LESSONS LEARNED IN DEVELOPING MULTIPLE DISTRIBUTED PLANNING SYSTEMS FOR THE INTERNATIONAL SPACE STATION

Theresa G. Maxwell

National Aeronautics and Space Administration (NASA)
Mission Support Systems Group (Mail Code: FD42)
Ground Systems Department
Flight Projects Directorate
Marshall Space Flight Center, Alabama 35812
E-mail: theresa.maxwell@msfc.nasa.gov

ABSTRACT

The planning processes for the International Space Station (ISS) Program are quite complex. Detailed mission planning for ISS on-orbit operations is a distributed function. Pieces of the on-orbit plan are developed by multiple planning organizations, located around the world, based on their respective expertise and responsibilities. The "pieces" are then integrated to yield the final detailed plan that will be executed onboard the ISS. Previous space programs have not distributed the planning and scheduling functions to this extent. Major ISS planning organizations are currently located in the United States (at both the NASA Johnson Space Center (JSC) and NASA Marshall Space Flight Center (MSFC)), in Russia, in Europe, and in Japan.

Software systems have been developed by each of these planning organizations to support their assigned planning and scheduling functions. Although there is some cooperative development and sharing of key software components, each planning system has been tailored to meet the unique requirements and operational environment of the facility in which it operates. However, all the systems must operate in a coordinated fashion in order to effectively and efficiently produce a single integrated plan of ISS operations, in accordance with the established planning processes.

This paper addresses lessons learned during the development of these multiple distributed planning systems, from the perspective of the developer of one of the software systems. The lessons focus on the coordination required to allow the multiple systems to operate together, rather than on the problems associated with the development of any particular system. Included in the paper is a discussion of typical problems faced during the development and coordination process, such as incompatible development schedules, difficulties in defining system interfaces, technical coordination and funding for shared tools, continually evolving planning concepts/requirements, programmatic and budget issues, and external influences. Techniques that mitigated some of these problems will also be addressed, along with recommendations for any future programs involving the development of multiple planning and scheduling systems. Many of these lessons learned are not unique to the area of planning and scheduling systems, so may be applied to other distributed ground systems that must operate in concert to successfully support space mission operations.

1.0 INTRODUCTION

THE ISS DISTRIBUTED PLANNING PROCESS

A "distributed planning" process allows multiple organizations to participate in the development of a single integrated plan. Detailed mission planning for ISS on-orbit operations is a highly distributed function, whereby pieces of the on-orbit plan are developed by multiple planning organizations, located around the world, based on their respective expertise and responsibilities. The "pieces" are then integrated to yield the final detailed plan (timeline) that will be executed onboard the ISS.

Major ISS planning organizations are provided by NASA Johnson Space Center (JSC), NASA Marshall Space Flight Center (MSFC), the Russian Aviation and Space Agency (RASA), European Space Agency (ESA), and National Space Development Agency of Japan (NASDA). The Canadian Space Agency (CSA) participates in the ISS planning process, but does not actually develop portions of the plan.

RASA, ESA, and NASDA control centers develop the plans associated with the ISS systems and payloads that those agencies provide. Within NASA, JSC develops the U.S. systems plans and MSFC develops the U.S. payload plans. JSC and MSFC also perform integration of the multiple plans. MSFC integrates the payload plans. JSC integrates the systems plans, and performs overall integration of the systems and payload plans. Figure 1 graphically depicts these responsibilities for the pre-increment planning products. See Reference #1 for more detail on this distributed planning process.

