Technical Expectations for a Full Scale Domain Engineering Demonstration Project

Lynn D. Stuckey, Jr. David C. Gross
Boeing Defense and Space Group

Greg D. Pryor
Naval Air Warfare Center Training Systems Division

ABSTRACT

STARS (Software Technology for Adaptable, Reliable Systems) is a long-term ARPA project aimed at advancing the management, quality, adaptability, and reliability of DoD software intensive systems Over the years, the STARS project has gradually focused on enabling a paradigm shift of DoD software practices to *megaprogramming.* The central megaprogramming concept is a process-driven, two-life-cycle approach to software development. One life-cycle spans the creation and enrichment of an organization's capabilities for a family of related products, or *domain engineering.* The other life-cycle spans the construction and delivery of individual products, or *instances* from the domain. This approach may provide substantial opportunity for *leveraged reuse*, that is, planned use of adapted software components in multiple products. Much of the effort to date has been for developing tools and processes that support megaprogramming. The STARS project is now in a transition and demonstration phase. One of the demonstration projects is in the domain of simulator-based training, specifically the U.S. Navy's domain of Air Vehicle Training Systems. If megaprogramming proves useful in this domain, it promises dramatic increases in productivity along with corresponding reductions in the cost of building simulations.

Previous experience reports have focused on pilot efforts in domain engineering for sub-domains of the Air Vehicle Training Systems (AVTS) domain such as environment and navigation simulation. These pilot efforts have demonstrated that the processes and tools are sufficiently mature for full scale domain engineering for AVTS -- which the demonstration project is proceeding to do. This paper summarizes the lessons learned from the pilot efforts and looks ahead to the technical challenges and opportunities we anticipate in the full scale demonstration. Expected technical challenges and opportunities include:

(a) Integrating many sub-domains,
(b) Relating to non-domain engineered models,
(c) Integrating dramatically larger staffs,
(d) Relating to a real product acquisition project,
(e) Controlling adaptation, and
(f) Leveraging extra-domain assets.

ABOUT THE AUTHORS

Lynn D. Stuckey, Jr. is a software systems engineer with the Missiles & Space Division of the Boeing Defense & Space Group. He has been responsible for software design, code, test, and integration on several Boeing simulation projects. He is currently involved in research and development activities dealing with software reuse in the domain of air vehicle training systems. Mr. Stuckey holds a Bachelor of Science degree in Electrical Engineering and a Master of Systems Engineering from the University of Alabama, in Huntsville. His thesis presents a systems engineering approach to software development. Mr. Stuckey is a doctoral student at the University of Central Florida. He can be reached as stuckey@plato.ds.boeing.com

David C. Gross is a software systems engineer with the Missiles & Space Division of the Boeing Defense & Space Group. He has worked in all life-cycle phases of simulation and training systems from requirements development through delivery. He is currently involved in applied research related to megaprogramming in the domain of training simulator systems. Mr. Gross holds a Bachelor of Science in Computer Science/Engineering from Auburn University and a Master of Operations Research at the University of Alabama at Huntsville. His thesis compares the utility of high level languages such as C++ and Ada for simulation. Mr. Gross is a doctoral student at the University of Central Florida. He can be reached as gross@plato.ds.boeing.com

Greg D. Pryor is a computer engineer with the Naval Air Warfare Center Training Systems Division (NAWCTSD). Prior to 1994, he worked at the Navy's Manned Flight Simulator. He was lead aerodynamics engineer for the MFS AH-1W Aircrew Procedures Trainer. His current assignments for NAWCTSD include the Navy/ARPA STARS demonstration project in the Air Vehicle Training Systems domain, and the AH-1W Cobra simulator. Mr. Pryor holds a Bachelor of Science in Aerospace Engineering from North Caroline State University, and a graduate of the United States Naval Test Pilots School. He can be reached as pryorg@ntsc.navy.mil

Technical Expectations for a Full Scale Domain Engineering Demonstration Project

Lynn D. Stuckey, Jr. David C. Gross
Boeing Defense and Space Group

Greg D. Pryor
Naval Air Warfare Center Training Systems Division

INTRODUCTION

The software community's experiences thus far in creating significant amounts of reusable software have seemed to be more wild goose chases than productive endeavors. Reuse projects to date seem to end up with endless lines of code that sit in a library and are never reused. The return on investment has rarely (if ever) justified the effort. Nevertheless, software reusability remains a hotly pursued goal for most organizations. The reasons are simple -- reusable software promises simultaneous progress toward the twin goals of any profit-oriented venture: (a) reduced cost of production, and (b) increased quality. The Software Technology for Adaptable, Reliable Systems (STARS) *project aims at reaching the reusability goal* through *megaprogramming.*

How is megaprogramming different than earlier efforts? Most reuse efforts have attempted to reuse existing software, either by adopting high quality software or re-engineering existing software. This is an intrinsically flawed approach. Many researchers have noted that reusability is not an accidental characteristic of software. "Reusability is first and foremost a *design issue. If a system is not designed with reusability* in mind, component interrelationships will be such that reusability cannot be obtained no matter how rigorously coding or documentation rules are followed." [Ausnit85] However, the recognition that creating reusable software is primarily a design issue is a far cry from defining the process for creating such software. And creating reusable software is only the first problem -- any consideration of the human nature of programmers will quickly lead to the conclusion that if the software is to be used, it must be easy to use. This implies some sort of automated, process-driven CASE tools. Finally, the most significant challenge is overcoming the cultural and business barriers to actually practicing reuse. Megaprogramming purports to address these problems.

