PURPOSE

This document is the design description for SERIM.EXE – the software that implements the Software Engineering Risk Management (SERIM) model.

REFERENCES

Karolak, Dale W. Software Engineering Risk Management. IEEE Computer Society Press. 1996.

BACKGROUND

The SERIM model is a formal project risk assessment model with a defined set of 81 questions divided into 10 categories. The equations and weightings are fixed, except for the three corners of the project triangle: Cost, Technical, and Schedule. The user selects the weights for how important each of these three facets is to the project. By default, the weights are equal.

In addition to the 10 categories, each of the questions also affects one or more of the six phases and one or more of the six risk management activities.

Categories

1. Organizational structure.
2. Estimation methods (cost & schedule).
3. Monitoring (project cost, schedule, and budget).
4. Development methodology.
5. Tools used for development.
6. Risk Culture of the company.
7. Usability of the software.
8. Correctness of the software (how well it matches the requirements).
9. Reliability of the software.
10. Personnel on the development team.

Development Phases

- Pre-Requirements.
- Requirements.
- Design.
- Code.
- Test.

- Delivery and Maintenance.

Risk Management Activities

- Identification.
- Strategy and Planning.
- Assessment.
- Mitigation/Avoidance.
- Reporting.
- Prediction.

INITIAL DESIGN

The design proceeded in incremental stages:

1. Identify the information flow through the software and identify the candidate objects and their roles.
2. Design the Question objects. These were designed and coded in a test shell to ensure that the interface was stable.
3. Design the User Interface for the ten categories.
4. Integrate the Question objects into the User Interface.
5. Design the Summary display.
6. Design the File Open/Save methods.
7. Design the project triangle weighting user interface.

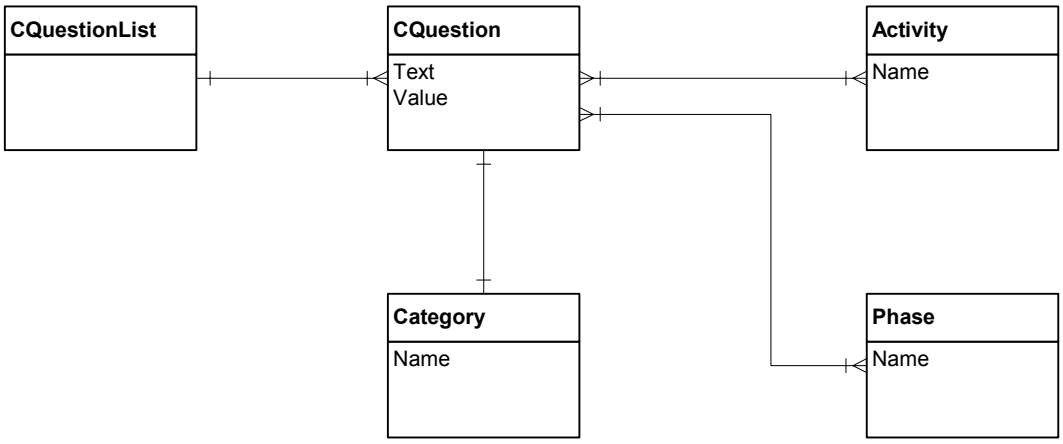
IMPLEMENTATION

CQuestion and CQuestionList

Functional Decomposition was used to create the Entity-Relation diagram of the information. This showed that each Question resided in only one Category, and affected one or more Phases and one or more Activities. This was implemented by defining a Flags field for each question, and OR-ing the bit-flags for the Category, Phases, and Actvities.

CQuestionList holds the list of 81 questions. The constructor creates the questions using a class factory, so that the rest of the application can obtain and use a CQuestion*, but cannot create new questions or delete the CQuestion object.

Figure 1 – E-R Diagram of Question Information



Resolving the many-to-many relationships was done by using bit fields in CQuestion to identify the Activities and Phases for that question.

