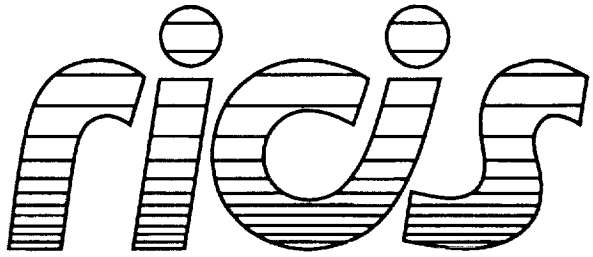
Mission and Safety Critical (MASC) Plans for the MASC Kernel Simulation

GHG Corporation

11/15/91

Cooperative Agreement NCC 9-16
Research Activity No. SE.16

NASA Johnson Space Center
Information Systems Directorate
Information Technology Division



Research Institute for Computing and Information Systems
University of Houston-Clear Lake

INTERIM REPORT

The RICIS Concept

The University of Houston-Clear Lake established the Research Institute for Computing and Information Systems (RICIS) in 1986 to encourage the NASA Johnson Space Center (JSC) and local industry to actively support research in the computing and information sciences. As part of this endeavor, UHCL proposed a partnership with JSC to jointly define and manage an integrated program of research in advanced data processing technology needed for JSC's main missions, including administrative, engineering and science responsibilities. JSC agreed and entered into a continuing cooperative agreement with UHCL beginning in May 1986, to jointly plan and execute such research through RICIS. Additionally, under Cooperative Agreement NCC 9-16, computing and educational facilities are shared by the two institutions to conduct the research.

The UHCL/RICIS mission is to conduct, coordinate, and disseminate research and professional level education in computing and information systems to serve the needs of the government, industry, community and academia. RICIS combines resources of UHCL and its gateway affiliates to research and develop materials, prototypes and publications on topics of mutual interest to its sponsors and researchers. Within UHCL, the mission is being implemented through interdisciplinary involvement of faculty and students from each of the four schools: Business and Public Administration, Education, Human Sciences and Humanities, and Natural and Applied Sciences. RICIS also collaborates with industry in a companion program. This program is focused on serving the research and advanced development needs of industry.

Moreover, UHCL established relationships with other universities and research organizations, having common research interests, to provide additional sources of expertise to conduct needed research. For example, UHCL has entered into a special partnership with Texas A&M University to help oversee RICIS research and education programs, while other research organizations are involved via the "gateway" concept.

A major role of RICIS then is to find the best match of sponsors, researchers and research objectives to advance knowledge in the computing and information sciences. RICIS, working jointly with its sponsors, advises on research needs, recommends principals for conducting the research, provides technical and administrative support to coordinate the research and integrates technical results into the goals of UHCL, NASA/JSC and industry.

RICIS Preface

This research was conducted under auspices of the Research Institute for Computing and Information Systems by GHG Corporation. Mr. Charlie Randall served as GHG Project Manager. Dr. Charles McKay served as RICIS research coordinator.

Funding has been provided by the Information Systems Directorate, NASA/JSC through Cooperative Agreement NCC 9-16 between the NASA Johnson Space Center and the University of Houston-Clear Lake. The NASA technical monitor for this activity was Ernest M. Fridge III, Deputy Chief of the Software Technology Branch, Information Technology Division, Information Systems Directorate, NASA/JSC.

The views and conclusions contained in this report are those of the author and should not be interpreted as representative of the official policies, either express or implied, of UHCL, RICIS, NASA or the United States Government.

Plans for the MASC Kernel Simulation

1.0 Introduction

The purpose of this report is to discuss the MASC Kernel simulation. It explains the intended approach and how the simulation will be used. Also it introduces the scenario used in the simulation and gives details about how the simulation works. Finally it discusses the expected results of this work and outlines future work directions.

In the long term, this simulation will form the basis for both a simulator and a kernel. In the final MISSION system, there will be a simulator which allows users to try out new ideas and test new components and programs before their integration into the working system. The experience gained from the current simulation effort will form the basis for the simulator. The majority of the current simulation though will focus on the kernel. Much of the simulation work at this level will concentrate on simulating the kernel and interactions with the kernel.

