Chapter 2
Example Modeling and Forecasting Scenario

This scenario is for a hypothetical project that aims to re-launch a website. It demonstrates the thinking process and practical implementation of using modeling to quickly forecast staffing levels, go-live dates, and how to interact with senior management to get good decisions made. At the moment, you may not understand all of the terminology and the software used to generate the reports; we cover this material later in the book. This scenario is one of any number of ways of estimating, modeling and forecasting a software project. It's meant to spark your creativity.

The Cast

You – You have the job of managing a team to produce and re-launch the next generation website that will save "The Company."

Charlie – Your CTO. He holds the keys to the budget for staffing requests.

Boris – Your business owner. He has the grand vision and the ear of the CEO.

Your Team – A group of Graphic designers, web developers, server-side developers and testers. At the moment, the exact number of resources you need isn't clear. You have a core group already consisting of -

1 – Graphic Designer
2 – Web Developers
3 – Java Developers
1 - QA Engineer

Forecasting Staff and Dates – Getting the Go-Ahead

Your project is one of three options for next year's budget the CEO is considering. He likes all of them and has asked for a better view on the cost of development and the delivery forecasts. The CEO wants a new website to go live before the end of the calendar year. Rumors are your major competitor has a new site to launch end of Q1 next year, and the CEO wants to beat them to it. He also wants to limit investment to $250K in labor costs.

You need to get a cost and delivery date forecast quickly to answer these questions. The first step is to gather the information needed without disrupting the development teams current project. You need to quickly generate the following numbers –

1. Staffing estimate in order to hit at least a partial deliverable by end of calendar year
2. Cost estimate to deliver (forecast)
3. Probability of hitting certain dates (forecast)

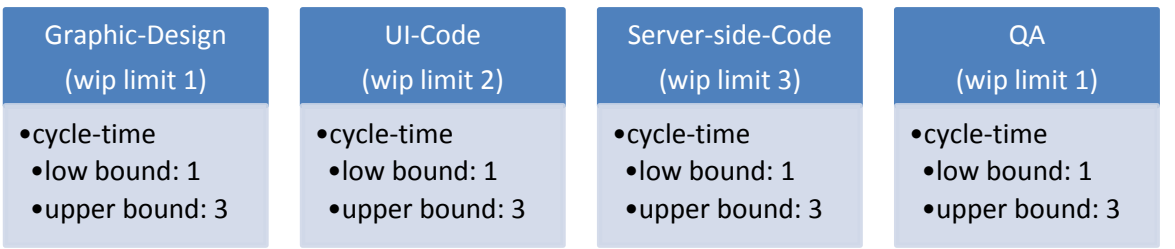
Note:

Appendix A is listing of the full SimML model file.

Act 1 – Backlog segmentation and Kanban Board Design

To begin, you need a backlog of work to model and estimate. Reviewing the initial backlog of 31 stories you have been given, and roughly modeling defects and blocking events using occurrence data from prior projects, you run a quick forecast date simulation with a very basic Kanban board definition and cycle-time estimates (as shown in Figure 2-1). You want to see how close to January 1st the most basic of simulation model gets you.

Figure 2-1
Basic starting point for Kanban board and estimation.



You execute a forecast date simulation run for 31 backlog stories. Listing 2-1 shows the SimML code that performs a Monte-carlo simulation and extrapolates the completion date, and likelihood percentage of hitting that date, and a simple cost estimate (simple estimate ($100K / 52 / 5) x number of staff, now 7). The results of this simulation are shown in Output 2-1.

Listing 2-1

SimML command to run a forecast date simulation.

```
<execute  deliverables="Must-Haves"  dateFormat="ddMMMyyyy">
    <forecastDate  startDate="01Oct2011" intervalsToOneDay="1"
     workDays="monday,tuesday,wednesday,thursday,friday"
     costPerDay="2700" />
</execute>
```

Output 2-1 clearly shows you have your work cut-out for you to deliver by the beginning of the year, and the cost is high, your target is around $250K. At the moment, you only feel confident in saying March 1st, which is approximately your 95th percentile result.