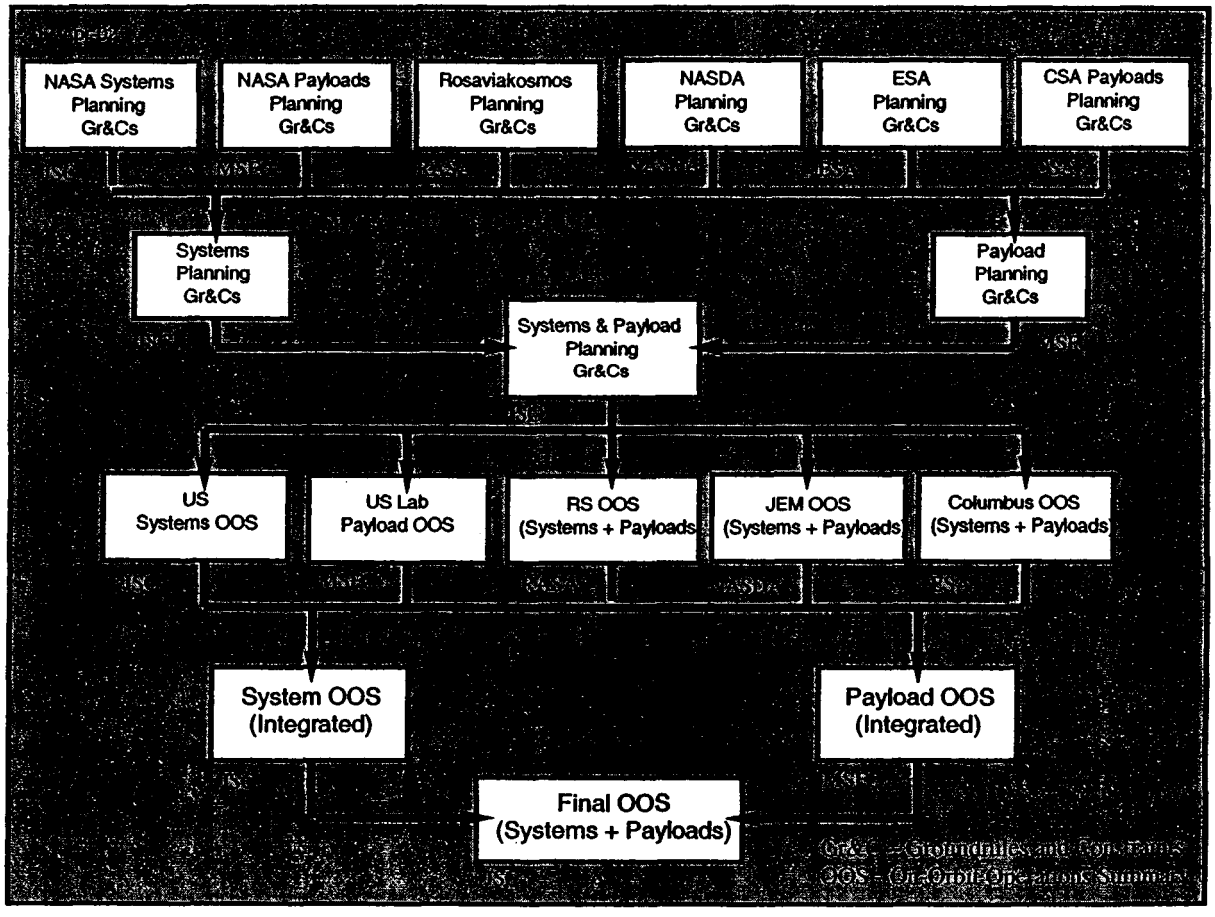


Figure 1 Pre-Increment Execute Planning Product Tree

THE NEED FOR MULTIPLE PLANNING SYSTEMS

Software systems have been developed by each of these planning organizations to support their assigned planning and scheduling functions. Although there is some cooperative development and sharing of key software components, each planning system has been tailored to meet the unique requirements and operational environment of the facility in which it operates. For example, differences in the systems may be driven by control center interfaces, unique product needs, language considerations, or the need to be accessible by remote payload users. Figure 2 shows the major ISS planning systems, and their locations.

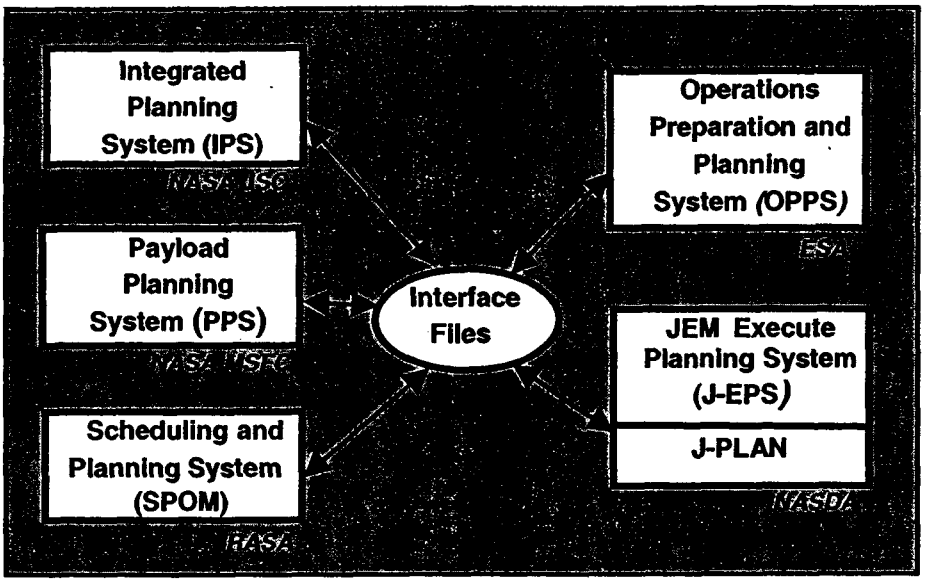


Figure 2 ISS Planning Systems

COORDINATION CHALLENGES

The distributed planning systems must effectively work together to produce a single integrated plan for ISS onboard operations, in accordance with the established planning processes. Close coordination of system requirements, interfaces, implementation, and development/deployment schedules is therefore mandatory. Unfortunately, this coordination is complicated, either directly or indirectly, by a number of factors that are inherent in a very large, multi-national endeavor, such as the International Space Station:

- The various systems are controlled by different political and/or organizational entities, so may have different funding sources or budget cycles.
- There are unique operating environments and requirements at each site.
- Some systems must support multiple programs. For example, the JSC Integrated Planning System supports both the ISS and Space Shuttle programs.
- Each system has a different development schedule.
- On an international development effort, must consider the geographical distribution of the various development organizations, language differences, and export control issues.
- Planning systems are highly dependent on many external factors, such as changes in planning concepts, onboard systems, and other ground systems.

2.0 LESSONS LEARNED

The lessons learned in this paper are from the perspective of a developer of one of the planning systems – the Payload Planning System (PPS) at Marshall Space Flight Center. They are based primarily on the author's experiences and observations while working on this planning system for the past 15 years, although some of the lessons are based on inputs from Johnson Space Center (see Acknowledgements). The developers of other systems, as well as the ISS operations community (the software users) may have additional lessons that are not reflected here. Also, these lessons focus on the coordination necessary to make the various systems work together in an integrated fashion, not on the actual development of any particular software system. For the reader who is also interested in lessons learned pertaining to the development of a particular planning system, there is a complementary paper in this conference dealing with the lessons learned in developing the Consolidated Planning System (CPS) (See Reference #2).

The lessons are grouped into five major categories: Planning Concepts and System Requirements, Sharing of Tools, Interface Definition and Control, Coordination of Development Efforts, and Applying Changes Across Systems.

PLANNING CONCEPTS AND SYSTEM REQUIREMENTS

The distributed planning concepts for the ISS are a major requirements driver for the various planning and scheduling systems. The planning concepts define the distribution of planning functions to each site, as well as the detailed planning processes, templates, and products. Each planning system must support the planning functions allocated to the site at which it operates. Unfortunately, the ISS planning concepts have evolved significantly and continually over the life of the Program, and continue to do so. On a major long-term program such as ISS, with multiple planning participants, complex political and budgetary considerations, and a continuously operating vehicle, this volatility is to be expected.

LESSON LEARNED: Recognize that planning/operations concepts will change, and plan accordingly.
As planning concepts have evolved, software requirements have had to change accordingly. Sometimes, the changes have impacted software that was well into development, thereby requiring significant software rework, with resultant cost/schedule impacts. Programs need to:

- Plan for this uncertainty in planning system budgets and schedules. Consider a phased system development, which will allow rework of requirements as the concepts change. Provide adequate sustaining budgets to accommodate the inevitable changes in planning concepts.
- Design the software to be as flexible as possible. Avoid implementing very specific operations concepts into the software. Provide for flexibility in functionality as well as in report formats to minimize rework when the concepts do change.

