DataTime Processing Framework
Project Plan And Schedule

Revisions:

CGP: 6/22/05; Revised 4A in accordance with discussion as of 6/22/05.

I. Plan

The development of DTPF is planned in several stages. The first phase of development involves creating the core framework and 'system' level applications. This will also include creating some standard or utility nodes (handling informational/error messages, framework testing, etc.) as well as a simple client application to allow testing of the client interface. The next phase of development will involve creating a more full fledged client application suitable for end users and implementing nodes intended for direct use in models.

II. Schedule

Phase 1: Core Framework

Task	Completion
1A. Create revision control reposirory (SourceForge CVS, subversion, local?) SourceForge project has been established, website documents now live in CVS. 1B. Draft formal requirements Draft document has been prepared, going through stages of editing. 1C. Update project plans Updated several times. This will continue for the duration of the project. 1D. Investigate YARP/ACE features YARP appears to provide the basics for message passing. Combined YARP/ACE provide a very capable API that should make it possible to avoid OS dependencies. YARP's message container format leaves a bit to be desired, MUSCLE looks like a good candidate for this. 1E. Explore OSX port feasibility Apple seems happy to provide information and resources to help people port to OSX. There appears to have been some work done with YARP and Darwin (Darwin is OSX's kernel, sometimes used to refer to the OSX UNIX environment). MUSLCE provides handy utility functions for endianess operations. Should only be a factor when moving data from opposite endian platforms.	End February

2A. Meet with Dr. Colburn and Dr. Shute, et. al. Want to have some feedback on what we are doing from 'outside' parties. 2B. Refine requirements for framework library, module management roster and module loading applications to a reasonable point of completeness and stability. Comparison to MediaKit, JMF, etc. will help with this.	Late March / Early April
3A. Define some possible applications for DTPF: SenseStream implemented in terms of DTPF, SoDiBot interfacing to the DTPF version of SenseStream, ESMA (June 2005 NIH grant modeling). *Goal*: Ensure that facilities planned for DTPF will enable implementation of these possible applications. 3B. Further refinement of requirements for framework library. *Goal*: move design portions of the requirements document into a design document. 3C. Investigate how ACE can support our requirements. *Goal*: Read ACE papers, and try using (i.e., compiling and running sample programs) particular ACE functionality on Mac OS X to resolve questions. (E.g., use of Streams in ACE).	May-June 2005 (Tasks to be completed by end of June)
4A. Add detail to the design for (i) Roster, (ii) Nodes, (iii) data I/O formats for Nodes (Format Models), (iv) Node parameters (Parameter Models), (v) Plugin loader, and (vi) Buffer channel communications (including sending/receiving messages and buffer groups). *Goal*: Create application programming interfaces (APIs) in DOxygen, and create primary data structures in C++ for these parts of the DTPF. (*Note*: By "Node parameters" I mean the mechanism by which Nodes specify to the GUI what information a user can change or view regarding a Node). *Some details*: Roster API will include loading a Node, unloading a Node, Node registration, querying for list of Nodes that are presently loaded, querying for a list of Nodes that can be loaded, querying a particular loaded Node for its input or output format requirement, run/pause/resume/stop for a loaded Node, establishing a communication connection between an upstream Node and a downstream Node, querying a Node for its parameters, changing a Node's parameters. 4B. Continued refinement of requirements for framework library. *Goal*: Keep requirements document up-to-date with regards to changes that arise as the design progresses. 4C. Program, debug, and carryout component tests for wrapper interface for use of ACE. *Goal*: API specified in DOxygen, and C++ class(es) that enable use of the parts of ACE that we need.	June-July 2005 (Tasks to be completed by end of July)

Hosted on