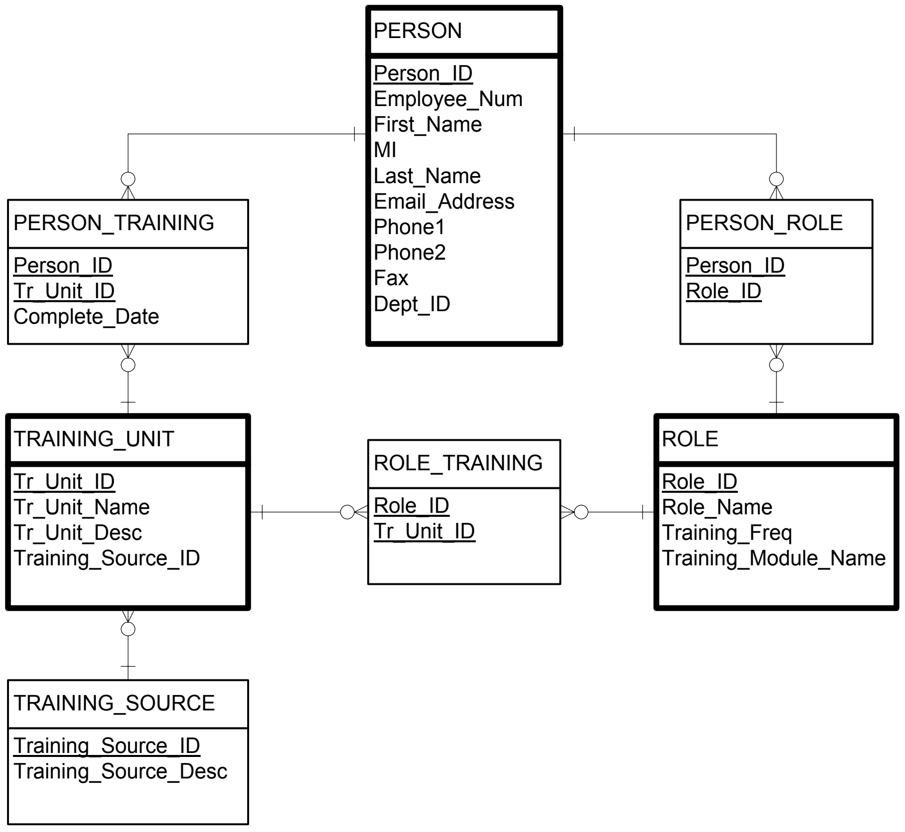
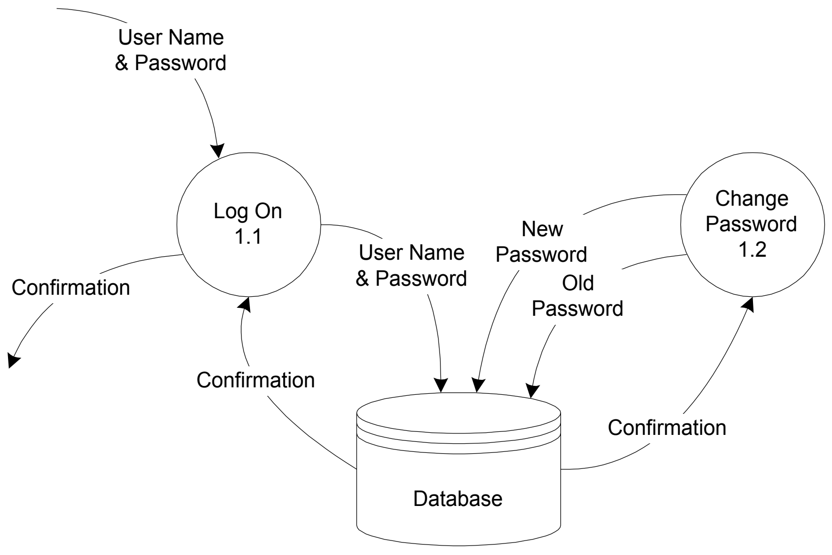


Figure 1. Entity-Relationship Diagram

The ERD tells us that there will be several tables in the database, and what those tables are.

Data Flow Diagram

And here is an example of part of a Data Flow diagram (DFD):



DFD 1: Logon

Figure 2. Data Flow Diagram

The DFD tells us that there will be several pages in the web site and what those pages are. In addition, there may be back-end processing that is not revealed directly on a web page. This will be modeled in the DFD along with the web pages.

