

COMP 145 UNC-Chapel Hill

Contract II

KALMAN FILTER LEARNING TOOL

Submitted to

_____ **GREG WELCH, Client**

and Prof. Greg Welch

February 13, 2001

_____ **Chris Riley, Director**

_____ **Tom Bodenheimer, Producer**

_____ **Erin Parker, Admin. Leader**

_____ **John Carpenter, Librarian**

PREFACE

The purpose of this document is to describe the requirements for a piece of software that will be written for client Greg Welch by the team consisting of Thomas Bodenheimer, John Carpenter, Erin Parker, and Chris Riley. The requirements described herein are meant to be a formal agreement of what will be delivered from the team to the client. This document is only a preliminary report of such an agreement. After receiving this document on or before February 6, 2001, the client will review the contract with the team. A second contract representing the proposed final agreement will be presented to the client by February 13, 2001 that will include additional information on verification and validation.

1 Introduction

The client for this project is Greg Welch, a professor in the computer science department at the University of Chapel Hill. He uses Kalman filters in his research and maintains a popular web page on the subject. The Kalman filter is a set of mathematical equations that provides an efficient computational estimate of the state of a process (e.g., the position and orientation of an airplane) given a time-varying sequence of noisy measurements (e.g., air speed, pressure, temperature, engine thrust). Our goal is to develop a web-based tool to help develop the intuition and insight of novice users regarding the behavior of the Kalman filter.

2 User-level Requirements Specification

2.1 Context Model

The Kalman Filter Learning Tool is composed of the four components shown in Figure 1: a user interface, a Kalman filter engine, a data repository, and a plotting tool.

2.1-A User Interface

The user interface sets the parameters governing the execution of the Kalman filter for a specified test problem, graphically displays the state and error estimates generated by the filter, and provides access to the filter's iteration history and intermediate values.

2.1-B Kalman Filter Engine

The Kalman filter engine implements the Kalman filter equations and is responsible for computing both single and multiple iterations of the filter.

2.1-C Data Repository

The data repository stores a record of the filter's iterations and the values computed.

2.1-D Plotting Tool

The plotting tool generates plots of the state and error estimates produced by the filter. The user interface is responsible for displaying these plots.

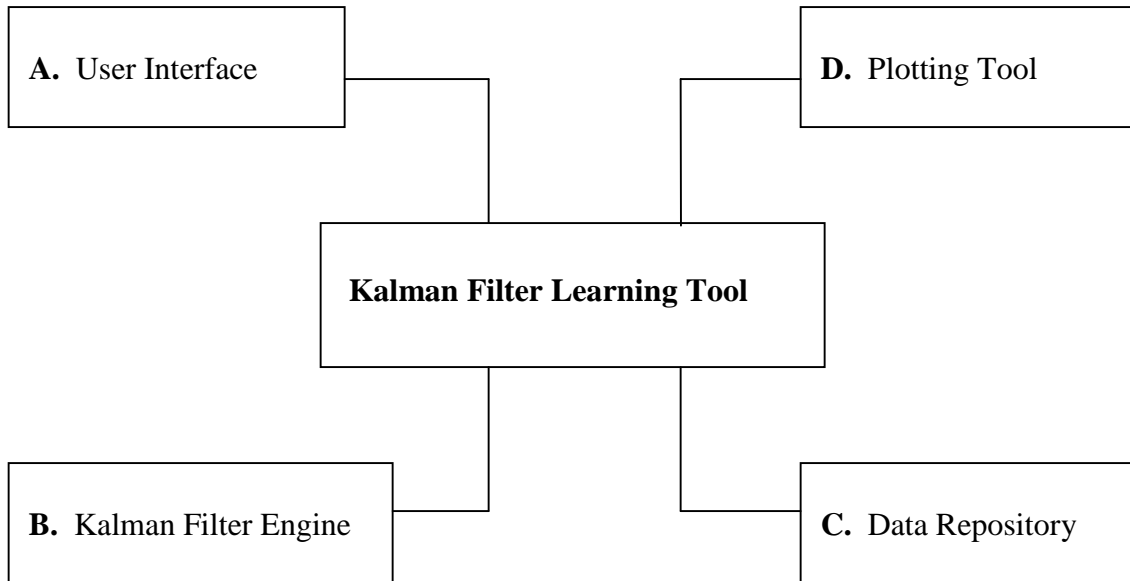


Figure 1: Context Model Diagram.

