Planning for the Future of the IUE

Original Draft by:
J.L. Mundy
Chair: IUE Technical Committee

Revised by:
W.E.L. Grimson
Chair: IUE Advisory Board

Draft 2.0
September 14, 1996

1 Introduction

1.1 Purpose

Now that Version 1.0 of the IUE Core system is nearing completion, it is useful to identify a number of stages for the future evolution of the IUE. The purpose of this document is to identify the main issues associated with the IUE release and areas where future development is needed to enhance its success.

1.2 Overview

The goals of the IUE are:

1. provide a common software environment for image understanding research and development;
2. provide a standard format for data and computational structures to facilitate the exchange of datasets and algorithms;

3. provide a common demonstration platform to facilitate technology transfer;

4. provide a mechanism for system integration, both within single research sites, and across sites.

Of these goals, the last is perhaps of lesser importance than the first three, although it is still of relevance to considerations of the future health of the IUE.

The IUE was designed by the IUE Technical Committee and is being implemented by Amerinex AI. The IUE is an object-oriented system implemented in C++ which is partially generated directly from the LaTex IUE class specification. The implementation is based on the goal of using free, public domain software where possible.

The initial release of the IUE is targeted at the UNIX operating system, using LINUX to support PC hardware. It is planned to support the major UNIX hardware platforms, such as SUN, SiliconGraphics and HP. Currently, SUNOS, Solaris, and LINUX are supported by AAI.

The IUE is being developed using the gnu C++ compiler, gcc version 2.7.0 and libg++ version 2.7.0a. In addition to it being free, the decision to use gcc is based on its broad availability across platforms and its relatively good compliance with the current ANSI C++ standard.

The IUE is based on the C++ Standard Template Library (STL) which is a set of base classes, such as list and set. STL is now part of the C++ standard and provides a style of iteration which is used in IUE container classes.

The IUE provides an ASCII data exchange format called DEX (Data EXchange). DEX encodes C++ class attributes with a self-documenting format specification which is part of the DEX file. The IUE provides DEX readers and writers for each IUE class. A library of C++ support utilities for reading and writing DEX format is also available.

2 User Models

In order to plan for the dissemination and evolution of the IUE, it is important to identify the likely modes of utilization of the IUE and the requirements which must be met to effectively support these user modes. In particular, it is important to consider what additional elements of the IUE are necessary to gain wide usage and acceptance by the user community. It is the view of the Advisory Board that wide spread usage by a range of IU sites is essential to the future growth of the IUE.

A (non-exhaustive) list of important user models is as follows. The user models include both applications as well as focused research activities.

2.1 Image Display and Image Processing

This mode is similar to that provided by XV or by KHOROS. On the one hand, there is no strong motivation to provide yet another implementation of this mode of use. On the other hand, it is such a common mode that an IUE user would want such operations in conjunction with data browsing and educational uses. This mode should support chaining a series of image processing stages, including user-coded processes. Furthermore, there should be a relatively seamless interface between this component and the current structures of the IUE. There are many applications of IU work that require the ability on the part of the programmer to gain efficient access to primitive image elements (such as pixels) with minimum overhead. The lack of a seamless interface stands as a potential barrier to wide acceptance of the IUE within this important part of the user community.

2.2 2D Image Feature Extraction

This user mode introduces the use of the *spatial-object* which is the root of the central portion of the IUE. The spatial object hierarchy encodes the geometric and topological attributes and operations which are at the center of image understanding algorithms. It seems feasible to restrict the use to 2D geometry where research is focused on the extraction of image features and the development of feature grouping algorithms. It is very important, however, to provide a GUI which supports image display with 2D spatial object overlays. In conjunction with such a GUI, new feature extraction algorithms can be built on the standard IUE image feature classes, and then easily compared with standard segmentation operations, such as the Canny edge detector.