LESSON LEARNED: Involve the software users in all phases of development.
Active involvement of the software users, throughout the development cycle, will allow changes in planning concepts and requirements to be captured as early as possible. It will also ensure that the subsequent software implementation (functionality/user interfaces) meets their needs, prior to final software release.

LESSON LEARNED: Agree on requirements allocations across systems as early as possible.
Need to define the basic functions that each system will implement as early as possible, so that the individual systems can proceed with development. The requirements allocation across systems is generally based on the specific planning functions to be performed at each site, which makes the

allocation fairly straightforward. However, there are circumstances in which a specific function could be assigned to one of several planning systems. For example, on ISS, the MSFC Payload Planning System contains some software provided by the JSC Integrated Planning System. Decisions on which system would implement certain payload planning functions were unresolved for quite some time, resulting in cost/schedule impacts to the PPS when the functions were finally added to the PPS very late in its development cycle.

LESSON LEARNED: Make firm decisions on languages early in the requirements phase.
In a multi-national program, decisions on which languages to support must be made early. For example, the ISS program originally had a requirement to conduct all operations in English. Years later, a decision was made to support Russian Cyrillic in many tools and products, including the planning system products. Unfortunately, by this time, the planning systems development was quite far along, and the Cyrillic requirement could not be accommodated without significant cost and schedule impacts. Because of budget limitations, some systems have been updated for Cyrillic, and others have not. Any software disconnects due to Cyrillic are being managed via operational workarounds and other means. If the Cyrillic requirement had been defined much earlier, these impacts could have been minimized.

SHARING OF TOOLS

Sharing of tools occurs when software developed for one planning system is included as a component of another system. Sharing is possible when two different systems must implement very similar functions, but may or may not be practical due to operational or hardware differences, language considerations, etc. On ISS, the various planning systems developed by MSFC, RASA, ESA, and NASDA, are all utilizing the Consolidated Planning System (CPS) component of the JSC IPS to some degree. The CPS provides capabilities needed to perform the detailed scheduling of the ISS on-orbit activities.

LESSON LEARNED: Sharing of tools can yield significant benefits.
For the ISS program, the adoption of the CPS as the common scheduling tool provided several benefits:

- Scheduling tools are generally quite complex, and therefore costly, so sharing of these tools has the potential to reduce overall program costs.
- Sharing of tools promotes commonality across systems. This is especially true when the shared component is the scheduling tool, which is typically the heart of a planning system. The design of the scheduler usually determines how data is modeled throughout a planning system. Therefore, sharing of this key component across systems promotes commonality in data modeling, which may in turn simplify the interface definition.
- Commonality benefits the software end-users. The planning personnel, located at control centers around the world, can refer to common displays and data when coordinating their planning activities.
- Utilization of a common tool forces the various planning system developers to coordinate even more closely than they might have otherwise done.

LESSON LEARNED: Sharing of tools requires compromise.
Complications arise when multiple parties must share a common tool. Each party has unique requirements and expectations on the software development, which must be accommodated. Sometimes, these conflict. (See reference #2 for additional discussion on this topic.)

- All parties must be willing to live with additional complexity in the tool in order to realize the benefits of tool sharing. Extra complexity is introduced as numerous parties' unique requirements are accommodated. For example, the ISS program requires capabilities, data, or displays in CPS not

needed by the Shuttle program, and vice versa. Also, other planning systems may levy unique requirements on the shared tool (e.g., additional data parameters) to allow it to effectively operate in conjunction with their other components. MSFC has levied many unique requirements on the CPS.

- All parties must be willing to compromise when debating and prioritizing proposed changes to the shared software.
- All parties must realize that they will not get everything they want, when they want it, and may have to accept things they don't particularly like.

LESSON LEARNED: Multi-party funding of common tools can facilitate requirements satisfaction.
For example, when MSFC adopted JSC's CPS as its scheduling tool, it did not meet all the MSFC unique requirements. MSFC's requirements were folded into the ongoing CPS development, but the MSFC requirements could not compete with the more immediate needs of the JSC Space Shuttle and ISS systems planners. When it became apparent that the MSFC needs could not be accommodated within the baseline budget/schedule, the ISS program allocated additional funds to the CPS project specifically for the implementation of the MSFC payload planning requirements. From the MSFC perspective, this worked out very well. Having some separate funding under the control of each planning community mitigated the inevitable conflicts over implementation priorities/schedules, and facilitated the satisfaction of the MSFC requirements.

LESSON LEARNED: Decide on tool sharing early in the development process.
For example, the CPS tool was adopted as the scheduling tool in the Payload Planning System after the system was already well into development. This necessitated a major rework of the PPS design, because the CPS data structures were quite different from the existing PPS data. The PPS system development did not fully recover from this late impact for several years.

LESSON LEARNED: Need formal mechanisms for coordinating common tool development.
With multiple parties relying on a common tool, formal coordination mechanisms are needed to ensure that all parties have adequate participation in its development. Participation is needed in requirements definition, review and approval of proposed changes, design reviews, software testing, and problem reporting and prioritization. With the ISS program, these processes evolved over many years. The processes for reviewing and approving CPS changes are formally baselined in a Work Instruction under the control of the Ground Segment Control Board (GSCB). See Reference #3.

LESSON LEARNED: Need an established export control process, with properly trained personnel.
With international software sharing, export control issues must be considered. Export control applies to software, hardware, documentation, videotapes, and even Web pages. Having a well-documented export control process, with trained personnel and clear points of contact, allows time-critical shipments to be made on schedule.

INTERFACE DEFINITION AND CONTROL

The interface definition controls the content and format of the data to be exchanged between the various planning systems, so it is the key to success. Good processes for defining and controlling the interface are therefore crucial.

LESSON LEARNED: IT security issues may drive the interface, so address them early on.
Each planning system is associated with an ISS ground control center, which dictates a certain level of Information Technology (IT) security and access control. These constraints may force the planning

systems into a particular means of data exchange (e.g., file exchange via drop boxes, rather than direct database-to-database communications), so address them as early as possible.

LESSON LEARNED: Get formal agreement from all parties on the interface definition.
Obviously, all development organizations must commit to meet the defined interface, or disconnects between the systems may occur. Establish a process that allows all parties to review and approve any change to the interface definition before it is implemented, to ensure that each system can accommodate the change. Also keep in mind that any bilateral agreements require multilateral review and approval.