2.2 Process Models

The Kalman Filter Learning Tool has two modes of operation. The first is a *batch* mode in which an entire simulation is computed and the results are then displayed to the user. The processes involved in this mode are shown in Figure 2. The second is a *step* mode in which the user may step through a completed simulation and access intermediate values computed by the Kalman filter at a local iteration. These processes are shown in Figure 3.

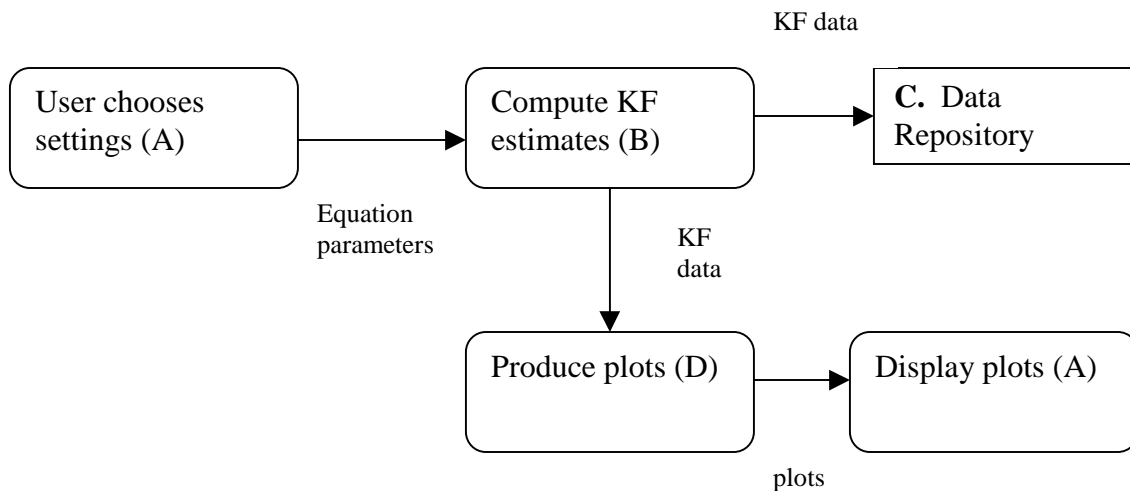


Figure 2: Kalman Filter Simulation (batch mode). Letters denote module responsible for process.

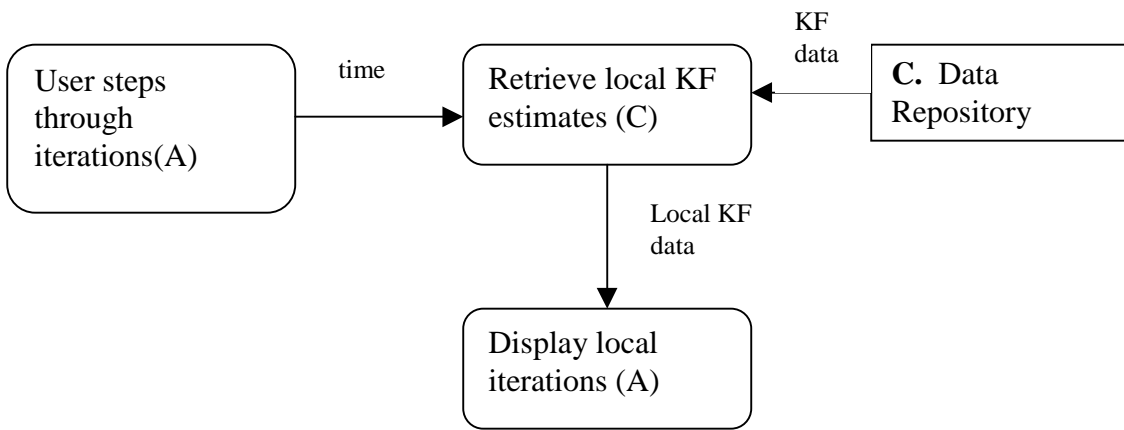


Figure 3: Kalman Filter Simulation (step mode). Letters denote module responsible for process.