BACKGROUND

STARS

In June 1983, the Department of Defense endorsed *the DoD Software Initiative which had been proposed* to offset growing manpower shortfalls and to improve capabilities for supporting software development and maintenance. This new initiative was comprised of three components. One component was the Ada Joint Program Office (AJPO). The second component was a Federally Funded Research and Development Center (FFRDC) later to be called the Software Engineering Institute (SEI). The third component was a project in software technology, which became STARS.

STARS is sponsored by the Advanced Research Projects Agency (ARPA), and involves three cooperating prime contractors -- Boeing, IBM, Paramax -- and a large number of subcontractors including DUAL, Incorporated and Enzian, Incorporated. Figure 1 illustrates the STARS Approach, based on megaprogramming. Megaprogramming is a product line approach to the creation and maintenance of software intensive systems. It is characterized by a product line view of the reuse of software life-cycle assets (architecture, components, and processes) and a disciplined, process-driven approach to development and evolution of application systems and the product line. The principle benefit to be derived from *megaprogramming is improved predictability and* quality in software and system development.

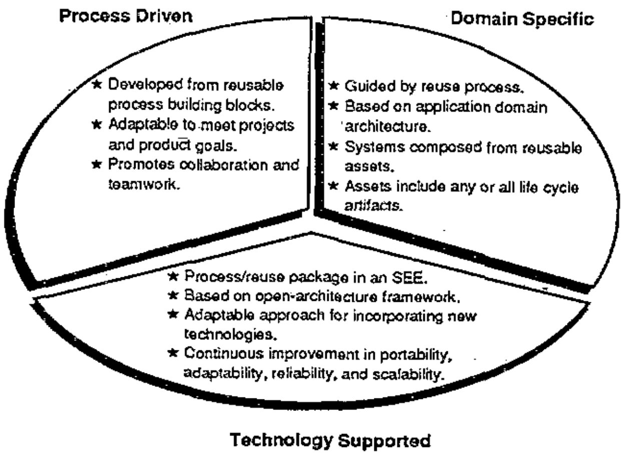
Process Driven
★ Developed from reusable process building blocks.
★ Adaptable to meet projects and product goals.
★ Promotes collaboration and teamwork.
Domain Specific
★ Guided by reuse process.
★ Based on application domain architecture.
★ Systems composed from reusable assets.
★ Assets include any or all life cycle artifacts.
★ Process/reuse package in an SEE.
★ Based on open-architecture framework.
★ Adaptable approach for incorporating new technologies.
★ Continuous improvement in portability, adaptability, reliability, and scalability.
Technology Supported

Figure 1: The STARS Approach

Synthesis Process

Engineering knowledge is a standardization of the decision process and its products. If this process captures the commonalities and variabilities between *specific systems within a product family, we can create* an opportunity to leverage reuse within the product family. In a sense, this is simply formally capturing the engineering knowledge about a family of systems, or domain. The Virginia Center of Excellence (VCOE) has *created a process called Synthesis for defining a* domain, specifying the products and adaptation for instances for the domain, and implementing designs that leverage whole and adapted components for reuse in new systems. [VCOE93] Figure 2 illustrates the Synthesis process.

Synthesis is a process developed for *leveraged* reuse. Leveraged reuse is one of five approaches to reusing software: (a) ad hoc, (b) opportunistic, (c) integrated, (d) leveraged, and (e) anticipated. Leveraged reuse assumes that a given product is actually a member of a product family, the members of which share some degree of commonality. Synthesis involves the definition, analysis, specification, and

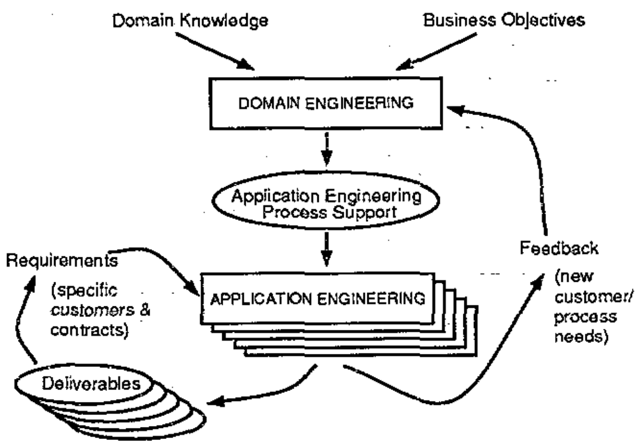

Figure 2: The Synthesis Process

implementation of a domain which encompasses a *viable* product family -- a family which shares sufficient commonality (and predictable variability) to justify an investment in the domain. Individual products are developed as instances of the domain, which reuse common elements of the domain and adapt variable elements using a defined, repeatable process. Synthesis defines the effort associated with creating the assets as *domain* engineering, and the effort in creating a specific product as *application* engineering. The central concept is that domain engineering creates the assets and processes by which application engineering can consider and select the best fitting instance of the domain.

