INTEGRATING PAYLOAD DESIGN, PLANNING AND CONTROL IN THE DUTCH UTILISATION CENTRE

T. J. Grant

BSO/Aerospace & Systems b.v.
P. O. Box 1444, 3430 BK NIEUWEGEIN, The Netherlands
Tel: ++31-3402 88888 Fax: ++31-3402 60577

ABSTRACT

Spacecraft payload design, experiment planning and scheduling, and payload control are traditionally separate areas of activity. This paper describes the development under Dutch Government contract of a prototype software tool - the Activity Scheduling System (ASS) - which integrates these activity areas. ASS is part of a larger project to build a Dutch Utilisation Centre (DUC), intended eventually to support all space utilisation activities in The Netherlands. ASS has been tested on the High Performance Capillary Electrophoresis payload. The paper outlines the integrated preparation and operations concept embodied in ASS. It describes the ASS prototype, including a typical session. The results of testing are summarised. Possible enhancement of ASS, including integration into DUC, is sketched.

Key Words: Spacecraft payload, design, planning, control, integration, Dutch Utilisation Centre.

1. INTRODUCTION

Spacecraft payload design, experiment planning and scheduling, and payload control are traditionally separate areas of activity. Software tools supporting one area are rarely integrated with tools supporting another area. The purpose of this paper is to describe the development of a prototype integrated software tool for designing scientific payloads, for planning and scheduling experiments using payloads, and for controlling payloads using the resulting schedules. The prototype - known as the Activity Scheduling System (ASS) - has been implemented under a Netherlands Agency for Aerospace Programs contract, as part of a larger project to build a pilot Dutch Utilisation Centre (DUC). DUC is intended eventually to support all space utilisation activities in The Netherlands. The DUC user may be either the payload's Principal Investigator (PI) or its Facility Expert (FE).

ASS is being tested on the High Performance Capillary Electrophoresis (HPCE) payload, which is the standard case study for developing DUC-Pilot software. HPCE is a general analytical technique for separating molecules by transporting charged particles through an electrolyte fluid in a fine capillary under the influence of an electric field (Ref 1). The HPCE payload is designed to be used for a wide variety of experiments in microgravity conditions, covering physical, chemical and biological processes, as a flexible, multi-user measurement facility serving many experiments on-board Columbus and many experimenters, both in-orbit and ground-based. HPCE can also be flown on other spacecraft, including ballistic sounding rockets, unmanned satellites (eg Eureca), and the Space Shuttle. In ASS, the HPCE payload is currently modelled as 27 entity-classes and 52 relation-classes, resulting in three sets of planning operators for payload assembly, preparation and operation.

The paper outlines the integrated preparation and operations concept embodied in ASS. It describes the ASS prototype, including a typical HPCE session. The results of ASS testing are summarised. Finally, possible enhancement of ASS, including integration into DUC, is sketched.

2. CONCEPT

2.1 Columbus USO and DUC

ESA has identified the need for an organisation to coordinate the use of Columbus. Earlier this year an international team reported to ESA on the definition of

a User Support Organisation (USO) for Columbus. The Columbus USO they proposed is a hierarchy, headed by a single Payload Operations Control Centre (POCC). Below the POCC, there could be one User Support Operations Centre (USOC) per nation. Each PI would have a User Home Base (UHB), which would be connected by communications networks to his/her nation's USOC. Mirroring the ESA-wide (E-)USO there would be national USOs (N-USOs). Nations could choose to implement their USOC with UHB functionality for PIs without their own UHB facilities; the USOC-UHB combination would be known as a Utilisation Centre (UC).

The Dutch User Support Organisation (DUSO) is seen as The Netherlands' N-USO. The DUSO would have as its primary goal to maximise the scientific output of microgravity research in The Netherlands. The Dutch user support concept (Ref 2) is based on performing as much as possible of the experiment preparation and operation in the user's own laboratory. There would be a Dutch Utilisation Centre which would support selected USOC-level functions and which could also provide UHB-level functions for those users who do not have equivalent facilities in their own laboratories. The DUC should be mobile, eg for positioning at a PI's location during an operations campaign. At a meeting in March 1992, representatives of the Dutch user community agreed that - funding permitting - a phased programme should be started to realise the DUSO and its associated DUC.

In a linked initiative, a consortium of Dutch organisations and companies, led by the Nationaal Lucht- en Ruimtevaartlaboratorium (the Dutch National Aerospace Laboratory), has started the detailed definition of the DUSO. The development concept (Ref 3) distinguishes two approaches:

> *The 'top-down' or 'formal' approach*, which would result in a DUC developed according to a standard systems development method.
>
> *The 'bottom-up' approach*, which would begin with the choice of a candidate experiment, ie a case study. DUC functions would be demonstrated by re-using existing infrastructure and applications in a pilot DUC environment.

