N86-23315

SPACE STATION - THE ROLE OF SOFTWARE

Dana Hall*
NASA Office of Space Station
Washington, DC

ABSTRACT

Software will play a critical role throughout the Space Station Program. This presentation is intended to set the stage and prompt participant interaction at the Software Issues Forum. The presentation is structured into three major topics:

- an overview of the concept and status of the Space Station Program;
- several charts designed to lay out the scope and role of software;
- and information addressing the four specific areas selected for focus at the forum, specifically: software management, the software development environment, languages, and standards. The presentation attempted to highlight NASA's current thinking and to raise some of the relevant critical issues.

*Dr. Dana Hall is the Level A Space Station software manager and is responsible for oversight of the planning, implementation, and integration of all Space Station Program software. Prior to joining the Program in October 1984, Dr. Hall served as a data system and software advisor within NASA's Office of the Chief Engineer. His prior experience is with MITRE and TRW where he has worked with projects ranging from airline operations models to missile trajectory simulations. Dr. Hall has also been involved with NASA data system advanced development and in the ground system design of several NASA spaceflight programs.

NEXT LOGICAL STEP

Given the advent of an operational space transportation system, the Space Shuttle, the development of a space station is the next logical step in mankind's exploration of the surrounding universe.

STATE OF UNION

The Space Station Program traces its official beginning to the January 1984 State of the Union message by President Reagan in which he directed that NASA proceed to develop a "permanently manned space station and do it within a decade." This official start builds upon many years of prior analyses and considerations that together laid the basic guidelines that now comprise the Space Station Program.

"We can follow our dreams to distant stars, living and working in space for peaceful, economic and scientific gain. Tonight, I am directing NASA to develop a permanently manned space station and to do it within a decade.

A space station will permit quantum leaps in our research in science, communications and in metals and life-saving medicines which can be manufactured ... in space."

January 25, 1984

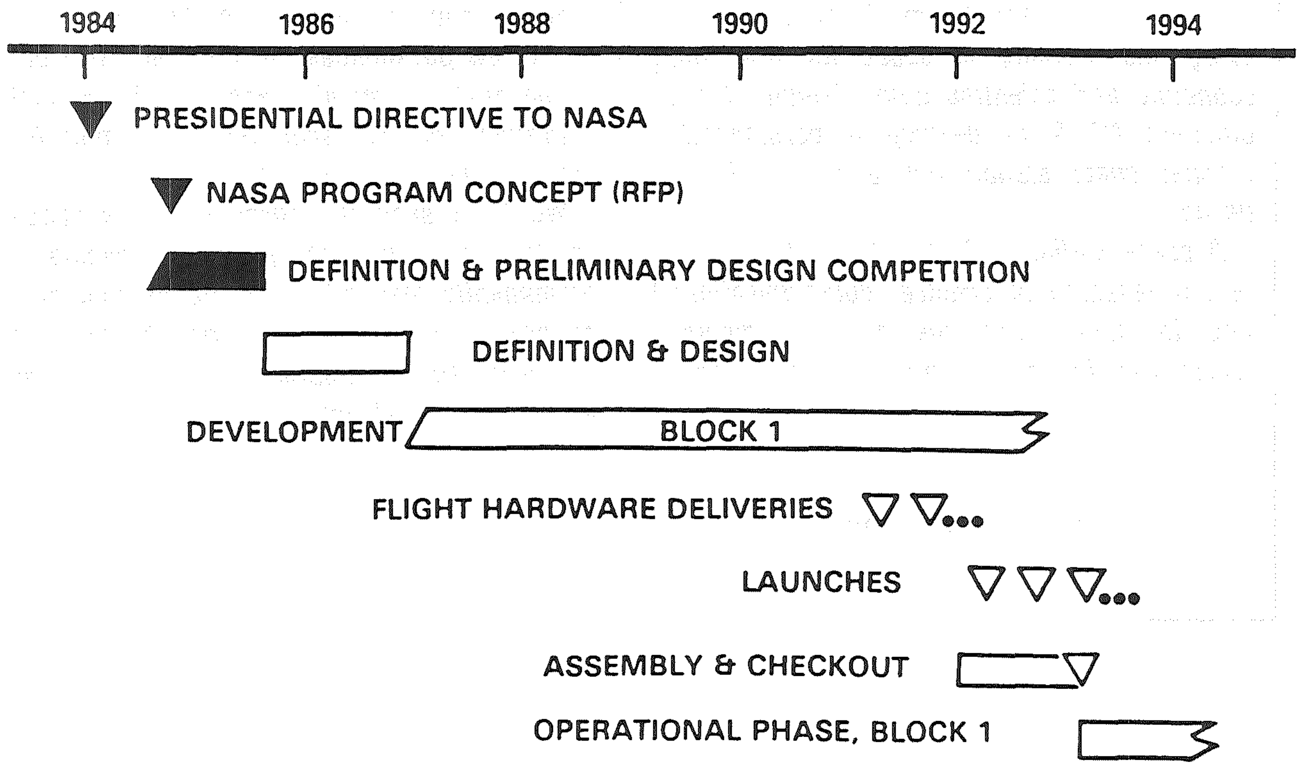
"Our Second American Revolution will push on to new possibilities not only on Earth but in the next frontier of space. Despite budget restraints, we will seek record funding for research and development.

We have seen the success of the space shuttle. Now we are going to develop a permanently manned space station and new opportunities for free enterprise because in the next decade, Americans and our friends around the world will be living and working together in space."

February 6, 1985

MILESTONES

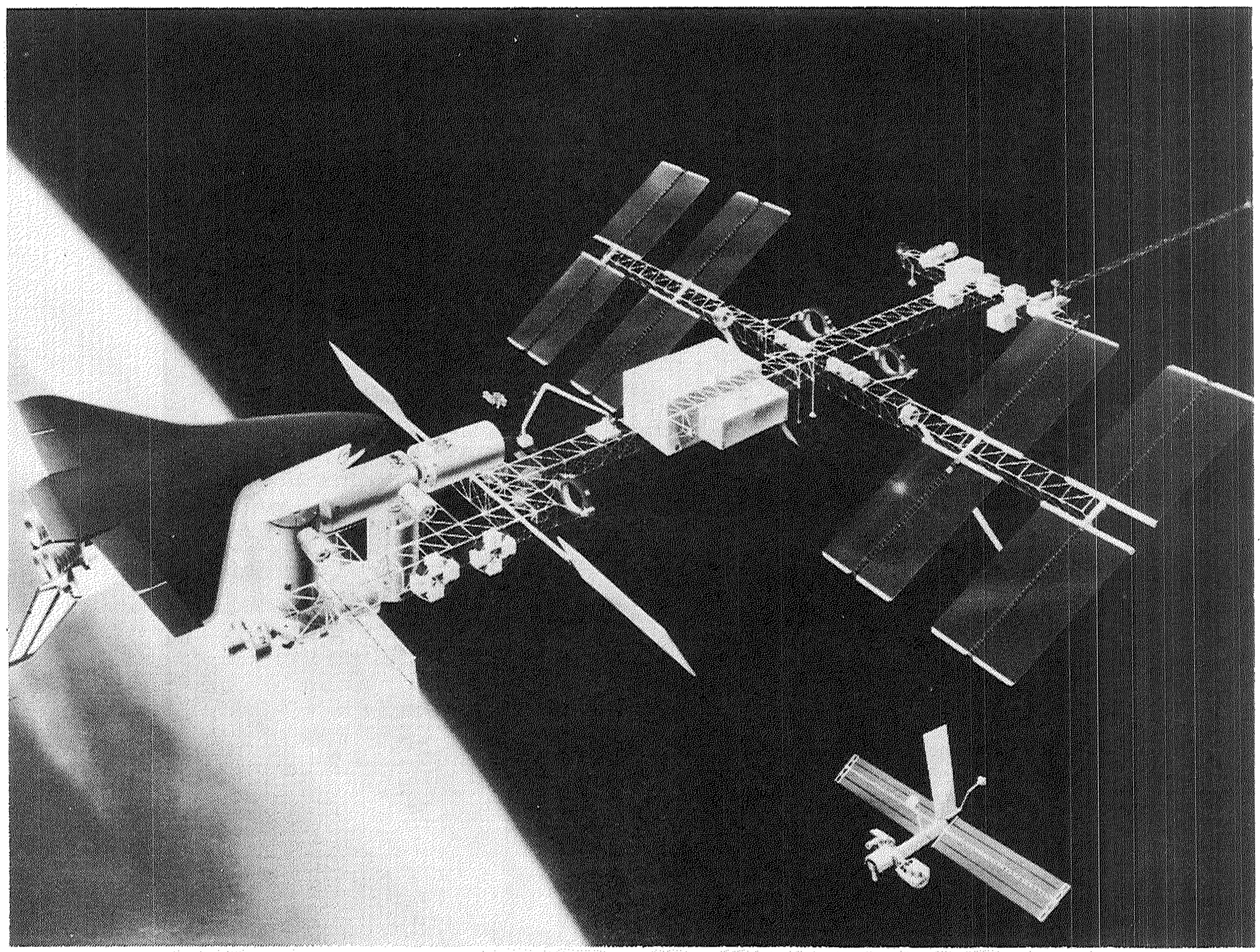
At the time of this forum, the program had just completed the competition for the definition and preliminary design of the Space Station elements. This competition resulted in the award of eight major contracts distributed across four primary work packages. As shown on this schedule, it is planned that actual development (i.e., Phase C/D) will begin in 1987. Initial operational capability is forecast for the 1993-94 time frame.