Technology Support

Megaprogramming recognizes that the best reusable software in the world will not be used if it is not convenient to do so. Furthermore, the megaprogramming process encompasses advanced software engineering techniques, which lend themselves to the adoption of CASE tools. The STARS project has responded to these needs by incorporating technology support in the form of integrated Software Engineering Environments (SEE). STARS funding of process engineering and SEE development far in advance of the demonstration projects is evidence of its commitment to its three central concepts: (a) domain specific, (b) process driven, and (c) technology support.

The SEE utilized in this project was developed by the Boeing Company under ARPA funding. The Boeing SEE provides a set of functional services which are integrated in a standard framework, and can be driven by process definitions. Example functional services are those for requirement definition and code development. These functional services support activities in both life-cycles (domain and application engineering). The SEE includes typical computer environment services such as operating systems, editors, and language compilers. It goes beyond this to provide advanced CASE tools. All of these services are supported by commercial off-the-shelf products.

The Boeing SEE integrates the services via a process engine which enforces organizational policies and procedures. For example, the process engine uses defined work breakdown structures and activity precedence networks to schedule and enable software development activities. Naturally, these activities derive from the Synthesis process definition, so the SEE helps an organization "stay on course" by following its desired plan.

The Boeing SEE controls storage and access to adaptable components. Access to components is accomplished via a knowledge-based tool, which uses rules about the domain (identified by domain engineering) to guide selection and adaptation of components.

REVIEW OF PILOT EFFORTS

The STARS project is engaged in three demonstration projects (one per military service, each in cooperation with a prime contractor). The Navy's demonstration project is the only one within the field of simulation and modeling; selected because of its outstanding potential for systematic, long term product improvements. The fundamental concept in the demonstration project is a shift from viewing the construction of software intensive systems as "one at a time", to viewing a specific software system as an instance from a domain. In other words, a software system is really just one product out of a product line. The benefit of this approach is potentially a dramatic increase in the number and kind of reusable components. If their project is successful, the Navy intends to treat training simulators as a domain (product line) -- the Air Vehicle Training Systems (AVTS) domain.

The demonstration project has defined the scope of the AVTS domain, focused thus far on expanding the domain to address the needs of trainers for T-series aircraft. The T-series aircraft consist of the various types of training aircraft in current and future use by the Navy. These aircraft are used in the instruction of student pilots in the various aspects of naval aviation. This demonstration project will develop at least one instance from the domain, a T-34C Flight Instrument Trainer (FIT). If megaprogramming proves useful in this domain, it promises dramatic increases in productivity along with corresponding reductions in the cost of building simulations. This demonstration project has completed its pilot and preparatory efforts, and is entering full scale development. Progress thus far has raised certain expectations about the advantages and disadvantages of the megaprogramming approach for simulation. The demonstration project is organized into four phases as illustrated in Figure 3.

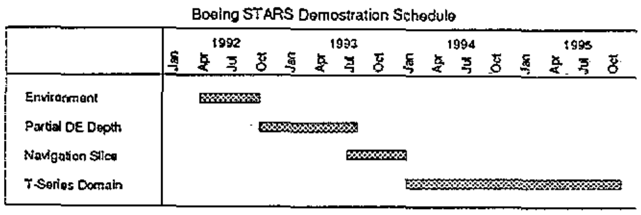

Figure 3: Demonstration Project Schedule

The demonstration project began in 1992 with the application of Synthesis to one aspect of the domain -- the Environment. This effort created a stand alone domain capable of supporting a few instances of the Environment. The next project phase involved application of the preliminary steps in Synthesis across the entire AVTS domain. This effort completed domain definitions, specifications, and preliminary design for the aspects most relevant to T-series aircraft. The third project phase finished the domain design and implementation for a slice of the Navigation sub-domain, and demonstrated the slice in the context of the AVTS architecture. The current phase of the project is to continue the domain engineering process in order to expand the AVTS domain to more completely support T-series aircraft, specifically a T-34C FIT instance. This phase will conclude with the demonstration of a functioning T-34C FIT in October of 1995. The following paragraphs briefly describe each of the pilot efforts along with some pertinent lessons learned.

Environment Segment Pilot

In 1992 Boeing Simulation & Training Systems (S&TS) performed a pilot project using Synthesis on a fragment of a training simulator -- the environmental model. Since the Synthesis process supports the STARS vision of reuse, ARPA wanted to evaluate Synthesis with a pilot project. The pilot project leveraged assets from the Modular Simulator (Mod Sim, or USAF designation HAVE MODULE) project. The Mod Sim project aimed at creating a generic, tailorable simulator. The central Mod Sim concept is a partitioning of the training simulator into twelve separate segments which encapsulate functionality of the system and relate through defined, controlled interface messages.

STARS determined that the systems engineering knowledge captured in the Mod Sim assets, and the flexible architecture that derived from Mod Sim, constituted a sufficient basis for megaprogramming in the AVTS domain.