The MASC Kernel has been explained in our earlier report on the MASC Kernel. This simulation is intended to be a prototype of the kernel. As such it will be a "quick and dirty" one. Its purpose is not so much to provide a truly working kernel in ever aspect, but instead provide a make-shift kernel to test the practicality of the ideas introduced in the MASC Kernel report. This prototype will be an evolving one starting with a forerunner of the kernel as it is now envisioned. Eventually the prototype, or more precisely the ideas and experience gained from it, will be transformed into a working kernel. This does not mean that the prototype will itself turn into the eventual MASC Kernel, but will probably serve as the basis for a redesign, and perhaps redesigns, of the MASC Kernel.

The simulation prototype will be built in Smalltalk although the final work, i.e. the kernel, will be implemented in Ada via Dragoon. The reason for choosing Smalltalk for the prototype is Smalltalk's usefulness in quickly building working models of systems and its object-oriented approach to software. The eventual system will use such an object-oriented approach in Ada through Dragoon. Using Smalltalk allows much greater flexibility in the prototype. Smalltalk is an interactive language and program development environment which facilitates experimental building and reusing of code without prolonged compile and link sessions. Smalltalk is much more conducive to a "design-prototype-refine" approach. (This same increase in flexibility though would cause a major maintenance dilemma in the final system.)

The simulation will introduce the objects of the kernel and allow the fine-tuning of their implementation and execution. The idea of an object-oriented kernel is a new one and some of our ideas have not been tried before. The prototype enables different options to be evaluated with regard to the various objects within the kernel and their interactions among themselves. Also the mechanics of using these objects and their associated messages will be more clearly identified through experimentation with the kernel simulation. The detailed operations of how these messages interact will be determined.

To implement the simulation, a scenario using elements typical of those in the Space Station, has been created. The main thrust of this scenario is to demonstrate the use of nested transactions. Use of the scenario with the simulation will demonstrate that the kernel appropriately manages nested transactions.

At this time the simulation is in the design stage. Some aspects of the design, much of which is defined in relation to the scenario, will be shown. The elements of the scenario are broken down and shown how they interact with components within the kernel. For example transactions within the scenario are identified. In turn it will be demonstrated how these transactions are associated with transaction objects within the kernel. Their relationship is detailed along with some implementation notes regarding the prototype.

The report concludes with a discussion of some of the future directions for the simulation. Specifically some of the plans for the prototype are examined. Using this as a springboard, the plans for turning this prototype into a design and the eventual implementation of the actual simulation are described.

2.0 Scenario

Eventually it is hoped to simulate all aspects of the MASC kernel, but the initial focus will be on the simulation of distributed, nested transactions. The example which will be simulated is a transaction required to safely implement part of the EVACS (Extra Vehicular Activity Control System), a subsystem of the Space Station responsible for monitoring and controlling astronauts' Extra Vehicular Activities (EVAs). For the purposes of this simulation we will be concerned with the part of the EVACS system responsible for maintaining communication channels between the space station and the astronauts engaged in EVAs.

In general there will be several astronauts working around the space station at any one time, each maneuvering by means of a special back-pack known as a Manned Maneuvering Unit (MMU). Amongst other things an MMU has an embedded computer which is responsible for activating thrusters in response to the astronauts commands and for facilitating communications with the space station. When an astronaut wishes to speak with colleagues on the station his/her voice is digitized and transmitted as a series of small packets from a radio antenna on the MMU to various radio antennas on the space station. Communicating in the opposite direction from space station to the MMUs is obviously achieved in the same way.

The software on the MMUs' embedded computers also uses this medium to communicate with software modules on the space station, and thus represent nodes of a large distributed system. The MMUs interact mainly with the so called "Central Control Unit" (CCU or central controller), a kind of mini "mission control" on the space station from which crew members can direct activities such as EVAs.