In mid-1991, the consortium decided to progress both approaches in parallel. The HPCE payload was selected as the case study, with development of the DUC-Pilot environment beginning in late 1991. The HPCE Case Study sub-project quickly resulted in a set of functional requirements (Ref 4) which served as an input to DUC-Pilot development. In the absence (in January 1992) of a space version, the consortium decided to baseline its activities on a commercially-available HPCE instrument (Ref 5) designed for use in (ground-based) laboratories.

DUC-Pilot Development re-uses the following sub-systems developed by consortium members:

- A generic Man-Machine Interface (MMI) tool.
- A payload simulator (Ref 6).
- A Planning and Control (P & C) sub-system, which became the Activity Scheduling System.
- A communications network simulator.
- A multimedia Document Filing System.
- A Multimedia Telesupport System for multimedia communication between PI/FE and crew.
- An interface to the Columbus Utilisation Information System.

These sub-systems were demonstrated as stand-alone applications, together with a Beckman HPCE instrument, at the 1992 European Conference on the International Space Year in Munich, Germany. In Phase 2 of DUC-Pilot Development the sub-systems are being integrated, for completion in December 1992.

2.2 Experiment Life-Cycle

The USO Definition Team has defined an experiment life-cycle which can be used to decide where N-USO support could be provided to Columbus users. The highest potential for N-USO support is to be found in those life-cycle processes which are generic (ie non-discipline specific), which are not better done at an international level, and which are not closely tied to the scientific peer judgement system. Such processes include payload design, experiment preparation, planning, and operations (including re-planning and maintenance).

At present, these processes involve the PI in generating large amounts of documentation: requirements documents, design documents, operating procedures, plans, post-flight reports, and so on. The one matter that all PIs agree on is that they would prefer to spend their time and effort in doing science, not in completing documents. Therefore, one good way of providing user support would be to reduce the amount of non-scientific documentation that the PIs must produce manually.

Documents are largely a means of exchanging information across system boundaries, whether the systems are people, organisations, or computers. One approach to reducing the manual production of documentation would be to identify the systems and the interfaces between them. There are two categories of interface: internal and external interfaces. Within an

N-USO, the internal interfaces are between the different processes in the experiment life-cycle. These interfaces could be made more transparent by integrating the processes. External interfaces, eg from an N-USO to the E-USO, could be made more transparent by making the processes open systems. In other words, nationally-provided tools must be able to exchange data with similar tools used at the international level; this would require a set of data-exchange standards. An enabling pre-requisite would be an agreement concerning the experiment preparation and operations concept underlying the processes. This paper proposes such a concept.

2.3 Preparation and Operations

2.3.1 Approach

In the development of ASS, attention has been focused from the start on making the internal interfaces between the life-cycle processes more transparent. The approach has been to determine the types of data that flow between them. The type of data output by one process must match the data-type input by the next.

2.3.2 Brief Theoretical Review

A very brief review of control, planning, and design theory is needed here. The process of control is intimately bound to the operation of a system, such as a payload. General systems theory regards a *system* as a process which behaves by taking inputs from its environment, processing them, and sending outputs back to its environment. *Control* is present when a feedback loop is added in which the outputs are inspected and compared with some objectives, and the results are added to the inputs. Controlling consists of making changes in the state of the operating system in order to influence what will occur in future and when it will occur. I distinguish between the subsystem under control and the controlling subsystem. The literature on control systems theory is vast; (Ref 7) is a good introductory text.

I define *planning* as the process of selecting and instantiating actions from the set of all known feasible actions for the system under control and logically ordering them into a sequence that will, on execution, transform a given initial state of the system under control into a desired goal state. This definition is consistent with that used in Artificial Intelligence (AI) (eg see (Ref 8)). Note that I make no stipulations regarding the constraints used in selecting, instantiating, and ordering the actions. Planning is a general process which can be specialised according to the type of constraint used. According to this view, *scheduling* is the specialisation with additional, time-related constraints. As defined, the planning process is a combination of *resource allocation* and *sequencing*.

