AI Planning for Just-In-Time Training in Space: ESA's Integrated Learning System

T. J. Grant, M. Verhoef, L. P. Gale

Origin Netherlands b.v.
P. O. Box 1444
NL-3430 BK Nieuwegein
The Netherlands
{Tim.Grant; Marcel.Verhoef; Leslie.Gale} @nl.origin-it.com
http://www.origin-it.com

Abstract

Advanced planning and scheduling technologies have been deployed largely in operational space applications to date. Taking the European Space Agency's Integrated Learning System[1] as an example, this paper demonstrates that operational support functions - most notably training of crew and ground support personnel - can also benefit from the application of such technologies. The Integrated Learning System combines AI planning and Web pages to deliver personalized training at geographically-distributed locations. AI planning is an enabling technology for Just-In-Time Training, and delivering Web pages over the Internet/Intranet is an enabling technology for On-the-Job Training. Designed to be domain- and platform-independent, the system has been implemented and applied to a full-size domain in support of the European Space Software Development Environment. By replacing the existing training material, the system could be readily applied to other domains. We propose training of the International Space Station crew and ground support personnel as an obvious future application.

Introduction

Advanced planning and scheduling technologies have been deployed to date largely in operational space applications, as shown by this workshop's Call for Participation. By contrast, this paper demonstrates that operational support functions, such as training, can also benefit from the application of advanced planning and scheduling technologies. In particular, we show that AI planning is an enabling technology for delivering Just-In-Time-Training (JITT) in-orbit and on-ground. The Integrated Learning System (ILS), implemented for the European Space Agency (ESA), is used as an illustrative example.

The ILS delivers training material at geographically-distributed locations via the Internet or Intranets. AI planning technology is employed to generate personalized training courses in real-time, making JITT feasible. When the training material is in the form of Web pages, the training courses can be delivered over the Internet or an Intranet for execution at the trainee's normal workplace using a standard Web browser, thus enabling On-the-Job Training (OJT).

Designed to be domain- and platform-independent, the ILS has been implemented fully. It has been applied to a full-size domain in support of the European Space Software Development Environment (ESSDE), namely the development of on-board software in Ada using ESA's Hierarchical Object-Oriented Design (HOOD) method and the LOTOS and RAISE formal methods. The ILS implementation, complete with this ESSDE training material, was delivered to ESA and accepted in March 1997.

The ILS could be readily applied to other domains by replacing the ESSDE training material with training material for another domain. In this paper we propose training of crew and ground support personnel training for the International Space Station as an obvious future application of the ILS.

This paper focuses on the AI planning aspects of the ILS. There are six chapters. Chapter 2 outlines the ESSDE. Chapter 3 describes the ILS, including its software architecture and the representation of training knowledge as learning units and instruction sequences. Chapter 4 discusses the AI planning issues arising during ILS development. Chapter 5 identifies other future applications, with an emphasis on ISS crew and ground support personnel training. Chapter 6 draws conclusions.

European Space Software Development Environment

Ada is ESA's preferred programming language for on-board software for the European element of the International Space Station: the Columbus Orbital Facility (COF). The Columbus Software Development Standards

[1] The views expressed in this paper are those of the authors and do not necessarily represent those of ESA.

(ESA, 1993) stipulate that SADT[2], HOOD, Petri Net, and Extended Entity-Relationship methods must be used in on-board software design.

Stating a preference for Ada and HOOD is in itself not enough to ensure that Ada will be used by on-board software developers. They also require a matching software development environment. Recognizing this, ESA instituted the European Space Software Development Environment (ESSDE) programme with two projects:

- An Ada/SADT/HOOD tool was developed in the ESSDE Reference Facility Project, and
- Ada training was a key theme of the ESSDE Advanced Methods and Tools (AMT) project, together with an investigation of the incorporation of formal methods such as LOTOS (Language of Temporal Ordering Specification) and RAISE (Rigorous Approach to Industrial Software Engineering).

The ILS has its roots in the Ada training programme defined in the ESSDE AMT project. The objective of the Ada training programme was to create the necessary conditions for optimal use of Ada. A survey of Ada users disclosed the following obstacles to optimal use of the language (Gale, 1993):

- The number of users familiar with Ada, HOOD and formal methods is vanishingly small.
- Users were very often unfamiliar with the full range of constructs available in Ada.
- Familiarity with other high-level languages is an interference.
- There are few methodologies for transferring from design notations to Ada constructs.
- Training of users in Ada is often inadequately planned and applied ad-hoc.
- Continuing support is rarely available when required.