ORIGINAL PAGE IS
OF POOR QUALITY

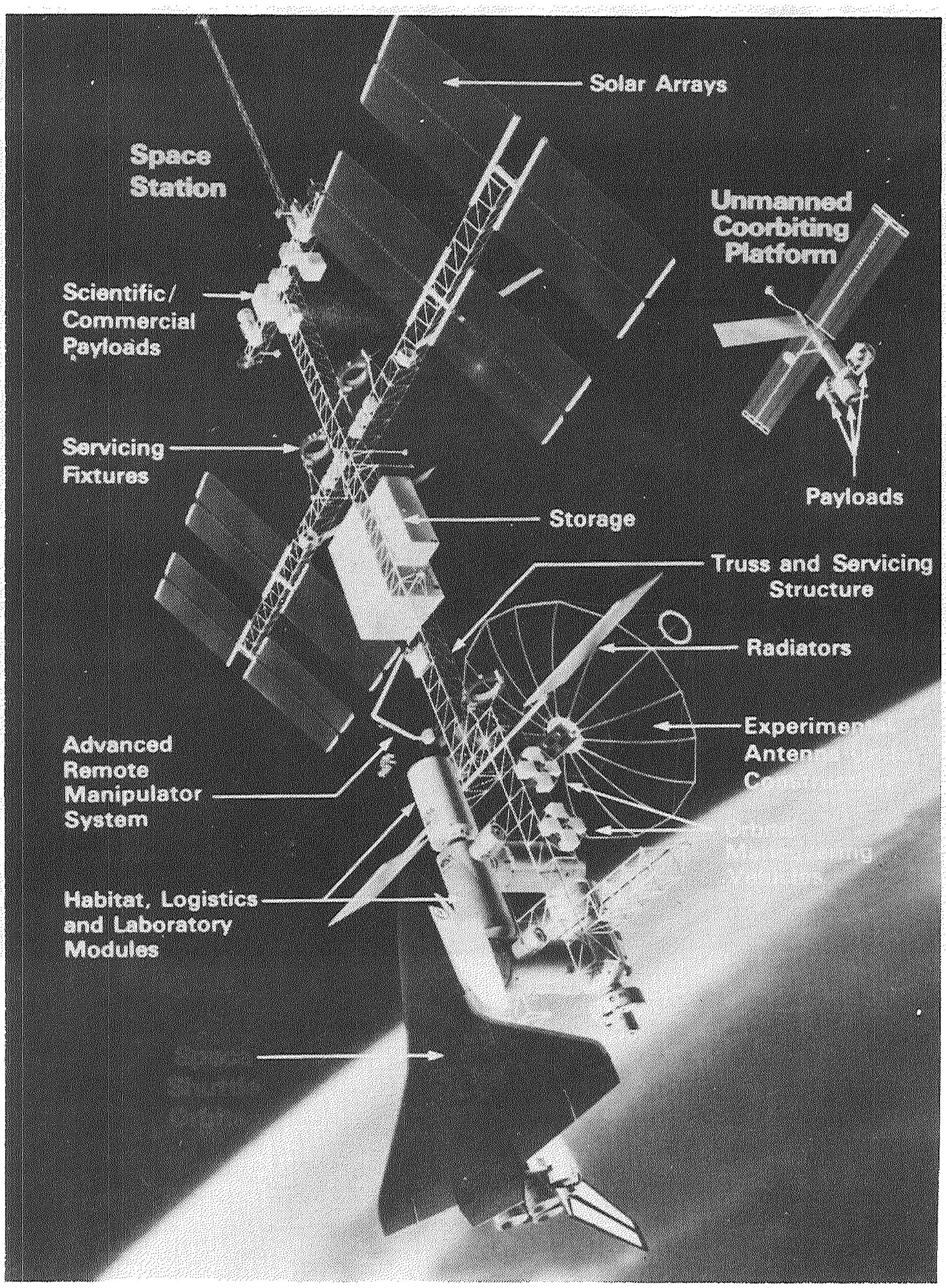
SPACE STATION DESIGN

The actual design of the Space Station is not known at present since the program is still in the requirements and definition part of its life cycle. However, NASA had adopted a reference configuration, as shown in this artist's concept.



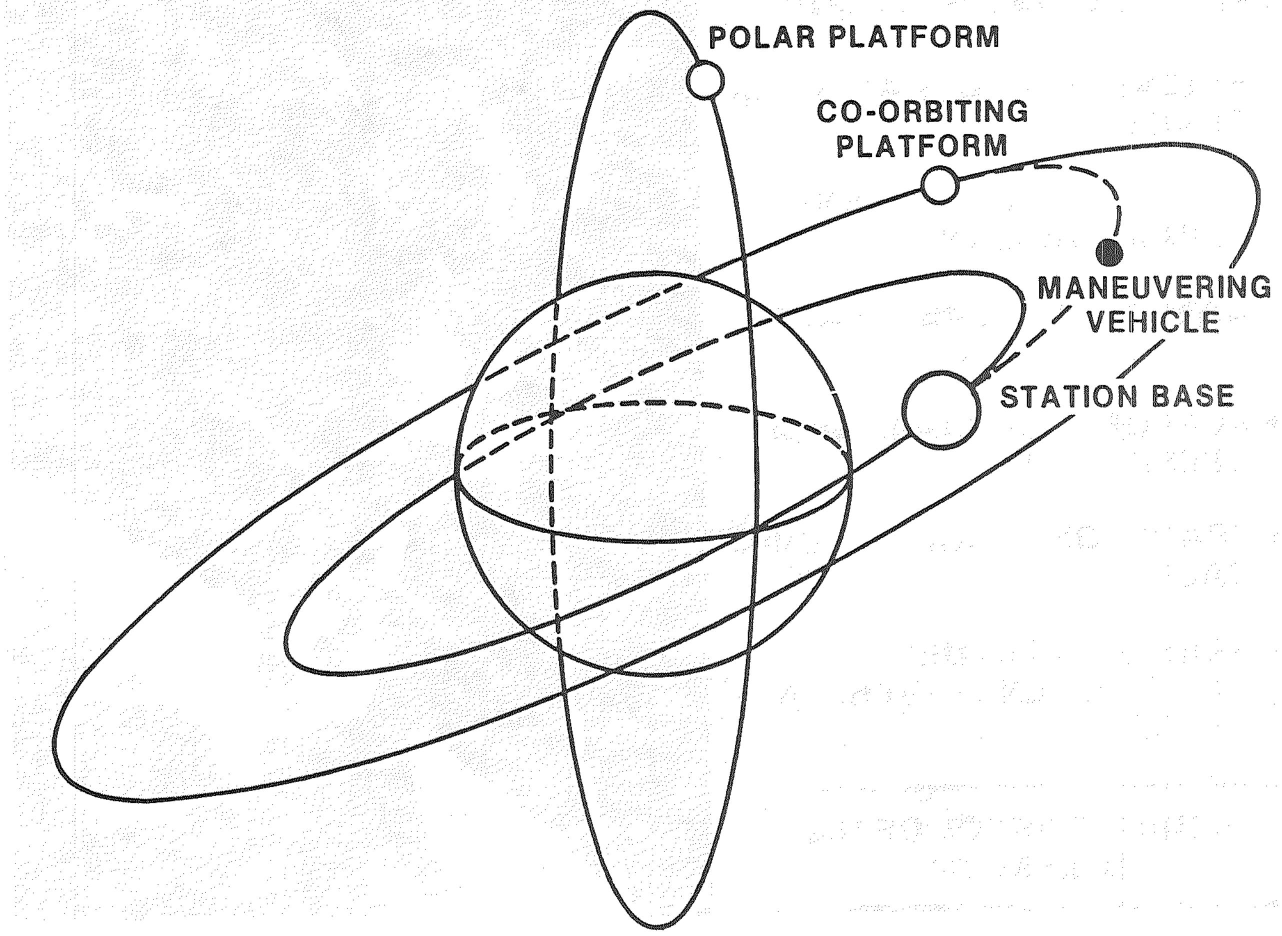
REFERENCE CONFIGURATION

As shown, the reference configuration is an elongated truss-like structure approximately 400 feet long and 200 feet wide. It will be maintained in a 250 n.mi. circular orbit inclined at 28.5 degrees to the equator. The station will be oriented in a gravity gradient attitude with Earth sensing payloads and the various modules located on the end closest to the Earth. The present concept is that the station will be powered by solar arrays. Also shown are two orbital maneuvering vehicles. These OMVs will be unmanned, remotely controlled spacecraft designed to ferry payloads and equipment in nearby ranges. One such destination might be a co-orbiting unmanned platform, as shown on the sketch.


Solar Arrays
Space Station
Unmanned Coorbiting Platform
Scientific/ Commercial Payloads
Servicing Fixtures
Payloads
Storage
Truss and Servicing Structure
Radiators
Experime
Anten
Advanced Remote Manipulator System
Habitat, Logistics and Laboratory Modules

SPACE STATION COMPLEX

The Space Station Complex consists of three major elements. Two of those elements are the Space Station Main Base, discussed in the previous figure, and, in that same 28.5 degree orbit, an unmanned platform. The third major element of the Space Station Complex is an unmanned Polar Platform. The Polar Platform will be the location for most Earth sensing instruments since that platform will survey all of the Earth's surface on a frequent basis. The figure also shows one of the orbital maneuvering vehicles traveling between the Space Station Main Base and the Co-Orbiting Platform.



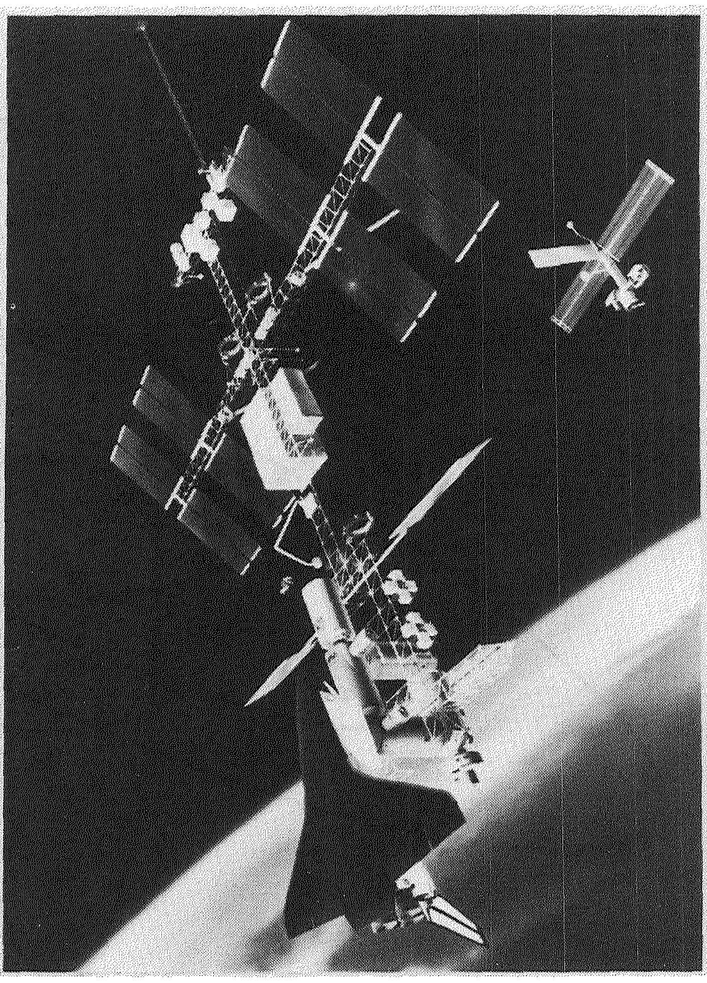
SPACE STATION

The Space Station will serve as a means for furthering our scientific research in space and will have a number of additional important functions. One will be as a satellite or instrument repair facility, a capability that has been demonstrated using the Shuttle Program. Space Station will also serve as a base to assemble large structures. It will be a facility to support the commercialization of space and a transportation staging base for missions to the Moon and beyond. Overall, the Space Station will be a visible symbol of U. S. strength.