It is obviously important that MMUs should be able to communicate with the space station at all times, and at any point in space around the station. Since each antenna on the space station will only provide radio coverage for part of the space surrounding the space station, to provide complete, uninterrupted coverage several antennas broadcasting the same signal have to be distributed around its exterior. Furthermore, to avoid interference each active MMU is allocated a unique frequency. A message exchange between a particular MMU and the station must be broadcast by every antenna simultaneously. Since there will often be several active MMUs, each antenna must therefore maintain a mapping between active MMUs and their allocated frequency. This suggests that antennas require intelligent controllers which form part of the station's distributed system and can interact with other nodes such as the CCU. As far as communications frequencies are concerned, however, an MMU is required only to record its own frequency.

Transactions become useful when one considers the problem of changing the frequency allocated to a particular MMU. This might be necessary to avoid interference from the Earth's magnetic fields or the Van Allen belt, or to reduce the chances of undesired eavesdropping. Assuming the need for such an operation, it clearly has properties that lend it to implementation as a distributed nested transactions. The overall action of changing the frequency allocated to one MMU requires state changes in several distributed objects; the antennas and the MMU concerned. If these state changes were implemented using normal procedural abstractions a failure in one of the processors or the communications system while a frequency change was being performed could result in one or more antennas or the MMU being in an undefined state, or being in mutually inconsistent states. Clearly the nature of the communications subsystem means that both these situations should be avoided at all costs since they either diminish, or completely eliminate the system's ability to communicate with the MMU involved. Should a fault or failure occur which obstructs the system's ability to successfully complete the frequency change it is clearly much more desirable for the system to maintain its old state rather than move to an erroneous or inconsistent one.

The frequency change operation is clearly best implemented as a distributed nested transaction, therefore, since this ensures that the system would only change state if it is able to do so successfully. The master transaction is the "global" frequency change operation, and the nested transactions are the operations to change the antennas and MMU individually. Figure 1 illustrates the objects participating in the master transaction, and the "operations" involved.

