Spk Version 0.1 Milestones

This document describes a set of milestones which will be reached by the Spk Software Team, on the way to releasing version 0.1 of the new Internet friendly Spk. A description of each milestone is provided, along with the name of the team member with primary responsibility for achieving it, and the estimated date of completion. As each milestone is reached, the actual date of completion will be added.

Table of Contents

Introduction

Spk is being developed as an open source project. One of the important benefits of this approach is that it makes available to us a broad array of free tools and it enables us to incorporate much free software. The tools and free software have accelerated our project substantially and will continue to do so. Given our limited resources, it would be impossible to accomplish this project in the time-frame required without these free assets.

Having access to a wealth of free tools and software changes the nature of the the development process. It places a premium on the ability to quickly evaluate, learn to use and to assimilate new products. It also introduces a scheduling uncertainty. Because we are learning so much in the process of development, it is difficult to know exactly how long any given task will take.

We have a commitment to deliver a usable system by the end of the second quarter of 2004. We are convinced that we can do this. Instead of attempting to do finely textured task scheduling, we have decided to focus on a set of milestones.

Milestones have been defined and divided up among the four developers. Many of the milestones require significant collaboration between two or more team members but, nevertheless, a single developer has accepted primary responsibility for each milestone. Each developer has provided a good-faith schedule for the milestones for which he or she is responsible, fully acknowledging that these estimates may include significant error.

In spite of the variation in the individual estimates, we believe that the cumulative effect will be to reach our goal on schedule. This document is meant both to show where we are going and to demonstrate the progress that we have made.

Milestone Table

Table 1. Spk Version 0.1 Milestones

Milestone Description	Responsible Developer	Estimated Date of Completion	Actual Date of Completion
Job History Model	Alan Westhagen	07/31/2003	07/31/2003

Milestone Description	Responsible Developer	Estimated Date of Completion	Actual Date of Completion
Model Capabilities Spec	Mitch Watrous	10/22/2003	10/22/2003
XML Models and Parameters	Sachiko Honda	11/04/2003	11/04/2003
Database E-R Model	Alan Westhagen	11/17/2003	11/17/2003
XML Results	Sachiko Honda	11/20/2003	12/05/2003
Database Schema	Alan Westhagen	11/21/2003	11/19/2003
XML Data	Sachiko Honda	12/05/2003	12/05/2003
Optimizer Replacement	Mitch Watrous	12/08/2003	01/12/2004
Aspk Compiler Output (C++)	Sachiko Honda	12/08/2003	12/17/2003
Database API	Alan Westhagen	12/12/2003	12/02/2003
Database API, Perl Binding	Alan Westhagen	12/18/2003	12/24/2003
User Output Spec	Jiaji Du	01/05/2004	01/08/2004
User Output	Jiaji Du	02/02/2004	02/02/2004
Database API, Java Binding	Alan Westhagen	02/09/2004	02/09/2004
Cspk Object Build	Alan Westhagen	02/29/2004	04/30/2004
Askp Compiler Output (Ind)	Sachiko Honda	02/19/2004	05/10/2004
MDA Submits Jobs	Jiaji Du	02/23/2004	03/01/2004
Network Security Spec	Alan Westhagen	03/01/2004	04/08/2004
Cspk Builds and Runs Jobs	Alan Westhagen	03/29/2004	05/17/2004
Web Site Functional Spec	Alan Westhagen	03/12/2004	04/30/2004
Cspk Generates Report	Alan Westhagen	03/29/2004	05/30/2004
Network Security	Alan Westhagen	03/26/2004	03/14/2004
All Open Source	Mitch Watrous	03/29/2004	03/15/2004
Web Site	Jiaji Du	03/29/2004	03/28/2004
Cspk Pred Class	Mitch Watrous	03/29/2004	05/05/2004
Cspk Covariance Class	Mitch Watrous	03/29/2004	06/04/2004
Aspk Compiler Output (Pop)	Sachiko Honda	03/29/2004	06/04/2004
MDA Presents Results (Pop)	Jiaji Du	03/29/2004	06/04/2004

Description of Milestones

In this section, the individual milestones are described.

All Open Source

All of the proprietary code in the Spk library is replaced with open source code.

Aspk Compiler Output (C++)

The Aspk compiler transforms model specifications submitted by the MDA into C++ source code and places this code in the run queue, for subsequent retrieval by the Cspk for building and execution.

This document is now available for viewing here[1]. Return to table.

Aspk Compiler Output for Individual Analysis

The Aspk compiler takes jobs from the compiler queue of the database, compiles them, and places in the database the archive of files needed by computational servers to perform population analysis. The compiler outputs the files required by computational servers to perform individual analysis.

Aspk Compiler Output for Population Analysis

The Aspk compiler takes jobs from the compiler queue of the database, compiles them, and places in the database the archive of files needed by computational servers to perform population analysis. The compiler outputs the files required by computational servers to perform population analysis.