SPACE STATION

- SCIENTIFIC LABORATORY IN SPACE
- SATELLITE/INSTRUMENT REPAIR FACILITY
- BASE TO ASSEMBLE LARGE STRUCTURES
- SUPPORT FOR COMMERCIAL OPERATIONS
- TRANSPORTATION STAGING BASE
- BRIDGE TO FUTURE NATIONAL ENDEAVERS IN SPACE

VISIBLE SYMBOL OF U.S. STRENGTH



SPACE STATION PLANNING GUIDELINES

A number of management and engineering guidelines have been established for the Space Station Program. The management guidelines include provisions for an initial operational capability station within a decade. The program has very extensive user involvement both from our traditional communities of the scientific and application areas as well as from technology and from the commercial sector. On the technical side, the station must be evolutionary in nature and technology transparent. We are looking at a Space Station Program with a lifetime of something like 25 to 30 years and thus must be able to change our technology without impacting the users. The station elements will be serviced by the Shuttle. The Space Station Main Base will be continuously habitable.

MANAGEMENT RELATED

- Three year detailed definition (5-10% of program cost)
- NASA-wide participation
- Development funding in FY 1987
- IOC: "within a decade"
- Cost of initial capability: $8.0B
- Extensive user involvement
  - Science and applications
  - Technology
  - Commercial
- International participation

ENGINEERING RELATED

- Continuously habitable
- Shuttle dependent
- Manned and unmanned elements
- Evolutionary
- Maintainable/restorable
- Operationally semi-autonomous
- Customer friendly
- Technology transparent

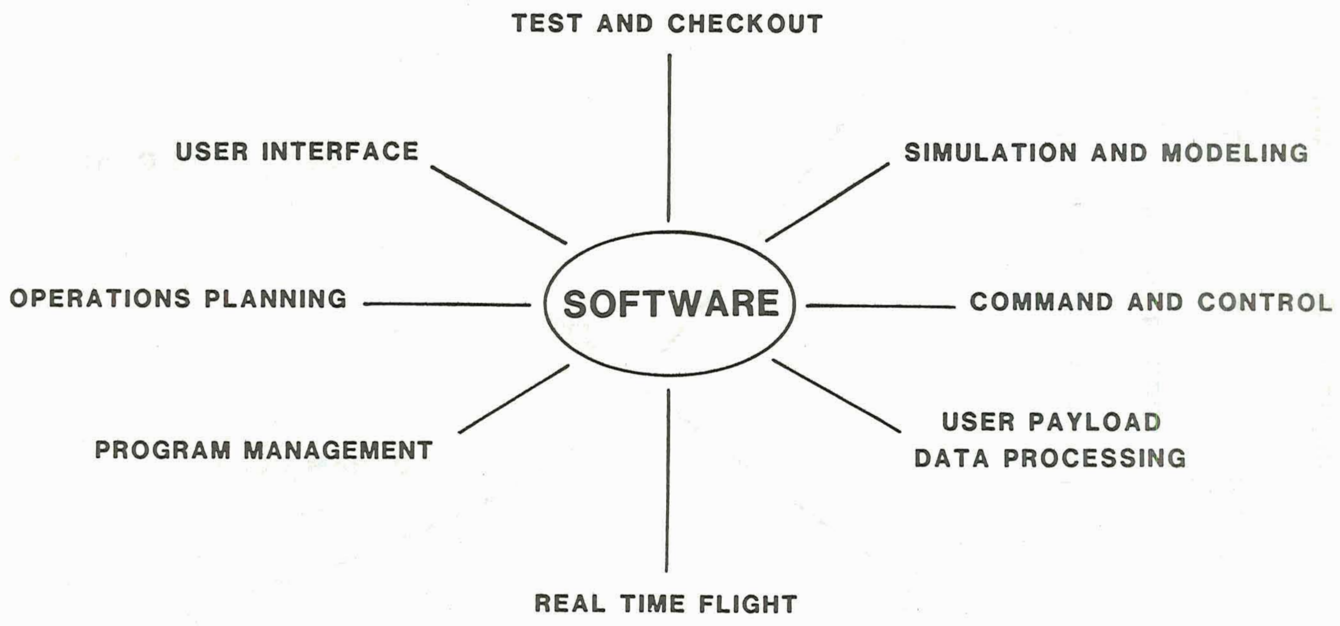
INTERNATIONAL COOPERATION

President Reagan as part of the Space Station initiation invited international participation. We are pleased to welcome the European Space Agency, Canada, and Japan to our team. The Memoranda of Understanding between ourselves and those participants are soon to be signed.

- **PRESIDENT REAGAN INVITED INTERNATIONAL PARTICIPATION**
- **ESA, CANADA AND JAPAN HAVE RESPONDED:**
  - **SOON TO SIGH MOU'S ON PHASE B COOPERATION**
- **SPACE STATION IS TO BE A TRULY COOPERATIVE ENDEAVOR:**
  - **DEVELOPMENT**
  - **UTILIZATION**
  - **OPERATIONS**
- **U.S. AND FOREIGN INDUSTRIES MAY COOPERATE TOO**

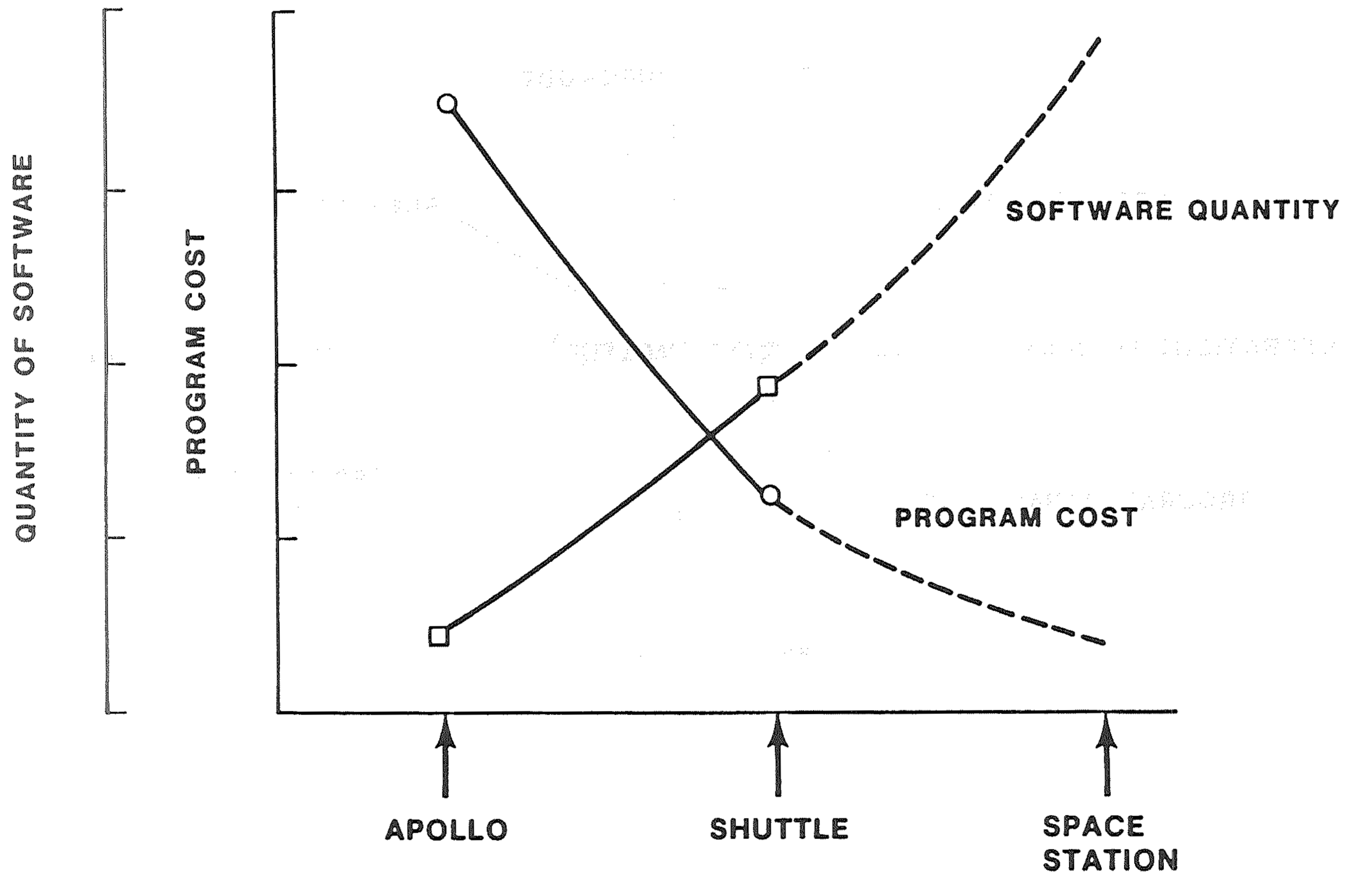
PROGRAM SUCCESS

Software will play a very critical role throughout the Space Station Program. This figure illustrates just a few of those major categories. They range from test and checkout to user interface support, payload processing, command and control, and of course management of the program itself.