Lifecycle Model

The lifecycle model tells you what the phases and stages of activity will be. For instance, following is a recommended lifecycle model for small to medium-sized web projects:

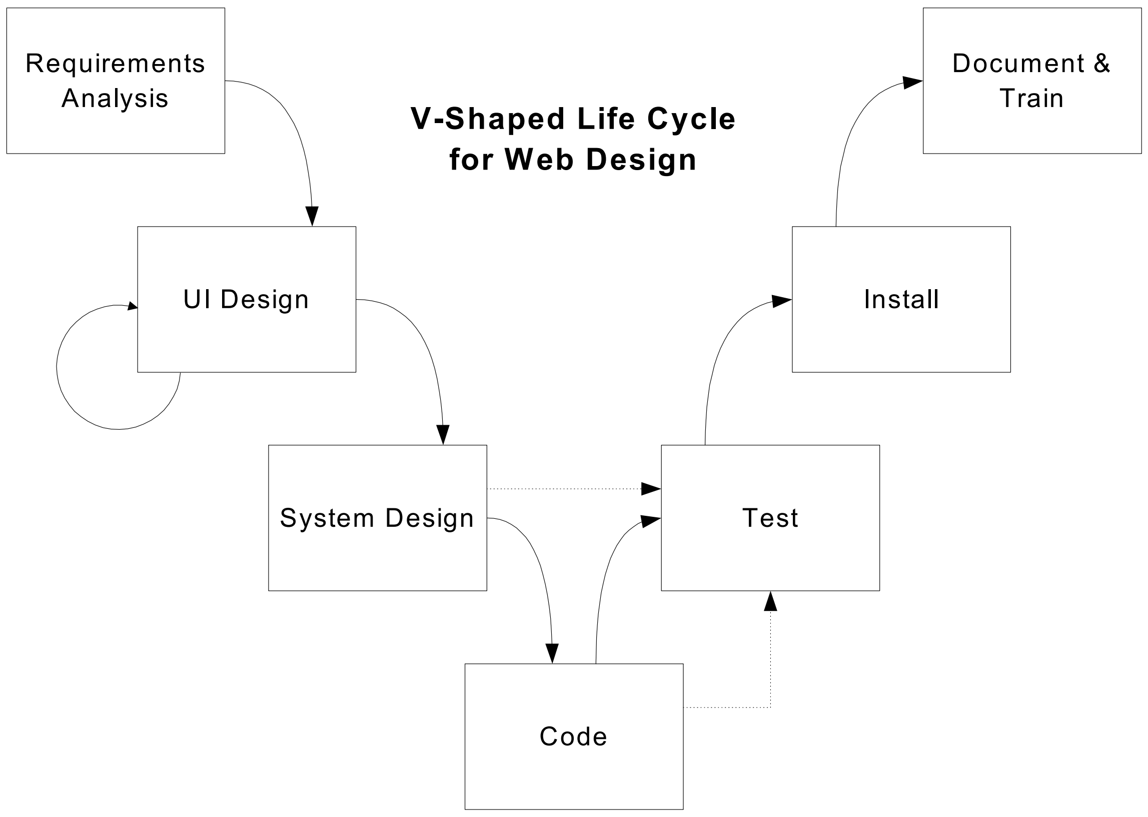


Figure 3. Lifecycle Model

The solid lines indicate succession of phases. The UI (User Interface) Design phase is iterative; the designer prototypes the visual appearance and site navigation using rapid web design tools, shows it to the client, gets feedback and makes revisions and shows it to the client again until the client is happy with the design. The dotted lines indicate that the test plan is written in the System Design and Code phases and then used in the Test phase.

Typical Effort

The only way you are going to have a firm grasp on how much effort it takes to develop each component is to keep careful track of the effort you expend as you build a system and record that data for later use. The more often you do this, the more accurate will be the data you use for your estimates.

The templates distributed with this class include some typical effort estimates in them, but *you should not rely on those estimates*. Instead rely on your own experience. As time goes by, we ask our developers to provide us with actual data so we can compile it to help estimate new projects.

The Estimating Process

Having determined the software requirements, you know how many elements and of what type (data table, web page, etc.) the system will contain. For each element you want to determine the resources needed to accomplish the tasks of defining its requirements, building it, testing it, etc. The sum of these will be the resources needed to implement the project.

To determine the resources needed, you have to predict the size of the deliverable, i.e. how big it will be. Both estimating size and estimating effort involve prediction. <u>Estimating size</u> is the prediction of how big the deliverables will be. <u>Estimating effort</u> is the prediction of the resources needed to produce the deliverables. Accurate estimates of size and effort are important for the following reasons:

- To set realistic expectations. Many project failures are due to unrealistic expectations based on inaccurate estimates. Performance is not poor, but the prediction of performance is often poor.

- To improve software quality. Accurate estimates provide the foundation for project management. Accurate estimates provide for control. Inaccurate estimates will require adjustments in the project that typically introduce defects.

Estimating size and estimating effort go together. The reason for predicting the size of the deliverables is to be able to estimate accurately the effort needed to produce them.

Estimating Size

Here are some things to consider when estimating the size of a project.

- Choose a measure that is appropriate for each terminal element. You may have many types of measures, such as number of tables and web pages, number of pages of documentation, number of meetings, etc.

- Stick with the size measure for each element so you can compare actual data to planned estimates over time to provide feedback for improvement. This will also help you compare one project to another.