3 System-Level Requirements Specification

3.1 Functional Requirements

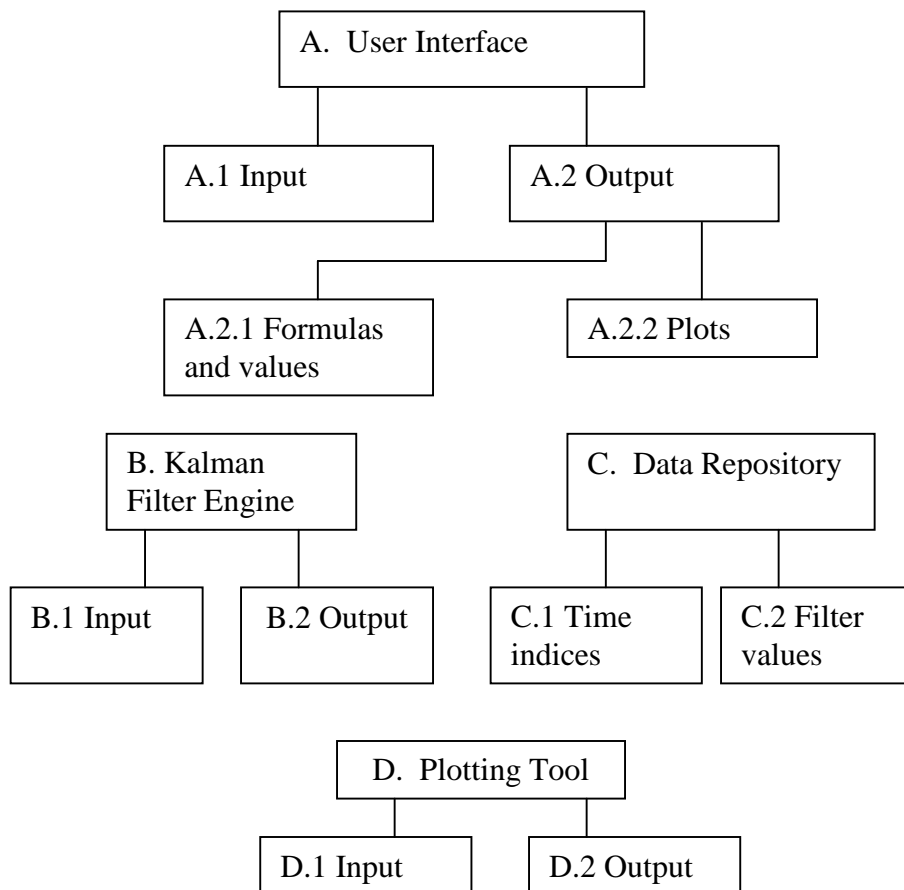


Figure 4: System-Level Context Diagram

3.1.A User Interface

3.1.A.1 Input

Input to the user interface includes the parameters to be used by the Kalman filter. (Parameter names will have *hyperlinks* to web pages with descriptions.) These include:

1. Fluid Level Dynamics for actual and model.
 - a. Constant (default, no buttons selected)
 - b. Filling (“add filling” button selected)
 - c. Sloshing (“add sloshing” button selected)
 - d. Filling and Sloshing (both buttons selected)
2. Measurement Noise (three choices of scale factors: 0.01, 1, 100)
3. Process Noise (three choices of scale factors: 0.01, 1, 100)
4. Measurement Type
 - a. Linear
 - b. Nonlinear

3.1.A.2 Output

3.1.A.2.1 Formulas and Values

The output from this section is yet to be determined and is still be discussed between client and team members. However, a static set of formulas will be displayed for each combination of input parameters (of which there is a finite number).

3.1.A.2.2 Plots

The plotted output of the user interface will be a window with the plots of the truth, residual, $\text{sqrt}(P)$, and estimated values from the Kalman filter.

3.1.B Kalman Filter Engine

3.1.B.1 Input

Input to the Kalman Filter Engine will consist of the user selected values from 3.1.A.1.

3.1.B.2 Output

Output from the Kalman Filter Engine will consist of the output data for the plot information, including the truth, estimated value, $\text{sqrt}(P)$, and residual.

3.1.C Data Repository

3.1.C.1 Indices

Each index into the data repository will correspond to the time-step of the filter for a simulation.

3.1.C.2 Filter values

The values of the data located at index t will correspond to the values of the Kalman filter at time t ($t=0$: initial conditions, $t=1$ first estimate).

3.1.D Plotter

3.1.D.1 Input

Input to the plotting tool will be the output values of the Kalman filter for an entire simulation (produced by the Kalman filter engine).

3.1.D.2 Output

The output of the plotter tool will be the graphed results of the current simulation in a window ready to be displayed by the user interface.