NASA SOFTWARE TRENDS

This figure tries to show in an unquantified manner the significant increase in software that we believe we will be working with in the Space Station Program compared to the amount that we developed for Apollo and Shuttle. It also shows that the Space Station effort will be built with substantially less dollars than were available on those past major programs. So the primary messages from this and the previous figure are that NASA must maximize the efficiency with which it uses its software resources. We must learn as much as we can from past lessons, be careful not to repeat mistakes, and use methodologies that worked well before.



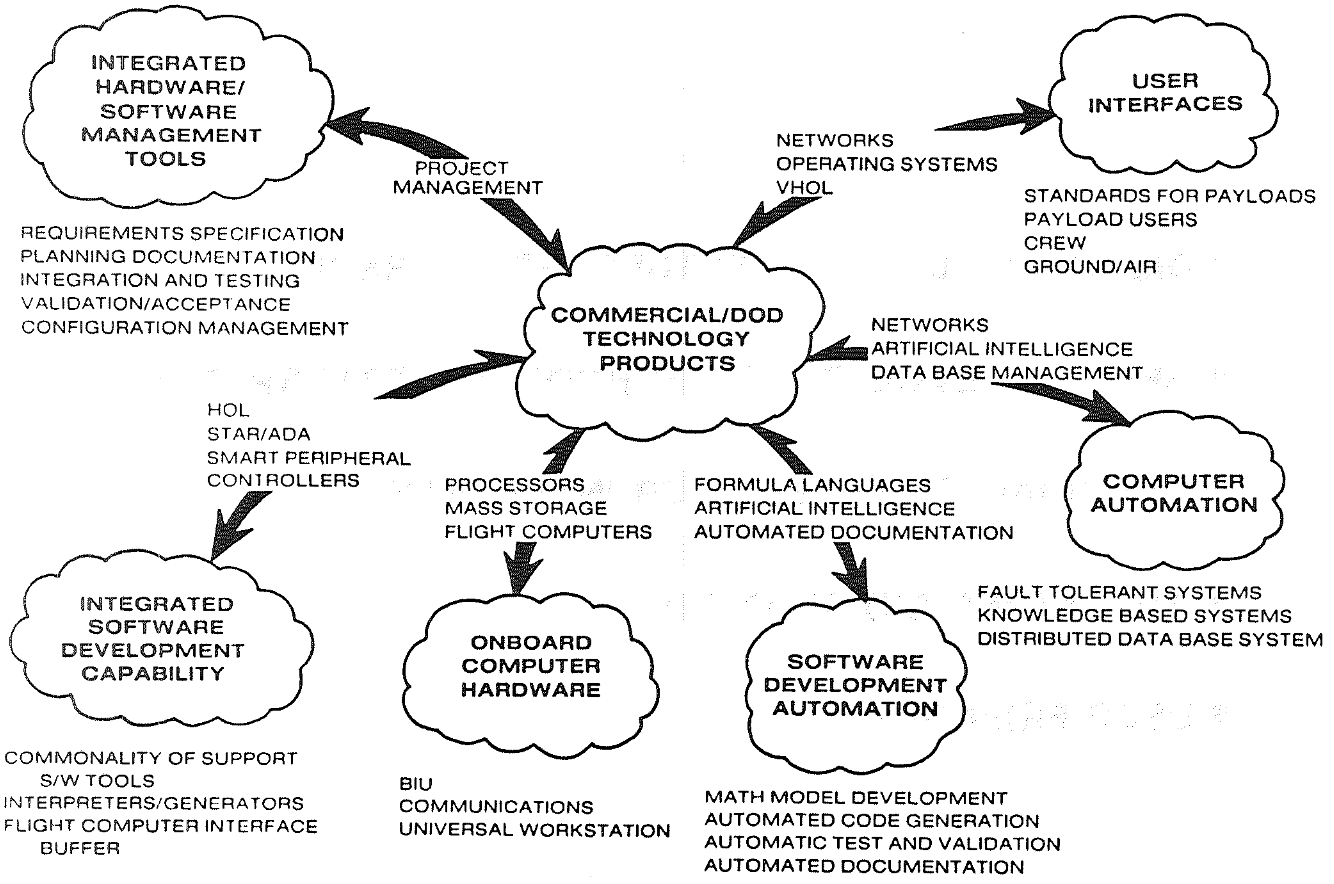
FUNDAMENTAL REQUIREMENTS

This figure lists a few of the major requirements that are software drivers. We recognize that we will be working with a highly distributed architecture and that networking will be prevalent throughout that architecture in the form of local area networks as well as wide area networks. As we said earlier, the station technology on-board and on the ground must be oriented for growth and evolution. Our users will be working from terminals via a space station information system that we plan will enable those users to operate just as if their instruments were in the laboratory next door. The Space Station will at least initially have a crew of somewhere between six to eight and therefore automation will be important. Many of the functions on-board the station must perform in an autonomous manner and since we are looking at a long term program, we must try to automate as much of the ground system as we can to minimize operating costs. Of course, the overall driving requirement is that the entire system be user friendly both for NASA operators running the station and for our customers.

- **DISTRIBUTED ARCHITECTURE, NETWORKING**
- **GROWTH, EVOLUTION, TECHNOLOGY TRANSPARENCY**
- **TERMINAL-ORIENTED USER INTERFACES**
- **AUTONOMY/AUTOMATION**
- **USER FRIENDLY**

TECHNOLOGY

There are a large number of commercial and Department of Defense technology products that can potentially be used to serve all of the areas on this figure. These include integrated hardware and software tools, on-board computer hardware, software development aids, computer automation, and aids for the user interface. Notice that the arrows go two ways. The two-direction arrows show that in some cases some of what NASA does with these products may influence the commercial and DOD sectors. However, that is not the main message. The bottom line is that we plan to maximize the use of commercial and DOD products.



SOFTWARE MANAGEMENT

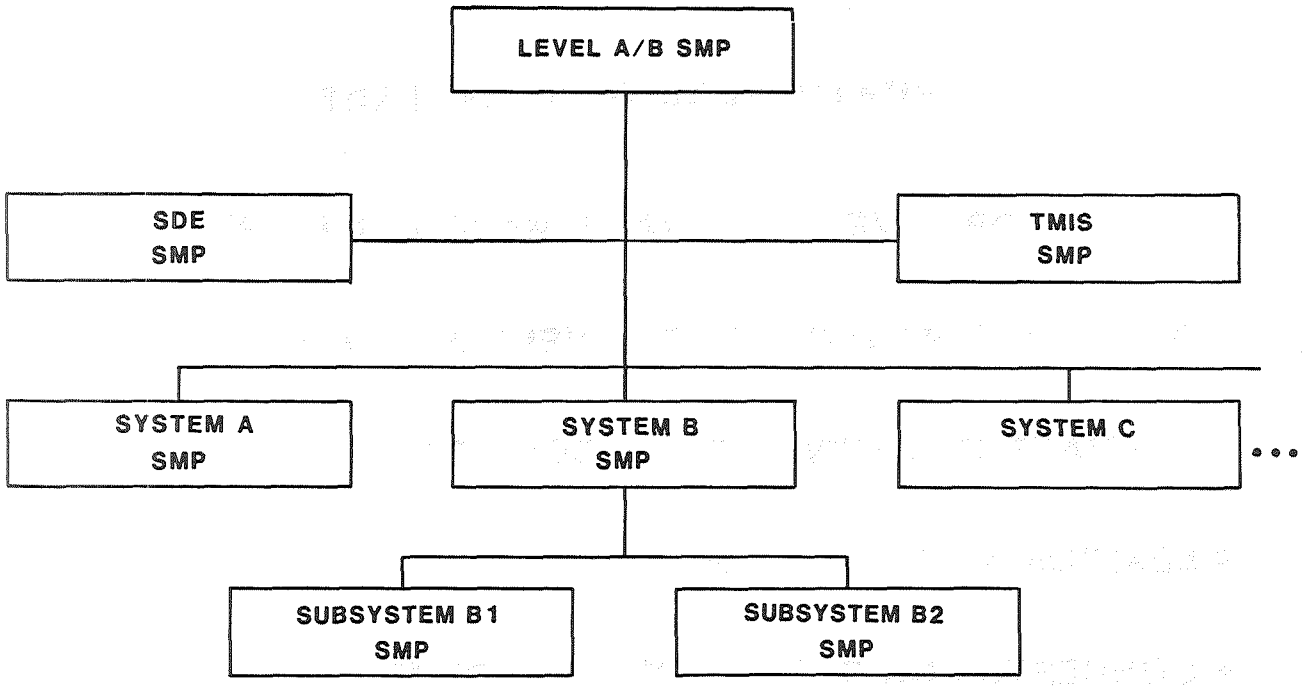
NASA has taken a number of software management steps that we think are positive and in the right direction. First of all, the top level (combined Levels A and B) software management plan has been drafted. That document will continue throughout the program's life to be the repository for the program's policies and procedures. We have also created positions and appointed people as designated software managers at Levels A and B. The Program is in the process of converting what has been an ad hoc software working group into a permanent software advisory panel. We are beginning to assemble software standards, the first of which will be a lexicon so that all participants will be using the same defintion of terms. And finally, we are in the latter stages of conceptualizing a software development environment. Now let me pause for a moment here and clarify that we also refer to the SDE as a software support environment, the idea being that the term "support" conveys a wider process than does development. We presently use both terms synonymously.

WHAT'S BEEN DONE SO FAR?