For ISS, the interface definition is designed around the CPS data structures, since the CPS tool is shared across multiple systems. Because of that, in the early years, the interface definition was basically controlled by the CPS development organization, but the ISS program has gradually evolved to a multilateral process, which seems to work better. The formal Interface Control Document (ICD) for the ISS planning systems is the Multilateral Distributed Planning Interface Specification (MUDPIS). It is baselined under the authority of the GSCB, which has multilateral representation. See Reference #4.

LESSON LEARNED: ICD development must support development schedules for all systems.
Adequate time must be provided between ICD baselining and software coding/deployment. The ICD development schedules should be laid out carefully to ensure that no system is forced to accommodate an interface change late in a development cycle. This is complicated by the fact that all the systems have different schedules. Before the ISS program established standard review processes for the MuDPIS ICD, such schedule disconnects did cause some impacts to ongoing development activities. In addition, the standard ICD review cycles must accommodate the internal review processes at each center. The right balance must be found between the need for more review time, and the need to get the document out in time to meet the various development schedules.

LESSON LEARNED: Beware an interface definition that is explicitly tied to one application.
For ISS, the interface files are explicitly tied to the internal CPS database structures. This allowed for a fairly quick/cheap implementation, but such tight coupling can cause complications. For example, data used only by the CPS application, such as display formats, is an integral part of the interface definition, and must be dealt with by the other systems. Therefore, changes in this data can force a change in the interface definition, even if no other planning systems utilize the data.

LESSON LEARNED: Document data integrity rules in the ICD.
The ICD must document data integrity rules. This is to prevent "bad" data from entering the system. The data integrity rules include items such as mandatory data fields, value constraints on specific fields, or relationships between data elements. Missing rules caused problems for ISS more than once.

LESSON LEARNED: Document standards for user-controlled data.
In planning systems, the creation and naming of the planning data is under the control of the software user. This is because the activities to be planned, and the resources/conditions which constrain the planning function, are different for every flight/mission. But since these names are often identifying keys that the planning systems use to process and integrate the various plans, they must be consistent across all the systems. Such information should not go into the software ICD for several reasons: it is rather dynamic, and it is under the control of the software users, rather than the developers. For ISS, these data standards are baselined under the auspices of the Execute Planning Control Panel (ExPCP), a multilateral panel representing each of the distributed planning organizations. See Reference #5.

COORDINATION OF DEVELOPMENT EFFORTS

For the distributed planning systems to effectively work together, close coordination of the independent development efforts is mandatory.

LESSON LEARNED: Establish regular communications forums to facilitate effective coordination.
Need to establish mechanisms that promote regular communications, such as monthly telecons, e-mail, and periodic face-to-face meetings. With ISS, the various parties are geographically distributed across the globe, so communications must consider time zone and language differences.

LESSON LEARNED: Encourage participation in each other's technical reviews.
Such participation was not as extensive as it could have been for ISS, but where it was done, it was extremely beneficial. From the PPS perspective, detailed technical participation in the development of the common CPS tool provided an in-depth understanding of the CPS capabilities, interface definition, and associated implications on the PPS system. Therefore, any potential disconnects between the two systems could be quickly identified and resolved. Also, such participation is a great way to share ideas.

LESSON LEARNED: Need strong integration function to lead the coordination.
Need one party to serve as the prime coordinator, responsible for setting up and implementing the day-to-day coordination processes. NASA JSC assumed this role for ISS. Also need a formal authority with multilateral representation to oversee/approve the coordinated plans. For ISS, this is the GSCB.

LESSON LEARNED: Need clear avenues for issue resolution.
Need to clearly identify paths/forums for resolving any issues that may arise, whether in the development process, or on into operations. Existing program board structures can be utilized for this purpose.

LESSON LEARNED: Need mechanisms to negotiate development/deployment schedules.
Establish mechanisms to negotiate and document key schedule milestones. This provides long-term visibility into the schedules, for planning purposes, and provides some level of commitment to the negotiated schedules. Address all critical events which impact more than one system, such as ICD completion dates, and system deployment dates. The individual systems can then plan their own development schedules around these milestones. For ISS, these agreements are documented in a Planning Facility Schedules document (Reference #6), which is under the control of the GSCB.

LESSON LEARNED: Decouple development schedules to the extent possible.
Development/deployment schedules for the various systems are not likely to line up, for many reasons. This can cause major problems when the interface definition changes. For example, the Payload Planning System once had to add an extra (unplanned) build in order to maintain compatibility with the JSC system when it deployed upgraded software (containing interface changes). This resulted in a cost/schedule impact to the PPS. It is therefore of utmost importance to find some mechanism which allows the schedules for the various systems to be decoupled. The ISS program utilizes the concepts of "sync points" and "backward compatibility", which appear to work quite well.

A *sync point* is represented by a specific version of the interface definition, which all parties have agreed to meet for data exchange. Later versions of the interface definition may exist, but no system is required to utilize the later version, until it is declared to be the next sync point by the entire planning community. All the development organizations must then agree on the schedule for upgrading their software to meet the new sync point. A "no-later-than" date is established, by which each system must be upgraded. During the transition period, the old sync point interface definition is still valid, and data can be

exchanged in either format. In other words, the software is *backward compatible* with the previous sync point. For ISS, the agreements on sync points and upgrade/deployment schedules are documented in the Planning Facility Schedules document (Reference #6).

LESSON LEARNED: Coordinate platform and COTS requirements as far out as possible.
This coordination is mandatory when sharing software tools, to prevent any "surprises" that might impact schedules. Identify any platform and COTS (Commercial Off-The-Shelf software) requirements as early as possible to accommodate the long lead times needed for procurements and installation.

LESSON LEARNED: Joint testing is required for successful checkout of the software.
To ensure that the various systems effectively work together, plan for some joint testing where data is flowed through all the systems. Enlist the participation of the user organizations in performing an end-to-end test of the distributed planning process, to simulate the operational environment.

LESSON LEARNED: Joint effort is required to resolve operational problems.
When problems arise in the operational environment, a responsive development/user team is needed to identify, diagnose, and resolve the problems. Keep in mind that a problem may manifest in one system, but really be caused by another (e.g., through the exchange of "bad" data).