There are four key points here. First, the planning process takes place before the actions are performed. The planning process does not have to be completed before planned actions can be performed. As soon as an initial portion of the plan has been generated, then performance of that portion can begin. Second, the product output by the planning process is a data-structure - the *plan* - listing the intended actions by name, together with the resources to be used in performing those actions. Very often, the plan is realised as a (human-readable) document, or some electronic analogue. The plan is the input to the control process; in AI jargon, the plan is *executed* by the control process. Third, planning takes as its inputs (descriptions of) the feasible actions, the initial state, and the goal. Most importantly, the planning process is goal-directed. Notably, the goal is expressed in terms of the state that is desired. Fourth, planning does not itself satisfy the goal, but is necessary for its satisfaction. Only when the plan is executed are the plan's goals satisfied. Unplanned (ie non-directed) action gives no guarantee that any goals will be satisfied.

Design can be seen as another specialisation of planning in which actions have additional, geometric constraints. The inputs to design are the system's mission and a set of production constraints, eg for materials and facilities. The output - the *design* - is a specialised plan, which specifies the component parts of a system (cf resources) and the actions required to construct or assemble them. The design is executed by constructing or assembling the component parts to become the system as designed. The as-designed system represents the resources to be allocated during planning, and, after construction/assembly and planning, becomes the subsystem under control.

Note that the output of design does not specify the set of feasible actions that may be performed when operating the subsystem under control. However, planning needs a set of feasible actions as a part of its inputs. Therefore, some process must intervene between the design and planning processes to transform the output of design into the input of planning. The intervening process will be named *action-set generation*, which will also have operating constraints as an input.

2.3.3 Current Practice

Currently, spacecraft design makes extensive use of CAD techniques and tools. The "Mission" input to design takes the form of a document known either as

the Spacecraft Users Manual (SUM), or as the Operations Requirements Handbook (ORH). The SUM/ORH is authored by the spacecraft manufacturer using conventional word-processing facilities. There is an ESA standard for the layout and contents of an ORH, and ESA is currently prototyping an expert-system-based tool to support the authoring of ORHs to this standard. The output of the design process takes the form of documentation, often paper-based.

The design process is divorced from planning and control, because action-set generation is done manually. In the Western world, the space industry practice is to combine action-set generation with the planning process. Action-set generation results in the production of spacecraft operating procedures for routine and non-nominal operations. These procedures can be seen as sequences of action-descriptions, ie generic plans. For older spacecraft, these procedures are also published as paper-based documents. Planning then consists of retrieving the procedure appropriate to the current spacecraft status, instantiating it, and passing it to the Spacecraft Control System for execution. The planning and control processes are only linked electronically in the most recent spacecraft projects, and then only by file transfer. On-line links between planning and control are under development for Columbus.

2.3.4 Integration Concept

The proposed integration of design, planning and control is diagrammed in Figure 1 by means of the SADT notation (Ref 9). For clarity, the control and resource inputs to the SADT processes have been omitted. In the SADT notation, control inputs enter an SADT process from above, and resource inputs enter an SADT process from below. For example, the design output from the design process would also be an input to another SADT process (named "production", say). The output of the production process would be the subsystem under control, and this would become a resource input to the Control (& operation) SADT process. Moreover, each of the SADT processes could be supported by a tool, eg a CAD package for design, an action-set generation tool, a planning and scheduling tool, and a control system. These tools would have to be shown as resource inputs, and additional SADT processes would have to be added to represent the tool development processes.

There is one refinement not shown in Figure 1. Currently, spacecraft designers produce their designs in the form of documents, albeit aided by CAD tools. Later, the design document, together with the manually-produced operating procedures, would be used as the inputs to the development of a software simulator used by spacecraft operators for training and contingency recovery. In the proposed preparation and operations concept, the development of a simulator would come first. The designers would work with a simulation-authoring tool to produce their designs in the form of a software simulation. They would explore the behaviour of this simulation, adjusting it as necessary. When satisfied with the result, the designer would trigger the tool to generate the design documentation automatically. The generation algorithm would be designed to produce the document according to the appropriate standards. This principle of automated generation of documentation would be repeated in the action-set generation, planning, and control processes. Although there would be little saving in the amount of documentation produced, the PI/FE would be spared the effort of creating and structuring documents. Moreover, consistency between documents and simulator would be ensured.

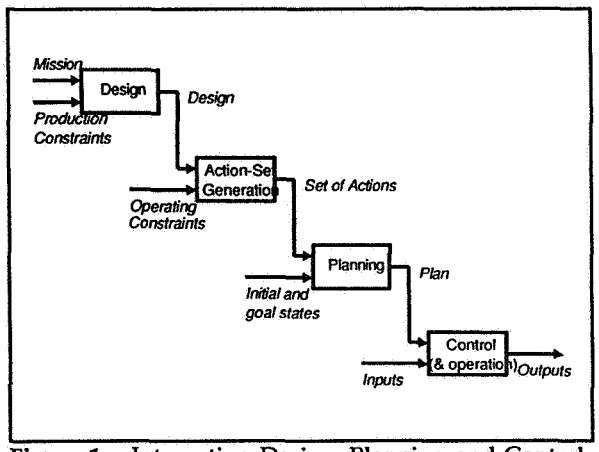

Figure 1. Integrating Design, Planning and Control.