- DRAFT TOP LEVEL SOFTWARE MANAGEMENT PLAN
- SOFTWARE MANAGERS AT LEVELS A AND B
- PERMANENT SOFTWARE ADVISORY PANEL
- LEXICON AS FIRST STANDARD
- CONCEPTS FOR SOFTWARE DEVELOPMENT ENVIRONMENT

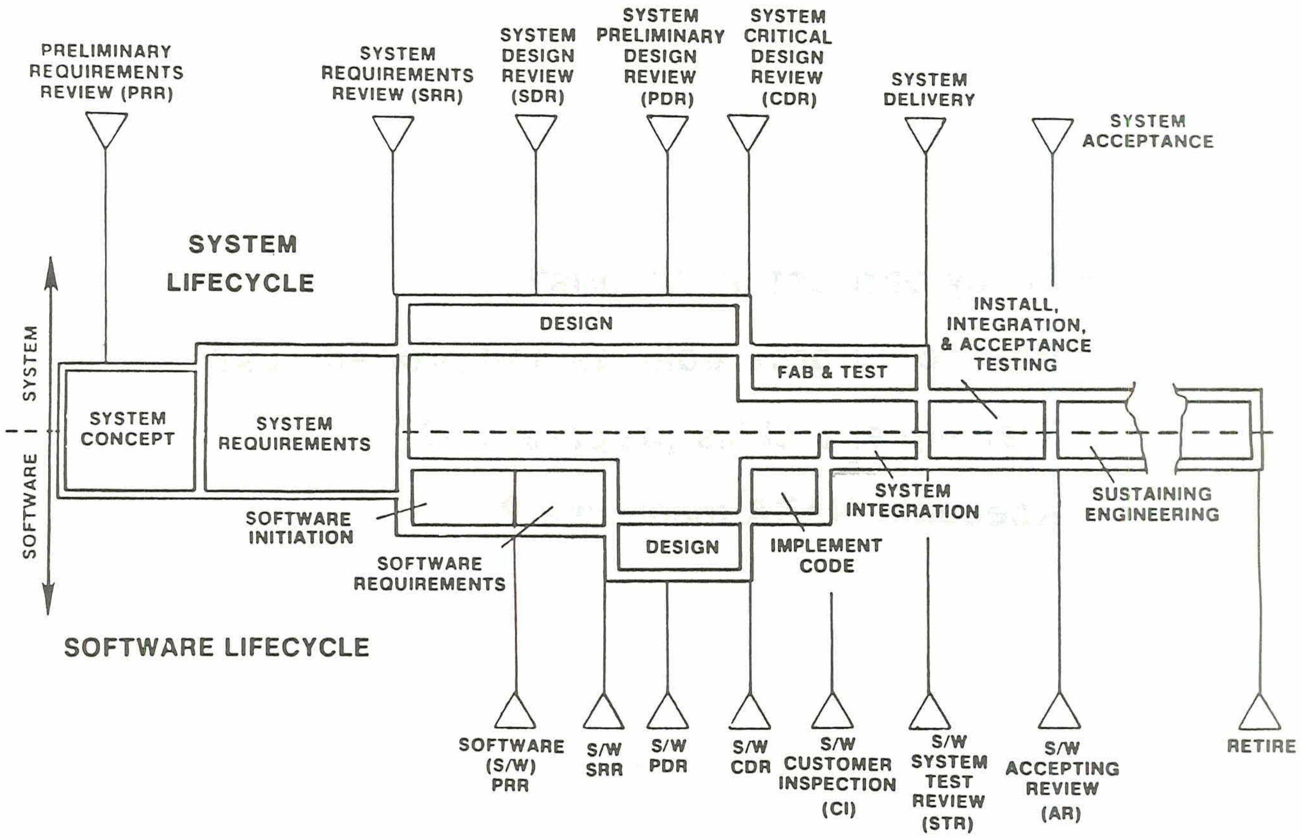
SOFTWARE MANAGEMENT PLANS

The Space Station Program envisions a hierarchy of software management plans, i.e., one plan per major software element. The plan at the top of this figure, the Level A/B software management plan, is the one that is presently in draft form and that will soon be undergoing formal review throughout the Space Station Program. Two other major elements that have been identified so far will also be required to have individual software management plans. One is the software development environment (or the software support environment) and the other is the Technical and Management Information System (TMIS). Since the other elements of the Space Station Program have not yet been identified (we are still in the requirement stage) they are shown on this chart simply as systems A, B, C, and so on.



SPACE STATION LIFE CYCLE

Pictured here is the standard Space Station System and Software Life Cycles that will be used within the Program. The top half of the figure shows what the systems phases are, and the bottom half gives the corresponding software phases. Shown as well are the major reviews and events that will take place across that life cycle. We will require that all space station software efforts utilize the life cycle regardless of whether the software is being developed or acquired. (Ed. note: This life cycle has been slightly modified in the approved Software Management Plan, which will be available from the Space Station Program Office in late 1985.)


PRELIMINARY REQUIREMENTS REVIEW (PRR)
SYSTEM REQUIREMENTS REVIEW (SRR)
SYSTEM DESIGN REVIEW (SDR)
SYSTEM PRELIMINARY DESIGN REVIEW (PDR)
SYSTEM CRITICAL DESIGN REVIEW (CDR)
SYSTEM DELIVERY
SYSTEM ACCEPTANCE
SYSTEM LIFECYCLE
SYSTEM
SOFTWARE
DESIGN
FAB & TEST
INSTALL, INTEGRATION, & ACCEPTANCE TESTING
SYSTEM CONCEPT
SYSTEM REQUIREMENTS
SOFTWARE INITIATION
SOFTWARE REQUIREMENTS
DESIGN
IMPLEMENT CODE
SYSTEM INTEGRATION
SUSTAINING ENGINEERING
SOFTWARE LIFECYCLE
SOFTWARE (S/W) PRR
S/W SRR
S/W PDR
S/W CDR
S/W CUSTOMER INSPECTION (CI)
S/W SYSTEM TEST REVIEW (STR)
S/W ACCEPTING REVIEW (AR)
RETIRE

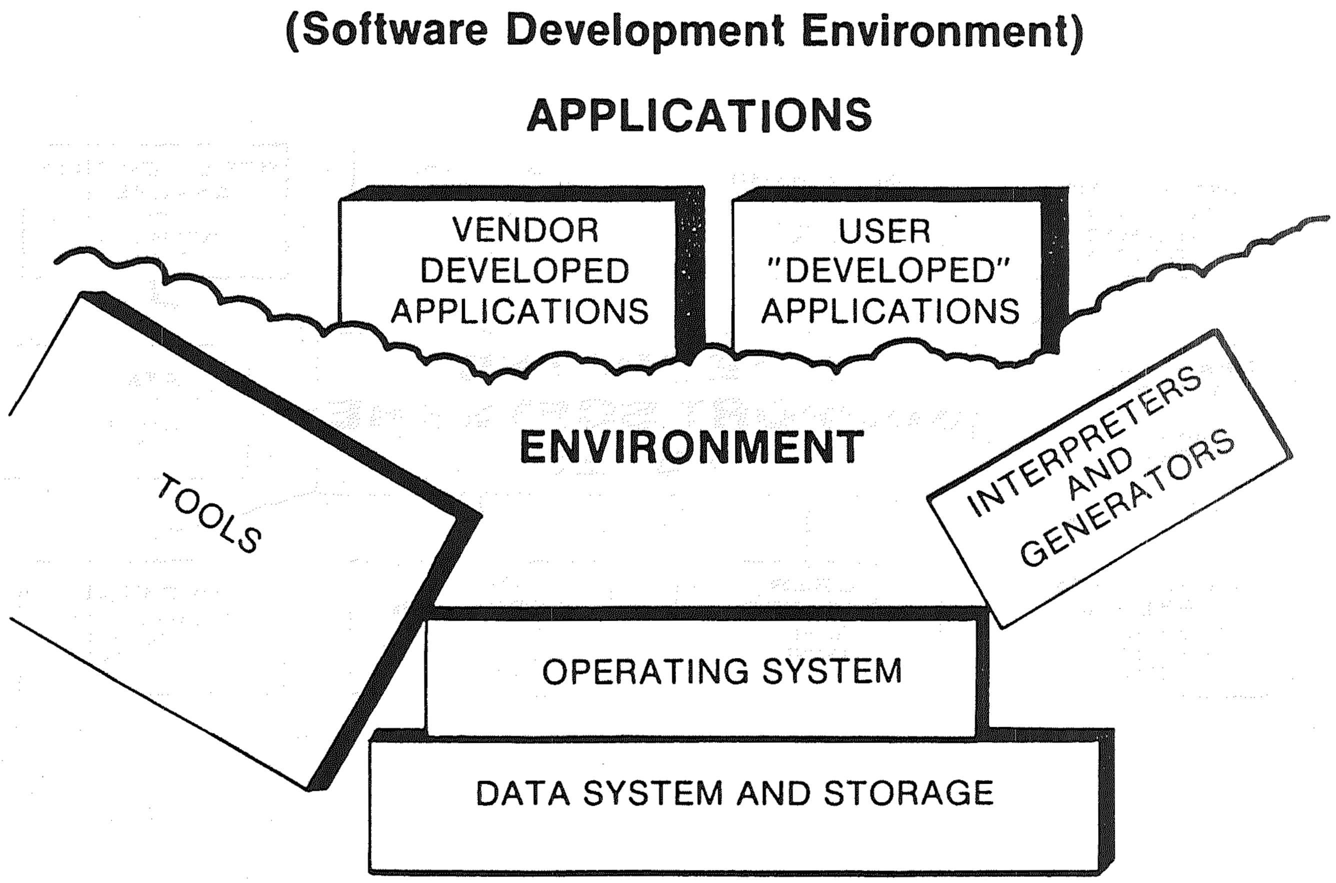
EXAMPLES ISSUES

Although a number of steps have been taken, many additional software management issues remain. This figure lists a few of them. One such issue concerns application criteria, i.e., to what depth and to what extent should our policies and procedures apply? We certainly don't want to impose all these rules on the technical person working in an office with a personal computer. By what criteria do we decide how much of the policies and procedures apply to each element and each situation? Another issue, and a very important one, is how do we enforce these policies and procedures as well as the supporting standards that serve to implement the policies and procedures? What enforcement mechanism should be used? A third issue is training and skills preparation. Is it adequate simply to send our people to management courses, or is additional preparation needed? Should we consider staff rotation into different jobs? A fourth issue is the question of whether NASA has adequate manpower to do the system and software engineering and integration job, and if the answer is no, then what should NASA do?