The solution to these obstacles was seen as the integration of a training system (now the ILS) into the software development environment, capable of providing (Gale and Mol, 1995):

- individualised and group training.
- trainee-, task-, and role-oriented training, i.e., responsive respectively to the trainee's skills, to the skills needed to perform the current task, and to the level at which the skills are to be performed.
- feedback and self-assessment.
- structured assistance in the management of training.

Integrated Learning System

Software architecture

The ILS has been implemented according to ESA's Software Engineering standards (ESA, 1991) using object-oriented design methods. Commercial Off-The-Shelf (COTS) development tools have been used, resulting in a low-cost, portable and extendible Computer-Based Training (CBT) solution. Runtime versions are available for SunOS 4.1.3, Solaris 2.5, and Windows NT/95. As delivered to ESA in March 1997, the ILS contained some 340 learning units covering software engineering in Ada, HOOD and LOTOS.

The ILS has become a generic CBT tool. Training can be provided on-the-job and just-in-time. Training sessions can be tailored to specific trainee needs depending on the trainee's existing skills and on the task he/she faces. Standardized training sessions can also be delivered to many trainees, either simultaneously or at times that suit the individual trainees.

The ILS is so named because it can be integrated into a larger system. In the ESSDE application, the ILS would be integrated into the ESSDE toolset. Integration is made possible by the ILS capturing data during the execution of training. The data includes time spent in each part of a training course, the number of failures made in each test, and the trainee's inputs. This data could be used to score the trainee's performance in known training courses, enabling successful trainees to be certified for mission-critical operations. Additionally, the data could be used on-line to bring in instructor assistance when a trainee needed it, or off-line to provide feedback to curriculum and courseware designers. Uniquely, the data enables an AI planning algorithm to modify the training course in real-time in response to the trainee's needs.

An ILS installation consists of the following elements:

- *Browsing and Tutoring Client.* Each trainee would have a COTS Web browser such as Netscape or Internet Explorer.
- *Browser and Tutoring Server.* A COTS Web server makes the training material available on the WWW or an Intranet to which the trainees have access.
- *Learning Unit Data Base (LUDB).* The LUDB contains the training material, represented as a collection of LUs. The training domain of the ILS installation is changed simply by changing the contents of the LUDB.
- *Authoring Support Environment (ASE).* The ASE is used to create the LUs, if necessary by conversion from existing documentation. A COTS HTML editor may be used.
- *Knowledge Data Base.* The knowledge data base contains the training courses, represented as instruction sequences. The curriculum can be changed without changing the training material in the LUDB by adding or removing courses in the Knowledge Data Base.
- *Planning and Evaluation Environment (PEE).* The PEE provides the facilities for defining instruction sequences and for evaluating the results of training sessions. It has been implemented in C with TCL/TK for the GUI and runs on X-Windows under SunOS and Solaris. At present, the PEE supports manual construction of training courses. Automated construction using an AI planning algorithm has been

[2] Also known as IDEF0.

prototyped, but - because of funding restrictions - has not yet been integrated into the PEE.

Selected ILS-related project documents are available on the European Space Technology and Research Centre (ESTEC) Web site at[3]:

http://www.estec.esa.nl/wmwww/WME/CBTraining/atp/index.html

ILS functionality

The ILS provides functionality to support two types of end-user:

- *Courseware designer.* The courseware designer would create and edit training knowledge using the ASE, and define curricula and individual courses using the PEE. In addition, the courseware designer would evaluate the results of training sessions also using the PEE.
- *Trainee.* Trainees would follow guided training courses using their Web browser and the ILS's Knowledge Data Base and LUDB. For refresher training, they would perform structured browsing using their Web browser and the LUDB. Given the appropriate permissions, they could also perform unstructured browsing of the training material outside the ILS.

Additional end-user roles could be identified. For example, the instructor role might be distinguished from that of the courseware designer, e.g. by making test result evaluation an instructor task. Considerations of practicality will also lead to the identification of maintenance roles, such as data base administrator.