APPLYING CHANGES ACROSS SYSTEMS

Because of the interdependence of the distributed planning systems, changes tend to ripple from one system to the next. It is therefore of critical importance to coordinate and manage these changes. Many external factors can force changes in the planning systems, including the planning concepts, which have already been discussed. By their nature, planning systems are also quite sensitive to changes in onboard systems and other ground systems, such as operations control centers. This is because they either model those systems, for planning purposes, or have software interfaces for the exchange of planning data.

LESSON LEARNED: Approve/fund changes across ALL affected planning systems.
When the distributed systems have different funding sources, as they do on ISS, it is always possible that a change may be approved/funded for one system, but not for others. A prime example of this is the implementation of Russian Cyrillic, which was funded for some, but not all, planning systems.

LESSON LEARNED: Consider impacts to planning systems when changing onboard systems.
Impacts to planning systems are not always considered when changes to onboard systems are being assessed. Any impacts to the planning systems should be identified and funded at the same time the onboard change is approved.

3.0 CONCLUSIONS AND RECOMMENDATIONS

The International Space Station Program has successfully demonstrated that it is possible to develop multiple planning systems that operate together in the development of a single integrated plan. However, the road to success has not been straight or smooth. Other programs considering distributed planning and scheduling should learn from the ISS experiences. Also, many of these lessons learned are not unique to the area of planning and scheduling systems, and may be applied to other distributed ground systems that must operate in concert to successfully support space mission operations.

The overriding lesson learned is the need to take a global perspective when developing multiple planning systems to support distributed planning. The entire collection of systems must be considered when defining requirements and budgets, approving changes, and coordinating development activities.

4.0 REFERENCES

1. SSP 50501, Pre-Increment Execute Planning Process Definition, Generic, April 2002.
2. Saint, R., "Lessons Learned in Developing an International Planning Software System", Space Ops 2002, the Seventh International Symposium on Space Mission Operations and Ground Data Systems, Houston, Texas, October 2002.
3. GSCB 0400, Work Instruction for Integrated Planning System (IPS) Change Request (CR) Protocol Process, April 2002.
4. SSP 50401-C15, Multilateral Distributed Planning Interface Specification (MuDPIS), Consolidated Planning System (CPS) Cycle 15, Rev A, February 2002.
5. ExPCP Planning Data Standards – Generic, November 27, 2001.
6. SSP 50606, Planning Facility Schedules Document, February 2002.

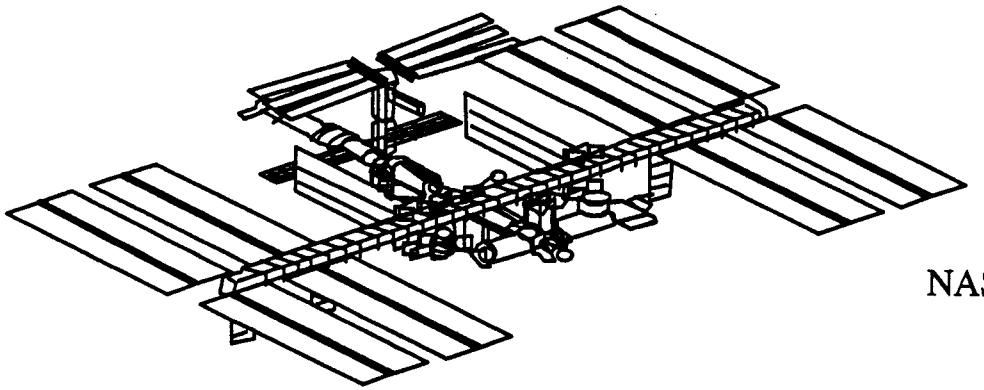
5.0 ACKNOWLEDGEMENTS

The author would like to thank Ms. Dena Hess of NASA Johnson Space Center for reviewing this paper, and for providing invaluable inputs on the lessons learned, from the perspective of the Integrated Planning System (IPS) development organization. Thanks, also, to Mr. Gary Rowe of Teledyne Brown Engineering, for providing current information related to the ISS distributed planning processes.

6.0 ACRONYMS

COTS	Commercial Off-The-Shelf
CPS	Consolidated Planning System
CSA	Canadian Space Agency
ESA	European Space Agency
ExPCP	Execute Planning Control Panel
Gr&C	Groundrules and Constraints
GSCB	Ground Segment Control Board
ICD	Interface Control Document
IPS	Integrated Planning System
ISS	International Space Station
IT	Information Technology
JEM	Japanese Experiment Module
J-EPS	JEM Execute Planning System
JSC	Johnson Space Center
MSFC	Marshall Space Flight Center
MuDPIS	Multilateral Distributed Planning Interface Specification
NASA	National Aeronautics and Space Administration
NASDA	National Space Development Agency of Japan
OOS	On-Orbit Operations Summary
OPPS	Operations Preparation and Planning System
PPS	Payload Planning System
RASA	Russian Aviation and Space Agency (Rosaviakosmos)
SPOM	Scheduling and Planning System

Lessons Learned In Developing Multiple Distributed Planning Systems for the International Space Station (ISS)



Theresa G. Maxwell
NASA Marshall Space Flight Center
Flight Projects Directorate
Ground Systems Department
256-544-2232
theresa.maxwell@msfc.nasa.gov

Scope of Paper

- ***Lessons Learned*** while developing multiple planning systems for the International Space Station (ISS)
  - From the perspective of a system developer
    - Payload Planning System (PPS) at NASA's Marshall Space Flight Center (MSFC)
    - Experiences and observations since 1987
    - Includes inputs from planning system developers at NASA Johnson Space Center (JSC)
  - Focus is *coordination* across systems so they can work together to support ISS "distributed planning"

