MDMC Users Guide Outline

A. Demo User Guide
    a. Batch files – ordering and timing.
    b. Batch files – configuration.
    c. Configuration files – customization.
    d. Example:
B. Component Building User Guide
    a. Specific component construction
        i. ProgramComponent
            1. Constructor – ComponentSetup
            2. Actions
        ii. Program
    b. Map the elements in a running system to a component designed to solve the given problem
        i. Selecting relevant components
            1. TargetTemplate match functions
            2. Protocol Groups
        ii. Identify the proper scope for a given relevant component
            1. Protocol group and scope identification
            2. Protocol group and new additions
    c. Add/Remove, or maintain existing tolerance components
        i. Protocol and new additions
        ii. Individual component customization.
        iii. Accessing existing components.
C. General Design Elements
    a. Common interface to distribute components across a system.
    b. Common interface to identify components that meet criteria.
    c. Mechanism to add components to new criteria
    d. Subscription to new component additions
    e. State-based action semantics
    f. Busy-wait loops

A. Demo User Guide

a. The demo is run by batch files. Resources must be started before their corresponding consumers, and r1 must be started before any other resource. Furthermore, each component must wait for the previous component to announce that its setup has been completed. (It will display setup completed once the initialization work is done). This may require patience.
b. Each batch file will run correctly if it is run from the same directory as the executable (and all of the libraries). They will run from any directory if the user changes the `SET RUN_PATH=%CD%` line to `SET RUN_PATH =[pathname]` where `[pathname]` is string that represents the directory where the executable file is located.
c. Each batch file uses a configuration file. These may rearranged in any way as long as each consumer refers to a resource that has previously been launched and all resources are connected.
d. Example:
    i. Go to the DGenericServer\bin\debug directory
    ii. Click on r1.bat Wait for the screen to prompt "setup complete"
    iii. Do the same for all of the remaining batch files, r2, r3, c1, c2, c3, c4, c21, c22, c23, c24, mutx
    iv. All accesses will eventually be controlled exclusively, except for r3, which is configured as Green, which is not recognized as requiring exclusion.

B. Component Building User Guide
    a. Specific component construction
        i. ProgramComponent
            1. Constructor - Program components are constructed using component setup objects. This infrastructure was created so that the physical interface of component construction can reflect the required interface of components as closely as possible.
            2. Actions - Actions are made by assembling Boolean functions with void functions and a string identifier that uniquely identifies actions within programs. This identifier may be used to update the guard of any action using the update guard member function of the component.
        v. Programs – Each target and proxy can have a program associated with its send and receive interceptors. Components may be added to these programs and actions to the components.

    g. Map the elements in a running system to a component designed to solve the given problem
        i. Selecting relevant components
            1. TargetTemplate match functions

2. Protocol Groups

ii. Identify the proper scope for a given relevant component
    1. Protocol group handles scope (protocol instance) identification, and forwards instances to protocols
    2. Protocol group attaches itself to a list of listeners for the platform.

h. Add/Remove, or maintain existing tolerance components
    i. Protocol handles additions and losses to components controlled by its own instance
    ii. Each local component within a protocol constructs itself based on input parameters. The set of these performs the logic that provides the desired tolerance to the underlying system.
    iii. For modification or removal, each action can be accessed by the unique identifier it possesses within its component by use of the updateguard command (The current version only modifies synchronous actions). This can be used to inactivate all of the actions of a component, or effectively, remove it.

D. General Design Elements
    a. Common interface to distribute components across a system. - This requires that the system become connected somehow. The current implementation uses synchronous calls to fully connect the network to achieve this and is a definite area for improvement.
    b. Common interface to identify components that meet criteria. - This involves associating components with their types and users. Some form of this is necessary in order to analyze semantic aspects of a system that has already been started.
    c. Mechanism to add components to new criteria - This is just an interface that formalizes the combination of a and b.
    d. Subscription to new component additions - Each platform is given a listener in step c.
    e. State-based action semantics - This facilitates debugging and the reuse of state-based components within the framework. It adds a little overhead to save state as events of interest occur.
    f. Busy-wait loops - Some form of retry logic is part of most if not all stabilizing protocols. Our literal implementation of this is the major cause of slowness within the system. Discovering a nice mechanism to perform retries in a DRSS type system with out bogging down the entire system is a very interesting and probably open topic. I will let you all know if I get a chance to find anything here.