Cspk Builds and Runs Jobs

The Cspk removes source files from the run queue that were place there by the Aspk compiler. It builds the source into executable objects and then runs them.

Cspk Generates Report

Cspk generates the user's output report, and stores it in the database for eventual retrieval by the MDA.

Cspk Object Build

This milestone builds object code, ready for execution, from source code received from the Aspk compiler.

Database API

The application programming interface (API) defines a set of functions that can be used by software processes to interact with the database. This specification is language independent.

This document is now complete and can be viewed here[2]. Return to table.

Database API, Java Binding

The Java binding is an implementation of the API for use by processes programmed in the Java language.

This milestone is now complete. Documentation can be viewed here[3]. Return to table.

Database API, Perl Binding

The Perl binding is an implementation of the API for use by processes programmed in the Perl language.

This milestone document is now complete. Documentation can be viewed here[4]. Return to table.

Database Entity-Relationship Model

Spk is implemented as a set of independent processes, which communicate with each other by means of a relation database. The entity-relationship model is the high-level design document for this database.

This document is now available for viewing here[5]. Return to table.

Database Schema

The schema is a sequence of statements in the Data Definition Language (DDL) which is a subset of the Structured Query Language (SQL). The schema can be executed by the MySQL relational database management system to create a working database that implements the entity-relationship model.

This document is now available for viewing here[6]. Return to table.

Job History Model

From a usability standpoint, Spk is organized around the notion of a *job*. In order to run a population kinetics model against a set of data, the user employs the MDA to submit model, data, and certain directives to Spk for processing. Taken together, the model, data and directives comprise a new job. On its route to completion, the job passes through a sequence of states, which constitute its history. The Job History Model defines the set of job state sequences that are possible.

This document is now complete and can be viewed here[7]. Return to table.

MDA Submits Jobs

The MDA submits jobs, by using the Java API to place XML versions of a model and data into the Aspk compiler input queue.

Model Capabilities Specification

A medium-term goal of the Spk project is to provide support for population kinetic modeling that is fully equivalent to that provided by Nonmem. Version 0.1 will support only of limited subset of these capabilities, as detailed in this specification.

This document is now complete and can be viewed here[8]. Return to table.

Network Security Implementation

This milestone is the implementation of the Network Security Specification.

Network Security Specification

This specification lays out the requirements for network security and describes a practical solution which meets the requirements.

This document is now complete and can be viewed here[9]. Return to table.

Replacement of the NAG Optimizer

The Spk function library contains a proprietary optimizer from an organization called NAG. Since version 0.1 will be licensed as open source, it is necessary to replace the NAG optimizer with one developed by the RFPK Mathematics Team. This milestone consists of integrating the new optimizer with the Spk library and testing it on against the Spk test suites.

User Output Options Implemented

User output options are implemented in the MDA.

User Output Options Specification

The output that comes back to the MDA from the computational server is XML. This specification describes what the MDA will do in terms of transforming this data into input for other analysis tools as well as doing a simple presentation of data on the user's screen.

This document is now complete and can be viewed here[10]. Return to table.

Web Site Functional Specification

The functional specification describes the web site from the point of view of the services and usability. It does not specify visual design.

This document is now complete and can be viewed here[11]. Return to table.

Web Site Implementation

This milestone is the implementation of a web site which conforms to the Web Site Functional Specification.

XML Representations of Data Set

A data set is submitted as part of a job. This data is structured as XML and is stored in the *xml_data* field of the *job* table of the database.

This document is available for viewing here[12]. Return to table.

XML Representations of Models and Control Parameters

A job contains specifications from which the ASpk Compiler creates a C++ program. The MDA submits these specifications in the form of an XML representation of a model and associated control parameters. This XML file is stored in the *xml_source* field of the *job* table of the database.

This document is available for viewing here[13]. Return to table.

XML Representation of Spk Results and Other Reports

When an Spk job runs to completion, whether successfully or not, a report is formatted as XML and is stored in the *report* field of the *job* table of the database.

This document is now available for viewing here[14]. Return to table.

Cspk Pred Class

C++ class which represents the part of the mixed-effects model which remains the same, from model to model.

Cspk Covariance Class

Software to calculate and return the covariance.

MDA Presents Population Results

The MDA presents results to a user for a population model.

Notes

1. ../../specs/sourceCpp/sourceCpp.html
2. ../../specs/dbAPI/dbAPI.html
3. ../../specs/dbAPIjava/index.html
4. ../..//specs/dbAPIperl/Spkdb.html
5. ../..//specs/erModel/erModel.html
6. ../../specs/dbSchema/dbSchema.html
7. ../..//specs/jobHistory/jobHistory.html
8. ../../specs/modelCap/html/modelCap.html
9. ../..//specs/netSecure/netSecure.html
10. ../../specs/userOutput/userOutput.html
11. ../../specs/website/website.html
12. ../../specs/dataML/dataML.html
13. ../../specs/sourceML/sourceML.html
14. ../../specs/reportML/reportML.html