Kaleidoscope Training System

Basis of Estimate-

Reassessment

Version 1.0

Release Date: 06 July 2020

CDRL: NA

Prepared for: Kaleidoscope Case Study, JHU 605.60.81.SU20

Authors:

Brian Hoelzer

Emily List

Brian Loughran

Jason Zhao

# Task a:

We still agree with our original choice for using the Organic Mode and the Detailed Model. Our main improvement with this reassessment was to break out the effort multipliers for each CSCI for each of the three builds. This allowed us to build a more detailed estimate, the primary benefit of using the Detailed Model.

Justification for Organic Mode

Organic uses a small team in a familiar environment and with a familiar application. This project contains two subsystems, a PC CSCI and a SUN CSCI running on Windows XP and Unix environments respectfully, and all programming is being done in C++. The expectation is that even junior members of the team would be familiar with this sort of development environment. The project is to create a training system, where groups of students work through units of training material over the course of a week. So, the team is working to a set of less-than-rigid requirements, e.g., data reduction, scientific models, business models, simple inventory control, or simple production control.

Nothing from the SOW indicates that either subsystem would have any embedded mode characteristics. The system seems straight forward, like several other training systems, and does not seem to have complex command and control structures.

Justification for Detailed Model

The Detailed Model adds the phases of the project and subsystems to use the multipliers in the cost driver equations for each phase of project and/or for each subsystem. This model makes the most sense for this project since there are 3 builds, each including the two subsystems. By using the Detailed Model, the team was able to more accurately determine the schedule for each of the builds.

# 

# Task b:

## SUN CSCI

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Attributes** | **Build 1** | **Build 2** | **Build 3** | **Justification** |
| RELY | Nominal | Nominal | Nominal | The complexity of the OS would make losses recoverable but not easily recoverable |
| DATA | High | High | Very High | Builds 0 & 1: Archive management and report generation could produce databases greater than 5MB  Build 2: Very high due to statistical analysis |
| CPLX | Nominal | Nominal | Nominal | Nominal control operations, low computational operations and device-dependent operations, and very low data management operations. The highest rating is nominal |
| TIME | Nominal | Nominal | Nominal | Nothing in the SOW indicates that any of the operations would be particularly time intensive, so absent any other information- assume the average |
| STOR | Nominal | Nominal | Nominal | Nothing in the SOW indicates that any of the operations would be particularly storage intensive, so absent any other information- assume the average |
| VIRT | High | Nominal | Nominal | Build 0: Most major changes to the VM will happen during the early stages of coding while the VM is being developed  Build 1 & 2: Assuming one major change to the VM once per build, or about 6 months |
| TURN | Low | Low | Low | Interactive system |
| ACAP | Low | Low | Nominal | Build 0 & 1: A more complicated OS that is less familiar means that the team would likely have a lower level of analyst capability at the end of the project  Build 2: After 18 months of the project you would assume the team would get more proficient |
| AEXP | Nominal | Nominal | High | Build 0 & 1: Assuming at least 3 years of experience with the application domain  Build 2: After a year and a half average experience for the team may have gone above 6 years |
| PCAP | Low | Low | Nominal | Build 0 & 1: A more complicated OS that is less familiar means that the team would likely have a lower level of analyst capability at the end of the project  Build 2: After 18 months of the project you would assume the team would get more efficient |
| VEXP | Nominal | Nominal | Nominal | Assuming at least one year of experience working with VM |
| LEXP | High | High | High | Assuming at least 3 years of experience with C++. Given that C++ is a common language for degree course work, even new graduates would be familiar with the language |
| MODP | High | High | High | General use of modern programming practices |
| TOOL | Nominal | Nominal | Nominal | Strong use of small tools and basic use of large tools |
| SCED | Nominal | Nominal | Nominal | Assuming familiarity with theyse types of training systems, nothing seems like the system should be particularly difficult or easy so assuming that it should take the average amount of time to deliver |