The domain for the pilot project could have been as large as Boeing S&TS's entire product line, or as small as a single instrument on the control panel of a simulator. It was determined that one of the twelve Mod Sim segments was the proper size for the study in that it would be large enough to provide scope for significant testing of Synthesis, while being small enough to be doable in the time available.

The Environment domain was selected because it best met these criteria. Environment includes the tactical and natural environments external to the simulated vehicle. Since the original Mod Sim implementation did not include the Environment segment there could be assurance that the segment's design and implementation were not preconceived.

The main conclusion was that the Synthesis process is effective when applied to real-time vehicle simulators. This process supports the main tenets of the STARS reuse philosophy. The Mod Sim architecture is suited for adaptability and reusability and when used in conjunction with the Synthesis process most of the work in defining the domain is already completed. Having the interfaces between segments defined greatly reduces the time required to assemble the assumptions and therefore is the key to a successful domain definition the decision model. This pilot effort concluded that Synthesis could be scaled to encompass the entire AVTS Domain.

Full Domain Analysis Pilot

When signing of the Memorandum of Agreement (MOA) between Advanced Research Projects Agency (ARPA) and Naval Air Systems Command (NAVAIR) for the STARS demonstration project was delayed, the STARS program manager decided to front load the Domain Engineering effort using Boeing S&TS. This became a pilot program to conduct initial domain analysis across the entire AVTS domain.

The pilot program adopted the Mod Sim partitions for the AVTS domain of twelve sub-domains: Propulsion

(PRO), Flight Controls (FC), Flight Station (FS), Instructor/Operator Station (IOS), Physical Cues (PHC), Environment (ENV), Visual (VIS), Navigation and Communication (NAV), Flight Dynamics (FD), Radar (RDR), Weapons (WPN), and Electronic Warfare (EW). Each sub-domain represented an area of expertise. Eight of these sub-domains apply to the T-34C FIT: Propulsion, Flight Controls, Flight Station, Instructor/Operator Station, Physical Cues, Environment, Navigation/Communication, and Flight Dynamics.

When the work required to define and specify the AVTS *Domain was defined, a thirteenth sub-domain* was identified: the AVTS sub-domain (in reality a super-domain of the other twelve) which contains information about the performance and fidelity of the simulator and the inter-process communication and sub-domain executives required by the software architecture.

Synthesis assumes as an operational concept a single commercial organization owning a product line. The organization defines its business goals and objectives as part of domain management. NAWCTSD, in cooperation with NAVAIR, has served as project owner, assisted by a team of contractors. We elected to proceed with the Domain Analysis by assuming the T-34C FIT was the first product in the Domain Evolution Plan.

At the conclusion of this phase, we had developed a first iteration (refined as needed) of the entire Domain Definition for the eight sub-domains relevant to the T-34C, plus the AVTS sub-domain. The Domain Specification, Process Requirements, and Product Requirements were also completed. Product Design *was in draft form. These work products exceed the* stated goal of supplying Software Requirements Specification level requirements, but fall short of the complete design desired.

Navigation/Communication Sub-Domain Pilot

After the partial trial DE effort, the project committed to a final pilot effort. The purpose of this effort was to gather further lessons learned and to provide technology transfer to the DE team intended for the full scale demonstration project. The focus of this effort was on a slice of the NAV sub-domain, particularly the radio aids components of Tactical Air Navigation (TACAN) and VOR.

A central difficulty with megaprogramming is determining how viable a domain may be. Obviously, the megaprogramming approach may not provide an economic payback for all product lines. An example of the difficulty of evaluating domain viability arose in the sub-domain. Casual consideration of the viability of the Navigation domain makes one somewhat skeptical due to its large number of inherent variabilities. For instance, the mode and power logic for aircraft equipment simulations will vary widely. These variabilities produced an extremely intricate network of decisions required to describe the operational characteristics of aircraft equipment, for example TACANs and VORs. However, moving past these surface difficulties one finds a rich set of commonalities in the underlying computations required for TACAN and VOR. For instance, geographic equations are the same whether they are used for TACAN or VOR navigation problem solutions. One hint of the hidden commonalities was the similarity of the data flowing in and out of these models. The domain engineering work products produced thus far in the project will *continue to evolve, through additional iterations of* Synthesis, into a more robust system that better leverages the lower level commonalities.

Pilot Lessons Learned

Analysis of variability and commonality results in generic architectural components that would not be created had a single point approach been taken

Use of metalanguage within components to implement adaptability breaks down the barriers that a generic Ada component provides. Metalanguage-based adaptability provides almost unlimited flexibility.

Automation of decision models to adapt code can be complex and costly. Careful metrics must/will be collected to provide feasibility information.

Commonalities and variabilities associated with this experience only involved the T-34, T-44, and T-45 series aircraft. As other aircraft are considered, the domain complexity will significantly increase during each iteration through the Synthesis process. Defin*ing the domain (especially commonalities and variabil*ities) is time consuming, but with the right documentation and personnel experience, the process leads to many unexpected and helpful results.

The experience using the TACAN and VOR brought to light commonalities throughout AVTS. These include the computation of bearing, range, course deviation, and so forth. The variabilities are apparent when you consider the physical components and models. Switches, lights, and functionality differ between aircraft.