Output 2-1

Most basic simulation of 31 stories using default column cycle-time.

```
<forecastDate startDate="01Oct2011" intervalsToOneDay="1"
      workdays="monday,tuesday,wednesday,thursday,friday"
      costPerDay="2700"  >
  <dates>
    <date intervals="96" date="13Feb2012"
          likelihood="1.60 %" cost="$259,200.00" />
    …… removed from brevity ……
    <date intervals="108" date="29Feb2012"
          likelihood="92.00 %" cost="$291,600.00" />
    <date intervals="109" date="01Mar2012"
          likelihood="94.80 %" cost="$294,300.00" />
    <date intervals="110" date="02Mar2012"
          likelihood="98.40 %" cost="$297,000.00" />
    <date intervals="111" date="05Mar2012"
          likelihood="99.20 %" cost="$299,700.00" />
    <date intervals="112" date="06Mar2012"
          likelihood="99.60 %" cost="$302,400.00" />
    <date intervals="113" date="07Mar2012"
          likelihood="100.00 %" cost="$305,100.00" />
  </dates>
</forecastDate>
```

You work with Boris, and segment the backlog into two deliverables: Must-Haves, and Everything-Remaining. You then work with your developers and partition each group further. You make a fast pass through the stories and put them into one of four categories-

1. *Small*: Those stories that are possibly under a days work, but certainly less than 2 days work in each column.
2. *Medium*: Those stories that are more than 2 days work but less than 3 days.
3. *UI-Intensive*: Those stories where lots of graphics design and UI coding is necessary.
4. *Server-Side-Intensive*: Those stories where lots of Java service (or .NET, we are all friends here) code is required.

Table 2-1
Estimates grouped by deliverable and then by size and skill specialty.

Delivery Group	
Must-haves	

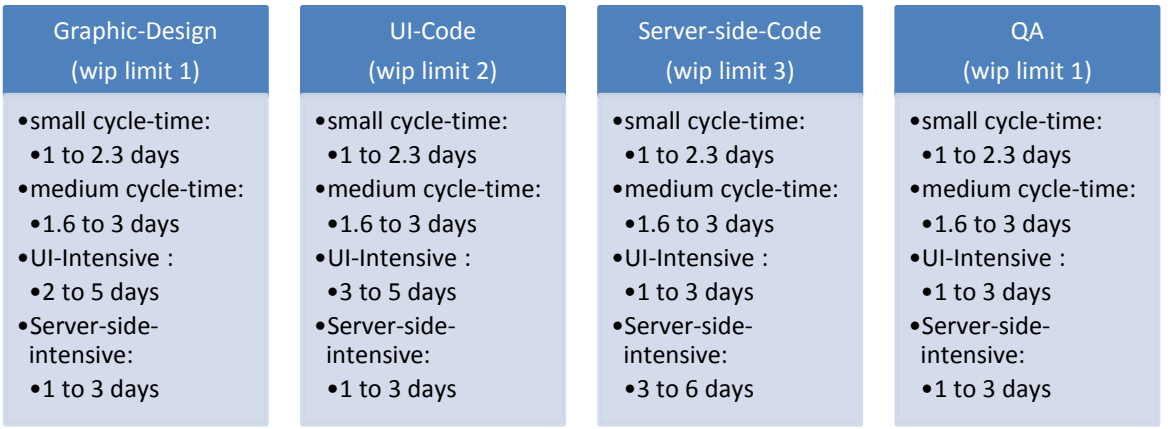
Developer Estimate Group	**Number of Stories**
Small	6
Medium	4
UI-Intensive	5
Server-side Intensive	6

Everything-remaining	Developer Estimate Group	Number of Stories
	Small	2
	Medium	3
	UI-Intensive	2
	Server-side Intensive	3

Act 2 – Cycle-time estimates

For each of the story categories you ask the development team to come up with 90th percentile ranges for cycle time, meaning the cycle-time they feel the actual cycle-time is between the range they give (5% below, 5% above). The final model for Kanban columns and the cycle-times is shown in Figure 2-2.

Figure 2-2
Final Kanban columns, Wip limits and cycle-time estimates for the backlog groups



With this story partitioning, and estimated cycle-times for each category of story in each Kanban column, you look at the forecast for just the "Must-Have" group.

```
<date intervals="83" date="25Jan2012"
      likelihood="94.80 %" cost="$224,100.00" />
<date intervals="84" date="26Jan2012"
      likelihood="96.80 %" cost="$226,800.00" />
```

Act 3 – Staff tuning to bring in the date

You are still beyond the target date needed to get a go-ahead. You must find what extra staff will bring that date into range. You perform a staff sensitivity simulation on the model (called the `addStaff` command in SimML) and find what it recommends. Listing 2-2 shows the SimML command to determine what the best three column WIP limit changes have the most impact.