- The chosen size measures should be easy to use early in the software life cycle.

- The chosen size measures should be readily observable after project completion.

Some examples of size measures include the following:

- Lines of Code (LOC)
- Formal methods such as Function Points or Feature Points
- Other measures of the system's functionality such as
  - Number of bubbles on a Data Flow Diagram (DFD)
  - Number of entities on an Entity Relationship Diagram (ERD)
  - Number of classes, attributes and services on an Class Diagram

Following is some discussion of each of these methods.

Lines of Code as a Unit of Size

We do *not* recommend lines of code as a measure, particularly for applications built with a visual tool. The number of lines of code in a system has historically been used as a measure of its size, mainly because it is easy to count lines of code. But there are a number of disadvantages:

- It is hard to know how many lines of code will be needed before the system is developed.
- Most modern application development environments are highly visual and the code generated is hidden from the developer.
- Source instructions vary with the type of coding languages.
- Software involves many costs that may not be considered when just estimating code size.
- LOC count should distinguish between generated code and hand crafted code because code generators often produce excess code that skews the LOC count and generated code takes a different amount of effort to produce.

Other Formal Methods

Because of the disadvantages of LOC as a unit of measure, people have devised other ways of estimating the size of a software system. These include formal methods such as Function Point Analysis, Feature Point Analysis and Constructive Cost Modeling (COCOMO). Such formal methods can be very useful when used rigorously on large projects. For small projects in which the schedule is tight, they are too complicated. They are not covered in this series.

Functionality as a Unit of Measure

In our experience, the best way to estimate the size of a small to medium project is to look at the software requirements of the system. Count the artifacts on the models. How many tables (entities) are there on the Entity-Relationship Diagram? How many web pages will there be? How many bubbles on Level 1 of the Data Flow Diagram? How many classes on the Class Diagram? How many data repositories, relationships, attributes, services? How many simple queries? How many complex, multi-table-joins? How many simple updates? How many complex updates? How many different business rules need to be accounted for?

Estimating Effort

Once you have estimated the size of the project, in whatever units are appropriate, you can estimate how much effort it will take to create the system. We discuss effort instead of time because often the overall effort may be spread among several team members or even among several projects, therefore bearing little or no relationship to elapsed time.

For all but very small projects, we recommend that you estimate effort based on activities to be done as well as things to be produced. Your sizing estimates tell you how many things (tables, web pages, etc.) will be produced. You can then look at the specific activities needed to produce them, such as design, coding, testing, fixing bugs, etc. This will also help you set milestones such as delivery of the first prototype, delivery of the final version, etc. For each activity, multiply the effort needed for each unit times the number of units. This is explained in more detail in Step-by-Step: Estimating a Project by Phases, page 17, and in Appendix A – Estimation Spreadsheet by Phases, page 26.

For very small projects, you can estimate effort based solely on the deliverables without breaking your activities out by phases. This approach is explained in Step-by-Step: Estimating a Simple Project, page 22, and in Appendix B – Estimation Spreadsheet for a Simple Project, page 28.

Remember that effort includes all activities, managerial, integral, administrative as well as developmental[1]. Either increase the effort estimate for each activity to account for the other activities or add them in as a separate line item at the bottom. The template spreadsheets include a percentage for project management at the bottom.

How do you know how much effort will be needed? There are no hard and fast rules here. It will depend on the experience you or others have had in similar environments, using similar tools, and creating similar applications in the past.

You can often estimate managerial, integral and administrative tasks by looking at comparable past projects.

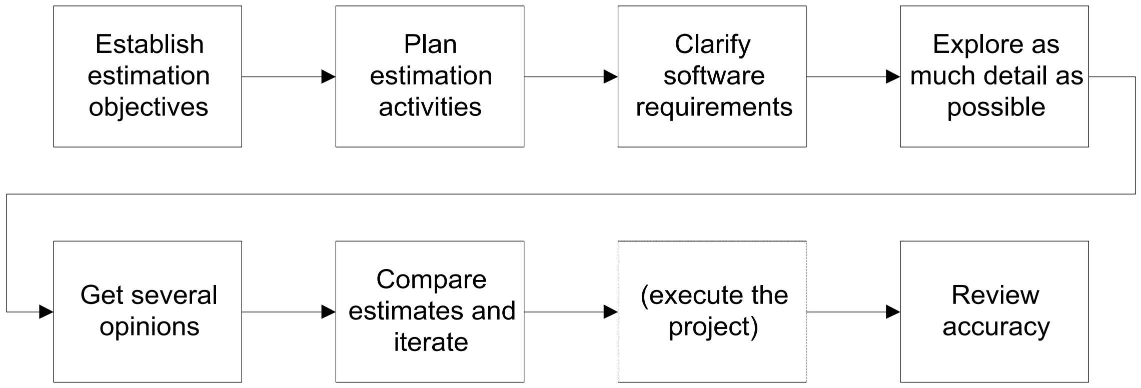
If you do not have past experience to draw on, make an estimate anyway. Estimate how much effort will be required per entity, per bubble, per class, per

[1] Development tasks include such things as Define Requirements, Build Prototype, Build Final Version, Test, etc. Managerial tasks include project planning, tracking, control, risk analysis, etc. Administrative tasks include word processing, review meeting room scheduling, etc. Integral tasks are those that span development phases and provide support for the project, such as configuration management. maintaining software development tools, collecting and analyzing metrics, etc.

query, per update, per business rule, etc. Record the estimates, and as the project goes on record the actual effort spent per entity, bubble, etc. When the project is over you will know how much effort was spent compared to the estimates and you will have started a useful database of historical measures that you can use to estimate future projects.