The ILS retains the useful functionality of Web-based training systems:

- Training material can be browsed at a distance from where it is stored.
- Conceptual links between items of training material can be implemented statically as hyperlinks.
- Externally-provided material can be incorporated in training courses.
- Access to the training material can be restricted using Intranet or Extranet techniques.
- Search engines can provide users with the facility to search large quantities of training material.

The advantages of the ILS over conventional Web-based training systems are as follows:

- The training material is structured by means of learning units, the role-task model, and intellectual skill and training base classes. This indexing structure can be used to guide browsing.
- Training material can be sequenced into training courses for guided training.
- A given item of training material (i.e. a learning unit) can be part of multiple training courses, minimising duplication and inconsistencies between training courses.
- The trainee's understanding of the training material can be tested, with tests being embedded into training courses.
- Test results and other data (e.g. dwell times in individual learning units) can be captured and made available for other systems. For example, the test results of individual trainees can be used for certification, and captured data from multiple trainees can be used to evaluate the effectiveness of the training material and courses.

Possible extra functionalities that could be implemented in the ILS include:

- Sending test results by email to an instructor, enabling the instructor to assist trainees that are having difficulties with a particular part of a training course.
- Synchronising the execution of a training course for trainee and instructor, enabling them to work on the course together at a distance.
- Coupling the execution of training courses with simulations, emulations, and operational applications.
- Automating the construction of training courses using AI planning techniques. This would enable JITT. The feasibility of automated course construction has been demonstrated.
- The ILS could be embedded into larger systems. Within the ESSDE project, the ILS is seen as a facility to be embedded in the ESSDE toolkit. In particular, the ILS could be embedded into an operational user-performance system for OJT.

Learning Units

The LUDB contains the training knowledge for a given domain. There are two types of elementary knowledge:

- *Learning Element (LE).* An LE delivers a piece of training material designed to impart one or more skills.
- *Assessment Element (AE).* An AE tests whether or not the trainee has acquired a skill.

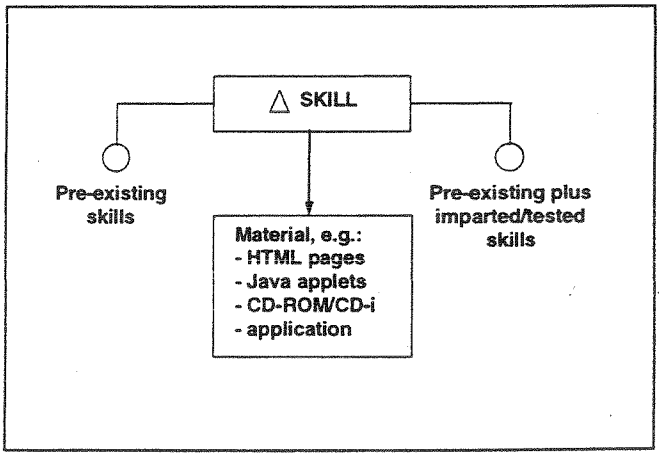

Figure 1. Schematic of Learning/Assessment Element.

The training material in LEs and the test material in AEs may be presented as web pages in HTML, Java or

[3] The correct upper and lower case is important.

Javascript code, (parts of) CD-ROM or CD-I presentations, (parts of) existing applications, etc. Figure 1 depicts an LE/AE schematically. Note that the LE or AE can be regarded as an indexing structure into the training/test material, which remains accessible (given the necessary permissions) for unguided browsing outside of the ILS.

A set of LEs and/or AEs related to one another is known as a *Learning Unit* (LU). Each LU is a linear sequence of one or more LEs and/or AEs. Usually, LUs consist of one or more LEs terminated by an AE, but other patterns are possible. In particular, an LU containing only AEs is known as an Assessment Unit (AU), and may be used:

- at the start of a training course to confirm that the trainee's existing skill-set subsumes the set of skills needed to assimilate the training material; and/or
- at the end of a training course to confirm that the trainee has acquired the skills needed to perform the task for which the course was designed.

The courseware designer determines the training and test material to be encapsulated in the LEs and AEs. The training and test material may well be pre-existing, as was the case for much of the ESSDE material.