Calculation

25 of the 28 SERIM equations (PA3 through PO, reference the SERIM book) are all an average of the responses from the questions in those Categories, Phases, or Activities. By using a bit-field to identify the question's Category, Phases, and Activities, the same calculation method could be used for these 25 equations. The logic is:

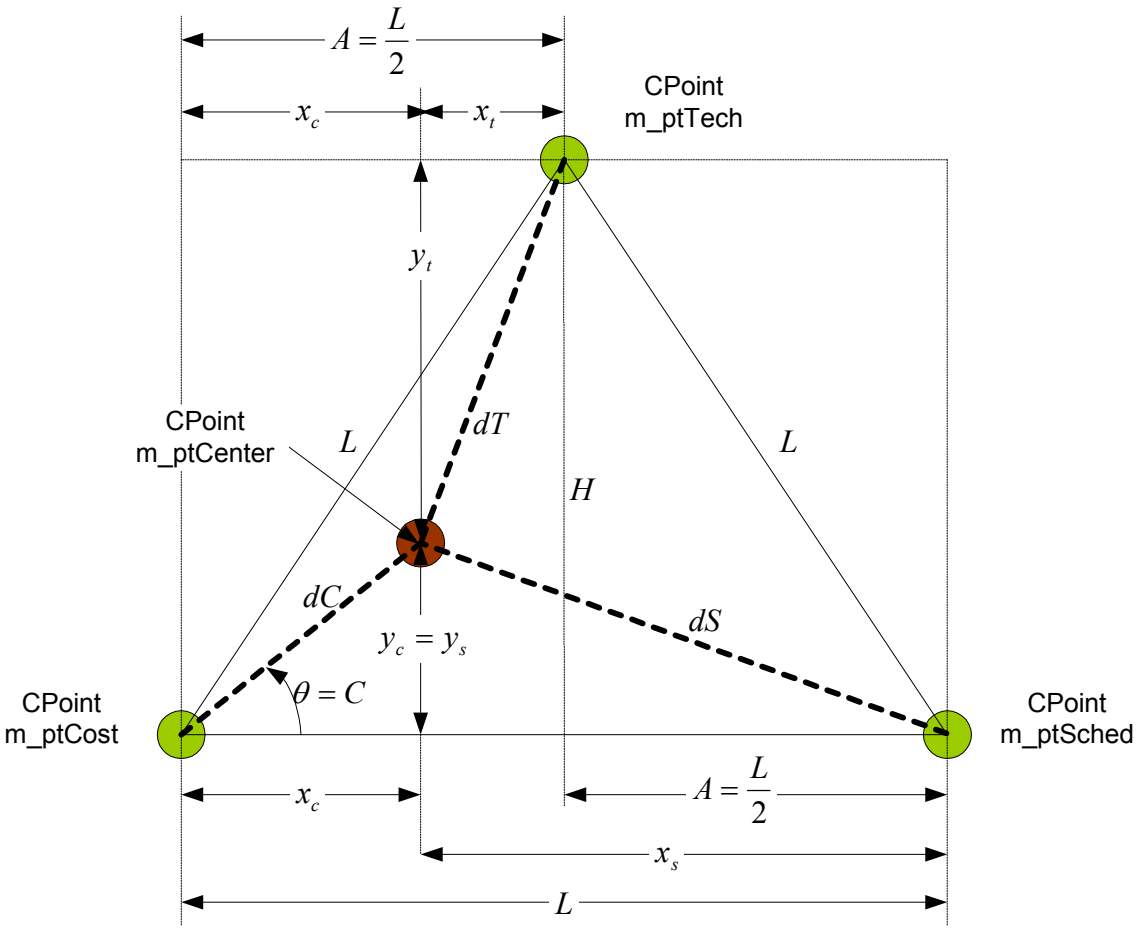
```
Calculate(EquationType)
        FOR each Question DO
                IF Question is in this EquationType THEN
                        Accumulate the Question's response to the average
                END IF
        END FOR
        Return Average
```

CDlgPriority – Setting the Project Triangle

The challenge here was to create a user interface that allowed the user to adjust three interactive values: 1) Cost Priority, 2) Schedule Priority, and 3) Technical Priority. The three priorities must add to 100%. These three priorities represent the three corners of the Project Triangle, so I decided to use a literal triangle on screen. Each corner would be one of the priorities. The user would then drag a red ball around within the triangle. The ball's distance to each corner determined the priority of that corner.