*Table 1: SUN COCOMO Parameters Justifications*

## PC CSCI

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Attributes** | **Build 1** | **Build 2** | **Build 3** | **Justification** |
| RELY | Nominal | Low | Low | Build 0: Loss of registration module brings the reliability up to nominal  Build 1 & 2: Loss of the system is easily recoverable |
| DATA | Low | Low | Low | Any databases would be small, containing data like students scores and progress, which would only need to be saved for a single secession |
| CPLX | Low | Low | Low | Low control operations, low computational operations and device-dependent operations, and very low data management operations. The highest rating is nominal |
| TIME | Nominal | Nominal | Nominal | Nothing in the SOW indicates that any of the operations would be particularly time intensive, so absent any other information- assume the average |
| STOR | Nominal | Nominal | Nominal | Nothing in the SOW indicates that any of the operations would be particularly storage intensive, so absent any other information- assume the average |
| VIRT | High | Nominal | Nominal | Build 0: Most major changes to the VM will happen during the early stages of coding while the VM is being developed  Build 1 & 2: Assuming one major change to the VM once per build, or about 6 months |
| TURN | Low | Low | Low | Interactive system |
| ACAP | Nominal | Nominal | high | Build 0 & 1: A fairly common OS, with no further information we would assume the team is at least at an average level of capability  Build 2: After 18 months of the project you would assume the team would get more proficient |
| AEXP | Nominal | Nominal | High | Build 0 & 1: Assuming at least 3 years of experience with the application domain  Build 2: After 18 months, average experience with a very common system for the team may go above 6 years |
| PCAP | Nominal | Nominal | High | Build 0 & 1: A fairly common OS, with no further information we would assume the team is at least at an average level of capability  Build 2: After 18 months of the project you would assume the team would get more proficient |
| VEXP | Nominal | Nominal | Nominal | Assuming at least one year of experience working with VM |
| LEXP | High | High | High | Assuming at least 3 years of experience with C++. Given that C++ is a common language for degree course work, even new graduates would be familiar with the language |
| MODP | High | High | High | General use of modern programming practices |
| TOOL | Nominal | Nominal | Nominal | Strong use of small tools and basic use of large tools |
| SCED | Nominal | Nominal | Nominal | Assuming familiarity with theyse types of training systems, nothing seems like the system should be particularly difficult or easy so assuming that it should take the average amount of time to deliver |

*Table 2: PC COCOMO Parameters Justifications*

# Task c:

Based on the above assumptions of COCOMO parameters, we can map the effort multiplier ratings into the effort multiplier chart. Each selected value is highlighted and the effort multiplier value is shown in the rightmost column. This exercise is performed for each of the builds and platforms.

Build 0

SUN Effort Multipliers

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Attributes** | **Very Low** | **Low** | **Nominal** | **High** | **Very High** | **Extra High** | **Multiplier** |
| RELY | 0.75 | 0.88 | 1 | 1.15 | 1.4 |  | 1 |
| DATA |  | 0.94 | 1 | 1.08 | 1.16 |  | 1.08 |
| CPLX | 0.7 | 0.85 | 1 | 1.15 | 1.3 | 1.65 | 1 |
| TIME |  |  | 1 | 1.11 | 1.3 | 1.66 | 1 |
| STOR |  |  | 1 | 1.06 | 1.21 | 1.56 | 1 |
| VIRT |  | 0.87 | 1 | 1.15 | 1.3 |  | 1.15 |
| TURN |  | 0.87 | 1 | 1.07 | 1.15 |  | 0.87 |
| ACAP | 1.46 | 1.19 | 1 | 0.86 | 0.71 |  | 1.19 |
| AEXP | 1.29 | 1.13 | 1 | 0.91 | 0.82 |  | 1 |
| PCAP | 1.42 | 1.17 | 1 | 0.86 | 0.7 |  | 1.17 |
| VEXP | 1.21 | 1.1 | 1 | 0.9 |  |  | 1 |
| LEXP | 1.14 | 1.07 | 1 | 0.95 |  |  | 0.95 |
| MODP | 1.24 | 1.1 | 1 | 0.91 | 0.82 |  | 0.91 |
| TOOL | 1.24 | 1.1 | 1 | 0.91 | 0.83 |  | 1 |
| SCED | 1.23 | 1.08 | 1 | 1.04 | 1.1 |  | 1 |
| Multiplier |  |  |  |  |  |  | 1.30 |