The Estimation Paradigm

In general, the steps in the software effort and cost estimation process are those shown below in Figure 4.



Establish estimation objectives
Plan estimation activities
Clarify software requirements
Explore as much detail as possible
Get several opinions
Compare estimates and iterate
(execute the project)
Review accuracy

Figure 4. Software Estimation Process

Step 1. Establish Estimation Objectives

Understand how the estimates will be used. If you are doing an initial feasibility study, the estimates do not have to be as rigorous as they would be for a project that is about to be executed. Some other things to consider:

- Absolute estimates are necessary for labor or resource planning.
- Relative estimates may be used for either/or decisions.
- Liberal versus conservative estimates heighten decision confidence.

Step 2. Plan Estimation Activities

A plan sets accurate expectations for the cost and value of the estimation process. View the estimation activities as a “mini-project.” Estimation, like all other software activities, serves better when more effort is put into it — you get what you pay for. An estimate produced in 30 minutes will not be as accurate as one built upon individual estimates of component parts of the system based on a carefully constructed work breakdown structure.

We recommend, if at all possible, that you sell your engagement in two parts. First contract with your client for 40 to 160 hours or more of scoping and estimating (the Requirements Analysis phase of the recommended life cycle). In this phase you will gather the functional requirements and come up with the software requirements in enough detail to be able to make an accurate estimate for the rest of the engagement. Then contract with your client for the rest of the work.

How do you know how many hours to propose for the Requirements Analysis phase? Obviously this requires some estimating in itself. You will need to get a feeling for the size of the project from your initial contacts with the client. Some things to look for are the following:

- Functionality. Try to get an idea of how many web pages will be involved, how complex the database will be, how many external systems the system will need to interact with, etc.. The more complex the eventual system, the more time you should spend on estimating.

- Client availability. Find out who the players are in the client's environment. How many are there? Are they all physically in the same place? Do they all have time to talk to you? One of the most important success factors in delivering a winning system is to identify all the requirements, and that means talking to all the people who have a stake in the system. The harder it looks like it will be to do this, the more time you should allot for scoping and estimating.

- Clarity of purpose. Does the client have a good, clear idea of what they want to accomplish? Or do they have a marketing idea but not much vision of the detail? The more you will have to help define the goals of the project in addition to the methods for achieving the goals, the more time the estimating part of the engagement will take.

- Client process maturity. Find out how much the client appreciates the value of good process and good project management. Does the environment seem orderly and well-managed, or is it chaotic and frenzied? Does the client have an appreciation of the need for careful planning or do they just want you to start coding and producing something right away? The less process maturity, the more time you will need to spend educating the client while you are scoping and estimating.

Step 3. Clarify Software Requirements

Clarify the requirements to the extent necessary to meet your estimating objectives. The more you understand about the requirements, the better your estimates will be. When you can't clarify, document any assumptions!

Step 4. Explore as Much Detail as Feasible

Remain consistent with estimating objectives. The more detail you explore, the better your technical understanding will be. The more you think through all of the software functions, the less likely you will be to miss the costs of the less-obvious functions.

Step 5. Get Several Opinions

Have several people develop estimates independently and then compare and talk about the results. At the very least, have someone else review your estimates and give you a sanity check. Also, use more than one technique, for instance consultation with others and a statistical approach.

Step 6. Compare, Understand and Iterate Estimates

People have differences in experience, roles and incentives. Some will be optimistic and some pessimistic. Pay attention to the source of your estimates and adjust accordingly. Also pay attention to Pareto's Law: 80% of the cost is contained in 20% of the components. Examine and iterate these components in greater detail.

Step 7. Review Estimate Accuracy

Compare your estimates against actual performance as the project progresses and at the end of the project. This will help you estimate better in the future.

Statistical Approach

You can get a better estimate if you provide an optimistic, pessimistic and realistic estimate. Then you can form a beta distribution by multiplying the realistic estimate by four, adding the optimistic and pessimistic, and dividing the total by six. For example, the time required to build a particular component might be estimated at between four and eight hours, with a belief that it will be closer to five. Thinking about optimistic and pessimistic scenarios might produce this final estimate.

$$\frac{4 + (5 \text{ x } 4) + 8}{6} = 5.33 \text{ hours}$$

If there are ten such components, the total estimate for those components would be 53 or 54 hours.