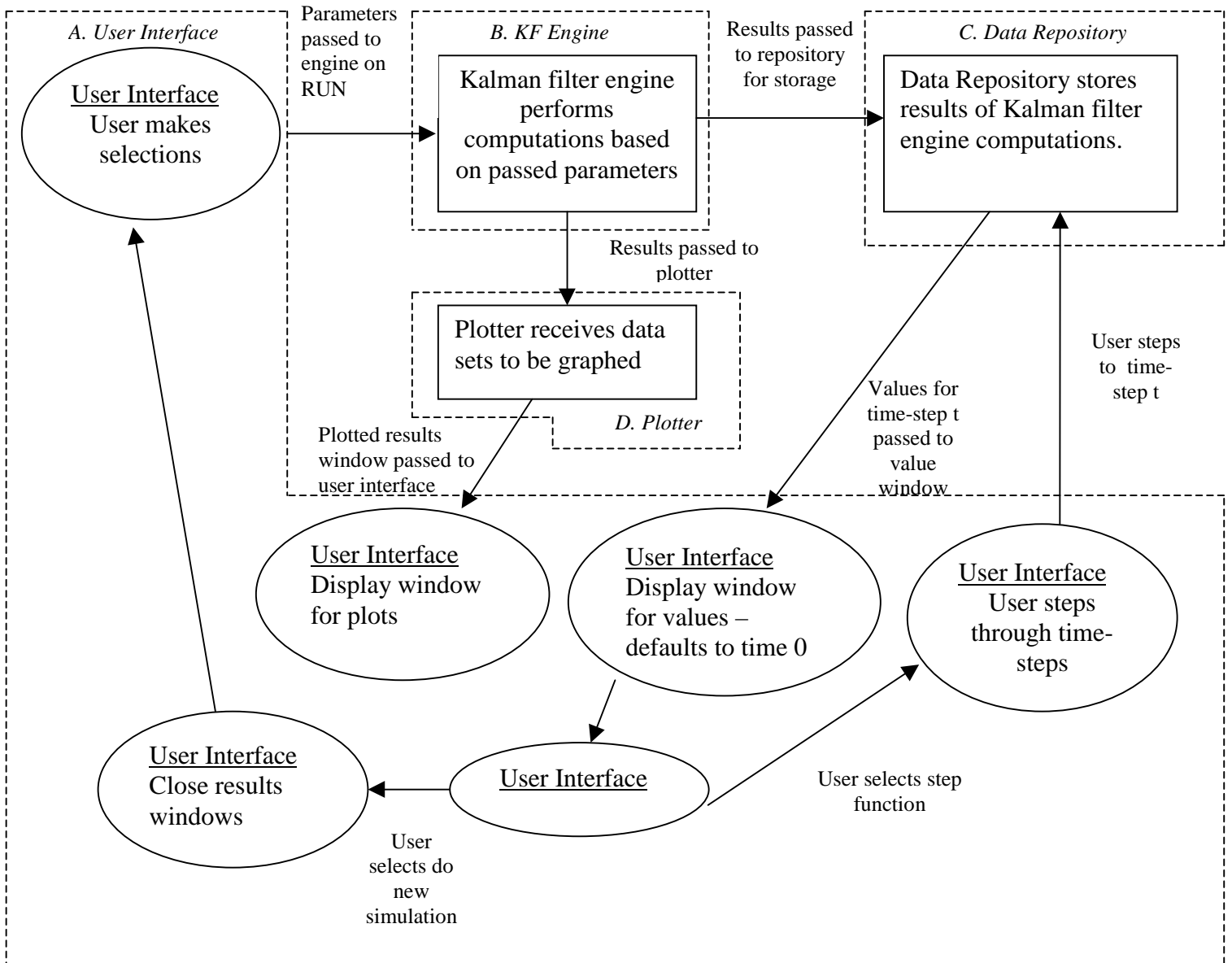


Figure 5: Data-Flow Diagram

3.2 Non-Functional Requirements

3.2.1 Product Requirements

Temporal constraints. This software is intended to be used as an interactive tutorial and thus will have some (not very stringent) real-time requirements. After

performing a time-step, the user will need to see the appropriate state displayed in a reasonable amount of time. It is of course impossible to determine an exact figure because of the different platforms on which this program will be run (see Portability requirements).