Automatically-generated documentation has the further advantage that it has a regular structure that could be parsed by another algorithm, enabling automation of its input by another tool. In essence, such documentation is readable both by humans and by machines. For example, a design document generated automatically by the simulation-authoring tool could then be automatically input to an action-set generation tool.

3. INTEGRATED TOOL

3.1 Implementation Status

The ASS prototype has been implemented in Smalltalk/V on 386-class PCs under MS-DOS. The intended coverage and implementation status of the ASS prototype can be seen from Figure 2. The experiment life-cycle is shown in summary form from

left to right, with the processes covered by AI planners and schedulers shown beneath it. The dashed parts of the range for ASS show the functionality that, at the time of writing (November 1992), has yet to be implemented.

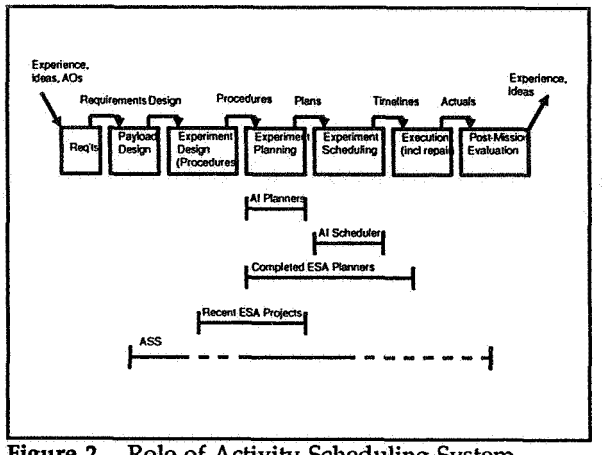

Figure 2. Role of Activity Scheduling System.

The implemented functionality is as follows. ASS supports payload design by representing it as an Entity-Relation model (Ref 10), extended with domain constraints. The user lists the classes of entities in the payload domain, the relations between those classes, and the instances of each entity-class. ASS then prompts the user for the constraints between relation-pairs. From these inputs, ASS generates a payload simulator whose behaviour the user can explore. When satisfied, the user can instruct ASS to induce an STN from the payload simulator. This induction process is computationally intensive. When the STN has been induced, the user can associate the use of external resources with each state and transition. Payload operating plans, together with summarised resource-usages, can be extracted from the STN for user-designated start and end states. At any time, the user can trigger the generation of a detailed design document. In addition, the user can initiate the automated generation of an ASCII file for input to an object-oriented analysis (OOA) tool which supports the Coad and Yourdon OOA method (Ref 11). Alternatively, the user may load the lists of entities and relations from an ASCII file prepared using the OOA tool.

The functionality still to be implemented is as follows. State- and transition-classes will be generalised from the STN, with transition-classes being represented as AI planning operators. (This functionality has been proven in another application.) Selected plans will be generalised as procedures. Schedules, timelines, and resource profiles will be obtained from selected plans by allocating external resources, as required by the parent state- and transition-classes. The schedules or timelines will be ranked according to resource usage. Execution of the plans or re-instantiated operators and procedures will be modelled on the payload simulator. If the simulator is set to an invalid state, ASS will automatically detect this, and recommend a recovery plan (if one exists). If a recovery plan does not exist, this would indicate that one or more of the domain constraints had been violated. In this case, ASS would identify the set of violated constraints, modify the payload simulator, re-induce the STN, extract the new recovery plan, and execute it.

3.2 Technical Issues

ASS makes extensive use of AI algorithms, set in an object-oriented framework. Expert systems, AI planning, Truth Maintenance Systems, and machine induction all have a role to play. Domain constraints are represented as production rules, and constraint violation will be detected by forward-chaining inference. STN induction is performed using a variant of the *version spaces and candidate elimination* algorithm (Ref 12), with post-processing to identify valid transitions. No-good sets are obtained from the domain constraints to use in candidate elimination.

