Scenario:

“Your organisation has standardised on the Unified Modeling Language (UML) and you are using an iterative software development process.

You’re planning the first production iteration of your software development project and now it’s time to publish a firm delivery date.”

**A daunting task?**

Actually, a *simple* task using a standard toolkit:

- A **UML CASE Tool** such as Enterprise Architect, Sparx Systems - [www.sparxsystems.com.au](http://www.sparxsystems.com.au)

- **Circa**, Tassc - [www.tassc-solutions.com](http://www.tassc-solutions.com)

Here’s how I did it in 7 easy steps*....

*Keep this a secret – you won’t want to tell your colleagues it’s this easy!*
We are developing a custom software application to manage the process of classified advertising for our publishing business.

During the requirements gathering stage of our project Rachel and Susan, the business analysts on my team, worked with our customer representatives to identify and document the use cases. Working closely with our user sponsor they prioritised the application requirements. Now it's time to agree what will get delivered first. That's where I come in.

This is a medium-sized project so it will not be possible, or even desirable, to deliver all of the software in a single iteration. Using Circa at project inception, I produced a first-cut estimate for the development schedule that indicated an optimum overall project duration of around 80 days (4 months) could be achieved with a team of 3 to 4 developers, Figure 1.

### Producing early estimates with Circa:

Using Circa it is possible to produce first-cut estimates at the very early stages of a software project – earlier than by any other formal estimation technique.

Using the auto populate mechanism, an initial understanding of project scope can be entered into Circa. Simply enter the approximate number of subsystems, components, use cases and classes. They don't need any detailed analysis, and don't even need to be named at this stage.

From this summary scope information, the built-in productivity metrics within Circa can be used to derive an instant calculation of overall project effort. The Optimum Resource chart plots duration against the number of people working on the project and quickly identifies the optimum project duration and team size.

Clearly, at this early stage, an appropriate level of contingency should be built into the results. However, these first estimates can be invaluable input at the bid stage of a project or during cost benefit analysis to determine project feasibility.
John and Richard, my most experienced software engineers modelled the use cases in Enterprise Architect and identified the related high-level business classes. They simply analysed each use case description and extracted the core business concepts – the nouns of the problem domain.

It was not essential to understand exactly how these relate to each other as yet – this will become clearer during the analysis and design time allocated for the iterations. Figure 2 is an example of some of the use cases and high-level classes that now exist in Enterprise Architect.

![Figure 2. Use cases and high-level classes in Enterprise Architect](image)

I need to develop a detailed schedule for the first production iteration. I can easily and quickly import the use cases and classes from Enterprise Architect into Circa using XMI. In Enterprise Architect, I select the Model root node in the Project Browser, and choose ‘Export Model to XMI’ from the popup menu. Then I open Circa and select ‘Import XMI File from CASE Tool’ from the Software menu. I click on the ‘Browse’ button to locate the .xml file produced by Enterprise Architect, Figure 3. Then click the ‘Import’ button.

![Figure 3. Import scope from Enterprise Architect into Circa](image)
When the use cases and classes are imported they are created as concrete software artifacts and the built-in concrete metrics are applied (these are designed to be used at the detailed design stage of a project). This is fine for our use cases which are detailed and well-defined. However, our classes are just a first guess based on the business vocabulary. Therefore I can select all the classes and convert them to concept. Now the built-in concept metrics are applied (these are designed to be used with elements at the early analysis stage of a project).

This done, my first task is to identify those use cases and classes to be developed during this first iteration. As a general guideline I will base this on the pre-assigned priority levels and select those use cases that are critical to the application. I work with Rachel and in minutes we have recorded the users’ priority levels against each use case and class, Figure 4.

![Figure 4. Prioritise scope in Circa](image)

**Step 2**

**Account for size and complexity (allow 10 minutes)**

I can now qualify each of the use cases and classes. In discussion with Rachel and Susan, we quickly reach consensus on the levels of size and complexity for each use case and class, Figure 5.

My business analysts can give me invaluable insight from their understanding of the problem domain reached during the requirements gathering stage of the project, and documented in the use case descriptions. By adjusting effort based on these qualifier levels Circa will be able to calculate a more accurate estimate.

![Figure 5. Qualifying the project scope in Circa](image)
For now, I simply set the level of reuse and generlicity to medium. Once my software engineers get more involved in the high level design they will be able to shed some light on the level of reuse we can reasonably expect to achieve in this project, and I can further refine my estimates.