General Tips for Estimating

- Avoid off-the-cuff estimates. The safest policy is never to give them. You never know how far they will travel, and people never remember the conditions you put on them.

- Allow time for the estimate and plan for it. Treat the estimation process as a project in itself. Hasty estimates will be inaccurate.

- Use data from previous projects. This is the single most important factor in providing accurate estimates and avoiding cost and schedule overruns.

- Estimate at a low level of detail. Figure out all the pieces of the project in as much detail as you can and estimate each piece, preferably by task (development phase).

- If you will have more than one person on the project, make the estimates a group process:

    1. Have each member of the team estimate each piece individually as Optimistic, Likely and Pessimistic

    2. Meet as a group to compare and discuss the estimates. Discuss and understand the sources of the differences. Come to a consensus regarding the estimate for each piece.

    3. Use the formula given above under Statistical Approach, to factor the Optimistic, Likely and Pessimistic estimates for each piece.

    4. Add up all the pieces.

Software estimation always involves risk and uncertainty. A good estimate captures that risk and uncertainty so that your client is aware of it. The way you present the estimate to your client can have a big impact on how the client will respond to a request to change the estimate later. Here are some ideas for how to present your estimates:

- Use plus-or-minus qualifiers to indicate the amount and direction of uncertainty. For example:

2000 hours, + 500 hours, – 500 hours	A confident estimate
2000 hours, +1000 hours, – 0 hours	A risky estimate

- State the estimate as a range to avoid people ignoring the plus and minus parts.

1500 – 2500 hours	A confident estimate
2000 – 3000 hours	A risky estimate

- Add percentage factors to indicate the amount of your confidence in the various estimates

1500 hours	5% confidence
2000 hours	60%
2500 hours	75%
3000 hours	95%

Step-by-Step: Estimating a Project by Phases

Following are the steps for using the spreadsheet template for estimating a project by both deliverable component and development phase.

Define Phases

1. Decide what your life cycle phases are. The template accompanying this class assumes the seven phases listed on page 7 under Lifecycle Model. If you need to change the phase names, do so on both the Standard Levels of Effort tab and the Feature Point Detail tab.

   If you need to add phases, copy a whole column and paste it, as follows:

   a. Select one of the middle columns that define the phases, not the first or the last. Select the entire column, not just certain cells, and copy it. (Do Ctrl+C or select Edit / Copy on the spreadsheet menu.)

   b. Without changing your selection, do Insert / Copied Cells on the spreadsheet menu.

   c. Change the phase name of the new column as needed.

   d. Do the exact same procedure with the same column on both the Standard Levels of Effort tab and the Feature Point Detail tab.

   This will ensure that the formulas remain correct.

Define Deliverables

2. Decide what your deliverables are. Gather functional requirements and define the software requirements. (These activities are defined in detail in the Software Development Process class in this series.)

3. Group the deliverables into categories. Following are suggested categories for small to medium-size web development projects. If you need to change the categories, do so on both the Standard Levels of Effort tab and the Feature Point Detail tab.

Deliverable	Description
Data tables	Tables in the database.
Web Pages	Pages seen by the end user. This category includes both producing the visual appearance of the pages – the html, the images, etc. – and designing and coding the logic of the dynamic functionality, such as interacting with the database, accepting user input on forms, etc. Note – If the visual appearance is of particular importance, as it is in a public e-commerce site, for instance, then split this category into two, one for producing the pages and one for producing the logic.
Usability Design	Sometimes called Information Architecture or Cognitive Engineering, this is the activity of designing the site navigation and the visual appearance of the pages in order to make it easy to use and attractive to the end user. This is different from the raw functionality. The raw functionality can be delivered in many different ways, ugly or pretty, frustrating to use or easy and intuitive to use, or in between. Usability Design is concerned with how the raw functionality is presented to the end user. Typically, the only development phase this applies to is User Interface Design. It is not lumped in with Web Pages because the complexity factor may not be the same as that for creating the web pages.
Reports	This category refers to reports delivered directly to the site administrator or owner, not content delivered over the web.
Other	This category includes any system element that does not fall in the other categories, in particular back-end processing that is not directly visible on a web page. For instance, if the system has to send email alerts when certain conditions happen or has to read and parse XML files from an outside source, include that functionality here. You may need to split this into more than one category if more than one distinct type of processing is required.
Documents	This category includes technical documentation (requirements documents, design specs, etc.) as well as user documentation and installation instructions.

Enter New Categories If Needed

If you have more than one type of functionality that falls in the Other category, you will need to add the extra types to the spreadsheet in two places, on the Standard Levels of Effort worksheet and on the Feature Point Detail worksheet.

4. Add appropriate rows to the Standard Levels of Effort worksheet directly under Other. Make sure the Total column sums the columns for the development phases.

5. Add appropriate sections to the Feature Point Detail worksheet. Copy all the rows in a section and insert them under the Other section.