2.3 3D Scene Modeling

Here, the 3D portions of the hierarchy are added along with various sensor models which define the projection of 3D objects into 2D. The 3D hierarchy involves additional classes of surfaces and curves which do not exist in 2D. Geographic coordinate systems and transforms are required. The GUI is more extensive than for 2D where manipulation and construction of 3D objects must be supported. This mode is similar to that used in RADIUS by RCDE for the manual construction of 3D site models. The main use is to produce site models and experiment with IU algorithms for 3D object construction. It is necessary to support NITF 2.0 in order to effectively work on military applications in this area. It is also the case that the IUE should support Photogrammetry mechanisms for the required sensors. Although this capability is not part of the core IUE, it is important to add it if this user mode, and especially if the photogrammetry community and its associated applications users, are to be widely supported by the IUE.

2.4 Image Exploitation

This application is very broad, but again extensive use of most of the IUE hierarchy is expected to incorporate scene context as well as various levels of recognition and matching algorithms. The main goal is to define change of military significance in an archive or incoming stream of intelligence imagery. Again NITF 2.0 and sensor photogrammetry support is essential.

2.5 Automatic Target Recognition (ATR)

ATR applications, as currently practiced, are centered around interpretation of targets in an image as primarily signal events therefore the emphasis is on image processing. Statistically-derived matched filters are extensively used. Some model-based simulation is used to predict the appearance of objects, particularly in SAR and limited use has been made of model-based recognition. More advanced forms of ATR algorithms are beginning to use spatial context provided by site models and therefore will make good use of the geometric and geographic aspects of the IUE.

2.6 Model-based object recognition

Model-based object recognition, particularly the recognition of 3D objects from 2D intensity images is perhaps the most demanding user mode for the IUE. A tight integration of 2D image-features, sensors and 3D object models is required. Most of the IUE class hierarchy is required to support state-of-the-art research in object recognition. In this mode it is also expected that the IUE would import object models from other modeling systems, such as CAD modelers or existing target databases.

2.7 Physics-Based Vision

The goal here is accurate graphics rendering of 3D scenes, including color. The more advanced classes and methods associated with the class **Sensor** are required. The physical models of light scattering and the effects of optical systems are incorporated into a realistic simulation of image formation. Again, extensive use is made of the 3D spatial object class hierarchy and 3D coordinate systems and transforms.

2.8 DEX Browser

In this mode, the IUE is acting as a central data exchange format switch. This mode is patterned after the successful XV tool which has the goal of displaying and transforming a large number of image formats. In the DEX browser, this idea is extended to the exchange and display of 2D image features and 3D scene models. There is little processing done on the data items but a reasonable portion of IUE class hierarchy is required to import and construct various formats of 2D and 3D spatial data. There appear to be no cohesive formats for image features, such as edgels, so the IUE class format will serve as a target for users wishing to exchange such data. For 3D objects, it is necessary to import IGES and AutoCad's DXF as a minimum. Standard sensor model formats are just now emerging. The IUE follows the notation and semantics of the Photogrammetry handbook which is a good start. The DEX browser API should be consistent with HTML/Netscape interfaces so the browser can be used to support Internet data viewing.

We do not expect that all of these user models will immediately be supported by the IUE, nor do we believe that this is an exhaustive list of potential users. Nonetheless, we feel it is very important that likely user models be considered in developing plans for the near term and long term future of the IUE. If none of these user groups finds the IUE to be an important leveraging tool for their research and development, then

3 Programming Issues

Beyond the user application models just described, it is necessary to define issues related to the IUE as a programming environment per se. In the rapidly-evolving universe of programming languages, Internet operability and graphics support, it is crucial that the IUE provides an up-to-date interface to these mechanisms and at the same time maintains a stable class library for IU algorithms. Planning for the future of the IUE must also take these issues into account.

In this section issues related to the IUE as programming environment are discussed, particularly those issues which represent new extensions or new developments which affect the long-term evolution of the IUE.

3.1 Graphical User Interface

The recent explosion of Internet applications using new inter-operability mechanisms such as Netscape and Java now place requirements on the IUE which were not envisioned at the time the Spec was developed. Fortunately, the choice of FRESCO, an object-oriented GUI standard, for the IUE is likely to provide Java support and is also targeted to operate under the CORBA object broker standard. It remains to be seen if CORBA evolves to be a real object exchange standard. In any case, the original idea of the role of the GUI as a viewer for IU structures must now be extended to include Internet operability.