### How to assess qualifiers:

Assessing the levels for setting qualifiers is more subjective than enumerating the scope of a project. However, even during early requirements analysis it is possible to refine estimates based on qualifier information. The following guidelines provide ‘rules of thumb’ for choosing between the levels during the requirements gathering stage of the project.

When setting size for use cases, look at the length of the textual description, or if it has yet to be written, simply think how much would have to be written to comprehensively document this business activity:

<table>
<thead>
<tr>
<th>Qualifier</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>a couple of lines</td>
<td>tiny</td>
</tr>
<tr>
<td>a short paragraph</td>
<td>small</td>
</tr>
<tr>
<td>a couple of paragraphs</td>
<td>medium</td>
</tr>
<tr>
<td>a page</td>
<td>large</td>
</tr>
<tr>
<td>several pages</td>
<td>huge</td>
</tr>
</tbody>
</table>

When setting complexity for use cases, think about the number of decision points in the use case description, the number of business rules to be enforced and the number of exceptions or alternate paths to be handled:

<table>
<thead>
<tr>
<th>Qualifier</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>basic sequential steps</td>
<td>trivial</td>
</tr>
<tr>
<td>a single decision, rule or exception</td>
<td>simple</td>
</tr>
<tr>
<td>a couple of decisions, rules or exceptions</td>
<td>medium</td>
</tr>
<tr>
<td>several decisions, rules or exceptions</td>
<td>difficult</td>
</tr>
<tr>
<td>many decisions, rules and exceptions</td>
<td>complex</td>
</tr>
</tbody>
</table>

When setting size for classes, consider the amount of application data that needs to be held to adequately model the concept:

<table>
<thead>
<tr>
<th>Qualifier</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 to 3 attributes</td>
<td>tiny</td>
</tr>
<tr>
<td>4 to 6 attributes</td>
<td>small</td>
</tr>
<tr>
<td>7 to 9 attributes</td>
<td>medium</td>
</tr>
<tr>
<td>10 to 12 attributes</td>
<td>large</td>
</tr>
<tr>
<td>13 or more attributes</td>
<td>huge</td>
</tr>
</tbody>
</table>

When setting the complexity for classes, consider algorithms required to process application data and business rules that need to be encapsulated:

<table>
<thead>
<tr>
<th>Qualifier</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>no algorithms or rules</td>
<td>trivial</td>
</tr>
<tr>
<td>a single algorithm or rule</td>
<td>simple</td>
</tr>
<tr>
<td>a couple of algorithms or rules</td>
<td>medium</td>
</tr>
<tr>
<td>several algorithms or rules</td>
<td>difficult</td>
</tr>
<tr>
<td>many algorithms and rules</td>
<td>complex</td>
</tr>
</tbody>
</table>
Step 3  Apply technology metrics (allow 5 minutes)

Circa allows you to define technologies and then indicate the impact of specific technologies on each of the development activities. We are developing in Java, so I can specify Java as a technology in Circa to account for the impact of this programming language choice. I create a new technology metric for Java and indicate the impact on development activities — in my assessment a 20% improvement in coding efficiency, as well as a small improvement in testing and integration. I then select all the classes and assign Java as their technology, Figure 6.

The impact of technology:

Contrary to popular belief, technology typically has less of an impact on development effort and project duration than other factors, such as the size and complexity of project scope, and the skill level of developers. Technology metrics can be defined to reflect the technology choices on the project and to specify a delta impact on various development activities. For example, a choice of programming language may impact build, integration and testing depending on the richness of built-in class libraries. Likewise, the use of a UML CASE tool such as Enterprise Architect should have a positive impact on analysis, design and review activities.

Figure 6. Creating and applying a technology metric in Circa

Step 4  Plan the production activities (allow 10 minutes)

Like any system development project, deploying a new application involves a great deal more than simply writing the code. I need to account for other non-software engineering activities, such as developing an on-line help system as well as some end user documentation and training manuals, Figure 7. These activities will be assigned to our technical author, Tom.

Figure 7. Defining production artifacts in Circa
My job is to keep our customer happy. I will assist our business analysts in building a business case for future application requirements to secure future funding for the project. This activity will run in parallel with the software build during the first iteration. We will need to schedule time to perform user interviews and document feedback.