Listing 2-2

SimML command to find what staff additions make the most impact to delivery time. In this case, 3 staff recommendations will be made, and every Kanban column can be changed.

```
<addStaff count="3" cycles="250" />
```

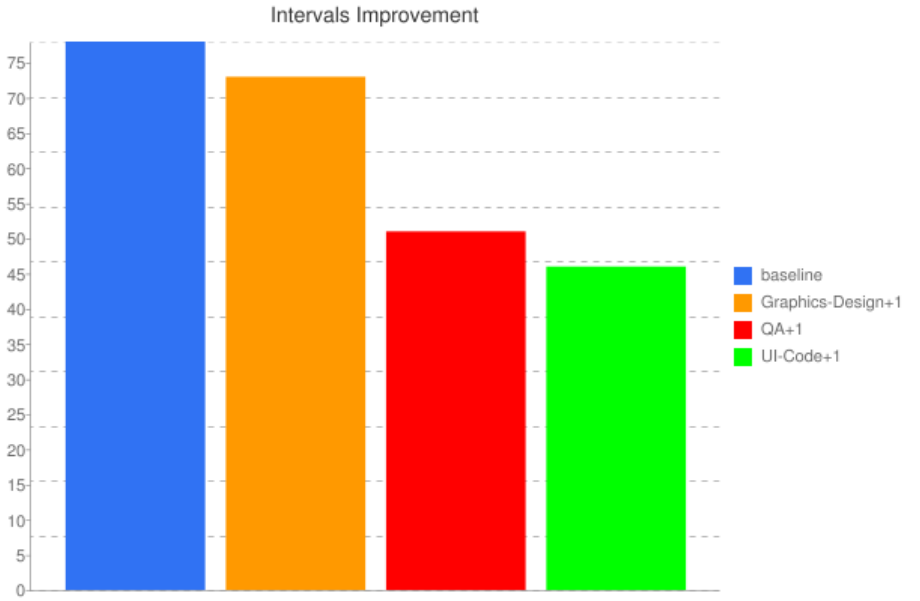
The `addStaff` simulation makes three recommendations, add a Graphics Designer, a QA engineer and then a UI Coder in that order. The recommendations are cumulative, and you notice that there is only a marginal benefit in adding the UI coder which is an expensive resource, so you decide not to add that resource. This simulation expected a 34% improvement (reduction) in intervals (days for this simulation).

Output 2-2

Results table for staff simulation. Improvements are cumulative, and improvement is over the original baseline.

WIP Suggestion	Graphics-Design by 1	the QA by 1	then UI-Code by 1
Original WIP	1	1	2
New WIP	2	2	3
Intervals to completion (original 78 intervals)	72	51	47
Cumulative Interval Improvement	7.3%	34.95%	40.68%

Figure 2-3
Simulation chart for recommendations of adding 3 people. Days to completion drop from 78, to 72, to 51, to 47.



Act 4 – Success; Getting to go-ahead

With the results from Output 2-2 and Figure 2-3 you re-run the forecast date simulation with increased column WIP's for Graphics Design and QA. In order to forecast the extra cost, you must increase the per day cost burn rate by 2 staff as well (using the same formula as before ($100K / 52 / 5) x number of staff, now 9). You present your findings to Charlie (the CTO) and get the approval to borrow a designer and add a QA engineer if your project gets the go-ahead.

Listing 2-3

SimML command for forecasting completion date with extra resources, and an increase in per-day cost to account for funding those resources.

```
<forecastDate cycles="250" startDate="01Oct2011"
    intervalsToOneDay="1"
    workDays="monday,tuesday,wednesday,thursday,friday"
    costPerDay="3461" />
```

By simulating the SimML command as shown in Listing 2-3, it is now possible to hit the end-of-year target with just the Must-Haves (a scenario Boris supports) with a very high degree of certainty as shown in Output 2-3.

Output 2-3

Forecast date estimate with the extra staff accounted for. This is for the Must-Haves ONLY. You can deliver before end-of year.

```
<date intervals="54" date="15Dec2011"
      likelihood="94.00 %" cost="$186,894.00" />
<date intervals="55" date="16Dec2011"
      likelihood="95.60 %" cost="$190,355.00" />
<date intervals="56" date="19Dec2011"
      likelihood="98.40 %" cost="$193,816.00" />
<date intervals="57" date="20Dec2011"
      likelihood="100.00 %" cost="$197,277.00" />
```

To deliver the entire project, you also feel confident in delivering by mid-January, and the total cost of very close to the $250K investment target as shown in Output 2-4.