3.2 The C++ Language

It is clear that C++ has achieved dominance as the object-oriented language of choice for most new large object-oriented systems. At the same time, the language is still evolving including the relationship of the Standard Template

Library(STL) to the C++ standard. It is still not the case that all C++ compilers fully handle, at least efficiently, the requirements of STL. Nevertheless, it was a good choice to base the IUE design on STL and to build the implementation on an existing implementations of STL, even though there are still serious limitations and bugs – this is where C++ is headed.

The IUE is committed to the use of gcc, a public domain C++ compiler. For a time, this compiler was as good as any under UNIX, if not the best available, in terms of supporting advanced features of the language, including support for templates. Now, there is some question about the continuing evolution of gcc. It is still our best choice, since it allows operability over a large number of hardware platforms. It may be the case that we will have limit the use of some C++ features or provide patches for gcc as part of the IUE release.

There is also an issue in that many IU labs have not yet caught up with the commercial world in their use of C++. It is the case that there are many C programmers and new C implementations of IU algorithms being developed. At the same time most leaders of IU labs see that they should move to C++ and would appreciate a low-cost entry into C++ development. It is important that the IUE provide levels of access for both the experienced and new C++ programmer. Currently, it is necessary to be a fairly sophisticated programmer to work with the IUE.

3.3 IUE Partitioning

The IUE class hierarchy is large, and thus compiling and linking a full IUE is a lengthy process. Similarly, a full executable is also large. The use of shared libraries has reduced this problem considerably. However, for the IUE to be broadly accepted, it will be necessary to partition the class libraries so that programmers only need compile and link to reasonably related functionality. This issue was not a primary consideration in the design of the class hierarchy in the initial design, particularly in the choice of sharing pointers to object instances. By just passing pointers it is often possible to separate libraries which are initially interdependent.

As the IUE moves towards broader use by the IU community, however, this issue becomes much more important. Initiale attempts to use the IUE by a number of experienced programmers was met mostly by frustration with the apparently inherent slowness of compiling, loading and executing code.

Thus, as we work on the next version of the IUE core, this issue should be a central consideration. The desired result is to be able to support the various user modes described in section 2 with small library partitions and executables when the desired functionality is also limited.

3.4 Memory Management

The IUE implementation does not currently support the reference counting and memory management mechanisms necessary for clean pointer-based programming styles and avoidance of memory leaks. Additional effort must be applied to defining and adapting the IUE to use these mechanisms. This iteration is a natural evolution in the maturation of the design.

3.5 Image Processing

There has been some effort in 1995 to define a basic set of image processing operators which the IUE will support. Further work on the specification of these operators and the general design of how an operator interacts with the IUE image class is needed. This library is important to implement perhaps jointly with KHOROS, but resources have not permitted the attention this area deserves. There may be other avenues by which image processing mechanisms can be seamlessly integrated into the IUE, the key issue is that this needs to be addressed in the near future.

3.6 Hierarchy Evaluation

Many areas of the IUE use novel designs. The IUE Technical Committee provided these designs in order to give the IUE a rational and expressive hierarchy. When using new designs, however, it is crucial to evaluate the designs in use, and where necessary, revise them. Matricies, image features, and user extensibility in general, are three areas that could benefit from design review. Ideally such a review should occur after we have more experience using the existing hierarchy, but before the IUE is so widely used that interface changes become a significant burden.

3.7 Documentation

It has always been recognized that for the IUE to be broadly accepted it will be necessary to encode the practice of IU algorithm theory and evaluation in terms of the programming structures of the IUE. Right from the start, the documentation rigor of the IUE has been one of its strong points. Nonetheless, it is very important that the documentation be maintained in an up-to-date manner, especially given the size and complexity of the full system.

Carrying forward to a well-thought-out set of examples in the IUE primer, and perhaps the long-planned IUE textbook, is also a key priority.

3.8 Independent Evaluation

The IUE Summer Camp program has been successful in encouraging the use and evaluation of the IUE by outside organizations. Originally presented as a six week course for six students, Summer Camp will be repeated this year as a two week course for ten students, with a follow-up session two months later. The follow-up session is intended to encourage the students to continue working with the IUE at their home sites and to ensure that the IUE committee receives their feedback. This program provides for direct feedback over a significant period of time in a face-to-face environment. This feedback is much more thorough than that gained through typical e-mail forums. This valuable tool should be continued throughout the development of the IUE.