Laying this out for the User Interface was initially done on paper. Because this was a math-intensive exercise, the equations are given in detail.

Figure 2 – Screen Layout for Project Triangle



Initial Assumptions:

The CPoint objects m_ptCost and m_ptSched are pre-defined points on the dialog box via the placement of an invisible static frame. The frame is used to provide those two anchor points for the bottom leg of the equilateral triangle. The frame's bottom left and right corners set the points m_ptCost and $m\_ptSched$, and the value of L.

The CPoint m_ptTech is calculated as follows:

Equation 1: $m\_ptTech.x = m\_ptCost.x + A$

Equation 2: $H = \sqrt{L^2 - A^2}$

Equation 3: $m\_ptTech.y = m\_ptCost.y - H$ (Note: MM_TEXT mapping mode is used)

This positions all three points on screen. The WM_PAINT handler *OnPaint()* is used to draw line L between all three points.

The ratios of the lengths dC, dS, and dT define the weights for Cost, Schedule, and Technical priorities. The total must always equal 100%.

Stage 1 – Designing The Logic Flow of the Priority Weighting User Interface

The user clicks on the red dot and drags it around. The distance to each corner of the triangle determines the relative weighting. The basic logic is:

```
On Left Mouse Click
        IF Click is inside the Red Dot THEN
                Set Tracking Mode
        END IF

On Left Mouse Release
        Clear Tracking Mode

On Mouse Move
        IF In Tracking Mode AND within bounds THEN
                Move Red Dot to the current mouse position (this sets m_ptCenter)
                Calculate Weights, given the new Centerpoint
        END IF
```

Stage 2 – Calculate Weights Given the Centerpoint

To calculate the three distances:

Equation 4 $x_c = m\_ptCenter.x - m\_ptCost.x$

Equation 5 $y_c = m\_ptCost.y - m\_ptCenter.y$

Equation 6 $dC = \sqrt{x_c^{\,2} + y_c^{\,2}}$

Equation 7 $x_s = L - x_c$

Equation 8 $dS = \sqrt{y_s^{\,2} + x_s^{\,2}}$

Equation 9 $y_t = H - y_s$

Equation 10 $x_t = A - x_c$

Equation 11 $dT = \sqrt{x_t^{\,2} + y_t^{\,2}}$

To calculate the weights, take the ratio of each distance to the total distance.

Equation 12 $$CostWeight = \frac{dC}{dC + dT + dS}$$

Equation 13 $$TechWeight = \frac{dT}{dC + dT + dS}$$

Equation 14 $$SchedWeight = \frac{dS}{dC + dT + dS}$$

The weights are calculated as the Red Dot moves, allowing the display to be updated. When OK is clicked, the new weights are returned to the caller.

Stage 3 – Calculate Centerpoint Given the Weights

When the CDlgPriority dialog is created, the caller can pass the current weights dC, dT, and dS. When the dialog box is displayed, it calculates the Centerpoint based on the weights. The Law of Cosines is used to calculate angle C, and then x_c and y_c can be calculated.

Since the weights are ratios, absolute measurements of the lengths dC, dS, and dT must be calculated. Since they are ratios, relate them to the percentage of the baseline.

Equation 15 $$dC = (1 - CostWeight) \times L$$

Equation 16 $$dT = (1 - TechWeight) \times L$$

Equation 17 $$dS = (1 - SchedWeight) \times L$$

By the Law of Cosines:

Equation 18 $$\theta = C = COS^{-1}\left[\frac{dS^2 - dC^2 - L^2}{2 \times dC \times L}\right]$$

Equation 19 $$y_c = dC \times SIN(\theta)$$

Equation 20 $$x_c = dC \times COS(\theta)$$

Equation 21 $$m\_ptCenter.x = m\_ptCost.x + x_c$$

Equation 22 $$m\_ptCenter.y = m\_ptCost.y - y_c$$