Output 2-4

Forecast date estimate with the extra staff accounted for. This is for the entire project. Cost is now in-line with expectations.

```
<date intervals="72" date="10Jan2012"
      likelihood="94.00 %" cost="$249,192.00" />
<date intervals="73" date="11Jan2012"
      likelihood="97.20 %" cost="$252,653.00" />
<date intervals="74" date="12Jan2012"
      likelihood="98.80 %" cost="$256,114.00" />
<date intervals="75" date="13Jan2012"
      likelihood="100.00 %" cost="$259,575.00" />
```

Boris and Charlie present the business case to the CEO. The CEO is pleasantly surprised that not only did you hit the targets he asked for; he can see that there was method to the calculations and trusts that you can manage to those numbers. Your project gets the go ahead.

From the developers' perspective, they gave a total of 4 estimates of cycle-time, and a single pass through the backlog categorizing the work. They thank you for not having to estimate all 31 stories, and then re-estimate them again when the target wasn't achieved.

Hitting a Date Promised

The project has now been underway, and ten stories have been completed. It's time to make sure that your project is tracking to the model by comparing actual data versus the model's estimates.

Act 5 – Comparing Actual versus modeled

After the first month and a half, actual data can and should be used to refine the cycle-time estimates. The original estimates were very wide in order to capture the group's consensus quickly on what the 90th percentile range would be. Now with a number of stories completed, with a variety of things that went well, and things that went wrong, a narrower range is likely to emerge.

The first step of analyzing actual data is to confirm that the actuals fell within the ranges estimated for cycle-time. Table 2-1 lists the cycle times estimated by the developers, and as you can see in Table 2-2 there were three actuals that fell outside of that range. A 5% variance above or below the range expected for 40 samples would be 2 stories above and 2 stories below; the actuals are showing 2 cards above, which is on the line, but within the bounds, and 1 card below, which is also below the 5% limit. It is too early to be making changes, but keeping an eye on the Medium category cycle-time is in order, another cycle-time above the limit would mean you should increase the cycle-times for simulation.

Table 2-2

Actual column cycle-times for the project after the first 10 stories. 3 cells fell outside of the range estimated.

			Cycle-Time Actuals			
Story	**Delivery Group**	**Dev. Estimate Group**	**Graphics Design**	**UI Code**	**Server Side Code**	**QA**
Story 1	Must-Haves	Small	2.21	0.97	1.22	2.13
Story 2	Must-Haves	UI-Intensive	4.28	4.11	1.23	1.31
Story 3	Must-Haves	Medium	2.26	**0.89**	**1.39**	1.73
Story 4	Must-Haves	UI-Intensive	2.47	3.77	1.22	1.38
Story 5	Must-Haves	Server-Side-Intensive	1.84	1.99	5.30	1.93
Story 6	Must-Haves	Small	1.34	1.24	1.34	1.44
Story 7	Must-Haves	Server-Side-Intensive	1.99	1.85	5.08	2.36
Story 8	Must-Haves	Small	1.78	**3.08**	1.77	1.63
Story 9	Must-Haves	UI-Intensive	3.91	3.24	1.63	1.50
Story 10	Must-Haves	Medium	2.37	1.66	1.83	1.71

Defect rates and blocking event rates do not look as promising. The original estimates in the model compared to actual are shown in Table 2-3. UI Defects are occurring every story. This is twice the rate expected, and is impacting the delivery date.

Table 2-3

Defect and blocking event estimate versus actual occurrence rates after 10 stories.

	Occurrence Estimate	Occurrence Actual
UI Defect	1 in 2 to 3 stories	**1 in 1 (10 so far out of 10 stories)**
Server-side Defect	1 in 3 to 6 stories	1 in 5 (2 so far out of 10 stories)
Block Spec Question	1 in 5 to 10 stories	1 in 3 (3 so far out of 10 stories)
Block Testing Environment	1 in 4 to 8 stories	1 in 5 (2 so far out of 10 stories)

Act 6 – Rallying the Team to Reduce Defect Rates

When you initially modeled your project, you took the defect rates and the blocking event rates from a previous project. You have identified by comparing actual data on this project that one of your defect types, specifically UI related defects is occurring more often than you modeled.