It is important that the domain decision model capture and group the choices between variations of domain instances according to some logical scheme. For example, instead of one large decision model called TACAN decision group, we decided to generate smaller decision groups, which consisted of logical groupings (e.g., Power, Self test, TACAN outputs, etc.).

We organized the decisions groups in a hierarchical structure to capture dependencies. This enabled the decision model to directly model the real world -- if the TACAN does not exist in this instance, then there is no need to exercise any of its related decision groups.

Synthesis requires identification of software components and their hierarchial relationship (product architecture) before beginning component design. This emphasis on careful partitioning of the software system is similar to that found in object-oriented design.

Synthesis tends to be life-cycle oriented rather than technique oriented. Therefore, traditional tools such as structural analysis, object-oriented analysis, program design language (PDL), pseudo-code, etc. were useful techniques for accomplishing Synthesis' goals.

TECHNICAL CHALLENGES

The project is now in a full scale demonstration phase. The objective of this phase is to complete sufficient domain engineering in a selected set of sub-domains, to an extent sufficient to support application engineering of a range of training simulators for T-series aircraft. The experiences derived from the pilot project have raised a series of concerns, or technical challenges that we anticipate will arise in the course of the full scale demonstration project.

Integrating Many Sub-Domains

One of the keys to our successful application of the DE process so far, has been our partitioning of the domain into related sub-domains. However, the pilot projects thus far have focused on only a small section of the domain or a partial DE cycle. Either focus faces fewer coordination problems than attempting full scale DE in many sub-domains. As we proceed, there is no reason to expect the DE process to proceed at the same rate in all of the sub-domains. One can envision DE progressing at different rate creating sub-domains that can not operate together. This problem has been called *domain integration*. Our response to this challenge has technical and management aspects.

Technical Aspects

Our technical approach within the domain contributes substantially to resolving this problem. First, the subdomains are *well-partitioned*. The pilot project based analysis (as well as the leveraged Mod Sim systems engineering legacy) suggests that only a very small fraction of subsystems may be reasonably assigned to multiple sub-domains. This fraction may be as low as 2 percent. Second, our technical approach depends on well-defined, controlled *interfaces*. The interfaces are defined in compilable Ada types. Each interface message is generated by a single subsystem (and as noted, a subsystem is associated with only one subdomain). Changes to these interfaces require agreement by the consuming sub-domains.

The third element of our technical approach that addresses the domain integration problem is that the sub-domains are highly *modular* and *localized*. The internal design of each sub-domain is required to conform to the Domain Architecture for Reusable Training Systems (DARTS), which specifies a hierarchial structure of modules (subsystems, components, and units) within segments. Figure 4 illustrates DARTS. Experience has shown that similar and related models are closely located. For example, the TACAN and VOR models are each encapsulated by DARTS components, and together (along with similar navigation equipment) in the Radio Aids subsystem. Strong localization means that functional changes tend to effect a small area of the system, rather than a broad effect.

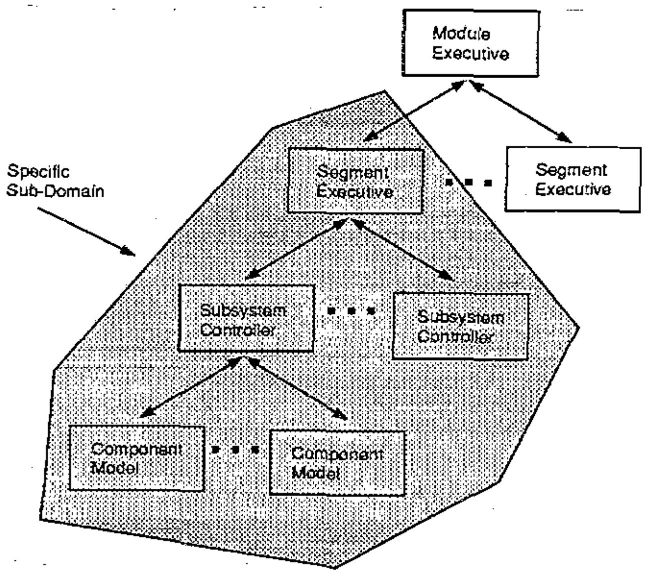

Figure 4: DARTS/Sub-Domain Relationship

The final element of our technical approach that addresses the domain integration problem is our adoption of *simulation strategies*, which guide the design and implementation of simulation models. These strategies capture the design decisions, which permeate throughout the simulator. The strategies create a common vision in every domain engineer's mind on how ubiquitous simulation problems will be addressed. Furthermore, they tend to reduce the number of nonessential variations to which the domain would otherwise have to adapt. The simulation strategies identified so far include: notification of electrical power, notification of malfunctions, assessing ownship damage, tracking real versus computed values, and tracking commanded versus current values.

Management Aspects

How can our project management approach help address the problem of domain integration? We expect there to be little sharing of subject matter experts between the different sub-domains, so cooperation between the sub-domains will be primarily a function of schedule and organization.