*Table 7: Sun Build 0 Effort Multipliers*

PC Effort Multipliers

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Attributes** | **Very Low** | **Low** | **Nominal** | **High** | **Very High** | **Extra High** | **Multiplier** |
| RELY | 0.75 | 0.88 | 1 | 1.15 | 1.4 |  | 1 |
| DATA |  | 0.94 | 1 | 1.08 | 1.16 |  | 0.94 |
| CPLX | 0.7 | 0.85 | 1 | 1.15 | 1.3 | 1.65 | 0.85 |
| TIME |  |  | 1 | 1.11 | 1.3 | 1.66 | 1 |
| STOR |  |  | 1 | 1.06 | 1.21 | 1.56 | 1 |
| VIRT |  | 0.87 | 1 | 1.15 | 1.3 |  | 1.15 |
| TURN |  | 0.87 | 1 | 1.07 | 1.15 |  | 0.87 |
| ACAP | 1.46 | 1.19 | 1 | 0.86 | 0.71 |  | 1 |
| AEXP | 1.29 | 1.13 | 1 | 0.91 | 0.82 |  | 1 |
| PCAP | 1.42 | 1.17 | 1 | 0.86 | 0.7 |  | 1 |
| VEXP | 1.21 | 1.1 | 1 | 0.9 |  |  | 1 |
| LEXP | 1.14 | 1.07 | 1 | 0.95 |  |  | 0.95 |
| MODP | 1.24 | 1.1 | 1 | 0.91 | 0.82 |  | 0.91 |
| TOOL | 1.24 | 1.1 | 1 | 0.91 | 0.83 |  | 1 |
| SCED | 1.23 | 1.08 | 1 | 1.04 | 1.1 |  | 1 |
| Multiplier |  |  |  |  |  |  | 0.69 |

*Table 8: PC Build 0 Effort Multipliers*

Build 1

SUN Effort Multipliers

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Attributes** | **Very Low** | **Low** | **Nominal** | **High** | **Very High** | **Extra High** | **Multiplier** |
| RELY | 0.75 | 0.88 | 1 | 1.15 | 1.4 |  | 1 |
| DATA |  | 0.94 | 1 | 1.08 | 1.16 |  | 1.08 |
| CPLX | 0.7 | 0.85 | 1 | 1.15 | 1.3 | 1.65 | 1 |
| TIME |  |  | 1 | 1.11 | 1.3 | 1.66 | 1 |
| STOR |  |  | 1 | 1.06 | 1.21 | 1.56 | 1 |
| VIRT |  | 0.87 | 1 | 1.15 | 1.3 |  | 1 |
| TURN |  | 0.87 | 1 | 1.07 | 1.15 |  | 0.87 |
| ACAP | 1.46 | 1.19 | 1 | 0.86 | 0.71 |  | 1.19 |
| AEXP | 1.29 | 1.13 | 1 | 0.91 | 0.82 |  | 1 |
| PCAP | 1.42 | 1.17 | 1 | 0.86 | 0.7 |  | 1.17 |
| VEXP | 1.21 | 1.1 | 1 | 0.9 |  |  | 1 |
| LEXP | 1.14 | 1.07 | 1 | 0.95 |  |  | 0.95 |
| MODP | 1.24 | 1.1 | 1 | 0.91 | 0.82 |  | 0.91 |
| TOOL | 1.24 | 1.1 | 1 | 0.91 | 0.83 |  | 1 |
| SCED | 1.23 | 1.08 | 1 | 1.04 | 1.1 |  | 1 |
| Multiplier |  |  |  |  |  |  | 1.13 |