Listing 2-4

SimML model for the UI Defect definition where actual occurrence is once in every story.

```
<!-- Originally modeled UI defects, found in QA, fixed in UI Dev-->
<defect columnId="4" startsInColumnId="2"
        occurrenceLowBound="2" occurrenceHighBound="3">UI Defect
</defect>
<!-- New model based on actual occurrence rates -->
<defect columnId="4" startsInColumnId="2"
        occurrenceLowBound="1" occurrenceHighBound="1">UI Defect
</defect>
```

You update your model definition for this defect from once in every 2 to 3 stories to once every 1 story, matching the actual measured occurrence rate you are seeing from the project. Running a date forecast simulation using this new definition for the Must Haves delivery group shows you will miss the end of year target as shown in Output 2-5.

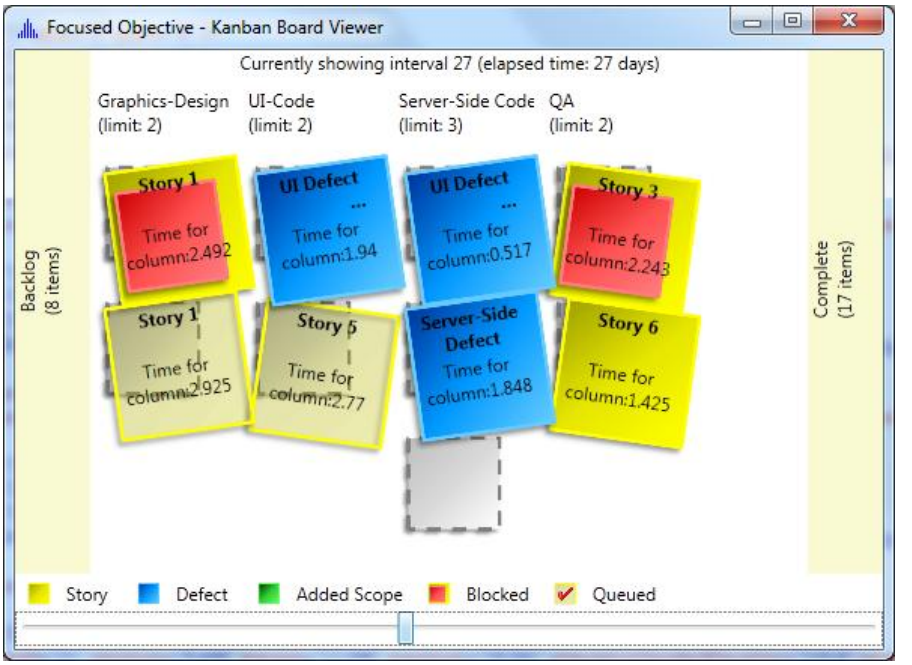
Output 2-5

New date forecast showing the impact of the UI Defects.

```
<date intervals="66" date="02Jan2012"
      likelihood="93.20 %" cost="$228,426.00" />
<date intervals="67" date="03Jan2012"
      likelihood="97.60 %" cost="$231,887.00" />
<date intervals="68" date="04Jan2012"
      likelihood="98.40 %" cost="$235,348.00" />
<date intervals="69" date="05Jan2012"
      likelihood="99.60 %" cost="$238,809.00" />
<date intervals="71" date="09Jan2012"
      likelihood="100.00 %" cost="$245,731.00" />
```

You pull your team together and explain the observation of UI defects. You share the new date forecast and show them the visual video simulation of the Kanban flow as seen in Figure 2-4.

Figure 2-4
Video screenshot of simulation showing the team the impact of the defects on throughput.



You then demonstrate that adding a **full day to the cycle-time of every story in the UI-Code** column has less impact than the UI Defects currently are causing (as shown in Output 2-6).

Output 2-6

Impacts of increasing the cycle-time for UI-Dev by 1 day (lower and upper bounds). Less than the impact of UI Defects alone!

```
<date intervals="62" date="27Dec2011"
      likelihood="94.40 %" cost="$214,582.00" />
<date intervals="63" date="28Dec2011"
      likelihood="96.80 %" cost="$218,043.00" />
<date intervals="64" date="29Dec2011"
      likelihood="99.60 %" cost="$221,504.00" />
<date intervals="66" date="02Jan2012"
      likelihood="100.00 %" cost="$228,426.00" />
```

On seeing and understanding the impact of these defects, the team brainstorms ideas on how to bring the occurrence rate back into the once every 2 to 3 stories. Their ideas of giving the business owner (Boris) a demonstration and feedback before checking in, getting a peer-review, and spending an extra half-day testing the stories themselves are effective and occurrence rates return to those expected over the next few stories.