Portability. This software must be able to run on a Macintosh, PC, or UNIX box running a popular web browser such as Netscape Navigator. It must look appropriate on a monitor with 1024x768 pixel resolution. The software should be able to be run over the web. A standalone version should also be available for inclusion in a CD-ROM that will accompany a tutorial on the Kalman filter that will be presented at SIGGRAPH 2001. Since Java applets will be used for much of the software, the product will not be able to write any files on the client machine. This will cause certain memory constraints. At this time it is difficult to predict the exact free memory requirements necessary on the client machine, but this will be a consideration in the development process.

3.2.2 Organizational Requirements

This product will be delivered in accordance with the process and deliverables set forth by Prof. Greg Welch in the COMP 145 class in the Computer Science Department of the University of North Carolina at Chapel Thrill.

3.2.3 External Requirements

This product will not address any concerns about safety, ethics, legislation, privacy, or inter-operability.

3.3 Domain Requirements

This product will not adhere to any standards developed by any third party organization nor will it co-exist with any external products.

3.4 Goals

The goal of this software is that it be easy to use. An assumption about this product is that a typical user will interact with it once or twice for a maximum of 2 hours. Since this is not a commercial piece of software, it is assumed that any potential user will spend very little time learning any “customizable features” and will very quickly become frustrated with any non-intuitive features of the user interface. It should thus be appropriately simple.

Although all work by the group will be completed in late April of 2001, the client may alter the software after this date. The documentation and modularity of this software will be given special consideration for ease of future maintenance.

4 Hardware and Software Resource Requirements

There are three areas of the Kalman Filter Learning Tool in which the use of pre-existing software packages would simplify and speed development. First, the development of the user interface would benefit from the use of an Integrated Development Environment (IDE). Second, the use of a plotting package saves the

time and effort of developing our own. Third, an object-oriented matrix package simplifies the development of the Kalman filter engine. Because we have chosen to develop the Kalman Filter Learning Tool in Java, all packages must be in Java as well. The following sections discuss procurement plans and alternatives for each.

4.1 Java IDE

We have already obtained Forte for Java, an IDE developed by Sun Microsystems, to use in the development of the graphical user interface. The IDE was chosen because it is specifically designed to work with Java, it runs on the Linux platform, and it is freely available. Alternatives to using this IDE include using other IDEs such as Visual J++ by Microsoft as well as developing the user interface from scratch. The danger in using any of these tools is that the tool will be difficult to learn and will actually slow down development.

4.2 Java Plotting Package

We are currently searching for an appropriate Java plotting package to use. Our desire is to find one that is relatively simple since we require only X-Y plots and is also easy to integrate with the user interface. The alternative to using an existing package is to write our own set of plotting routines.

4.3 Java Matrix Package

We already identified the object-oriented matrix package JAMA as a possible candidate to use in the implementation of the Kalman filter equations. The risk in using this package is low in that the matrices involved in the Kalman filter equations are small and the matrix routines could easily be coded by hand.

5 Preliminary Schedule (Task Budgeting)

5.1 Development Items

The major phases of the schedule are the following.

- Procurement/learning phase: procure tools for interfacing and plotting, learn the tools and the Kalman filter (activities a, b, c, d, e, g, h)
- Design phase: work with the client to specify a design for the project (activities i, k)
- Planning phase: develop plans for implementation (activities f, j, l, m, n)
- Implementation/integration phase: implement project pieces and integrate them (activities o, p, q, r, s, t, u)
- Testing phase: test and validate project (activity v)
- Final phase: finalize implementation and documentation (activities w, x)

The major milestones of the schedule are the following.

- *Contract I/Schedule I* delivered to client
- *Contract II/Schedule II* signed by client
- *Design Specification* signed and delivered to boss

- *User Manual* signed and delivered to boss
- *Prototype* presented to client
- *Implementation Manual* signed and delivered to boss
- Kalman filter engine passes test cases (successfully reproducing MATLAB results)

The tasks and dependencies are exhibited in the PERT Chart below. Deliverables are denoted in the PERT Chart by italics.