I will also assign a small team of engineers, probably just Jane and Mark, to work on a GUI prototype to illustrate some of system’s future capabilities. This will allow my core team to focus, and make progress with the first real iteration in relative isolation.

### Accounting for non-software engineering activities:

A software development project is a production that involves the services of a variety of specialists - in addition to the teams involved in the construction of the software. Non-software engineering tasks are often specialist in nature and may require a significant degree of effort or incur a monetary cost. In order to obtain an accurate estimate for an entire project, these production artifacts need to be accounted for, alongside the software construction activities. Production artifacts typically include development of multimedia content, on-line help, user manuals, training materials, business documents, proposals and reports.

### Step 5

**Profile my development team (allow 10 minutes)**

Now I can enter my development team details. I have to be able to account for various differences in my staff. Richard is an expert but William has recently joined us and is still coming up to speed. Most of my staff work full-time, whereas I am expected to manage a number teams and only dedicate about 40% of my time to this project. This information is critical in determining an accurate estimate of project duration and cost.

Furthermore, how I organise my people into teams will affect the level of communication and management overhead. Fortunately this complex computation has been taken care of by Circa. I plan to have 4 specialist teams:

- Help team: Tom, technical author
- Business team, myself, project manager; Rachel and Susan, business analysts
- Prototype team: Jane and Mark, software engineers
- Software team: Richard, William and Alison, software engineers

All I have to do is to enter the characteristics of each person and Circa will do the rest, Figure 8. It can even plot an Optimum Resource chart, which illustrates the optimum number of developers I need in order to deliver in the shortest possible duration. This also shows me whether I am over or under resourced on the project.

![Figure 8. Profiling the project team in Circa](image-url)
Calculating project duration:

The team size is a significant factor in establishing project duration. Software development remains a labour intensive activity. Where there is a certain amount of effort required, clearly two or more developers will complete the task more quickly than one. Team members can co-operate and delegate activities enabling the team to perform better and more efficiently than any individual.

This is true to up to a point. As the team size is increased, the communication levels also increase and channels of communication become more formal. A small team can communicate on a fairly informal basis whereas a large team will require more formal documentation and regular formal meetings. Therefore adding additional engineers will decrease duration until the inter-personal communication and management overhead outweighs the advantage of further partitioning the work.

Step 6  Scope software for the first iteration (allow 10 minutes)

I now need to scope the software for the first iteration. I want to focus on what the team can achieve within a 10-week time-box. To gauge how long I need for core software development I can quickly and easily temporarily exclude all other teams and all production artifacts from my working estimate. I then begin a process of elimination by selectively excluding low priority software artifacts until the duration that Circa calculates for software construction is within my parameters, Figure 9.

Scope definition and stability:

Incremental development actively encourages participation from the user community and it is reasonable to expect that they will bring fresh ideas to the project. They may suggest additional requirements or influence development priorities. The management challenge is to control the stability of the project and measure and account for the impact of change requests. Where scope specification changes as a result of new ideas and requirements, estimates can be produced to illustrate the impact of these changes. Therefore, if and when the scope of the project ‘creeps’, the customer will understand and be able to accept the impact on project schedules and costs.
Step 7  
Schedule the First Development Iteration (allow 10 minutes)

I can now create the first ‘core features’ iteration within the first phase of my project. In this iteration I can define tasks that allow me to assign workers to specific activities. I plan to have 4 tasks: Business Case task assigned to the business team workers, Prototype task assigned to the prototype team workers, Development task assigned to the software team workers and Help task assigned to my technical author.

I re-include all the software and production artifacts, and begin populating the tasks with software, production and worker information. I plan to start the Development task immediately – I can mark it as ‘in progress’, indicated by the yellow folder in Figure 10. I leave the other tasks as ‘planned’, indicated by the red folders.

As tasks are populated with appropriate software, production and worker information, Circa dynamically calculates estimates of effort, duration and cost for each task, which are combined into estimates for the iteration. The project plan tells me that the shortest possible duration of the iteration depends on the Development task – this task is initially on the critical path as it is the longest.

Using the Gantt chart I can create a dependency between the Business Case task ending and the Prototype starting, Figure 11. Circa calculates that if the iteration started today, August 28th, the earliest possible delivery date is October 26th – around 9 weeks elapsed. Perfect!