The courseware designer also determines the grouping of LEs and AEs into LUs. There are organizational advantages in partitioning the training domain so that there is a one-to-one relationship between training topics and LUs. Each LU can be assigned an organizational owner who is responsible for keeping the training material in the LU up-to-date. To avoid the duplication of training material relating to any given topic, the ILS is designed so that an LU can be contained in multiple courses.

In addition to the LE/AE indexing structure, there are additional means in ILS for structuring the LUs including:

- grouping LUs into categories according to the Promesse role-task-skill model.
- allocating LUs to one of four intellectual skill classes (discrimate, apply-concepts, apply-rules, problem-solving).
- assigning LUs to one of four training base classes (process, methods, technical, domain-specific).

Inside the ILS, this indexing structure provides guidance in browsing the LUDB to facilitate refresher training. Moreover, the indexing structure is the key to the manual and automated construction of training courses.

Instruction Sequences

Training courses are represented as wholly- or partially-ordered sequences of LUs. Such an *instruction sequence* consists of a directed acyclic graph of LUs (see Figure 2).

A trainee may already have the skill knowledge that would be imparted by some of the LUs in the full instruction sequence. It would be both demotivating and wasteful of time and energy to teach these topics again. An instruction sequence can be tailored to deliver only those LUs the trainee really needs. Figure 2 depicts a case in which the trainee already knows items a, b and c and whose current task does not require him/her to complete the sequence.

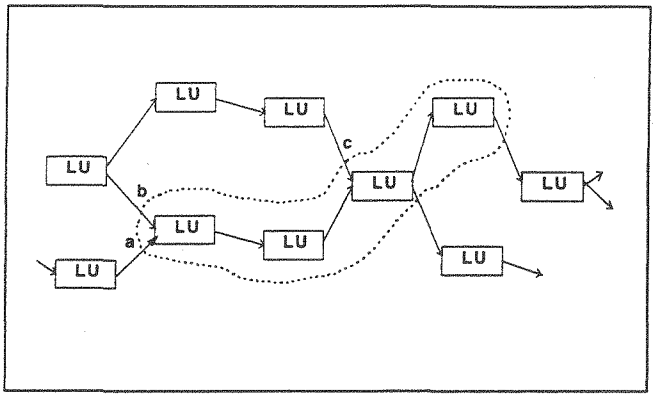

Figure 2. Instruction sequence as network of LUs.

Instruction sequences may be constructed either automatically using an AI planning algorithm or manually by an instructor or courseware designer. Automated construction has been demonstrated as a stand-alone application during Phase 2 of the ESSDE project. The currently-implemented PEE supports manual construction of instruction sequences by exploiting the knowledge in the indexing structure of LEs, AEs, and LUs. Mixed-initiative planning (i.e., the combination of automated and manual sequence construction) is under development in a separate, defence-related project; Grant (1997) describes a suitable graphic user-interface concept.

Comparison with other CBT frameworks

Other comparable CBT frameworks in European space applications include:

- *POINTER*. The Portable Intelligent Trainer for External Robotics (POINTER) uses expert systems techniques for domain knowledge and student modelling. POINTER has been applied to the European Robot Arm, to be flown on ISS. It is not Web-based and does not use AI planning techniques.
- *ASTRO*. The Advanced Software Tool for the tRaining of Operators (ASTRO) is a Web-based tool for distributing interactive multimedia training material (Wolff and Sanchez, 1996). ASTRO has been applied to the Tubular furnace with Integrated Thermal analysis Under Space conditions (TITUS). ASTRO does not use any AI techniques.
- *DMTS*. The Distributed Multimedia Training System enables instructors and students at geographically-separated locations to communicate in real-time by means of whiteboarding and video conferencing facilities. It is unclear whether DMTS uses Web-based technology; it does not use AI techniques.
- *ATIS*. Astronauts training: an intelligent system (ATIS) is a project to develop a knowledge-based system for supporting system and procedural self-

training of astronauts in the operation of payloads (Bertotti et al, 1995). The RAMSES payload, flown on the Spacelab IML-2 mission in 1992, serves as the illustrative application. Although ATIS uses expert systems techniques for domain and pedagogic knowledge and for student modelling, it does not appear to employ either AI planning or Web-based technology.