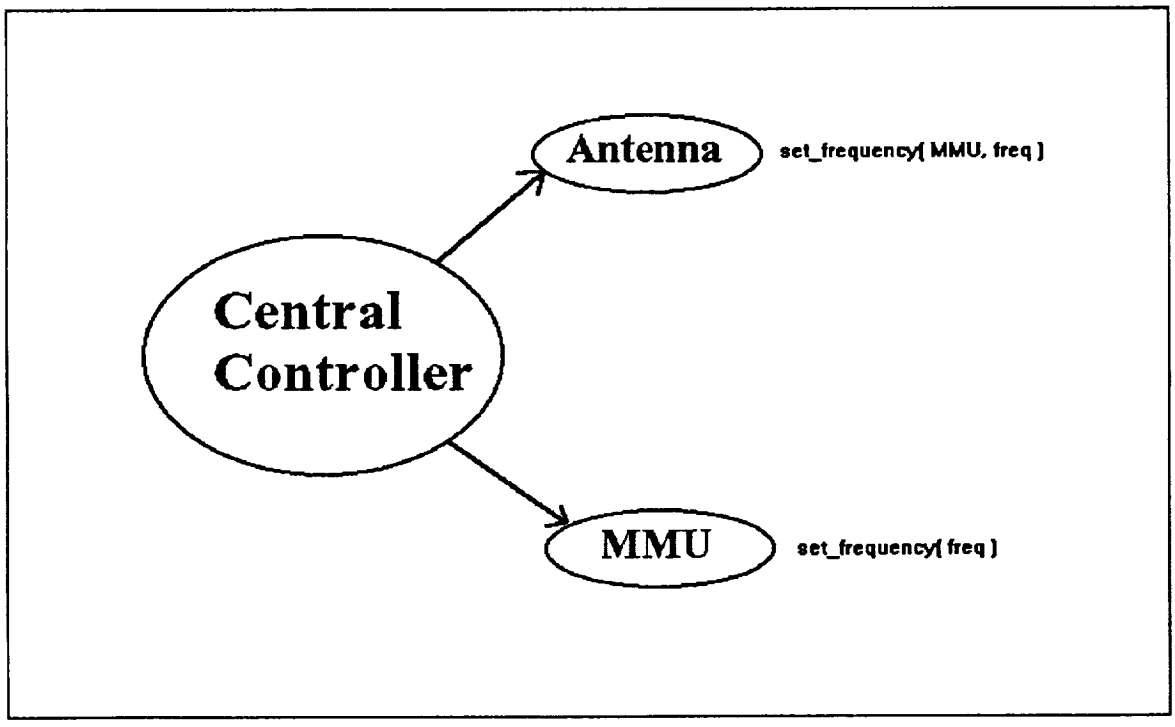


Figure 1

In order to demonstrate the successful implementation of this distributed nest transaction it is necessary to illustrate how the state variables storing the communication frequencies of the various participant objects change during the execution of the simulation. For each

active MMU and antenna, therefore, a window is created which displays the current value of the relevant state variables. In the case of the MMUs this is simply a variable of type FLOAT which stores the frequency at which the MMU thinks it is currently communicating. In the case of the antennas, however, this is an array of FLOATS which stores the frequency for each currently active MMU. By displaying these values prominently on the screen the viewer will be able to observe whether consistent frequencies are maintained.

To complete the simulation it is necessary to simulate the effect of processor and communication failures. This is achieved by providing the antenna, MMU, and CCU objects with an additional method which can be used to set them in a failure mode. A failure in the simulation is simulated by a message not being returned. There are several possible cases. One is where the receiver object successfully does the requested operation and then fails without returning any indication that the operation has been carried out. Another is when it fails prior to doing the operation or never acquiring the message through a communications failure or the message not being sent or addressed properly.

Any of the objects participating in the transaction may be set to simulate any of these failure modes from a "simulation control object" which provides the user's input interface. (See Figure 2)