*Table 9: Sun Build 1 Effort Multipliers*

PC Effort Multipliers

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Attributes** | **Very Low** | **Low** | **Nominal** | **High** | **Very High** | **Extra High** | **Multiplier** |
| RELY | 0.75 | 0.88 | 1 | 1.15 | 1.4 |  | 0.88 |
| DATA |  | 0.94 | 1 | 1.08 | 1.16 |  | 0.94 |
| CPLX | 0.7 | 0.85 | 1 | 1.15 | 1.3 | 1.65 | 0.85 |
| TIME |  |  | 1 | 1.11 | 1.3 | 1.66 | 1 |
| STOR |  |  | 1 | 1.06 | 1.21 | 1.56 | 1 |
| VIRT |  | 0.87 | 1 | 1.15 | 1.3 |  | 1 |
| TURN |  | 0.87 | 1 | 1.07 | 1.15 |  | 0.87 |
| ACAP | 1.46 | 1.19 | 1 | 0.86 | 0.71 |  | 1 |
| AEXP | 1.29 | 1.13 | 1 | 0.91 | 0.82 |  | 1 |
| PCAP | 1.42 | 1.17 | 1 | 0.86 | 0.7 |  | 1 |
| VEXP | 1.21 | 1.1 | 1 | 0.9 |  |  | 1 |
| LEXP | 1.14 | 1.07 | 1 | 0.95 |  |  | 0.95 |
| MODP | 1.24 | 1.1 | 1 | 0.91 | 0.82 |  | 0.91 |
| TOOL | 1.24 | 1.1 | 1 | 0.91 | 0.83 |  | 1 |
| SCED | 1.23 | 1.08 | 1 | 1.04 | 1.1 |  | 1 |
| Multiplier |  |  |  |  |  |  | 0.53 |

*Table 10: PC Build 1 Effort Multipliers*

Build 2

SUN Effort Multipliers

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Attributes** | **Very Low** | **Low** | **Nominal** | **High** | **Very High** | **Extra High** | **Multiplier** |
| RELY | 0.75 | 0.88 | 1 | 1.15 | 1.4 |  | 1 |
| DATA |  | 0.94 | 1 | 1.08 | 1.16 |  | 1.16 |
| CPLX | 0.7 | 0.85 | 1 | 1.15 | 1.3 | 1.65 | 1 |
| TIME |  |  | 1 | 1.11 | 1.3 | 1.66 | 1 |
| STOR |  |  | 1 | 1.06 | 1.21 | 1.56 | 1 |
| VIRT |  | 0.87 | 1 | 1.15 | 1.3 |  | 1 |
| TURN |  | 0.87 | 1 | 1.07 | 1.15 |  | 0.87 |
| ACAP | 1.46 | 1.19 | 1 | 0.86 | 0.71 |  | 1 |
| AEXP | 1.29 | 1.13 | 1 | 0.91 | 0.82 |  | 0.91 |
| PCAP | 1.42 | 1.17 | 1 | 0.86 | 0.7 |  | 1 |
| VEXP | 1.21 | 1.1 | 1 | 0.9 |  |  | 1 |
| LEXP | 1.14 | 1.07 | 1 | 0.95 |  |  | 0.95 |
| MODP | 1.24 | 1.1 | 1 | 0.91 | 0.82 |  | 0.91 |
| TOOL | 1.24 | 1.1 | 1 | 0.91 | 0.83 |  | 1 |
| SCED | 1.23 | 1.08 | 1 | 1.04 | 1.1 |  | 1 |
| Multiplier |  |  |  |  |  |  | 0.79 |