- *MOTE.* The Modular On-board Training Environment (MOTE) project is intended to prove that a CBT platform is well suited to changing environments, including off-nominal situations (Auferil and Bessone, 1997). Like ASTRO, MOTE uses the TITUS furnace as its illustrative application. MOTE is Web-based, but does not use any AI techniques.

The ILS is unique in combining Web and AI planning technologies for OJT and JITT in space applications.

AI Planning as Enabling Technology

In this chapter we justify the use of AI planning techniques as novel issues arise by comparison with operational applications. See (Grant, 1994) for technical details.

The first issue concerns the knowledge representation adopted. LEs and AEs are fundamental here; we consider LEs first. The purpose of LEs is to impart knowledge which the trainee can employ, i.e. to add to his/her repertoire of skills. The knowledge to be imparted is often known as the LE's *training objectives*. To achieve these objectives, the trainee must already possess certain other, more basic skills, often known as *training prerequisites*. Prerequisites and objectives can be expressed in terms of the trainee's skills. The prerequisites are the pre-conditions for delivering the training, and the post-conditions from delivering the training are the prerequisites plus the training objectives. This suggests that:

- LEs can be represented as planning operators, with pre- and post-conditions.
- the construction of instruction sequences can be represented as planning in a state-space expressed in terms of skills.

A second issue arises as to whether the instruction sequences are partially or fully ordered. We must consider execution to obtain an answer. Instruction sequences are intended to be delivered to trainees. Each trainee must undergo the complete instruction sequence. Moreover, each trainee can only do one thing at a time. Therefore, irrespective of whether the potential exists in an instruction sequence to deliver some LEs in parallel, the sequence must be linear when it is delivered. Hence, a linear plan-generation algorithm suffices.

Given that plan generation is state-based, uses operators with pre- and post-conditions, and outputs linear sequences, the STRIPS plan-generation algorithm (Fikes and Nilsson, 1971) is a reasonable choice. The classical planning assumptions embodied in STRIPS have been questioned, but, as McDermott (1994) points out, they are reasonable for many engineering applications.

The mapping from ILS to AI planning concepts is then as follows:

ILS concept	AI planning concept
Learning unit	Operator instance
Prerequisites	Operator pre-conditions
Training objectives	STRIPS-style add-list
Training prerequisites plus objectives	Post-conditions (a.k.a. effects)
Contents of LUDB	Operator-set
Trainee's existing skills	Current state (a.k.a. initial state)
Trainee's desired skills	Goal state
Instruction sequence	Plan
Instruction sequence construction	Plan generation

There remains a third issue. The imparting of knowledge by LEs would appear at first sight to be a monotonic process. Executed correctly, an LE adds skills to the trainee's store of knowledge and never removes them. In terms of the STRIPS representation, the LEs all have empty delete-lists. If so, AI planning techniques might seem to be an over-complication, with rule-based expert systems or decision tree techniques sufficing. Each LE could then be represented as a situation-action rule with the prerequisites as its antecedents and the training material as its consequents.

However, there are two factors which can rule out simple techniques:

- *Trainees can forget what they have previously learned.* Implicitly, there exist virtual operators with non-empty delete-lists additional to those representing the LEs. For obvious reasons, the operators of forgetting would not be included in instruction-sequence generation. To counter forgetting, an AU can be added at the start of the instruction sequence to check that the trainee has the necessary skills for assimilating the knowledge in the remainder of the instruction sequence. The AU would contain one AE for each skill in the prerequisites.
- *Trainees can fail to assimilate the training material.* Presenting an LE to the trainee does not guarantee that he/she will successfully assimilate the training material as skills. Although the LE's post-conditions would have been added to the modelled world state within the planner after the LE's execution, the same changes may not have occurred in the real world. To counter assimilation failure, an AU can be added at the end of the instruction sequence to confirm that the trainee has indeed successfully assimilated the training material in the instruction sequence. The AU would contain one AE for each skill in the training objectives.