5.2 Schedule Diagrams

5.2.1 PERT Chart

	<u>Activity</u>	<u>Duration (days)</u>	<u>Dependency</u>
1. (a)	Meet with client	1	none
2. (b)	Read “An Introduction of the Kalman Filter”	2	none
3. (c)	Write <i>Preliminary Report</i>	2	1, 2
4. (d)	Choose/procure a Java IDE	3	none
5. (e)	Learn the Java IDE	5	4
6. (f)	Write <i>Contract I/Schedule I</i>	6	3
7. (g)	Choose/procure a plotting tool	3	none
8. (h)	Learn the plotting tool	5	7
9. (i)	Develop <i>Prototype</i>	10	5, 8
10. (j)	Write <i>Contract II/Schedule II</i>	4	6
11. (k)	Write <i>Design Specification</i>	5	9, 10
12. (l)	Write <i>User Manual</i>	7	11
13. (m)	Prepare for <i>Group Status Report</i>	5	9
14. (n)	Write <i>Implementation Manual</i>	7	12
15. (o)	Complete the Kalman filter engine	14	3
16. (p)	Integrate engine and plotter	7	8, 15
17. (q)	Complete the user interface	14	5, 11
18. (r)	Integrate UI and engine	7	15, 17
19. (s)	Integrate UI and plotter	7	17
20. (t)	Complete data repository	7	none
21. (u)	Integrate data repository to UI/engine/plotter	7	18, 19, 20
22. (v)	Test and validate KF On-line Tutorial	10	21
23. (w)	Prepare final <i>Group Presentation</i>	7	22
24. (x)	Prepare/write <i>Final Project Documentation</i>	14	13, 14, 23

Table 1: The PERT Chart. (Note: Deliverables are denoted by italics.)

developed software packages are coupled together, there is a risk that the modules either will not work together or individual modules will require rewriting to force them to fit together.

Testing Phase. Activity v involves testing the Kalman Filter Learning Tool. Because testing occurs late in the schedule, there is a risk that key elements of the tool may not be working and may be too difficult to fix before the end of the project.

5.3.2 Plans

Procurement/Learning Phase. To minimize the risks involved in procuring and learning the Java IDE and plotting tool, we have naturally scheduled these activities early in the project. In fact, we have already obtained and learned how to use the Java IDE, Forte for Java. Developing a prototype user interface with the IDE before all of the other modules are finished is another way to deal with this risk. In addition, the plotting tool is not truly needed until the end of the schedule when it is integrated with the user interface. Thus the schedule allows for the possibility of writing our own plotting routines if necessary.

Implementation/Integration Phase. The integration of the modules occurs in several steps. The order in which modules are integrated with the user interface, which is the primary module, is not critical. Thus progress on integrating the modules may proceed independently once the individual pieces are written.

Of course the final tool depends on the successful integration of all modules, but this incremental approach minimizes the impact to the schedule due to difficulty coupling any pair of modules.

Testing Phase. Developing a prototype of the user interface is our main way of minimizing risk in the testing phase. With a prototype, we hope to discover missing features and identify and correct coding and interface bugs relatively early in the design schedule. Testing the individual modules as they are developed also minimizes the number of coding errors and should reduce testing time later in the schedule.

6 Validation and Verification

6.1 Strategy

Developing a prototype of the user interface early in the design process will help validate and verify our Kalman Filter Learning Tool. As well as helping us arrive at a design that is acceptable to the client, working with a prototype will also help us detect flaws in the interface of the user interface with the other components (e.g., Plotter, Data Repository, Kalman Filter Engine). Testing of the individual components will occur as they are being developed and before they are integrated into the final product. For example, the plotter can be tested on generic X-Y plots of the form used in the learning tool. The output from the Kalman filter module can be

checked against pre-existing Kalman filter computations or the output from other codes. Testing of the integration of the various components with the user interface will occur relatively late in the schedule. Because there are relatively few parameters available to the user, we can test all possible combinations of input (see Section 3.1) for the complete learning tool.

6.2 Test Cases

Test cases that will be used to validate and verify the software are as follows:

1. Test Kalman filter module for all combinations of actual and modeled fluid level dynamics and measurement type for a specified amount of measurement noise and a specified measurement rate. This presently leads to 32 test cases. Output from each will be compared with output from MATLAB routines furnished by the client.
2. Test user interface by verifying that for each of the combinations listed above, the user interface passes the correct parameters to the Kalman filter module.
3. Test data repository by checking that the data returned to the user for a specific time index matches the data computed by the Kalman filter module (or MATLAB routines) at that time index.
4. Test plotting tool on data representative of the Kalman filter computations. The resulting plot will be checked for accuracy and appearance including axis limits, line width, and axis labels.
5. Test completed Kalman Filter Learning Tool in batch mode. Check that for each of the input combinations, the correct data is plotted for the user.
6. Test integrated tool in step mode. For each of the input combinations and a specified time index, check that the data returned to the user matches the correct Kalman filter computations at that time index.