N89-16374

Artificial Intelligence & the Space Station Software Support Environment

Gilbert Marlowe

In a software system the size of the Space Station Software Support Environment (SSE), no one software development or implementation methodology is presently powerful enough to provide safe, reliable, maintainable, cost-effective real-time or near real-time software. In an environment that must survive one of the harshest and lengthiest lifetimes, software must be produced that will perform as predicted, from the first time it is executed to the last. Many of the software challenges that will be faced will require strategies borrowed from "Artificial Intelligence (AI)." In the Statement of Work (SOW) for the SSE, AI is the only development area mentioned as an example of a legitimate reason for a waiver from the overall requirement to use the Ada™ programming language for software development. While it

Ada™ is a Registered Trademark of the United States Government, Ada Joint Program Office (AJPO)

is recognized that some solutions are not readily amenable to solution in contemporary Ada Programming Support Environments (APSEs), it is clearly the intent of the SOW that there be one development language for all of Space Station software so that configuration management, system definition and reuse of verified and validated software be as simple and as efficient as possible. This paper will attempt to define the limits of the applicability of the Ada language, APSEs (of which the SSE will be a special case), and software engineering to AI solutions by describing a scenario that involves many facets of AI methodologies.

The scenario itself is fairly simple. It involves the Space Station, an undocked Space Shuttle, and a robot unattached to either the Space Station or the nearby Shuttle (the robot is equipped with vision sensors, a propulsion system with translational and rotational jets, and manipulators/grapplers). The robot will start in proximity to the Station either stationkeeping or performing a low priority task that may be preempted. At the request of one of the specialists onboard the Station the robot begins to maneuver itself to the nearby Shuttle. If the Shuttle is near enough, the robot will be guided by the Station Trajectory Control Program. If the Shuttle is any appreciable distance away the robot will request Guidance, Navigation and Control (GN&C) programs necessary to compute and maintain a trajectory to the Shuttle. It may also request the Station Trajectory Control Program to calculate intermediate vectors that it will use to compare against during the rendezvous. While all of this was happening, the specialist onboard the Station identified and requested a software load in addition to the GN&C software being loaded by the robot. This software included a vision system, general GN&C programs to be used in proximity operations at the end of the rendezvous (this could be detailed enough to allow the robot to literally settle down in a specified position and attitude in the cargo deck of the specified Shuttle without any human intervention, or it might allow a specialist onboard the Shuttle to interactively guide the robot to the desired location and attitude) and a task identification that will establish whether or not this task may be preempted and, if it can, by what other tasks or levels of tasks. Once the necessary software has been loaded, the robot is essentially a free agent and must vie with other agents for Station computing resources. As soon as it begins an escape trajectory, the robot begins to interface with the Station Collision Avoidance Program (CAP) to establish and maintain a clear trajectory. Very likely the robot and the Station will enter a dialogue, with the robot proposing a trajectory and the CAP either accepting the proposed trajectory or denying it. If the trajectory is denied, it is the responsibility of the robot to calculate another trajectory, using Station computing facilities if necessary. This cycle of calculation, proposal, and verification will proceed until an acceptable trajectory is proposed, acceptable meaning that the proposed trajectory does not involve undue risk of collision between the robot and the Station or the robot and other free flyers, and that the proposed trajectory is reasonable given the amount of propulsive and non-propulsive consumables that have been budgeted for this task (a configuration item that will be maintained by the Station Object Base). The robot is responsible for calculating a trajectory that meets the specified goals :

that the rendezvous occur within a specified amount of time, that the rendezvous cost no more than a specified amount of non-reusable resources, and that the rendezvous occur with a specified object (rather than that the rendezvous occur at a specified place). The Station maintains configuration control over trajectories using the CAP and will not allow trajectories that violate safety standards.

After the robot has negotiated a safe trajectory, it still must maintain a dialogue with the Station so that both are aware of the robot's current and predicted position in any given time quanta. This dialogue is necessary to keep the CAP current and so that the robot may be informed of any changes in the trajectory or in the task.

When the robot arrives in near proximity to the Shuttle it has been assigned to rendezvous with, it will announce itself to the Shuttle computers. At this point, depending on the software loaded at the Station, the robot may or may not be able to proceed to dock without any human intervention. If it is capable, the robot will inform the

Shuttle computers, and begin a docking sequence. At any time the humans onboard may elect to override the automatic docking sequence and control the robot through their onboard computers. If the robot has not been loaded with the appropriate software, it will announce this and wait for further instructions. Shuttle specialists may decide to either load the software necessary for an automatic docking sequence into the robot, or manually control the docking sequence.