In each case, the AU enables the training system to gain knowledge about the trainee's skills. In AI planning terms, the AU is needed to satisfy knowledge subgoals (Steel, 1991). This is easier to show for the case of countering assimilation failure. Suppose that a given LE imparts knowledge p (see Figure 3, in Steel's "tadpole" notation), then the AE that tests for p would have p in its pre-conditions and p V -p in its post-conditions[4]. The planning algorithm ensures that there is a *protection* for the condition p from the LE's post-conditions to the AE's pre-conditions, shown in the diagram as a horizontal line joining them. In the case of countering forgetting, the LE would have been (implicitly or explicitly) in some preceding instruction sequence.

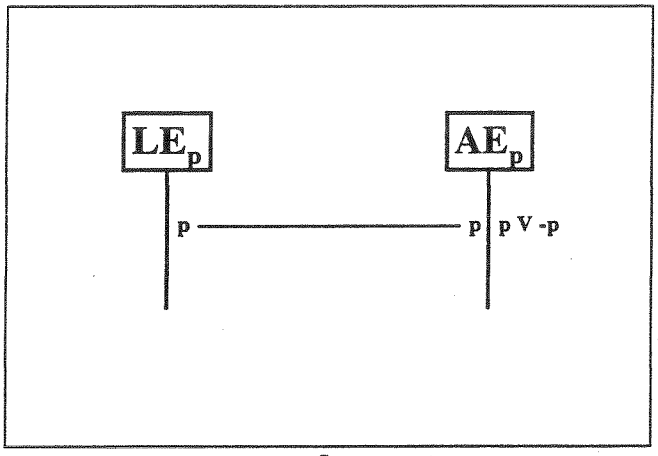

Figure 3. Dependency between LE and AE.

As the tadpole diagram shows, the world state after executing the AE may either contain the condition p or the condition -p. In AI planning terms, there are two possible worlds. The possible world containing -p is non-monotonic, because what was true (p) later becomes false (-p). Simple expert systems or decision-tree techniques are inadequate for constructing such instruction sequences, even though the non-monotonicity may only be a possibility. By contrast, a planning algorithm would insert into this possible world's instruction sequence another LE that re-asserts the condition p.

The exact state of the trainee's skills after executing the AE will only be known at execution-time. The crux of the problem is how to construct instruction sequences containing AEs. Possible solution approaches include:

- *Repeated replanning.* Construct the partial instruction sequence only up to the next AE. Execute this partial instruction sequence. Replan for the new initial state, and execute the new instruction sequence. Repeat until the training objectives are achieved. This approach depends on having a data-driven planning algorithm. However, most planning algorithms are goal-driven.
- *Optimistic planning.* Construct the instruction sequence assuming that all AEs will succeed. Execute the plan as long as the training objectives are confirmed. Handle negations in one of the following ways:
  - *Sudden death.* Terminate execution of the instruction sequence if the execution of any AE results in negation of a training objective. This approach has the disadvantage that the trainee's success is not guaranteed.
  - *Re-assertion.* Suspend execution of the instruction sequence, retrieve and execute a separate LE which will re-assert the negated training objective, and repeat the AE. This approach depends on providing the system with separate LEs for re-asserting all of the training objectives. The trainee's success within a finite time cannot guaranteed with this approach.
  - *Iterative planning.* Suspend execution of the instruction sequence, return the planner to the point in the plan just prior to an LE which asserts the negated training objective, and resume execution from that point. This approach depends on the segment of the instruction sequence preceding the AE containing an LE which asserts the negated training objective, as in Figure 3. Moreover, the planner must be *opportunistic*, in that it does not execute LEs for which the objectives have already been satisfied. The trainee's success within a finite time cannot guaranteed with this approach.
  - *Plan repair.* Suspend execution of the instruction sequence, repair the remainder of the instruction sequence for the new situation, and resume execution with the repaired instruction sequence. Plan repair is known to be a difficult problem.
- *Exhaustive conditional planning.* Construct the instruction sequences for all possible outcomes of the AEs. The instruction sequences can be merged into a single sequence with branch-points after each AE. At execution time, identify the correct branch to follow after executing each AE. This approach is impractical when there are many training objectives for the course, i.e., the instruction sequence has a high branching factor.

The ILS has been developed so as to support as many of these solution approaches as possible, with the courseware designer making the choice. Currently, the solution approaches must be executed manually, but could be automated by integrating planning into the ILS.