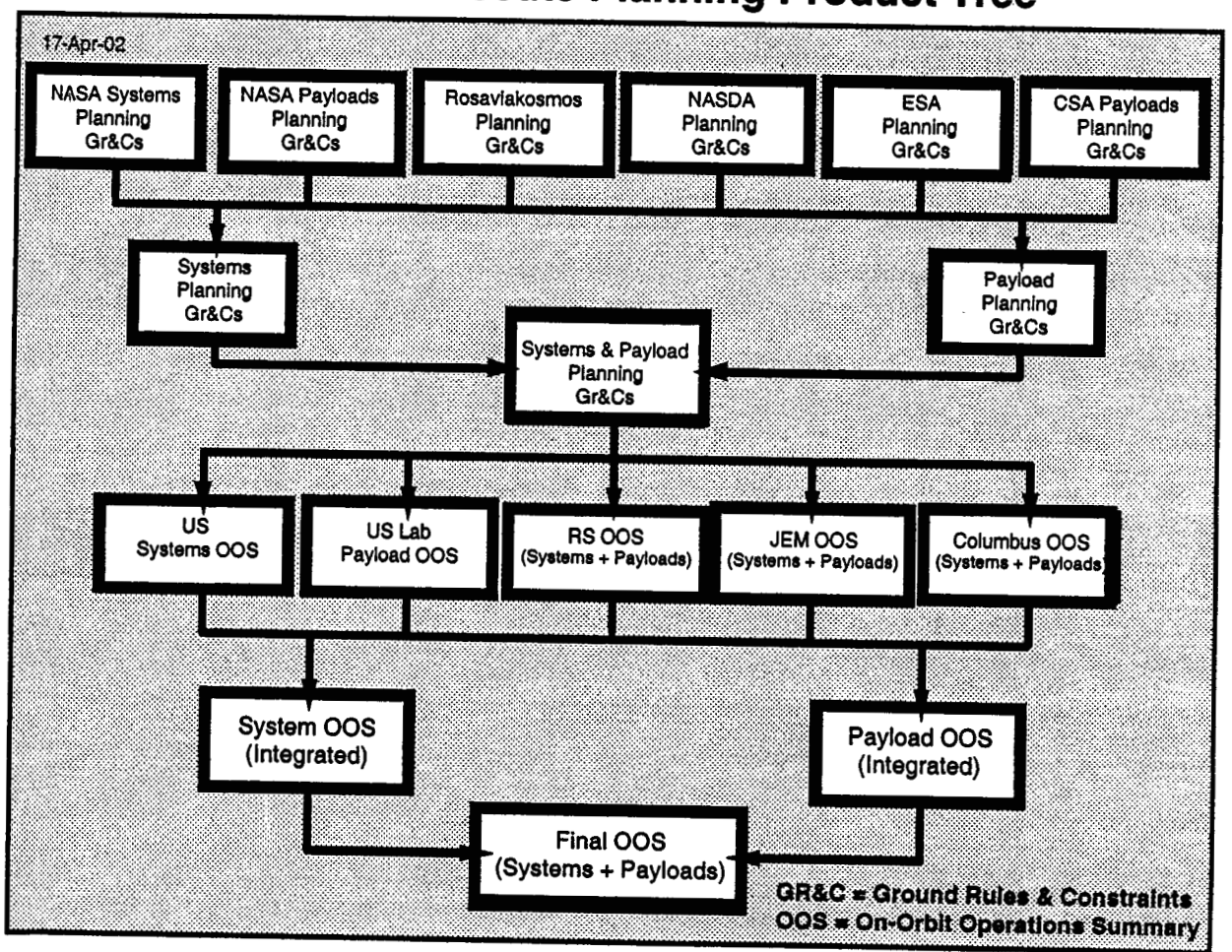
What is "Distributed Planning"?

- Multiple organizations participate in developing a single integrated plan
- "Pieces" developed by the distributed experts
  - Partner/segment plans
  - Systems plans
  - Payload plans
- Pieces then integrated

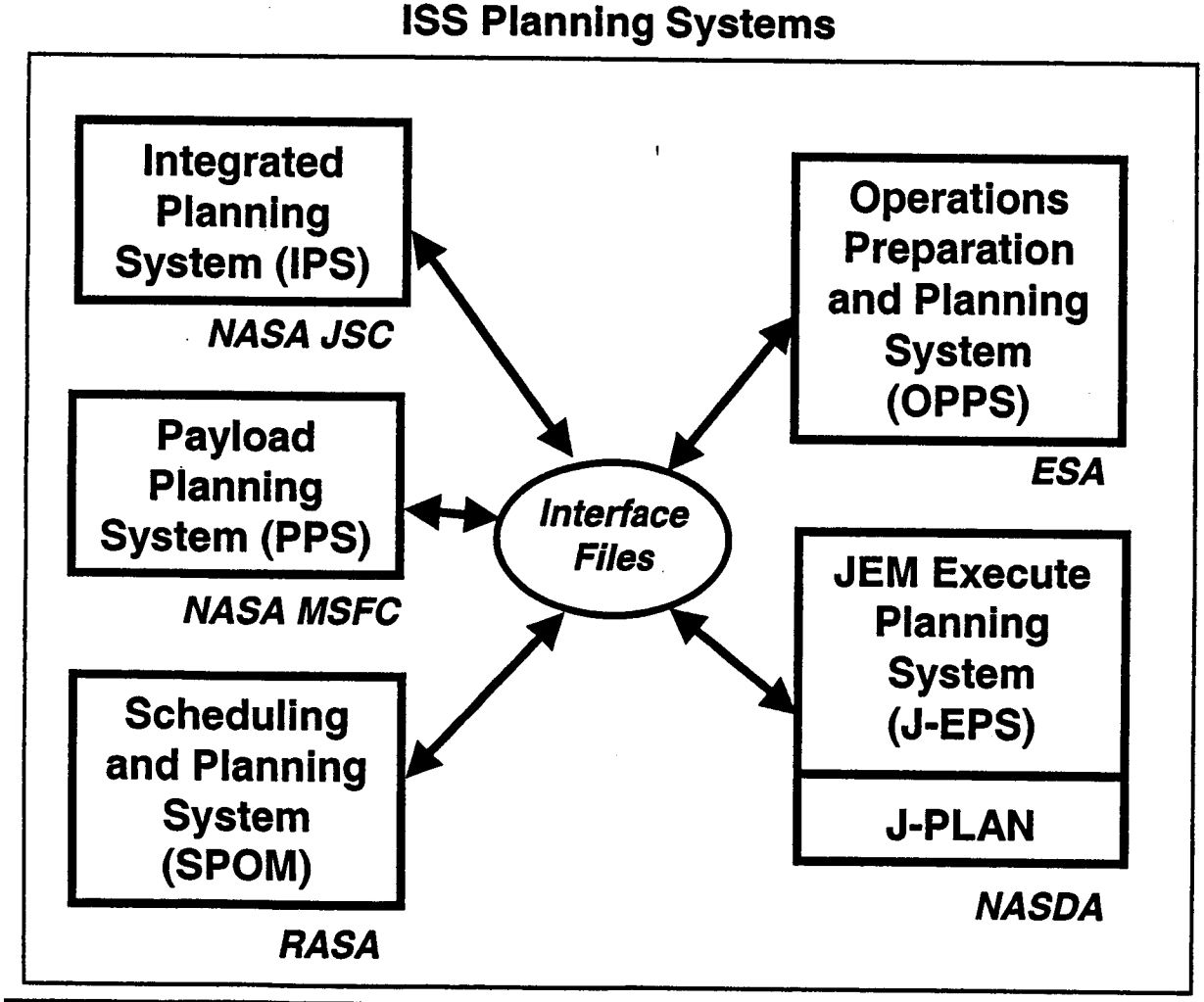
ISS Planning participants:

- NASA Johnson Space Center
- NASA Marshall Space Flight Center
- Russian Aviation & Space Agency
- National Space Development Agency of Japan
- European Space Agency
- Canadian Space Agency

Pre-Increment Execute Planning Product Tree



Why Multiple Planning Systems?



- Each planning system is tailored to the needs of its home site:
  - Supports allocated planning functions
  - Systems versus Payload focus
  - Unique control center constraints
  - Unique products to support operations
  - Language considerations

Challenges

- GOAL: The various planning systems must effectively work together to produce a single integrated plan for ISS onboard operations

- Complicating factors:
  - Each system controlled by a different agency/organization
  - Unique operating environments and requirements at each site
  - Some systems support multiple programs
  - Different development schedules
  - International development effort
  - External dependencies (changes in operations concepts, onboard systems, other ground systems, etc.)

Lessons Learned

- The Lessons Learned are grouped into five categories:
  - Planning Concepts and System Requirements
  - Sharing of Tools
  - Interface Definition and Control
  - Coordination of Development Efforts
  - Applying Changes Across Planning Systems

Planning Concepts & System Requirements

Planning concepts (roles, processes, products, etc.) drive the requirements. On ISS, these concepts have continually evolved.

LESSON: Recognize that planning concepts will change, and plan accordingly

- Plan for this uncertainty in planning system budgets and schedules
  - Phased development
  - Adequate sustaining budgets
- Design flexibility into the software (functionality, report formats, etc.)

LESSON: Involve the software users in all phases of development

- To capture changes in planning concepts and requirements
- To ensure functionality and user interfaces meet their needs

Planning Concepts & System Requirements

LESSON: Agree on requirements allocations across systems early on

- Which systems will implement which functions?
- Potential cost/schedule impacts if not decided early on
  - *Example*: Early debates over allocation of certain payload planning functions to the JSC or MSFC planning systems – when finally decided, was almost too late to implement

LESSON: Make firm decisions on languages early in requirements phase

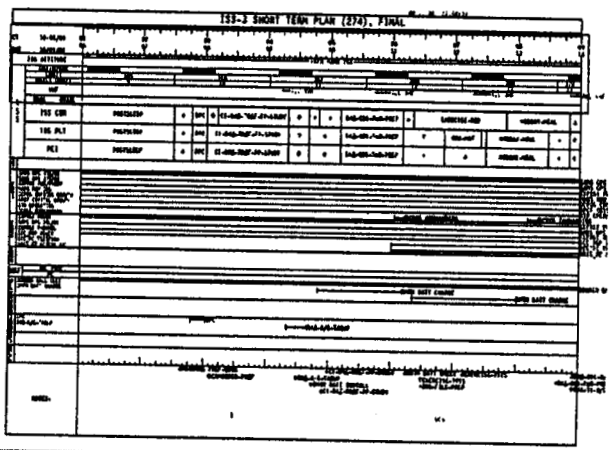
- In multi-national program, which languages to support?
- *Example:*
  - ISS began as "English-only" program
  - Russian Cyrillic added after years of software development
    - Software impacts to update fonts, displays, etc.
    - Some systems funded for updates; others not

Sharing of Tools

On ISS, all the planning systems utilize the Consolidated Planning System (CPS) component of the JSC software to some degree.

LESSON: Sharing of tools can yield significant benefits

- Potential to reduce overall program costs
- Promotes commonality across systems
  - Simplifies interface definition
  - Benefits end-users (common displays, etc.)
- Encourages close coordination between developers



LESSON: Sharing of tools requires compromise

- Additional tool complexity to support multiple parties' requirements
- Parties must jointly debate and prioritize proposed changes
- No party gets all they want; and must live with things they don't like

Sharing of Tools

LESSON: Multi-party funding can facilitate requirements satisfaction

- Helps to mitigate conflicts over use of limited development resources
- *Example:* Separate funding for MSFC-unique requirements on CPS

LESSON: Decide on tool sharing early in the development process

- *Example:* Retrofitting of CPS into MSFC Payload Planning System (PPS) caused significant rework

LESSON: Need formal mechanisms for coordinating shared tool development

- Multi-party participation in requirements definition, change approval, design reviews, software testing, and problem reporting/prioritization

LESSON: Need an established export control process, with trained personnel

- In multi-national program, must consider export control issues
- Established process allows time-critical shipments to be made on schedule

Interface Definition / Control

With multiple systems, the Interface Definition is the KEY to success.

LESSON: IT security issues may drive interface, so address them early on

- Database interfaces vs. file exchange/drop boxes, etc.

LESSON: Formally agree on definition via Interface Control Document (ICD)

- Multilateral review/approval of any changes
- Commitment to meet the interface
- Baselining authority with multilateral representation

LESSON: ICD development must support all individual system schedules

- Adequate time between ICD baselining and software coding/deployment
- Review cycles must accommodate internal review processes at each site

Interface Definition / Control

LESSON: Beware an interface definition explicitly tied to one application

- *Example:*
  - For ISS, interface files are tied to the CPS internal database structures
  - Changes in data used only by the CPS (e.g., display formats) force a change in the interface definition

LESSON: Document data integrity rules in the ICD

- Include mandatory data fields, data validation rules, etc.
- Prevents "bad" data from entering the system

LESSON: Document standards for user-controlled data

- In planning systems, creation/naming of resources and activities to be planned/scheduled is under control of the software user
- Document user agreements on these to synchronize data across systems

Coordination

Close coordination of the independent development efforts is mandatory.