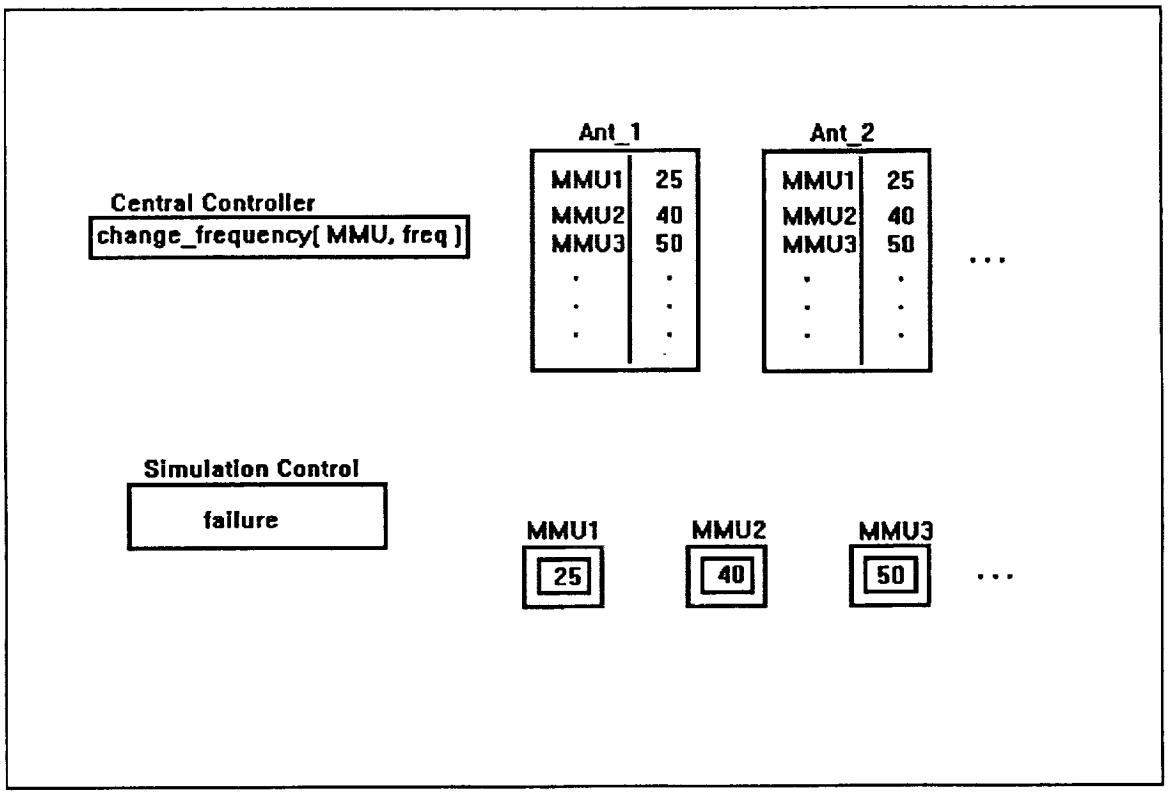


Figure 2

3.0 Simulation

This report reflects only the first pass of work on the simulation, which introduces the plan for the design of the simulation. The details and implementation will follow in

subsequent passes and reports. This plan is a high-level one, the purpose of which is to start the process of developing the simulation. This report does not go into any great detail describing the actual design, only an overview is given. The plan for the first pass breaks down into three steps:

1) Design of simulation, with scenario, at a high level.

2) Elaboration of each of the kernel elements.

3) Design of mechanisms necessary for simulation control and monitoring.

These steps are discussed in the following sections. To help in the discussion, two diagrams are introduced which are built upon the scenario given in the previous section. These figures are based on the current preliminary design for the simulation. Figure 3 shows the relationship between the central controller objects and the antenna objects. It is similar to the diagram used in the section describing the scenario. But notice that instead of illustrating just the elements of the scenario, it includes some of the kernel objects necessary to handle the scenario. Specifically these are the lock objects and the transaction objects. Notice that there is a lock object associated with each object that must be locked. There is a transaction object for each transaction between other objects.