- **Policy application criteria?**
- **How to enforce policies and procedures?**
- **Training and skills preparation?**
- **Adequate NASA manpower?**

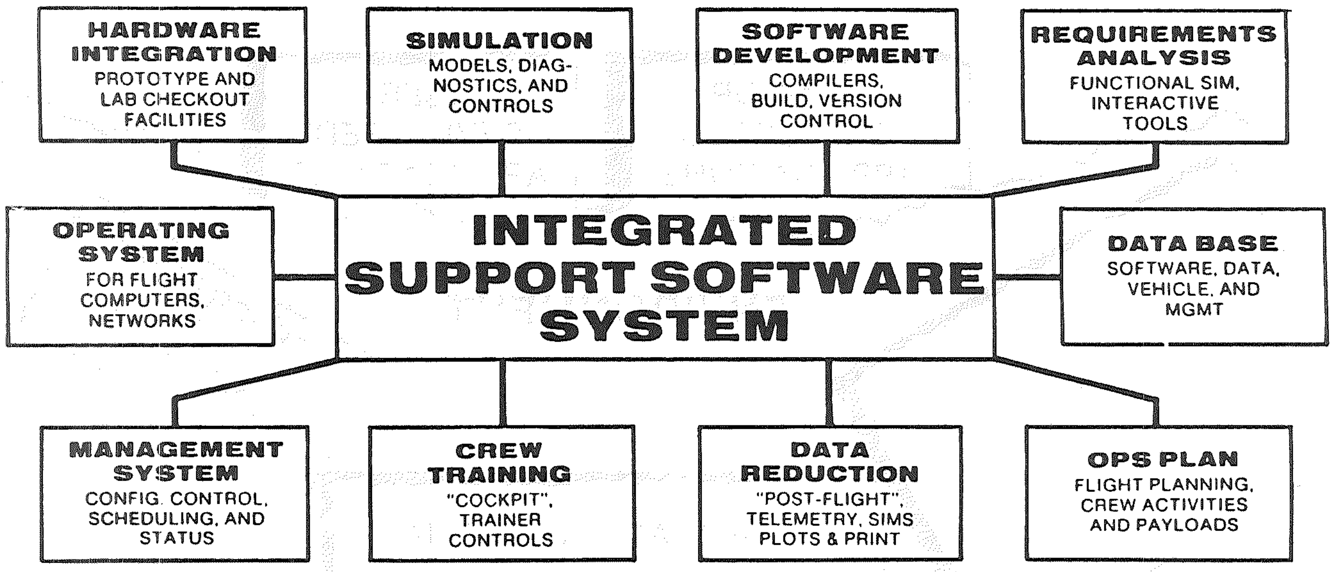
APPLICATIONS ENVIRONMENT

This figure illustrates a couple of important features about the software development environment. One is that the software development environment will consist of a standard set of tools, software packages, policies, and procedures which from one perspective will free the user and the application software from the operating system and data storage. Another important message from this figure is that the user and the application software will be provided a number of services by the software development environment via the tools, interpreters, code generators, operating system, etc., that comprise that software development environment.


(Software Development Environment)
APPLICATIONS
VENDOR DEVELOPED APPLICATIONS
USER "DEVELOPED" APPLICATIONS
ENVIRONMENT
TOOLS
INTERPRETERS AND GENERATORS
OPERATING SYSTEM
DATA SYSTEM AND STORAGE

COMMONALITY

This figure illustrates that an integrated software support system will consist of many different elements: aids for hardware integration, simulation models, diagnostics, control tools, software development aids, compilers, version control tools, packages to analyze requirements, operating systems, management systems, and so on. It is also important to realize that the integrated software support system will enable the user to select the particular support elements required and thus form a specific subset software support environment.


HARDWARE INTEGRATION
PROTOTYPE AND LAB CHECKOUT FACILITIES
SIMULATION
MODELS, DIAG-NOSTICS, AND CONTROLS
SOFTWARE DEVELOPMENT
COMPILERS, BUILD, VERSION CONTROL
REQUIREMENTS ANALYSIS
FUNCTIONAL SIM, INTERACTIVE TOOLS
OPERATING SYSTEM
FOR FLIGHT COMPUTERS, NETWORKS
INTEGRATED SUPPORT SOFTWARE SYSTEM
DATA BASE
SOFTWARE, DATA, VEHICLE, AND MGMT
MANAGEMENT SYSTEM
CONFIG. CONTROL, SCHEDULING, AND STATUS
CREW TRAINING
"COCKPIT", TRAINER CONTROLS
DATA REDUCTION
"POST-FLIGHT", TELEMETRY, SIMS PLOTS & PRINT
OPS PLAN
FLIGHT PLANNING, CREW ACTIVITIES AND PAYLOADS

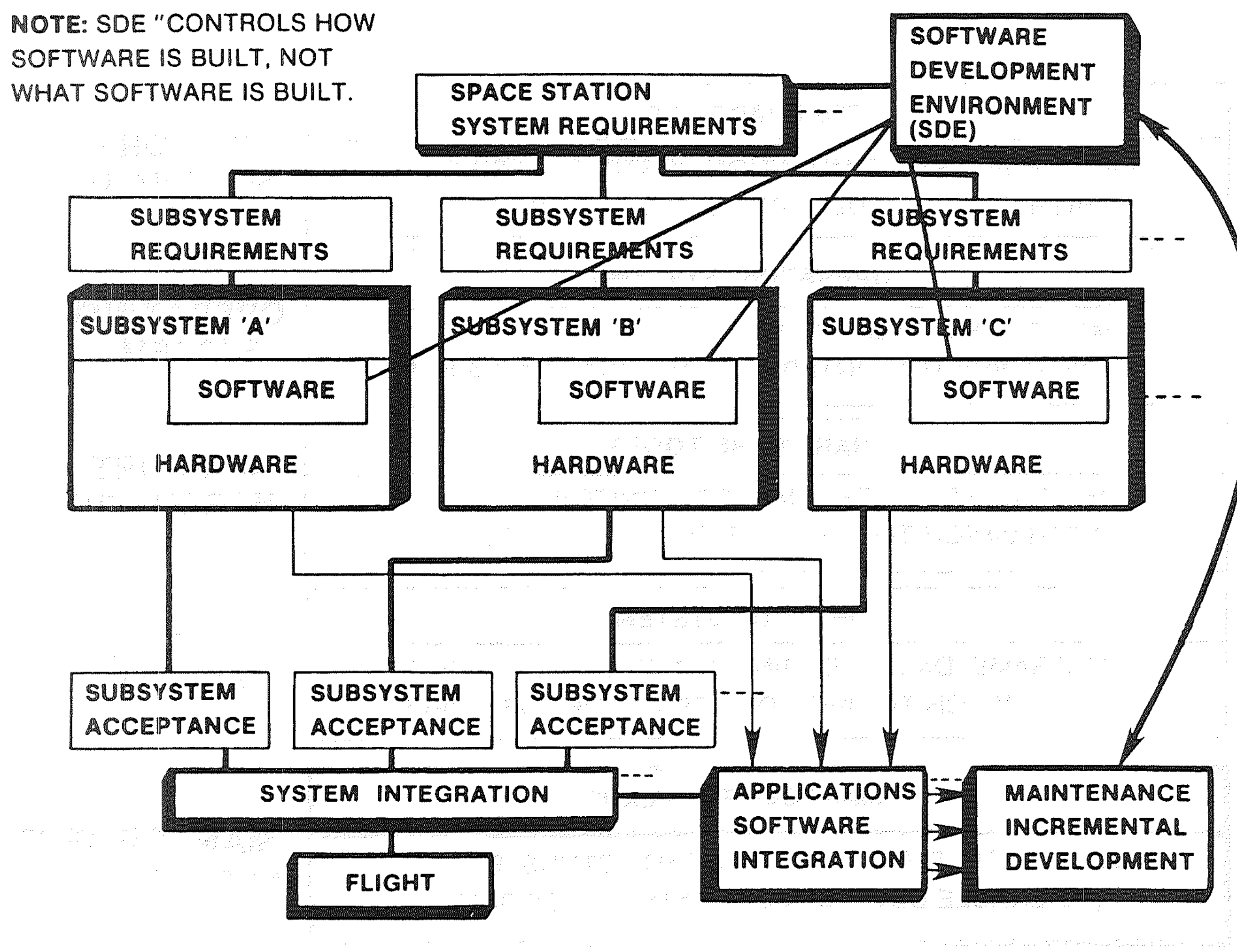
ENVIRONMENT COMPONENTS

The software support environment will consist of five major constituents. They are software tools, operating systems, various hardware tools (such as simulation interface buffers and performance monitors), a host data processing system, and then last, and certainly not least, overall management policies, procedures, and standards. The management of the software development and acquisition process is critically important. Thus this last category, the "management plan" box, is highlighted.

SOFTWARE TOOLS LANG, BUILD SYS, SIMULATION POSTPROCESSING. . . (WITH WHICH TO DESIGN & BUILD SOFTWARE)	**SUPPORT SOFTWARE**
OPERATING SYSTEMS DMS I/O, TIME/TASK MGMT., USER I/F, ETC. . . (WITH WHICH TO INTERFACE APPLICATIONS & H/W)	**OPERATING SYSTEMS**
HARDWARE TOOLS SIM I/F BUFFER, PERFORMANCE MONITOR. . . (WITH WHICH TO SIMULATE/TEST SOFTWARE)	**SUPPORT HARDWARE**
HOST DP SYSTEM MAINFRAME, DASD, TERMINALS, NETWORK, OUTPUT. . . (WITH WHICH TO DEVELOP/VERIFY THE SDE ITSELF)	**SDE LEVEL FACILITY**
"MANAGEMENT PLAN" DEVEL/INTG. FLOW, CONFIG. MGMT., STNDS, ETC. . . (TO ENABLE DEVELOPMENT AND INTG OF S/W)	**MANAGEMENT AND CONTROL**