LESSON: Establish regular communications forums

- Monthly telecons, e-mail, periodic face-to-face meetings, etc.
- With multi-national program, consider time zone and language differences

LESSON: Encourage participation in each other's technical reviews

- Can identify/resolve potential disconnects before they become problems
- Opportunity to share ideas

LESSON: Need strong integration function to lead the coordination

- One party should serve as prime coordinator
- Formal authority with multilateral representation to oversee/approve the coordinated plans (*Example:* for ISS, this is Ground Segment Control Board)

Coordination

LESSON: Need clear avenues for issue resolution

- Define avenues throughout the development process, and on into operations

LESSON: Need mechanisms to negotiate development/deployment schedules

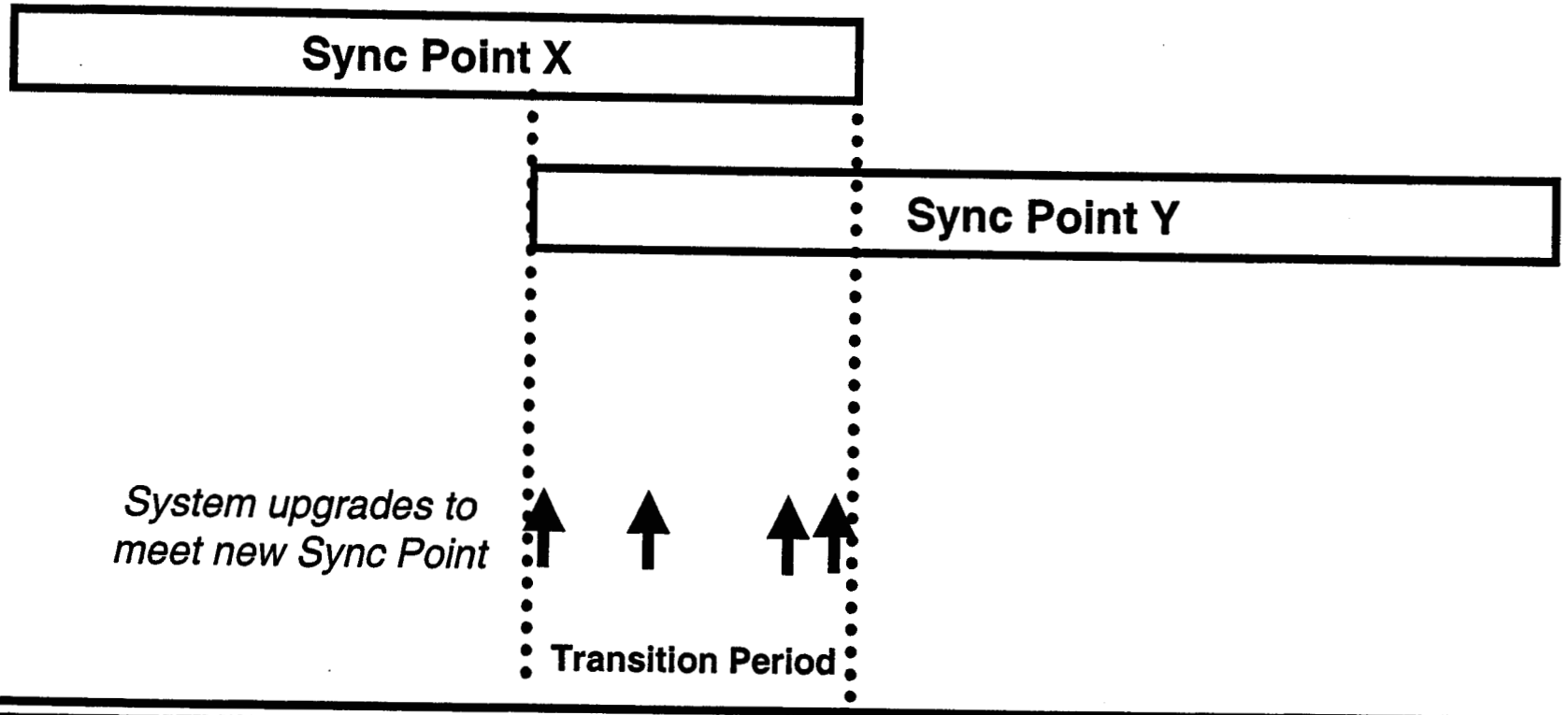
- All parties must commit to negotiated schedules
- Address all critical events: ICD development, system deployments, etc.
- *Example:* ISS has a "Planning Facility Schedules Document"

LESSON: Decouple individual development schedules to the extent possible

- Individual development/deployment schedules not likely to line up
- Can cause major problems when interface definition changes
- Find mechanism which allows schedule decoupling
  - *Example:* ISS utilizes "backward compatibility" and "Sync points"

Coordination

A ***Sync Point*** is a specific version of the interface definition that all systems have agreed to meet by a certain date. The software is ***Backwards Compatible*** with the previous sync point during the transition period.



Coordination

LESSON: Coordinate platform and COTS requirements as far out as possible

- Mandatory if sharing software tools
- Prevents "surprises" which can impact schedules
- Accommodates long lead times for procurements and installation

LESSON: Joint testing is required for successful software checkout

- Cooperative effort between development and user organizations
- Perform end-to-end testing to simulate the operational environment

LESSON: Joint effort is required to resolve operational problems

- Problem may manifest in one system, but be caused by another
- Responsive development/user team is needed to identify, diagnose, and resolve operational problems

Applying Changes Across Systems

Because of their interdependence,
changes tend to ripple across planning systems.

LESSON: Approve/fund changes across ALL affected planning systems

- With separate funding sources, a change may not be approved/funded across all affected planning systems
- *Example:* Implementation of Russian Cyrillic was not funded system-wide

LESSON: Consider impacts to planning systems when changing external systems

- Planning systems are sensitive to changes in onboard systems and other ground systems
- Programs should consider impacts to planning systems when assessing other changes, and fund accordingly

Conclusions

- Multiple planning systems can operate together in the development of a single integrated plan
- Other programs can learn from the ISS experiences
- Over-riding lesson: Need to take a ***global perspective*** in requirements, budgets, change approval, and system coordination