Our planning approach anticipates two distinct planning cycles: increments and iterations. *Increments* represent a management decision to invest sufficient resources to incorporate additional capability into the domain baseline. *Iterations* are complete or partial cycles through the Synthesis process. There may be many iterations within an increment. We do not plan to require sub-domains to proceed in concert for iterations -- but we do plan to require sub-domains to proceed in lock step through the increments. This is less structure than might first appear because the planned increments are fairly large and broad -- the current plan is one increment between now and October 1995. Figure 5 illustrates the increment/iteration relationship.

In contrast to schedule, we expect our organization to provide substantial assistance. Figure 6 illustrates the current DE organization. This figure indicates that the very heart of the DE organization is the AVTS Domain Integration Team. This team exists to control changes that effect multiple sub-domains, and

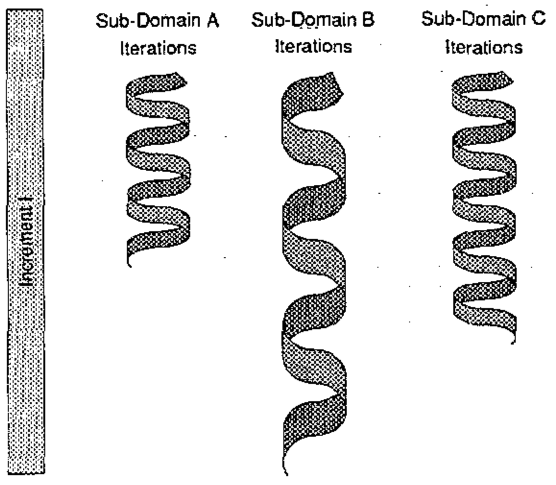

Figure 5: Increment/Iteration Relationship

resolve such issues for the benefit of the entire domain. As such, its members have to maintain a systems mindset -- never seeking to optimize one component at the expense of the system. Notice also, that the twelve distinct sub-domains are grouped into systems groups, again following the principle of localization. The systems group leaders are required to have experience across the sub-domains within their jurisdiction.

Relating to Non-Domain Engineered Models

The demonstration project is focusing on *leveraged* reuse, i.e., products and product fragments uniquely built to be reused via defined processes. However, it is not clear how well this approach will support the

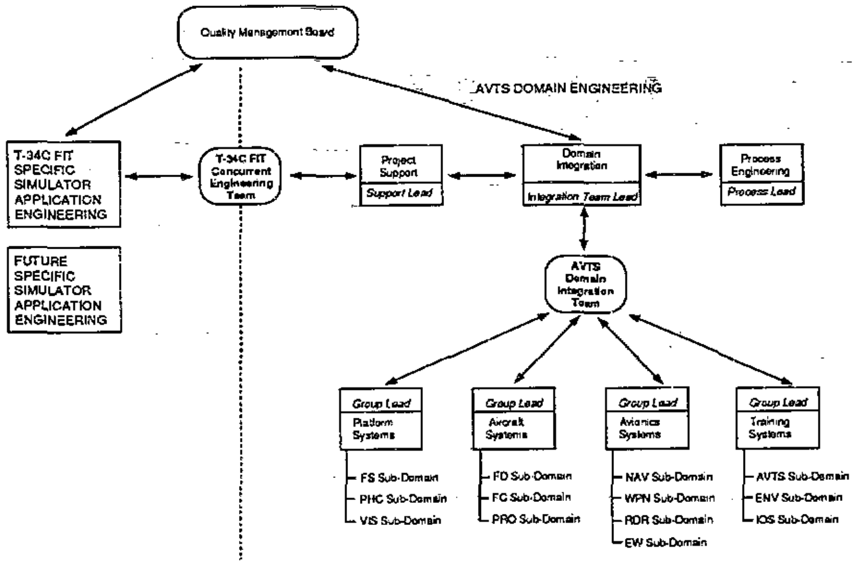

Figure 6: DE Organization

attempt to utilize *opportunistic* reuse, i.e., reuse based on the availability of existing assets. We should note that *re-engineering* of exists models does not in itself satisfy the AVTS requirement of providing *adaptable* models.

The central empowering concept in the domain is the insistence on a strong architecture. Only through such an architecture is leveraged reuse even possible, however, it does create specific limitations for opportunistic reuse; namely no component is usable which does not conform to the architectural constraints. If a component existed that conformed to the architecture, but was not developed through the DE process, it would be ideally suited for opportunistic reuse -- this however is very unlikely. Systems that impose a strong architecture for themselves are rare enough -- what is the chance that another system uses the derived architecture for AVTS?

Therefore, any component which is a candidate for opportunistic reuse will require some degree of re-engineering. In most cases, this may be relatively small. If an existing model cleanly encapsulates some functionality (say a single DARTS component), then the required re-engineering may be a simple as identifying and reconciling the interfaces. Once again, this is unlikely. There are as many ways to partition a simulator system as there are such systems, so even if a given model well encapsulates a problem within the context of its originating simulator, it probably crosses several partitioning lines in AVTS. The most likely case is that a single model will require partitioning to meet the AVTS partitions. In the worse case, a model may be based on assumptions that are dependent on data that in not available to an AVTS model. Such models would have to be re-designed.

None of these outcomes is satisfactory. Actual experience has shown that it is much better to use the existing assets as examples of what the leveraged sub-domains will have to provide rather than wholesale substitution. There is no substitute for actually performing the DE process, however, access to a rich collection of legacy assets can dramatically reduce the resources required and the risks incurred.