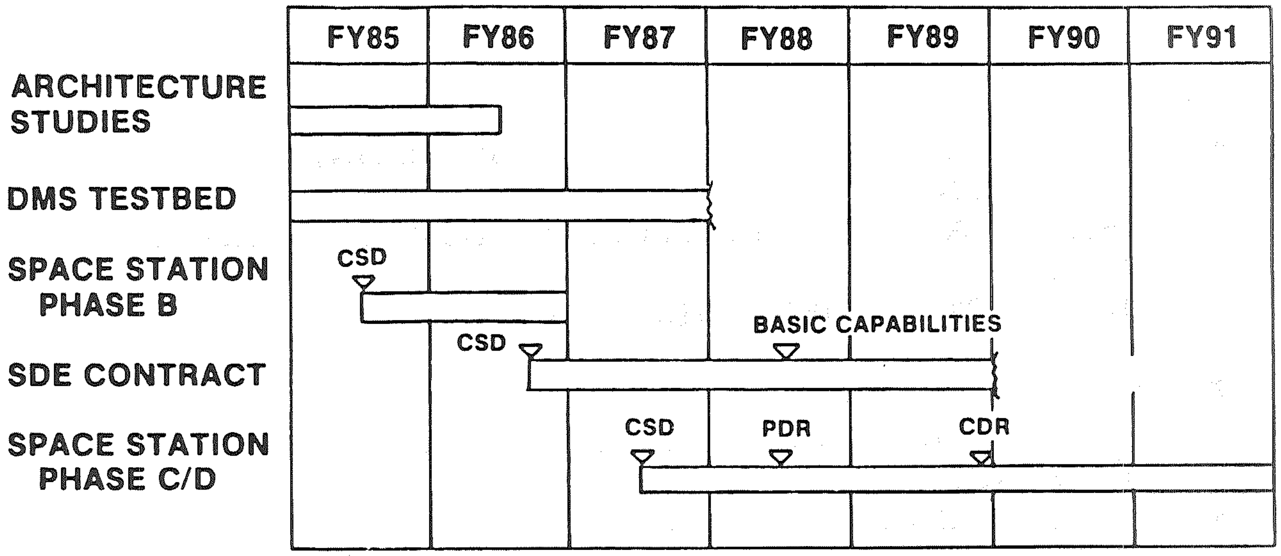
RELATIONSHIP

This figure tries to conceptualize how the software development environment will provide support throughout the system life cycle. First, the SDE will support each subsystem as it is being developed. That same software development environment will then provide support as those subsystems are integrated to form the system and then later on as that system moves into the long term maintenance and enhancement phase. A key driver behind the SDE concept is to minimize maintenance costs.



SCHEDULE

This figure illustrates the planned schedule for the software development environment. A separate contract for the software development environment will be issued in the latter part of FY 1986 so that we have a basic capability SDE in place by mid-year FY 1988. Note how that correlates with the Space Station Phase C/D mainstream development. Phase C/D is scheduled to start at mid-year FY 1987 so it is important that the SDE be in place, tested and checked out shortly thereafter, i.e., prior to the critical design review for the Space Station. We are on a very tight schedule.



PROS AND CONS

The software development environment has a number of advantages as well as a few disadvantages. Some of the disadvantages are that it will require a large investment up front. Certainly the establishment of a standard set of tools, practices, policies, and techniques will affect a number of previously established "sand boxes," by which I mean the ways people have traditionally been doing business both within NASA as well as within industry. In addition, the SDE must be designed for changes; it won't be a fixed set of tools. On the advantages side, however, we are firmly convinced that the SDE will substantially reduce the cost of ownership for our software, the ownership (maintenance) cost that we're worrying about being something like 70 to 80 percent of the total life cycle outlay. The SDE will also lend stability to our software process by helping to assure that all participants are using the same set of tools, standards and techniques and it will thus improve the integration and checkout process. We believe the advantages, particularly in the long run, far outweigh the disadvantages.

Cons

- MAY REQUIRE SUBSTANTIAL FRONT-MONEY INVESTMENT
- AFFECTS A LOT OF PREVIOUSLY ESTABLISHED "SAND BOXES"
- SDE HAS TO BE DESIGNED FOR CHANGE

Pros

- PROVIDES FOR REDUCED COST OF OWNERSHIP FOR SOFTWARE
- LENDS STABILITY TO THE SOFTWARE PROCESS
- IMPROVES THE INTEGRATION & CHECKOUT PROCESSES

Fact of Life:

THE CONTINUING RAPID EVOLUTION OF THE COMPUTING INDUSTRY

ISSUES

There are a number of issues associated with the software development environment. The first concerns the practicality of an SDE and whether NASA really should try to define and develop such a software support capability. Secondly, should we try to apply that software development environment to all software, both in-house and that which we contract for? What will be the impact of a NASA defined SDE on our contractor colleagues? Should NASA furnish the SDE, lock, stock and barrel, or only specify what it should be and allow each organization that wants a copy to procure their own software/hardware? Should the SDE be a single centralized facility or should we allow multiple copies of the SDE? Another very important question is how do we maintain configuration control? The SDE certainly won't be a static capability. What will be the government's liability? When software is late or has problems, will the developer be inclined to point to the SDE as a source of the problem? And then finally, a remaining issue is whether we really should be talking about two different kinds of SDE's, one that would support software development and the other that will support software acquisition and thus be largely a management SDE.

- **Should a uniform NASA SDE be defined and developed?**
- **Apply to all software development (in-house and contractors)?**
- **Relationship to established industry SDEs?**
- **NASA GFE or only specs?**
- **One central facility or multiple copies?**
- **How to configuration control?**
- **Government liability?**
- **Two SDEs: development and management?**

STANDARDS

The basic question concerning standards is what standards are needed. I have listed on this figure a few of the types of standards that we think we should have. This list ranges from types of documentation and formats for those documents down to terminology instruction, set architectures, standardized languages, standards for quality assurance, testing procedures, and a standardized life cycle. Now, in a couple of cases, we have already moved forward to begin the standardization process. We have established a standard life cycle, as shown on a previous figure. We are specifying a critical set of documents that should be required of most software projects. (It will always be possible to apply for a waiver, but we do have a standard set of documents that will normally be required.) We are also in the process of finalizing a software terminology standard. But what other categories should we be worrying about and what candidates exist to fill those needs?

- **WHAT STANDARDS ARE NEEDED?**
  - **Documentation types and formats?**
  - **Terminology?**
  - **16 bit and 32 bit instruction set architectures?**
  - **Languages?**
  - **Operating systems? Tools? DBMS?**
  - **Quality assurance?**
  - **Configuration management?**
  - **Testing procedures?**
  - **Life cycle (phases, events, products)?**
- **WHAT OTHERS?**
- **WHAT CANDIDATES FILL THE NEED?**

ARGUMENTS FOR AND AGAINST

There are a number of arguments leaning in favor of standards and of course some arguments against standards. Arguments that indicate that we should have standards point out that we will have greater compatibility in our equipment and data. It will be less costly to transfer information if we have standardized software/hardware and standardized documents. Systems and subsystems should be implemented more quickly. Standards should facilitate wider use of information, particularly across the large number of organizations that will comprise the Space Station Program. Standards in some areas at least will mean that we will need fewer skilled personnel. In other words, we won't have to train and maintain so many specialists in so many different areas. Arguments against standardization include the possibility of discouraging individual preference, moving us away from the leading edge of technology, and lowering the competitiveness of hardware and software.

The Argument for:

- **COMPATIBILITY FOR EQUIPMENT AND DATA**
- **LESS COSTLY TO TRANSFER INFORMATION**
    - NO NEED TO PURCHASE S/W, H/W BRIDGES
    - LESS PROGRAMMER TIME REQUIRED
- **FASTER IMPLEMENTATION**
- **WIDER USAGE OF INFORMATION**
- **LESS SKILLED PERSONNEL REQUIRED**

The Argument Against:

- **DISCOURAGES INDIVIDUAL PREFERENCE**
- **MOVES AWAY FROM "LEADING EDGE" OF TECHNOLOGY**
- **LOWERS COMPETITIVENESS OF HARDWARE AND SOFTWARE**

LANGUAGES

It has been the intent of the Space Station Program for some time now to standardize on a very few computer languages: one or two languages in the implementation category and a similar small set of languages in each of the other categories. But there are some basic questions that we must ask ourselves. One is should the Space Station Program try to standardize on languages at all? And if you agree that we should, then by what criteria? How is it that we should select one language versus another? And in each category of application, should we focus on one single language or a small set? Some considerations to fold in to our thinking about those questions include the fact that we want to minimize life cycle cost. This is a program that will stretch out over 25 or 30 years. The languages that we pick must be easy to use, and must be robust and have a wide range of functional capabilities. Of course, we would like a language that's reasonably mature and therefore has a good tool support and experience base. The languages must be compatible with the types of computers that we will use, the environments within which that hardware will be exercised, and the existing software. The latter is a very important point for Space Station because we must interface with a number of software applications that are already existing and are written in a number of different languages. Another consideration is programmer availability.

- **QUESTIONS**
  - **Should Space Station Program standardize languages at all?**
  - **If so, by what criteria?**
  - **One language or several?**

- **CONSIDERATIONS**
  - **Minimize life cycle costs**
  - **Ease of use**
  - **Richness and functional capabilities**
  - **Maturity and support base**
  - **Compatibility to machines, environments, other languages**
  - **Programmer availability**

CATEGORIES

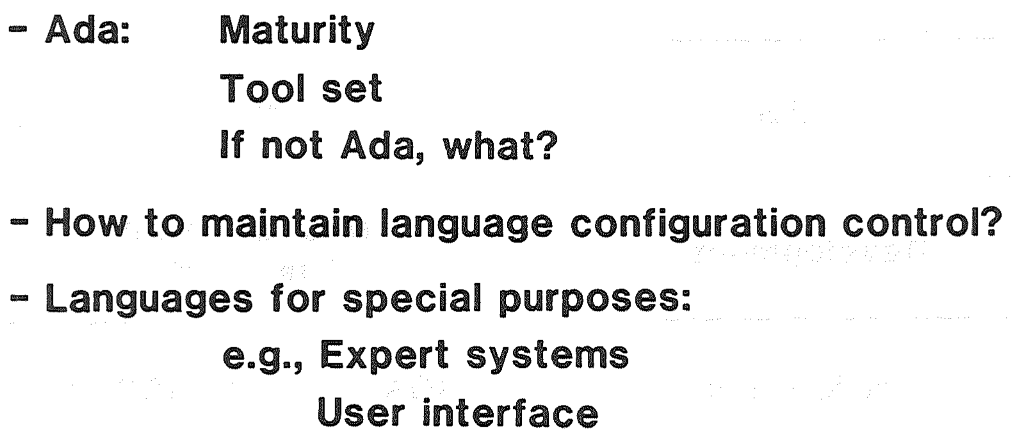
Listed on this figure are the probable major categories for language standardization and a few of the possible candidates that might be suitable for each category. Now, that list of candiates is by no means complete but at least some of the major ones are listed. The categories are requirements and specifications, design, development (which is of course the language standardization area that people most often think of), the user interface, and artificial intelligence and expert systems.