A further solution approach has also been implemented in the ILS. Within a given AE, the courseware designer can specify one of the following actions per attempt if the trainee fails:

- Repeat the test.
- Repeat the test, giving the trainee a hint.
- Note the failure, but continue the remainder of the instruction sequence as if the trainee had succeeded.

[4] The LE would be part of an LU, and the AE would be part of an AU.

For example, the test might be repeated on the first failed attempt, repeated with a hint on the second failed attempt, and the instruction sequence continued on the third failed attempt.

Application to ISS Training

Several authors have been considering ways of reducing International Space Station crew training time and increasing its effectiveness; see Muccio, Ockels and Gibson (1992), Griner, Lewis and Smith (1992), and Noneman (1992). Two options are relevant to this paper. The first option would be to increase training on scientific background and experiment objectives at the expense of procedural training. The second option would be to perform more training on-board.

Crew training on-board the ISS is a form of OJT. This could be implemented by incorporating training facilities in the crew terminals. Astronauts would then receive training using the same hardware and user interfaces that they use for operations. The third-generation Advanced Crew Terminal (Gale and Wolff, 1996), which is compliant with the ISS training hardware standards (ITCB, 1997), makes provision for a Simulation service for this reason. The ILS's learning units and instruction sequences can be structured by the courseware designer to be compliant with the ISS training software standards. Integration of a CBT framework with the Advanced Crew Terminal toolkit has been studied as a part of the POINTER-2 project (de Voogt, 1997). The ILS is one of the CBT frameworks under consideration, and is unique in combining AI planning and Web technologies (van der Arend, Kuiper and de Voogt, 1997). Moreover, the facility to integrate the ILS in a larger system can be used on the ground for the certification of crew members, planners, controllers, instructors, translators, Principal Investigators, and others who actively participate on console during a mission. This would be achieved by interfacing the ILS's LUDB and Knowledge Data Base to the European Astronaut Training Database and/or to the Training Administration and Management System.

The astronaut's mission will be several months long on the ISS. This means that the retention time expected of an astronaut since he/she was last trained on a payload on the ground is also of the order of months. Over this timescale, there is a substantial chance that payload planning will have changed. Even though the general rule on the ISS is that, if an astronaut has not trained for an operation on the ground then he/she does not perform the operation in-orbit, astronauts may still have to operate experiments for which his/her training is out-dated. This situation has already occurred on the MIR space station.

In particular, OJT aboard the ISS can reduce the retention time to a matter of days. Taking this trend further, the provision of JITT would minimize the time since the astronaut last trained on a payload. As its name suggests, JITT is intended to be delivered just prior to the moment when the knowledge gained is to be used. This has three benefits:

- The training effect is at its greatest, because the material is still fresh in the trainee's mind.
- Training is given only when it is needed, saving resources.
- The training material can be adapted to the precise circumstances in which the knowledge is to be used.

The ability of astronauts to react to unexpected situations is one of the key advantages of manned over unmanned spaceflight. On-board JITT can enhance these desirable capabilities of astronauts.

Each payload would be flown to the ISS together with its operating software, Virtual Control Panels, operating procedures, and training material (e.g. on a CD-ROM). The training material would be loaded on the crew terminals, as for the other payload-specific software.

The astronauts would consult the training material just prior to installing, commissioning, and using the payload. Although the crew activity plan would have to allow additional time for on-board training, the overall (i.e. space plus ground) training time would be reduced as a result of the heightened training effect. The heightened training effect would also increase the success rate in performing experiments, leading to an overall increase in the scientific utilization of the International Space Station.

Analogous arguments apply to the value of OJT and JITT for training ISS ground support personnel and users, as well as for crew training during Human Mars Exploration.

Conclusions

This paper has shown that operational support functions - most notably training of crew and ground support personnel - can benefit from advanced planning and scheduling technologies. The Integrated Learning System has been used as an illustrative example. The ILS is unique in that it combines AI planning techniques for constructing instruction sequences with Web-based training material. The techniques implemented in the ILS have been proven in the ESSDE application. Integration of the ILS with the third-generation Advanced Crew Terminal has been investigated so as to form a crew performance system with embedded OJT/JITT that is compliant with the ISS training standards for hardware and software. Embedding OJT in a crew performance system would reduce the retention time expected of astronauts from months to days. Adding JITT functionality, made possible by AI planning techniques, would minimise retention time and enhance the capability of astronauts to react to unexpected situations. The overall effect would be to increase the scientific utilization of the International Space Station.