The most useful non-engineered models will be those that apply a strict separation between physics and controls -- a long standing tenet of modeling technique. Obviously the physical model is based on a very slow changing baseline (the earth) while the control model varies with each specific aircraft. Therefore, there is more opportunity for standardization and commonality with physical models. Models that cleanly divide between physics and controls tend also to have a clear interface, which again reduces the challenge of integrating them into the AVTS architecture.

Relating To A Real Product Acquisition Project

Megaprogramming makes no pretense of being a viable economic model unless *both* life-cycles are healthy. Most of this discussion has related to the challenge of domain engineering, but this is only the first part of the story. The products and processes of the domain have to be used by some actual application for there ever to be a payback from domain engineering. In fact, the envisioned situation would be several application project competing for instances from the domain. As discussed, the AVTS demonstration project has the T-34C FIT for its application project. While this simulator is very attractive from scalability and resource conservation aspects, it provides limited benefit in validating the domain engineering objectives for increment one: supporting a range of T-series aircraft with this first increment of domain engineering. Furthermore, the project lacks sufficient resources to achieve full scale domain engineering in all of the thirteen sub-domains in increment one.

The *transition of megaprogramming of applications* to the esoteric world of academics to the real world is a challenge that requires an incremental approach to the depth of domain engineering. The concept of full exploitation of all of the separate sub-domains at one time flies in the face of both synthesis and common sense. The Navy wishes to grow a product line of synthesis based simulators which would include the T-45, T-44, and the T-34C. The necessity is to rigorously model the domains in a manner that allows them to be expanded to include all of the vehicles in the product line. The process of concurrently synthesizing thirteen highly interactive sub-domains in a manner broad enough to cover three air vehicles would be extremely risky. The objective of first iteration is to approach the different sub-domains with varying level of domain development. The limit on utilization of the T-34C FIT as a prototype effort affords the DE & AE team a simplistic first iteration approach to synthesis.

The domain engineering team has decided to develop certain areas in a point solution method. For the point solution methods the sub-domain is concentrating on creating a model which is T-34C specific. Concurrent to the development of the point specific solution the sub-domain engineers are developing standard interfaces which will afford later growth. An example of this concurrent development is the sub-domain of propulsion. The T-34C engine is being specifically modeled and will have no variation in that model. However the variables, and structure of the model will allow the development of the domain in later increments in a manner which includes other gas turbine engines.

The *domain engineering* team is fully developing other sub-domains. These fully developed sub-domains will give the application engineer the ability to create new instances of the family if they has the correct data. One example of a fully developed sub-domain is flight dynamics. The full development of the domain will allow the application engineer to develop the correct flight dynamics model if they have the correct aerodynamics data. The growth of the sub-domain to include the other instances of the family line will require only the inclusion of the aerodynamics. The full development of other sub-domains is also requirement and data driven.

There are certain instances of the sub-domains which are not required for the product line of trainers. An example of these components are weapons. The T-34C, and T-44 do not employ weapons and for that reason the sub-domain of weapons is deferred to a later date.

The delineation of the methodology to synthesize domains was a critical step to gaining the capability of developing a synthesis based training system within the restraints of money and time. To assist us in the conversion of esoteric academics to robust reality we have physically separated the applications engineers from the domain engineers. While at first glance this may appear to separate the customer from the producer it appeared to us to be essential. It was apparent to us that if we had the AE & DE in close proximity to each other it would be easy for the AE team to drive the DE team to the T-34C solution and not to the product line solution.

Finally, we are committed to building a graphical representation of alternative cockpits in the product line. This *virtual cockpit* will give us the capability to exercise instances from the AVTS domain intended for platforms planned in the domain evolution plan, without incurring the cost and risk associated with use of actual simulator hardware. Furthermore, a virtual cockpit, graphically based, with a rapid prototype capability provides us a way to perform meaningful module tests and accomplish preliminary hardware/software integration.

Controlling Adaptation

Performing domain analysis is like riding a roller coaster -- there are highs when the domain seems perfectly suited to the process, and lows when it seems that the domain will never come together. It seems that the highs and lows are related to the depth of analysis in layers. Consider the following example.

At first glance, AVTS seems much too complicated to be a viable domain -- too many different kinds of aircraft and types of simulators. However, a little analysis reveals some important commonalities -- a simulator has some kind of flight dynamics, flight controls, and environment models, and instructor capabilities. However, further analysis (particularly in the avionics sub-domains) tends to re-agitate worries about the viability of the domain -- there are so many different equipment suites, much less the dozens of each kind of equipment (how many models of TACAN are there in military aircraft?). But once again, further analysis reveals the viability question -- yes, there are lots of different TACANs, but everyone of them requires essentially the same bearing calculation.

Another aspect of the problem is the reasonable decisions for an application engineer to make. For example, one can imagine an application engineer that can select the correct TACAN part number, but can they be expected know what the correct sweep rate is for a compass bearing pointer driven by a particular model TACAN is while searching for a station lock? While a TACAN designer may know such things, will they have the breadth of knowledge to make other decisions in the sub-domains, much less the AVTS domain at large. A decision model this complex would likely require the application engineering organization to import design level experts -- in which case, how does the domain show a return on investment?