   To insert an additional section do the following:

   a. Select all the rows within a section and copy them. (Do Ctrl+C or select Edit / Copy on the spreadsheet menu.) Select entire rows, not just some of the columns.

   b. Select an entire row below the last section but above the Total for All Components row and insert the copied rows above it. (Select Insert / Copied Cells on the spreadsheet menu.)

   This is tricky, as you need to make sure that the formulas are correct. Check the following:

   - The Standard Levels of Effort cells for the section must point to the correct cells on the Standard Levels of Effort worksheet.

   - The formulas in each of the detail lines in the section must be correct. Compute each cell in the section = (the value in Complexity Factor on its row) * (the Standard Level of Effort cell for the section in its column).

   - The totals for each section must include all the rows in the section.

   - The total line below all the sections, titled "Total for all components" must include the total line of each section you add.

Enter Standard Levels of Effort

6. Look at the values on the Standard Levels of Effort worksheet and adjust them if needed.

   The <u>standard level of effort</u> is the staff time (person-hours) needed to perform one stage of activity on one simple deliverable, for instance to define the requirements for one simple table or build one simple web page.

Obviously, these are averages. If you have a site with a number of simple web pages, it is unlikely that each one will take exactly the same amount of effort. Take the amount of effort needed to produce all of them and divide by the number of pages to get an average.

How do you know what a reasonable standard level of effort is? From your own experience and that of others who develop similar applications using a similar process and similar tools. The spreadsheet template provided with this class includes our initial estimates, but do not take these as definitive. Instead collect metrics from your own project and adjust the standard levels of effort in future estimates to more accurately reflect your own experience.

The Project Management class in this series discusses in detail how to gather metrics on actual effort expended.

Enter Feature Point Detail

7. Enter the deliverables in the appropriate category on the Feature Point Detail worksheet. Insert additional rows to accommodate all the deliverables.

   To insert additional rows do the following:

   a. Select an entire row within a category and copy it. (Do Ctrl+C or select Edit / Copy on the spreadsheet menu.)

   b. Select an entire row in the same category and insert the copied row above it. (Select Insert / Copied Cells on the spreadsheet menu.)

   Adding rows this way ensures that the formulas remain intact.

8. When you are done adding rows, be sure the formulas remain correct.

   - Each cell in the section must be computed by multiplying the value in Complexity Factor on its row times the Standard Level of Effort cell for the section in its column.

   - The totals for each section must include all the rows in the section.

9. For each deliverable, enter the complexity factor in the Complexity Factor column.

   The complexity factor indicates how complex the deliverable is. For example, a simple database table, which has a complexity factor of 1, is one that has fewer than 20 columns, three or fewer constraints, and no triggers. If a table has more columns or more constraints or more triggers, increase its complexity factor. The idea is to calculate the effort by multiplying the complexity factor times the standard level of effort for a simple element, whose complexity is 1.

If you have several deliverables that have the same level of complexity, you can put them all on one line instead of listing each of them separately. Enter how many there are under Volume and the complexity level under Rate. Then compute Complexity Factor = Volume * Rate. Be sure to make reference to a requirements document that does list each one separately.

For instance, if you have eight simple tables you could list each one separately with a complexity of 1 or you could list them all on one line with a calculated complexity of 8.

10. If there are other tasks in the project that do not fit into this scheme, add a single line for each of them just above the Total for All Components line. Make sure they are added in to the Total for All Components.

    For instance, maybe you have to purchase and install hardware as part of the project or research and recommend an off-the-shelf component such as an accounting package. These tasks do not fit in neatly with software development phases, so add a line item for each one and put your estimate in the Total column. Make sure you add them into the Total for All Components.

11. Review your percentage for Project Management.

    You must include funding for Project Management! If you do not manage the project, you have very little reason to believe it will succeed. We suggest 15% of the total, but you should set this to an amount you feel comfortable with.

    We have had clients that do not want to pay for project management. This class should give you some ammunition for convincing them that it is necessary. If they are still not convinced, then omit the Project Management line item, but increase the Standard Levels of Effort to include the extra percentage.

12. Review your percentage for Contingency.

    You should include a percentage for Contingency. You can be virtually certain that things will not go entirely as planned.

13. Review the spreadsheet for accuracy.

    Get someone to check all the formulas and make sure the numbers add up right. You can get in big trouble if you underbid a job because certain line items are not included in the total.

14. Do a sanity check on your numbers and get others to review the estimate.

    Ask yourself if the final figure is in the ballpark based on your gut feelings. See how it compares to similar projects you have done in the past. Get others to give you their opinions.

Step-by-Step: Estimating a Simple Project

Following are the steps for using the spreadsheet template for estimating a project by both deliverable component only, without separating out the effort by phase. Use this spreadsheet if the project is small enough that more detail would be burdensome.

Define Deliverables

1. Decide what your deliverables are. Gather functional requirements and define the software requirements. (These activities are defined in detail in the Software Development Process class in this series.)