4 Evolution of the IUE

4.1 The ideal

How should the IUE evolve over the next five years? The original plan called for the IUE to be released and during 1996, at which point the IU community would work together to assemble IU algorithm libraries on top of the core.

This is exactly the plan formulated by the UK in their effort to adopt the IUE in Britain. A central hub, the University of Manchester, will take in code from various UK research centers and adapt it to the IUE environment and then redistribute it to the participating users. Funding is identified to support this activity, both at Manchester and at the "spokes".

There is no such organized plan, nor funding for such a plan in the US. In the current funding climate it does not seem likely that US IU researchers will have the resources to learn about and adopt the IUE in such a direct manner. Initial insertion will more likely happen at the research centers which are already participating in the design and development of the IUE, e.g. IUE committee members. For example, Terry Boult's lab and other groups at AAI are already developing in the IUE. For the long term health of the IUE, however, it is important that the IUE quickly reach a stage at which other user groups, without the vested interest and thus deep commitment to the IUE, can easily begin to use the IUE for development purposes. These means that the IUE development team must ascertain what constitutes a minimum threshold of acceptance, and then work to achieve that as quickly as possible.

4.2 Possible paths of development

There are at least two different models for how further IUE development can proceed. The first is to use applications to pull the future development of the IUE, the second is to perform more general tool building, and let that push the integration of the IUE into the broader IU community. The Advisory Board feels strongly that it is very important at this stage to the IUE to build strongs ties to the IU community beyond the Technical committee, and thus feels that it is probably best to pick a small number of applications areas, and build (perhaps together with the IU community) a base set of library routines needed to support these applications. Some examples of possible avenues are detailed below, but input from both DARPA and the Technical Committee on other possibilities should also be sought.

4.3 Co-Development With DARPA Programs

As noted, we believe that the IUE should be extended along lines which are appropriate to some set of applications. This model might be called the testbed model in analogy to the development of SRI's CME lisp-based environment, now called RCDE. Its development has been driven over the last few years primarily by requirements of RADIUS. Assuming that IUE funding and development will be structured around specific program requirements, two aspects can be identified:

1. enhancements and maintenance of the core IUE;
2. application algorithms and interoperability application system architectural elements.

Item 1 is probably best carried out by AAI, the original developer of the IUE. Some examples of type 1 work are given in section 3. To maintain stability and configuration of the code it is essential that AAI continue to play a key role in developing and maintaining the IUE core. AAI is also well positioned to adapt the IUE to new application areas, but it is essential that other application contractors be used as well to disseminate the IUE and broaden the community skills in its use.

For type 2 items it is assumed that funding will be earmarked from application programs to support the necessary developments and enhancements of the IUE to support the application requirements [1]. The IUE Technical Committee has begun investigating whether an initial application of the IUE within MSTAR would be profitable to both the IUE and to MSTAR. This needs to be pursued aggressively.

5 MSTAR

5.1 Overview

The MSTAR(Moving and Stationary Target Acquisition and Recognition) project is an extensive DARPA project to advance the state of the art in model-based ATR. The focus of the program is on SAR. The processing within MSTAR is divided into several stages:

- Modeling - Deriving suitable geometric models of ATR targets from a CAD model, selecting features such as corner reflectors.
- Focus of Attention - Using simple CFAR algorithms detect likely target indications.
- Indexing - Selecting an initial set of target hypotheses for each region of interest, prior to executing the PEMS loop (Predict, Extract, Match, Search).

[1] There is an existing model of this approach at GE CRD where a precursor to the IUE, TargetJr is maintained and developed in exactly this manner.

- Prediction - Predicting the appearance of a target in SAR by modeling the scattering of electro-magnetic radiation.
- Extraction - Extracting features from input SAR imagry.
- Matching - Finding a match between predicted features from stored models of targets and extracted features from input SAR imagry.
- Search - Narrowing the search for targets and managing recognition hypotheses.

During the first quarter of 1996, AAI and Ross Beveridge of the IUE TC investigated the MSTAR project with the goal of finding a good fit to applying the IUE. After some discussion it has been decided to work on the Matching function within MSTAR.