> **Note**
>
> Being able to show that working slower and more carefully (by not allowing defects to be raised in later columns) WILL reduce the

overall project time is a key lesson. Although most of this scenario is contrived, this lesson isn't – spend an extra half a day testing during development, as this case shows, even if you spent a day testing you would still be better off than allowing the UI defect rate be high.

Act 7 - What if we added staff now?

Charlie the CTO in your weekly meeting asks you if you want more staff to make sure you reach the promised delivery date. He offers as many developers as you need. You perform an add staff simulation (results shown in Output 2-7), and demonstrate that whilst adding one UI Developer would be the most benefit right now, an additional Graphics Designer and QA Engineer are then next in line most beneficial to throughput.

Output 2-7

The results of an addStaff SimML command showing the next three staff additions that have most impact on reducing days of work.

```
<wipSuggestion  column="UI-Code" originalWip="2" newWip="3"
                intervalImprovement="9.45" >
<wipSuggestion  column="Graphics-Design" originalWip="2" newWip="3"
                intervalImprovement="14.78">
<wipSuggestion  column="QA" originalWip="2" newWip="3"
                intervalImprovement="22.26" >
```

He assigns another UI Developer to your team, and lets you borrow a graphic designer from another team who is ramping down on another project.

Note

I've been around a few projects that underwent scrutiny for being late. The upper-management reflex was to offer and add developers, but it rarely works. The ability to show that just adding developers without increasing the capacity surrounding them is a waste of time is invaluable.

Act 8 - Impact (Sensitivity) Analysis

You want to see what most impacts the delivery date; is it the defects or the blocking events? Your graphics designer is complaining about the business being slow to respond to questions. You want to see if that would impact your delivery date by a meaningful amount. You perform a sensitivity analysis on the SimML model.

A sensitivity report gives you an ordered list of what input factors of the model change the delivery date the most. When managing a project it's good to know what factors to go-after and solve next, and a sensitivity reports makes your next opportunity clear. Listing 2-5 shows how to perform a sensitivity simulation.

Listing 2-5

SimML command to execute a sensitivity report. Returns an ordered list of what most influences the output.

```
<sensitivity cycles="250" sortOrder="descending"
estimateMultiplier="0.5" occurenceMultiplier="2"  />
```

The sensitivity analysis is enlightening (Table 2-4) for the development team. It shows that even if the occurrence rate for the specification questions was halved, it would make less than a single day difference to the final outcome. UI Defects remains the highest impacting defect. The development team stops complaining about the lack of response to questions and you get to re-iterate the importance of limiting the UI Defect occurrence rate.

Table 2-4
The results from executing a sensitivity analysis on the model

	UI Defect	Spec question (awaiting answer)	Spec question (awaiting answer)	Block testing (environment down)	Block testing (environment down)	Server-Side Defect
Object Type	Defect	BlockingEvent	BlockingEvent	BlockingEvent	BlockingEvent	Defect
Change Type	Occurence	Estimate	Occurrence	Estimate	Occurence	Occurence
Interval Delta	-3.344	-0.812	-0.744	-0.468	-0.328	-0.248

Project Retrospective

Although it was tight, the addition of a UI coder, a Graphics Designer and a QA Engineer helped quickly resolve some last minute defects and change of scope that Boris and his team wanted after seeing the completed site. The Must-Have features were delivered before the New Year, in mid-December. The Everything-Else delivery group was completed early in the New Year and the extra features helped raise considerable revenue leading up to and over the Christmas and New Year season.

From your perspective, you felt you always had a clear picture of what needed to be managed, and early indications of issues that might risk delivery date promises. Charlie, your boss was also impressed that for every meeting you came prepared with a list of impacting events (a sensitivity report), and knew what staff additions would have the most impact.

Summary

Although this was just a fictional scenario (or was it?) you can quickly see how the use of a model and simulation tools can quickly give you the ability to understand impact of various what-if questions, and the confidence to ask for what you need.