*Table 11: Sun Build 2 Effort Multipliers*

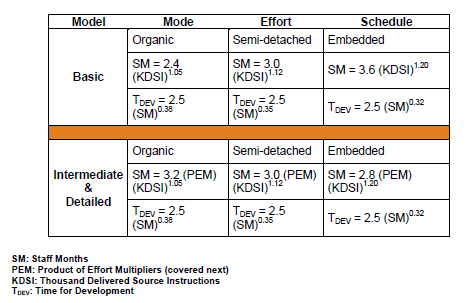
PC Effort Multipliers

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Attributes** | **Very Low** | **Low** | **Nominal** | **High** | **Very High** | **Extra High** | **Multiplier** |
| RELY | 0.75 | 0.88 | 1 | 1.15 | 1.4 |  | 0.88 |
| DATA |  | 0.94 | 1 | 1.08 | 1.16 |  | 0.94 |
| CPLX | 0.7 | 0.85 | 1 | 1.15 | 1.3 | 1.65 | 1 |
| TIME |  |  | 1 | 1.11 | 1.3 | 1.66 | 1 |
| STOR |  |  | 1 | 1.06 | 1.21 | 1.56 | 1 |
| VIRT |  | 0.87 | 1 | 1.15 | 1.3 |  | 1 |
| TURN |  | 0.87 | 1 | 1.07 | 1.15 |  | 0.87 |
| ACAP | 1.46 | 1.19 | 1 | 0.86 | 0.71 |  | 0.86 |
| AEXP | 1.29 | 1.13 | 1 | 0.91 | 0.82 |  | 0.91 |
| PCAP | 1.42 | 1.17 | 1 | 0.86 | 0.7 |  | 0.86 |
| VEXP | 1.21 | 1.1 | 1 | 0.9 |  |  | 1 |
| LEXP | 1.14 | 1.07 | 1 | 0.95 |  |  | 0.95 |
| MODP | 1.24 | 1.1 | 1 | 0.91 | 0.82 |  | 0.91 |
| TOOL | 1.24 | 1.1 | 1 | 0.91 | 0.83 |  | 1 |
| SCED | 1.23 | 1.08 | 1 | 1.04 | 1.1 |  | 1 |
| Multiplier |  |  |  |  |  |  | 0.42 |

*Table 12: PC Build 2 Effort Multipliers*

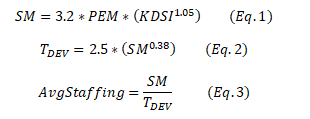
To get the total effort multiplier rating for each of the builds and platforms we can simply take the product of each effort multiplier rating. We can use this effort multiplier rating to compute staff months and time to develop for each of the three builds and two platforms.

As discussed in the above sections, we are using the organic development model and we are using the detailed model for each of the three builds. We can consult the following lookup table to determine the equations to use for the staff months and time for development, as well as the average staffing for the duration of each build.



*Table 13: Equation lookup table*

Using the organic model and the detailed equations we see that we can use the following two equations to compute staff months and development time. The equation for computing the average staffing is trivial, but is shown below as well.



The only remaining item to collect is the lines of code to write for each of the builds and platforms. We can take each of the tasks provided in the case study and sum the lines of code for each task into a total lines of code for each build. This information was also collected in the SDP. The total lines of code for each build and platform is shown in the table below:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Build | 0 | | 1 | | 2 | |
| Package | Sun | PC | Sun | PC | Sun | PC |
| LOC’s | 6500 | 4500 | 9500 | 3700 | 4000 | 3800 |

*Table 14: Summary of lines of code to be delivered for each build*

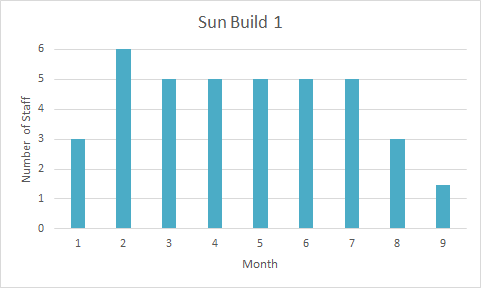
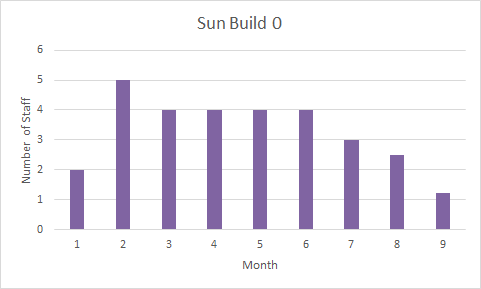
With the above information we can compute the number of staff months, the time to develop, and the average staffing for each of the builds. The results for each of the builds are shown in the below table:

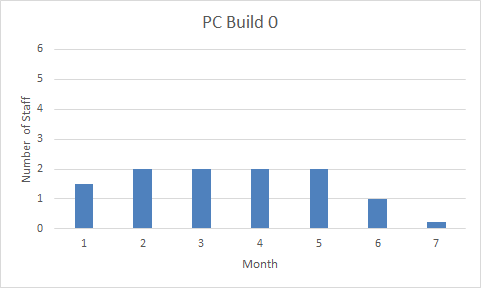
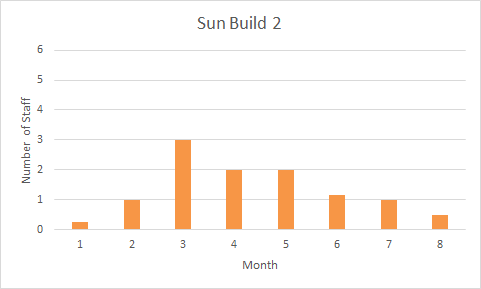
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Build | 0 | | 1 | | 2 | |
| Package | Sun | PC | Sun | PC | Sun | PC |
| SM | 29.71 | 10.73 | 38.48 | 6.68 | 10.89 | 5.44 |
| Tdev | 9.07 | 6.16 | 10.01 | 5.15 | 6.19 | 4.76 |
| Avg. Staff | 3.28 | 1.74 | 3.84 | 1.30 | 1.76 | 1.14 |

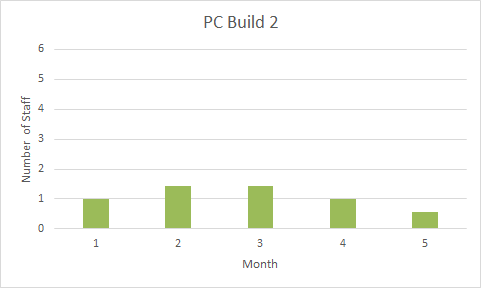
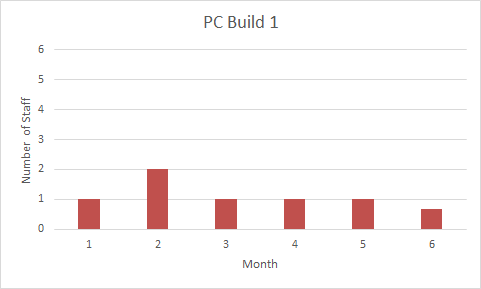
*Table 15: Tabulated results for staff months, development time and average staffing*

# Task d:

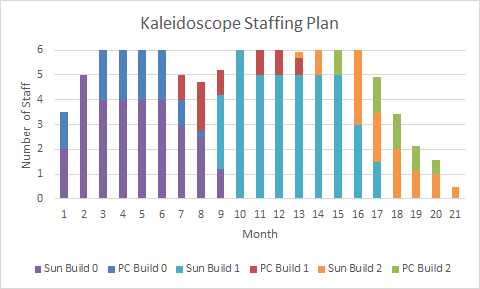
Overall, the estimations provided by the COCOMO method provide a clearer picture of when we can expect to deliver each build. In addition, the staffing requirements indicates a small team that validates our choice of the organic model. For each build, our staffing requirements are focused on ramping up the build during the first 2-3 months, culminating in the largest team size during the 2nd or 3rd month in order to facilitate the design phase of each build. Near the end of each build, the team size is tapered down as the work on the build finishes, leaving a smaller team to finish documentation as well as Breaking down each build by month gives us the following information:







Utilizing a phase-in/phase-out staffing model, we can stagger our build staffing and complete Build 0 in 9 months, Build 1 in 17 months, and Build 2 in 21 months. Because Sun Build 1 requires the most personnel in month 10 of 6 people, the staffing plans to allocate a maximum of 6 staff at any given work month. In addition, it prioritizes the current build, only releasing team members to the next build once the current build is fully resourced. Finally, because the Sun builds are much larger than their PC counterparts, resources are reallocated from the PC build to the Sun during each Sun build month 2 during the design portion of the build, when the staffing requirements of the build are much higher. The full project staffing plan would be as follows:



Alternative, if the entire project needs to be delivered as quickly as possible, it is possible to deliver Builds 1 and 2 at the end of month 17 with an increased staff size, as well as

Task 7:schedule parts of each build as aggressively as possible. In this case, our staffing plan will be as follows:

# 

This plan sacrifices build consistency of the incremental build process for a faster delivery of the final build

# Task e:

In the original Kaleidoscope BOE that our team developed, we only created COCOMO parameters for the project as a whole. This had not provided the level of detail that was necessary to get an accurate picture of what the project's timelines and staffing requirements would be. In the updated kaleidoscope BOE, the project was broken out into 6 parts, 3 builds for the PC and Sun Server. We also changed the parameters between builds to fit each builds characteristics as well as the changes in the team’s skills and knowledge as the development progressed. Mainly we made the assumption that the team would improve its productivity when working on the project over time as more experience was gained.