5.2 The IUE and MSTAR Matching

There are two contractors currently working on the Matching function: TASC/MIT, and NYU/ERIM. The TASC/MIT algorithm is already written in C++ while the NYU/ERIM algorithm is written in a LISP derivative, SETL. The MIT algorithm is based on maximizing mutual information and the NYU algorithm exploits geometric hashing.

The general idea is to integrate, both the MIT matching algorithm and the NYU algorithm into a common framework within the IUE. There are many common elements which can be shared and enhance the IUE class libraries and thus provide a unification of matching machinery. In this approach, the IUE development can be done in parallel and produce a more flexible and comprehensive matcher, perhaps combining the strengths of both.

Ideally, the implementation of the IUE matcher would be carried out in close collaboration with the MSTAR matcher developers. Bob Hummel of NYU is enthusiastic and has offered a student to work with AAI to define the geometric hashing algorithm in terms of generic IUE structures. MIT is also willing to pursue methods for integrating their algorithm into the IUE, and Gil Ettinger of TASC should be approached directly about this.

The rough timescale is that some initial implementation of these algorithms should be demonstrable by the end of 1996 to gain credibility for the IUE and to insure future evolution of the IUE implementation during the remainder of the MSTAR project. Thus, there is a clear need to push ahead in this direction as soon as possible.

6 The IUE and TargetJr

Another aspect of planning the evolution and development of the IUE is its relationship to TargetJr.

6.1 What is TargetJr?

TargetJr is a C++ software environment (500K lines of code) for carrying out Image Understanding research and applications. TargetJr's class libraries have been developed over the last seven years with emphasis on a geometric approach to image analysis. The system was initiated at GE's Corporate Research and Development (CRD) with initial emphasis on photo-interpretation and X-ray image analysis.

TargetJr has been used in research on object recognition based on geometric invariants under DARPA funding. TargetJr is also used in a project called PINPOINT which is developing a system for simulating weapon sensor-derived imagery. In this application, TargetJr plays a similar role as RCDE in RADIUS and provides site modeling and photogrammetry support.

TargetJr also is used in RADIUS as an environment for developing and integrating C and C++ IU algorithm implementations. To support this algorithm integration a small and lightweight blackboard architecture has been implemented which includes an in-memory relational database to support queries to the blackboard.

During 1996 the RADIUS applications developed in TargetJr are being applied to a internal R&D project with Lockheed Martin Corporation to apply MSE concepts to the retrieval of images from an image archive. The project will use event triggers to select relevant images for use in intelligence analysis.

The TargetJr libraries are divided into Packages, which are groups of class libraries with partitions to minimize cross-library dependency. Packages are available for the following general functions:

> **Image** - Libraries to support reading and writing various image formats, e.g., PGM, TIFF, MIT etc., as well as image processing.

Spatial Object - Representation of geometry and topology including digital curves which are derived from image segmentation.

Viewing - Support for graphical display of entities and menu driven interaction. TargetJr's graphics is based on the Interviews class library.

Photogrammetry - Models for various types of image projection such as perspective and graphics display cameras.

Segmentation - Classes and methods for extracting geometric descriptions from an image. Includes various Canny edge detectors and curve fitting algorithms.

Database - A simple relational database implementation for storing in-memory temporary data to be shared between classes. The database also supports a blackboard architecture for encapsulating large-grained processes which communicate via database relations and queries.

Morse - A suite of classes to support research in invariant object representation. Included is a recognition system, called Lewis, which represents planar shapes using algebraic invariants.

GeneralUtility - General programming support including:

> **DEX** - Several libraries to support ASCII import and export of TargetJr classes.
>
> **Solver** - Various numerical methods for solving systems of equations, including support for polynomials.
>
> **Stat** - Classes to support statistical analysis
>
> **Basics** - Really basic classes, e.g. UNIX directory parsing, reference counting, ..etc.

COOL - A templated container class library called C++ Object Oriented Library or COOL. COOL was originally developed the AI group at Texas Instruments in the late 1980's as a base for AI programming in C++.