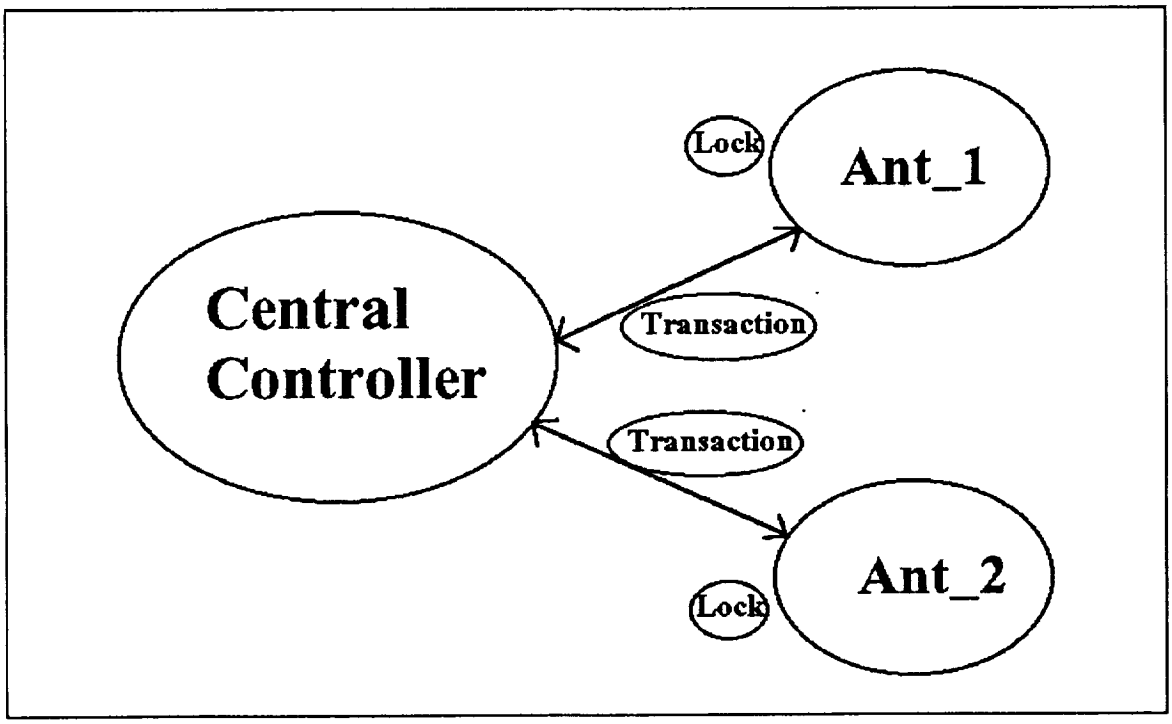


Figure 3

Figure 2 shows the preliminary user interface for the simulation. The user interface is typical of the screen display a user sees during a simulation session. This typical screen includes the various objects introduced by the scenario (the central controller object, the antenna objects, and the MMU objects) and the simulation (the simulation control object). Some of the elements of the display are strictly to show the current state of those

elements while others allow interaction with the element in question. For example, a failure can be initiated by simply selecting it on the screen, i.e., selecting the failure method within simulation control box. A window would then pop up at that point for the user to select further aspects of the desired failure (e.g., what message is to fail, between which two objects).

3.1 High level design of simulation

The first step is the design of the simulation, with the scenario, at a high level. This design, in the follow-up work to this report, will then be hand-translated into simulation code. Specifically in our case, it is translated into Smalltalk.

As described above in the description of the scenario, the scenario consists of three basic classes of objects: the central controller, the Antennas, and the MMUs. In addition the simulation itself introduces the simulation control object. Besides these four classes of objects there will be object classes for kernel elements. Ignoring the simulation control and kernel objects for the moment, the basic interaction within the scenario objects is as follows. The central controller is responsible for initiating any frequency changes. One of its operations is to change the frequency of a particular MMU to a specific frequency. A message to this effect is sent to all Antenna objects and to the MMU object in question. The mechanics of a nested transaction are used to insure the complete and consistent updating of all objects involved. These include the passing back and forth of pre-commit and commit messages between the central controller object and the Antenna and MMU objects.

3.2 Elaboration of kernel elements

In the second step, each of the kernel elements are elaborated. Each kernel element is described along with any special interactions that might be necessary for the simulation. Those elements involved in the actual simulation/scenario will be addressed first, and probably, more thoroughly. These elements were described in the MASC Kernel report.

As an example consider an interaction between the central controller object and an antenna object. Suppose that the central controller sends a message to the antenna changing the frequency for a specific MMU. Associated with the antenna object is a lock object that controls the changing of the frequency values that the antenna tracks. Associated with the transaction between the central controller object and the antenna object is a transaction object to control the logistics of the transaction. This situation is illustrated in figure 3. The full design will consider all of the necessary kernel objects and their interactions within the scenario.

3.3 Design of simulation mechanisms

The final step is the design of the mechanisms necessary for the simulation control and monitoring. This includes the introduction of errors and faults into the system. The simulation will interject these errors and faults and then facilitate the management of them. A user will be able to choose which error or fault is imposed upon the system and then control and monitor the resulting activity.

Again consider the situation depicted in Figure 2. A separate object, the simulation control object, oversees the introduction of errors and faults. In this case, an interactive user can choose to have a failure by simply invoking the related method within the simulation control object.

Other example mechanisms include those associated with debugging. One such mechanism would present the user with a window which tracks the system's current execution. The window might display a message indicating each method as it is invoked. This display might include the invoked method's name, any parameters passed with the message, and the originating object's identity.

4.0 Conclusions

In conclusion, the prototype for the kernel simulation will be done in Smalltalk. Smalltalk was chosen because of its "design-prototype-refine" flexibility. The eventual system will be a fully object-oriented one implemented in Ada via Dragoon. The Smalltalk system is only for the prototype, the other team members working on the MISSION project need not use it, although they can take it and use it if they so desire.

This report has given an overview of the plans for the prototype of the simulation with regards to the scenario. The next step is to complete the design of the prototype (based partially upon feedback to this report), implement it and report on its building and any lessons learned. The process will be to build the scenario related portion first. Once this is fully implemented and understood, it will serve as a basis for generating the full simulation.