LANGUAGES

CATEGORIES	CANDIDATES?
Requirements and specification	PSL/PSA, SREM, SADT, CADSAT
Design	PDL, SDDL
Development	HAL/S, Fortran, PL/I, Jovial, Ada, C, Modula-2, Pascal
User interface	GOAL, ATLAS, SCOL,STOL, Ada
AI/expert systems	LISP, PROLOG

ISSUES

There are a number of issues associated with selecting computer languages. The first one that comes to everyone's mind is Ada. Is Ada sufficiently mature? Does it have the proper set of tools available? If we decide not to follow the Ada route, at least for a period of time, then what languages or language should we be choosing temporarily? Another issue is how do we maintain language configuration control? How do we prevent or should we even try to prevent people from creating special versions of the selected standard language or languages? Other important issues revolve around the special application areas of expert systems, artificial intelligence, and the user interface. Do we need to select special languages for those categories or can the same standard language that we choose for implementation also suffice?



- **Ada:** **Maturity**
  Tool set
  If not Ada, what?
- **How to maintain language configuration control?**
- **Languages for special purposes:**
  e.g., Expert systems
  User interface

CONCLUSIONS

I have tried in this briefing to prompt your thinking. I have pointed out that software will be a very critical element of Space Station, prevalent throughout all aspects in space as well as on the ground. There are many open issues that the Program is now identifying and attempting to resolve. They range across the four major categories that will be the focus for this forum: software management planning, the software development environment, standards, and languages. We are requesting industry and university assistance and welcome your contributions.

- **SOFTWARE CRITICAL ELEMENT OF SPACE STATION**
- **MANY OPEN ISSUES**
  - **Management planning**
  - **Software development environment**
  - **Standards**
  - **Languages**
- **NASA REQUESTING INDUSTRY AND UNIVERSITY OPINION AND IDEAS**

SOFTWARE MANAGEMENT PANEL SUMMARY

Prior to the forum, the Software Management Panel reviewed the Space Station Software Issues report (ref. 1) and the draft Level A/B Software Management Plan (Table 1). During the forum, the panel experts and the audience made 30 specific recommendations for assuring the successful management of Space Station software. The following six recommendations are essential to the Program's success and are the basis for accomplishing the 30 specific recommendations.

1. The charters of the Level A and B Software Managers must be strengthened to assure that those positions have the decision and control authority to properly conduct their jobs. Specific actions are:

   a. Support the Level A and B Software Managers with increased software-experienced staff. (The panel notes with alarm the lack of any support staff at the present time for the Level A position.)

   b. The Level A and B Software Managers must each have significant discretionary budget to provide the appropriate guidance and support of the software management and acquisition functions below them.

   c. The hierarchy of software decision making and approval authority must be clearly established. The panel recommends that technical decisions with system engineering and integration impact be the responsibility of the Level B Software Manager with the concurrent involvement of the Level A Software Manager. However, the panel recognizes that there will be certain major decisions (such as the choice of a standard language and the overall concept for the software support environment) that will have major, long reaching impact, both within the Program and to organizations that interface to the Program. The panel recommends that such decisions be the responsibility of the Level A Software Manager with the concurrent involvement of the Level B Software Manager.

   d. The Software Management Plan needs to be modified as follows:

      - Develop and adopt charter statements for both the Level A and B Software Managers.
      - Specify items a, b, and c above in the charter statements.
      - Identify and provide a schedule for important decisions that need to be made.
      - Specify how the decisions will be made and by whom.
      - Specify who has control of the management functions, e.g., budget approval and product approvals.

2. The Software Management Plan policies and procedures are in-house development oriented, whereas in fact the task is the management of the acquisition of software (including in-house development). Large-scale software acquisition is new to some parts of NASA and is different from, and more difficult than, hardware acquisition. The plan must be reformulated to reflect this acquisition orientation. Various sections in the plan need to be revised to strengthen the policies and the ability of the Level A and B Software Managers to be effective in playing a role in software acquisition. The plan should call for in-house (NASA) software development to be managed in the same way as non-NASA

(contractor) acquisition/development, with appropriate tailoring to accommodate differences such as legal contracting procedures for external acquisitions.

3. The focus of the Software Management Plan needs to be revised to emphasize the maintenance/sustaining engineering considerations in more detail and earlier in the system life cycle process. The major cost of most long-life-cycle computer-based systems is in the post-delivery-to-operations phase (60-80% of total software life cycle costs). The role of the software managers in the early system definition and design phase should be expanded to provide for software allocation and software trade-offs. If the wrong decisions are made in this phase, it will be nearly impossible to reduce the maintenance/sustaining engineering costs later.

4. It is not clear what the boundaries of Space Station are. The specific management spheres of control are unclear and the procedures for accomplishing management interaction with non-Space Station services are not defined. Additionally, much of the inherited software appears to be outside the control of Space Station policies and standards. For example, interoperability design, interface design, and integrated schedule coordination need to be more clearly delineated. Policies and procedures for managing these issues must be specified as they impact Space Station software.

5. The Software Management Plan and stated NASA approach call for NASA to perform the top level software engineering and integration (SE&I) function. The panel observes that the scope of that task (multicenter, multicontractor and multi-subcontrator) is far beyond NASA's past experience. The panel suggests that the full scope of the SE&I job be re-assessed with special attention to integration. The plan must address more specifically the management of the many geographically dispersed organizations involved in the integration task. More detail is needed on policies (who, how, when) and on the specific responsibilities of developers and integration organizations.

6. It is the consensus of the panel that the Software Management Plan should be restructured. A new table of contents is recommended that provides for:

   - A more complete list of policies.
   - Charters for the Level A and B Software Managers that are sufficient and delimiting with respect to control and authority of the management process.
   - Special attention and focus on several significant procedures.

RECOMMENDATIONS

1. The Level A/B Software Management Plan structure does not focus sufficient emphasis on several areas and needs revision. (See Table 1, recommended Software Management Plan Table of Contents, p. 76.)

2. The interdisciplinary activities and interactions are not well defined. Their definition and control mechanisms should be specifically detailed in the Level A/B Software Management Plan.

3. The Level A/B Software Management Plan should emphasize the considerations of using existing (inherited) software as an alternative to totally new development.

4. The Level A/B Software Management Plan should specify the policies and procedures for control and feedback between the level A/B/C management functions for cost, schedule and technical content.

5. The Level A/B Software Management Plan should specify the policies and procedures for managing the risk issues.

6. The Level A/B Software Management Plan should specify the policies and procedures for managing the various technical performance items.

7. The Level A/B Software Management Plan should address the policies and procedures to accommodate modern, appropriate software development methodologies.

8. The Level A/B Software Management Plan should focus more emphasis on the early planning for the maintainability/sustainability aspects of acquired software.

9. The policies on independent verification and validation (IV & V) in the Level A/B Software Management do not put enough emphasis on its SELECTIVE use. The criteria for utilization of IV&V should be defined.

10. The policies and procedures for managing the acquisition and configuration management of FIRMWARE should be specified.

11. The Level A/B Software Management Plan policies and procedures for acquisition of software should emphasize QUALITY and should be formulated and reviewed to accommodate new paradigms as they may be accepted industry practice over the life of the project (30+ years).

12. The policies and procedures in the Level A/B Software Management Plan should specify how and when software and hardware trade-offs are made in the system life cycle, as well as how and when hardware/software interfaces are defined.

13. The Level A/B Software Management Plan policies and procedures for tailoring should set tailoring guidance based upon different identified categories of software and should provide different life cycles if appropriate.

14. The Level A/B Software Management Plan should define the policies and procedures for the various reviews addressing the who, why, what, and when. They should also provide for an evaluation of the review process and a mechanism for improving the review process.

15. The Level A/B Software Management Plan should specify the policies and procedures for contract incentives that are easily understood and administered and are directly tied to the cost, schedule and technical content, and quality of the product.

16. The Level A/B Software Management Plan needs to stress the policies and procedures for ACQUISITION of software rather than DEVELOPMENT of software.

17. The Level A/B Software Management Plan should rely heavily on existing government and industry standards such as the new DOD-STD 2167 (ref. 2) and IEEE standards.

18. The life cycle definition should expand its scope to include the system front-end definition and design, operations, and sustaining engineering, and to specify the products and reviews relevant to each phase.

19. The Level A/B Software Management Plan should specify the policies and procedures for defining and managing the support system interfaces and interoperability, such as TDRSS, Shuttle, Mission Control, etc.

20. The Level A/B Software Management Plan should first focus on the acquisition/development methods and languages and then choose the tools to support the methods for the Software Management Environment.

21. The Level A/B Software Management Plan should specify the procedures for its timely review, approval, and maintenance.

22. The Level A/B Software Management Plan should specify policies and procedures based on legal and government policies for managing the software on an international basis with respect to proprietary information and software and the export of key US technology.

23. The Level A/B Software Management Plan should address the policies and procedures for managing the security, sensitivity, privacy, and contamination/destruction issues of software acquisition and ownership.

24. The Level A/B Software Management Plan should specifiy the policies and procedures for the decision process and authority for decision making.

25. The Level A/B Software Management Plan should specify the policies and procedures for insuring non-loss of software and continuous operations due to inadvertent and/or catastropic loss of operational or support software.

26. The Level A/B Software Management Plan policies and procedures should focus on the management, control, quality, etc. of the PRODUCTS as opposed to the development process; i.e., acquisition management as opposed to development management.

27. The Level A/B Software Management Plan should specify the policies and procedures for "designing-to-cost" as a potential acquisition strategy.

28. The Level A/B Software Management Plan should specify the primary goals and objectives of the plans, policies, and procedures.

29. The Level A/B Software Management Plan should specify the policies and procedures for obtaining and utilizing software acquisition experience from past and future projects.

30. The Level A/B Software Management Plan should specify the policies and procedures for establishing standardization.