2. Group the deliverables into the categories such as Data Tables, Web Pages, Reports, etc. These categories are explained on page 17.

Enter New Sections If Needed

If you have additional elements, you will need to add them to the spreadsheet in two places, in the Effort Assumptions section and in the list of elements.

3. Add appropriate rows to the Effort Assumptions section.

4. Add appropriate sections in the list of elements. Copy all the rows in a section and insert them under the last section.

   To insert an additional section do the following:

   a. Select all the rows within a section and copy them. (Do Ctrl+C or select Edit / Copy on the spreadsheet menu.) Select entire rows, not just some of the columns.

   b. Select an entire row below the last section but above the Total All Features row and insert the copied rows above it. (Select Insert / Copied Cells on the spreadsheet menu.)

   This is tricky, as you need to make sure that the formulas are correct. The critical formula is in the Effort cells. Check the following:

   - The formula in the Effort cells for the section must point to the correct cells in the Effort Assumptions section.

   - The totals for each section must include all the rows in the section.

   - The total line below all the sections, titled "Total All Features" must include the total line of each section you add.

Enter Effort Assumptions

5. Look at the values in the Effort Assumptions section and adjust them if needed. These specify the staff time (person-hours) needed to produce one deliverable whose complexity is Simple, Medium or Complex.

   Obviously, these are averages. If you have a site with a number of simple web pages, it is unlikely that each one will take exactly the same amount of effort. Take the amount of effort needed to produce all of them and divide by the number of pages to get an average.

   How do you know what a reasonable effort assumption is? From your own experience and that of others who develop similar applications using a similar process and similar tools. The spreadsheet template provided with this class includes our initial assumptions, but do not take these as definitive. Instead collect metrics from your own project and adjust the standard levels of effort in future estimates to more accurately reflect your own experience.

   The Project Management class in this series discusses in detail how to gather metrics on actual effort expended.

Enter Feature Point Detail

6. Enter the deliverables in the appropriate category on the Feature Point Detail worksheet. Insert additional rows to accommodate all the deliverables.

   To insert additional rows do the following:

   a. Select an entire row within a category and copy it. (Do Ctrl+C or select Edit / Copy on the spreadsheet menu.)

   b. Select an entire row in the same category and insert the copied row above it. (Select Insert / Copied Cells on the spreadsheet menu.)

   Adding rows this way ensures that the formulas remain intact.

7. When you are done adding rows, be sure the formulas remain correct.

   - The formula in each Effort cell must point to the correct cells in the Effort Assumptions section.

   - The totals for each section must include all the rows in the section.

8. For each deliverable, enter the complexity factor in the Complexity column.

   Complexity indicates how complex the deliverable is. For example, a simple database table is one that has fewer than 20 columns, three or fewer constraints, and no triggers. If a table has more columns or more constraints or more triggers, increase its complexity factor to Medium or Complex

If you have several deliverables that have the same level of complexity, you can put them all on one line instead of listing each of them separately. Enter how many there are under Number. The formula in the Effort cells multiplies the number times the appropriate effort assumption. Be sure to make reference to a requirements document that does list each one separately.

For instance, if you have eight simple tables you could list each one separately with a Number of 1 or you could list them all on one line with a Number of 8.

9. If there are other tasks in the project that do not fit into this scheme, add a single line for each of them just above the Total All Features line. Make sure they are added in to the Total All Features.

   For instance, maybe you have to purchase and install hardware as part of the project or research and recommend an off-the-shelf component such as an accounting package. These tasks do not fit in neatly with software development phases, so add a line item for each one and put your estimate in the Total column. Make sure you add them into the Total All Features.

10. Review your percentage for Project Management.

    You must include funding for Project Management! If you do not manage the project, you have very little reason to believe it will succeed. We suggest 15% of the total, but you should set this to an amount you feel comfortable with.

    We have had clients that do not want to pay for project management. This class should give you some ammunition for convincing them that it is necessary. If they are still not convinced, then omit the Project Management line item, but increase the Standard Levels of Effort to include the extra percentage.

11. Review your percentage for Contingency.

    You should include a percentage for Contingency. You can be virtually certain that things will not go entirely as planned.

12. Review the spreadsheet for accuracy.

    Get someone to check all the formulas and make sure the numbers add up right. You can get in big trouble if you underbid a job because certain line items are not included in the total.

13. Do a sanity check on your numbers and get others to review the estimate.

    Ask yourself if the final figure is in the ballpark based on your gut feelings. See how it compares to similar projects you have done in the past. Get others to give you their opinions.

Appendix A – Estimation Spreadsheet by Phases

This is an abbreviated version of the spreadsheet used to estimate the effort needed by phase. It does not include all the phases of activity. See notes below.