Once the robot is securely docked, a specialist in the cargo bay begins refurbishment and outfitting of the robot. The old manipulators/grapplers are removed and new ones are attached. The robot is refilled with consumables for the next segment of its task and, in parallel with all of this activity, new software is loaded into the robot. This new software will guide the robot to a satellite at a geosynchronous altitude, direct the robot to grapple the satellite (which will require the robot to make contact with the satellite in a very specific attitude with very specific rotational and translational velocities as well as a sequence of grapple maneuvers that must be performed as directed to ensure stability), and return to the Shuttle or to the Station so that the satellite may be repaired. Alternatively, if the repair is simple enough (such as increasing the spin of the satellite) the robot may perform the indicated repair and return to the Station (if supplies of consumables allow return to the Station rather than refueling at the Shuttle). The close in proximity operations immediately preceding the grapple will require a number of real-time computations. The robot must visually confirm that the satellite is the correct one, that the approach is proceeding nominally, and that grapples are being manipulated in the correct sequence and towards the correct targets on the satellite. Trajectory programs in the robot must calculate burns that will match translational and rotational velocities of the two vehicles and manipulator control programs must monitor and guide grapplers from an unsteady platform toward targets that are moving. As soon as the manipulaotr control program confirms that the satellite has been securely grappled, the robot begins to contact the Station. It reports the successful completion of rendezvous, approach and grapple and again enters negotiation with the CAP, this time for a return trajectory. When a acceptable trajectory (which will be based on new mass properties and different consumables loadings that reflect the current robot/satellite pair's configuration characteristics) has been agreed to by both parties, the robot will begin its trip home to the Station. As before, the robot will maintain contact with the CAP and perform maneuvers as required or as requested by the CAP until it is docked at the Station.

This scenario illustrates the flexibiltiy offered by allowing a general-purpose robot to serve as an free agent to perform a task that would be uneconomical if performed by humans or if performed by a robot that could not perform unless guided by humans or Shuttle or Station computers. A robot may be treated as an agent and allowed to compete with other agents for computing and other shareable resources to maximize the efficient use of those resources. Obviously computing time and consumables will both be at a premium for the

Station since neither is a renewable resource. Just as obviously, it is more efficient to send a robot to do many tasks rather than sending a manned vehicle with the life-support system that it must provide. An added benefit to treating the robot as a separate agent is that in the event of a communication failure the robot would be able to continue the task until such time as communications are restored or the task requires communication (such as the negotiation for trajectories described before). This also makes efficient use of human resources and offloads computing work to the responsible agent - the robot. A subtle, but important, benefit is that this approach separates the specialists from details about how the robot fullfills the task assigned to it (similar to the way that object-oriented design hides implementation details from the user) allowing him/her to worry about the overall task rather than details that are subject to change dynamically (such as a trajectory that fullfills the task requirement without violating Station safety constraints).

All of the software discussed in this paper should be implemented in

Ada™ to ensure consistancy of interface between the software modules. The Ada™ construct of packages will allow software to be developed in modules that are additive to the total software functionality. The time is now to start deciding not <u>whether</u> Ada™ should be used for AI applications on the Station, but <u>how</u> to efficiently use the power of Ada™ to develop software modules that are sufficiently well engineered to meet real-time requirements in problem spaces that may not allow a complete description at any given time. To introduce another language on Station doubles the complexity of configuration management. To introduce another language on Station that cannot support strong typing will double again the configuration management task. It is clear that Ada™ is <u>sufficient</u> for many applications in AI, but it is not clear that another language is <u>necessary</u> for AI applications or that a trade off between power in expressing a solution using a "traditional" AI language (i.e. Lisp, Prolog) and the resources required to maintain any type of configuration control (including verification, validation, testing and safety data) over a configuration item produced using that language is worth the price. Perhaps it is too early to tell, but it is my hop that by discussing now what the Station will require in the future we may have a crystal vision of our intermediate and long-term goals and the tools we will use to reach those goals. I think that discussing scenarios such as the one above will prove fruitfull in determining the direction that the Space Station SSE will take.

Gilbert Marlowe
c/o Rockwell Shuttle Operations Company (RSOC)
600 Gemini Blvd.
Houston, Texas 77058
(713) 282-2760

NASA Mail code : RS16

Acknowledgements :

Special thanks to Dr. Charles W. McKay for his support and guidance.

Thanks also to Lisa Willingham for moral support and the abilty to listen to me ramble on long after a mere mortal would have gone to sleep.