The foregoing discussion illustrates that the specific level of detail captured in a decision model will play a large role in our sense of how viable a domain is -- and thereby what its likely return on investment will be. We have no satisfactory answers to this issue and plan to experiment with different solutions. Identification of a guiding principle for decision model complexity is a critical need for extending this approach to domains outside of the demonstration project.

Leveraging Extra-Domain Assets

A training simulator is (potentially) a very complex system. Upon casual reflection, one envisions the domain of training systems as a collection of various mathematical models interfacing with some kind of replicated crew interface controllable by some kind of operator/instructor device. While this is a reasonable simplified picture of the simulator, it does not begin to address the complete component set of the entire *system*. Figure 7 illustrates the broader view of a training simulator in which our domain exits.

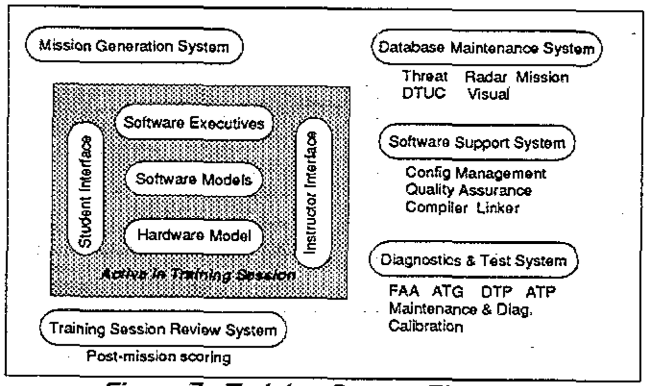

Figure 7: Training System Elements

While all of these system elements hold out good prospects for a reusable approach, the funding and scheduling constraints of the demonstration project would not permit examination of all possible elements. Therefore, we carefully considered what elements would constitute a minimal, yet complete system.

The operating hypothesis of the demonstration project is that while other system components such as mission generation, mission debriefs, etc., are significant resource consumers in training systems (and probably viable sub-domains), the appropriate focus for our work effort in the demonstration project are training system software elements active in a training session. There is no particular reason that those elements outside of the current AVTS domain boundaries could not be eventually included in the AVTS domain, or perhaps supported by a complementary domain.

Therefore, the demonstration project will have leveraged assets not created in the domain. For software assets, this does not present a serious problem. The SEE is fully capably of providing files that were not created in the course of domain engineering. The major problems will arise with acquisition of these assets, and interfacing to them. We expect that strong participation by NAWCTSD will open access to existing suitable assets. We plan to treat the interfaces to such assets as a routine variability of AVTS. Finally, the demonstration project has arranged for short run relaxation of some of the typical training system support requirements. For example, the increment one objectives include a sharply limited IOS requirement.

On the other hand, the non-domain hardware may present a substantially greater obstacle. Consider the simple example of the target computational system. The SEE exists on DEC engineering workstations using the VMS operating station. While some real-time applications have successfully utilized VMS (notably Manned Flight Simulator), it is unlikely to be a satisfactory answer for most applications in the domain. Applications will expect the SEE to provide cross-targeting capabilities, which are non-trivial to integrate.

A separate hardware problem is the stimulate/simulate debate. We plan to treat the decision to stimulate as a special case within the scope of the AVTS domain's variabilities. Of course, the various simulation approaches are routine aspects of the AVTS domain's variability.

CONCLUSION

The STARS goal is to increase software productivity, reliability, and quality via a paradigm shift; synergistically integrating computer-based support environments for modern software development processes and advanced software reuse concepts. We believe that the STARS approach may provide breakthroughs in the difficult problem of rapidly creating quality real-time simulator software. The STARS Program is entering the demonstration phase, with a Navy cosponsored effort to apply the STARS approach to the construction of a T-34C Flight Instrument Trainer.

However, these capabilities are not easy to deliver. Some of the difficulties in creating a domain of simulator software are: (a) the size and scope of the problem area (b) non-specific variabilities between users, and (c) the not-invented-here syndrome. The real world considerations of funding and schedule constraints, combined with constantly evolving requirements apply to the demonstration project, just as they do to more traditional projects. Never-the-less, experience deriving from the series of megaprogramming pilots has created high expectations for a technical success.

REFERENCES

[Ausnit85] Ausnit, Charles, et al. *Ada Reusability Guidelines.* April 1985. SoftTech Incorporated. Waltham, MA.

[Boehm92] Boehm, B. and B. Scherlis. "Megaprogramming", *Proceedings of the DARPA Software Technology Conference.* 1992. Meridien Corporation. Arlington, VA.

[NATOPST-34C88] Department of the Navy. 1988. *NATOPS Flight Manual Navy Model T-34C Aircraft.* NAVAIR 01-T34AAC-1. Naval Air Systems Command. Washington, D.C.

[VCOE93] Software Productivity Consortium Services Corporation. 1993. *Reuse-Driven Software Process Guidebook.* SPC-92019-CMC. Virginia Center of Excellence. Herndon, VA.