A major issue in ASS development has been to counter the combinatorial explosion in size of the version space during STN induction. Version space construction could be guided by heuristics, but only by sacrificing domain-independence and the guarantee of completeness. Object-oriented concepts have been more effective. The class-subclass relation, with inheritance, has been applied to entities to reduce the number of relation-classes needed to represent a given design. The class-instance relation has been exploited in partitioning entities, relations, constraints, states and transitions. Message-passing will be used to implement the payload simulation. The whole-part relation has a potential role to play in representing a payload as an assembly of sub-systems, but has not yet been incorporated in ASS. Despite these measures, it is recognised that STN generation remains NP-hard. There will still be some payload designs which are inherently so under-constrained that manual or heuristic methods will have to be used.

4. TEST RESULTS

Several toy domains are being used for development purposes, including the Dining Philosophers problem, a domain consisting of tanks and reactor-vessels (modelling fuel cells and refineries), and many versions of the blocks world. These domains vary from 2 to 6 entity-classes. Testing

with the HPCE domain has begun. The OOA tool has been used to construct a model consisting of 27 entity-classes and 52 relation-classes. (The same model was used in constructing the HPCE Payload Simulator (Ref 6)). This model has been loaded into ASS and a detailed design document generated. Two hours user-interaction was needed to determine the inter-relation constraints. Translation of the 1-to-1, 1-to-many, many-to-1, and many-to-many constraints captured by the OOA tool into ASS production rules would reduce this time substantially; this is being implemented. To date, only STNs for subsets of the 27 entity-classes have been induced. An algorithm for merging these "sub-STNs" into a complete STN is being designed. Hand simulation suggests that 20+ planning operators will result.

The existing procedure-oriented knowledge of operating the Beckman instrument is limited to a single procedure: the calibration of a newly-fitted capillary. The only existing "plan" is an abstract one shown on the instrument's front panel: RINSE, INJECT sample, SEPARATE sample constituents, and RINSE again. The instrument's manual states that empirical methods must be used to design experiments. Hence, any additional plans or procedures generated by ASS will be totally novel. We anticipate that the generated plans, procedures and planning operators will fall into three groups: instrument assembly (as performed in manufacture), experiment preparation, and experiment execution.

5. CONCLUSIONS

This paper has described the development of a prototype software tool which is designed to integrate spacecraft payload design, experiment planning, and experiment control. Traditionally, these have been separate areas of activity.

6. REFERENCES

1. Eckhard, F. 1992. High Performance Capillary Electrophoresis in the Microgravity Environment, *Advanced Space Research*, 12, 5: 247-255.

2. Visser, F.B. 1992. Dutch User Support Organisation Concept Description and User Requirements Document, *NLR CR 92097 L*, final issue, 16 Mar 92, Nationaal Lucht- en Ruimtevaartlaboratorium, Amsterdam, The Netherlands.

3. Pronk, C. N. A., N. Koopman, and D. de Hoop. 1992. Development Concept for Dutch User Support, Paper IAF-92-0711, *Proc 43rd Congress, International Astronautical Federation*, 28 August to 5 September 1992, Washington D.C., USA.

4. van Eenennaam, J. 1991. Capillary Electrophoresis in Space (CEIS): Payload Simulation, *HPCE Case Study Technical Note 1*, Issue 1, Comprimo Consulting Services b.v., NIVR Contract 2001CO, November 1991.

5. Beckman Instruments Inc. 1989. *P/ACE System 2000 Instrument Manual*. Beckman Instruments Nederland bv, Nijverheidsweg 21, 3640 AA Mijdrecht, The Netherlands.

6. Grant, T. J., M. H. Wigmans, F. Eckhard, and J. van Eenennaam. 1992. A Re-Useable Simulation for Space Payloads, *Proc Europeen Simulation Multiconference*, 1-3 June 1992, York, UK, pps 30 to 34.

7. Dorf, R.C. 1989. *Modern Control Systems*, Addison-Wesley, Reading, Mass., USA.

8. Tate, A., J. Hendler, and M. Drummond. 1990. A Review of AI Planning Techniques. In *Readings in Planning*, Allen, J., J. Hendler, and A. Tate, (eds), Morgan Kaufman, San Mateo, California, USA, 26-49.

9. Marca, and McGowen. 1988. *Systems Analysis and Design Technique*, McGraw-Hill, USA.

10. Chen, P. P. 1976. The Entity-Relationship Model: Towards a unified view of data, *ACM TODS*, 1, 9-36.

11. Coad, P., and E. Yourdon. 1991. *Object-Oriented Analysis*, 2nd ed, Yourdon Press Computing Series, Prentice Hall, Englewood Cliffs, New Jersey, USA.

12. Mitchell, T. M. 1982. Generalisation as Search, *Artificial Intelligence Journal*, 18, 2: 203-226.