References

Auferil, J., and Bessone, L. 1997. On-Board Training and Operational Support Tools Applying Web Technologies. *ESA Bulletin*, 89 (February).

Bertotti, L., Buciol, F., de Saint Vincent, A., Shafhouser, E., and Kass, J. 1995. ATIS - Astronauts training: an intelligent system.

de Voogt, E. J. 1997. Integration of CBT Framework and ACT. Netherlands Organisation for Applied Scientific Research (TNO), Physics and Electronics Laboratory, The Netherlands, version 0.3, 19 March 1997.

ESA. 1991. *European Space Agency Software Engineering standards*. ESA PSS-05-0, issue 2, 2 February 1991.

ESA 1993. *Columbus Software Development Standards*. EuroColumbus, document number STD 1213 800, issue 6, 25 June 1993.

Fikes, R. E., and Nilsson, N. J. 1971. STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving. *Artificial Intelligence Journal*, 2, pp. 189-208.

Gale, L. P. 1993. Development of an Ada Strategy for Space Industries. In Proceedings of the 4th Ada in Aerospace Conference, Brussels, 8 November 1993.

Gale, L. P., and Mol, F. 1995. ESSDE Advanced Methods and Technologies, Phase 3: User Requirements Document, Integrated Learning System. Document D/4000/1, version 1.0, BSO/Origin Nieuwegein bv, The Netherlands, 18 August 1995.

Gale, L. P., and Wolff, M. 1996. Advanced Crew Terminal: A generic framework for the realisation of system and payload crew terminals. In Guyenne, T.-D. (ed). Proceedings of the 1996 Spacecraft Operations (SpaceOps'96) conference, Munich, Germany. ESA document SP-394, Volume 2, Track 4, paper SO96.4.19, ISBN 92-9092-259-1, pp. 713-720. Also available at URL: http://www.op.dlr.de/SpaceOps/spops96/miscmpp/mm-4-19/4_19.pdf.

Grant, T. J. 1994. ESSDE Advanced Methods and Technologies, Phase 2: Knowledge-Based Planning of Ada Training. Document 1707094, version 1.0, BSO/Origin Nieuwegein bv, The Netherlands, 26 September 1994.

Grant, T. J. 1997. A GUI Concept for Mixed-Initiative Planning. In Fox, M. (ed). Proceedings of the 16th workshop of the UK Planning and Scheduling Special Interest Group, Durham University, UK. Forthcoming.

Griner, C. S., Lewis, C. M., and Smith, K. A. 1992. Payload Training for the Space Station Era. In Proceedings of the 43rd Congress of the International Astronautical Federation, paper IAF-92-0706.

ITCB. 1997. Multilateral Training Management Plan, Volume 1. International Training Control Board, International Space Station Program, 1 September 1997

McDermott, D. 1994. The Current State of AI Planning Research. Invited paper. In Proceedings of the International Conference on Industrial and Engineering Applications of AI and Expert Systems.

Muccio, J., Ockels, W. J., and Gibson, E. 1992. Training for International Space Station 'Freedom' - A New Perspective. *ESA Bulletin*, 68, pp. 85-92.

Noneman, S. R. 1992. Ground Tended Payload Operations of Space Station Freedom. In Proceedings of the 43rd Congress of the International Astronautical Federation, paper IAF-92-0714.

Steel, S. 1991. Knowledge Subgoals in Plans. In J. Hertzberg (ed.). Proceedings of the first European Workshop on Planning, March 1991, Sankt Augustin, Germany. Lecture Notes in Artificial Intelligence, Vol. 522, Springer-Verlag, Berlin, Germany, ISBN 3-540-54364-3, pp. 112-121.

Van der Arend, J. G. M., Kuiper, H., and de Voogt, E. J. 1997. POINTER-2: CBT Survey Document (WP 1200). Netherlands Organisation for Applied Scientific Research (TNO), Physics and Electronics Laboratory, The Netherlands, version 1.1, 21 March 1997.

Wolff, M., and Sanchez, M-L. 1996. ASTRO: Computer-based payload operations training. ESA, *Preparing for the Future*, vol. 6, no. 4.