Functional Analysis									
System Components	Difficulty	Volume	Complexity	Define Require-ments	Build Prototype	Build Final Version	Test	Install	Total
Database Tables									
Standard Level of Effort				0.5	1.5	0.0	0.5	0.5	3.0
Simple tables	1	8	8	4.0	12.0	0.0	4.0	4.0	24.0
More difficult tables	2	2	4	2.0	6.0	0.0	2.0	2.0	12.0
Total for Database Tables				6.0	18.0	0.0	6.0	6.0	36.0
Pages									
Standard Level of Effort				0.5	3.0	1.0	1.0	0.5	6.0
Main Menu	1	1	1	0.5	3.0	1.0	1.0	0.5	6.0
Update Products	4	1	4	2.0	12.0	4.0	4.0	2.0	24.0
Update Supplies	2	1	2	1.0	6.0	2.0	2.0	1.0	12.0
Update Customers	5	1	5	2.5	15.0	5.0	5.0	2.5	30.0
Total for Pages				6.0	36.0	12.0	12.0	6.0	72.0
Reports									
Standard Level of Effort				0.5	1.0	0.5	0.5	0.5	3.0
Summary Report	3	1	3	1.5	3.0	1.5	1.5	1.5	9.0
Products Report	1	1	1	0.5	1.0	0.5	0.5	0.5	3.0
Supplies Report	1	1	1	0.5	1.0	0.5	0.5	0.5	3.0
Customers Report	2	1	2	1.0	2.0	1.0	1.0	1.0	6.0
Total for Reports				3.5	7.0	3.5	3.5	3.5	21.0
Total for all functions				15.5	61.0	15.5	21.5	15.5	129.0
						Project Management		15%	19.35
						Contingency		25%	32.25
							Total Estimate		180.6

Notes to Estimation Spreadsheet

Functional elements such as tables, web pages and reports are listed down the left. Activities such as Analysis & Design and Final Test are listed across the top. Each intersection estimates the amount of effort needed to perform the activity on the element. You can list each functional element individually or you can lump all the elements of the same difficulty together. In the example above, web pages and reports are listed separately and tables are lumped together. For small projects it is good to list everything separately so you know you have not forgotten anything. For larger projects, this may be unwieldy and you may have to refer to a separate list of elements.

Standard Level of Effort is the amount of effort needed to perform the activity on one functional element of minimal difficulty. Include a page of definitions of "minimal difficulty." For example:

Data Tables	Fewer than 20 columns, three or fewer constraints.
Web pages	One table, fewer than 20 data items, minimal code behind the web page; includes stored procedures needed to facilitate data entry.
Reports	Tabular format, no calculations other than column totals; effort includes underlying query.

> **Note** – these Standard Levels of Effort are examples only. Do not blindly apply them to your project. Instead find records of actual effort from comparable projects or get several people to help you estimate.

Complexity is a function of how many elements there are of a certain type and how difficult they are. For instance, if there are two tables that are twice as difficult as a simple table, the complexity factor for them is four.

The individual cells are calculated as Complexity * Standard Level of Effort.

Always add Project Management at the bottom! Project management is absolutely crucial to completing the project on time and within budget.

If there are other things that are not easily listed in the function/activity format – for instance database administration or network capacity planning – add them at the bottom as well.

Always add a contingency. Nothing ever turns out exactly as planned.

This is a spreadsheet for small to medium projects. Large projects will have more complex methods of estimating effort. For instance, an element of difficulty n may not take exactly n times the standard level of effort for all phases.

Appendix B – Estimation Spreadsheet for a Simple Project

This is an abbreviated version of the spreadsheet used to estimate effort without splitting it out by phases. See notes below.

	Effort Assumptions		
	Simple	Medium	Complex
Data Tables	4	12	20
Web Pages	8	24	40
Reports	8.5	25	42

Data Tables	Complexity	Number	Effort
Table 1	S	1	4
Table 2	M	1	12
Table 3	C	1	20
Total Tables			36
Web Pages	Complexity	Number	Effort
Page 1	S	1	8
Page 2	M	1	24
Page 3	C	1	40
Total Pages			72
Reports	Complexity	Number	Effort
Report 1	S	1	8.5
Report 2	M	1	25
Report 3	C	1	42
Total Reports			75.5
Total All Features			184
Project Management		15%	28
Contingency		20%	37
Total Estimate			**248**
These estimates include design, code and test.			

Notes to Simple Estimation Spreadsheet

Functional elements such as tables, web pages and reports are listed down the left. Effort is estimated by element for the entire project, not broken out by phase. Each line lists the amount of effort needed to produce the element throughout all phases of the project.

Effort assumptions show the amount of effort needed to produce a simple, medium and complex functional element. The formulas in the spreadsheet plug in the correct number based on the code – S, M or C – in the Complexity column.

You can list each element separately with the Number set to 1 or you can lump several elements together on a line and put the number of elements in Number.

Always add Project Management at the bottom! Project management is absolutely crucial to completing the project on time and within budget.

If there are other things that are not easily listed in the function/activity format – for instance database administration or network capacity planning – add them at the bottom as well.

Always add a contingency. Nothing ever turns out exactly as planned.