Comparing the results of the original BOE with the updated BOE gives clear differences in staffing and scheduling. Where in the original BOE we had a staffing profile that started with an overall picture of about 3 staff for build 0 ramping up to about 4 staff for build 1, and then a ramp down of 2 staff for build 2, now we see in the reassess a similar trend albite with different numbers in the sun server development. With the differences in COCOMO parameters and breaking out the builds and platforms, the staffing requirements have changed with the Sun server taking up the most staff and the PCs taking up fewer. In the Sun server graphs, we see the same trend of starting with build 0 and then ramping staffing up a bit for build 1 and then dropping staff for build 2 as in the original. For the PC development, the outlook is a little more different and linear with the staffing being the highest at the start with build 0, and dropping little by little with subsequent builds. The original BOE staffing requirements seemed to be mostly driven by the Sun server development given the breaking out of COCOMO parameters in the reassessment.

Scheduling has also changed from the original BOE into the reassess. Build 0 has a development time and release at 9 months, Build 1 at 17 months, and Build 2 at 21 months. Comparing these new numbers to the original BOE shows that the new schedule has shrunk in size with the original BOE allocating development and release at ~9 months to build 0, ~18 months to build 1, and 26 months to build 2. This decrease in schedule may be due to our new BOE reassess taking into account the project team developing their skills over the course of the project and increasing their productivity. Here again, the Sun Server development has taken the most effort and therefore has the longest development time, pushing out the schedule as compared to the PC development.

These staffing requirements and scheduling make sense as given by the number of lines of code for each build (Build 1 requiring 11,000 LOC, Build 2 requiring 13,200 LOC, and Build 3 requiring 7,800 LOC) we would expect the staff requirements and scheduling to reflect this and they do. The staffing for build 2 and development time is the greatest of the three and therefore is predicted to have the most staff and the longest development time. Build 1 is the second longest and this is reflected with the staff and development time being the second longest. Build 3 has the least of amount of LOC and therefore has the least staff and smallest development time.

If we look at the rate of LOC per day predicted by the Basis of Estimate Assignment of 18 LOC per day and perform the following calculations:

11,000 LOC / 18 LOC per day = 611.11 staff days / 5 working days per week = 122.22 staff weeks / 13 weeks per staff quarter = 9.4 staff quarters \* 3 staff months = 28.21 staff months

13,200 LOC / 18 LOC per day = 733.33 staff days / 5 working days per week = 146.67 staff weeks / 13 weeks per staff quarter = 11.28 staff quarters \* 3 staff months = 33.85 staff months

7,800 LOC / 18 LOC per day = 433.33 staff days / 5 working days per week = 86.67 staff weeks / 13 weeks per staff quarter = 6.67 staff quarters \* 3 staff months = 20 staff months

Given these straight linear predictions for how long the development time would take in staff months, we see that they are close to what was predicted in Task 3, taking slightly longer than what was predicted given that we took the conservative number of 18 lines of code.

If we take the more optimistic rate of 20 LOC per day and perform the following calculations:

11,000 LOC / 20 LOC per day = 550 staff days / 5 working days per week = 110 staff weeks / 13 weeks per staff quarter = 8.46 staff quarters \* 3 staff months = 25.38 staff months

13,200 LOC / 20 LOC per day = 660 staff days / 5 working days per week = 132 staff weeks / 13 weeks per staff quarter = 10.15 staff quarters \* 3 staff months = 30.46 staff months

7,800 LOC / 20 LOC per day = 390 staff days / 5 working days per week = 78 staff weeks / 13 weeks per staff quarter = 6 staff quarters \* 3 staff months = 18 staff months

We see these calculated staff months are close to the predicted staff months from task 3 as well, being slightly under. We can be confident that values calculated in Task 3 are fairly accurate.