6.2 What does TargetJr have to do with the IUE?

Currently TargetJr is being used by a consortium of vision researchers in object recognition, including: University of Virginia, Oxford University, University of Leuven(Belgium) and INRIA(France). These groups and GE-CRD as well plan to migrate this recognition system development to the IUE with the goal of incorporating TargetJr functionality into the IUE over the next several years. At the end of the transition period, TargetJr would be adapted completely within the IUE.

6.3 How will the transition be carried out?

During March 19-24 a meeting of this consortium was held on Martha's Vineyard. In attendance were 17 participants from the US in Europe. Representatives of AAI and the IUE TC were also present as well as a representative of the UK IUE consortium. The main focus of the meeting was to define a plan for the evolution of TargetJr towards the IUE. The plan which emerged is as follows:

> **Phase I** (3Q 1996) - Replace the current TargetJr image classes with the IUE image classes. Work jointly with AAI to maintain consistency.
>
> **Phase II** (1Q 1997) - Implement the FRESCO graphical user interface design to be used in the IUE and integrate and replace TargetJr's current Interviews GUI with IUE interface components.
>
> **Phase III** (4Q 1997) - Replace the SpatialObject library in TargetJr with the IUE SpatialObject library. Emulate COOL with IUE base classes.

Some additional considerations which emerged from the Martha's Vineyard discussions were:

Image Processing The University of Manchester has developed an extensive C++ library for 2D image processing and image geometry called

RADIAL[2]. As mentioned earlier, Manchester is also the hub for adapting vision research software in the UK to the IUE. One possibility is that the image processing library in RADIAL can be adapted to the IUE and to TargetJr as an initial phase of working with the IUE image classes.

Multiview 3D reconstruction There is currently a European project, called Vanguard, with the goal of advancing the state of 3D reconstruction from image sequences using recent developments in projective camera theory based on the tri-linear tensor. These algorithms are being developed in TargetJr and are to be integrated with SiliconGraphics' OpenInventor library. This application will be a good testbed for the new IUE/TargetJr GUI based on FRESCO and Mesa-Graphics.

7 Overall Plan

The Advisory Board reiterates that in their view the most pressing immediate need is for the IUE to pass the minimum threshold necessary to be come usable by portions of the community outside of those with a committed stake in the system, i.e., the Technical Committee and AAI. This, in our view, should be the overarching theme throughout considerations of detailed plans. This implies a need:

- to define a small number of target applications of interest to DARPA,
- to develop a basic set of library routines needed to support these applications,
- to ensure that system development work on the IUE infrastructure proceeds along lines that will enhance performance characteristics, so that the system is usable, i.e., reasonably fast processing, compilation, loading, etc.,
- to connect, through well designed interfaces, with other utilities that are needed, but not supported by the IUE, e.g., image processing routines, graphics, etc.

[2] Jeff Hunter, of AAI, was a key participant in the RADIAL project while he was at Manchester.

7.1 Overview

From these considerations, the proposed five-year plan for further evolution of the IUE (1996 - 2000) is as follows:
Year 1 - 1996 Complete and strengthen the IUE core. Library implementations by AAI, the IUE TC and the UK IUE consortium. Provide better library partitioning and compilation efficiency. Tighten the class hierarchy and methods in response to real-world algorithm development. Advance the GUI along lines useful for the IU program applications. Demonstrate an integrated model matching library and matching application to support MSTAR as proof of operation. Demonstrate interoperability with KHOROS in the MSTAR integration.

Year 2 - 1997 Develop and extend the IUE hierarchy to support the evolution of the MSTAR matcher. Work in collaboration with LMC Denver to integrate the IUE more extensively into MSTAR. Collect additions to the IU libraries from the UK consortium as well as the IUE/TargetJr transition project. By the end of 1997, the spatial object core of TargetJr will be replaced by the IUE spatial object classes, and TargetJr 3D libraries supplementing the IUE. It is also expected that in 1997, new BAA work will be initiated by some IU teams who propose using the IUE in their research. It is essential that the IUE be developed to the point where such groups can easily begin working with it. The IUE will be extended and revised in directions supportive of these development needs. Evaluate and redesign portions of the IUE hierarchy based on field experience. Monitor the developments of SEDRIS, VRML, STEP, JAVA and CORBA to insure IUE compatibility with these rapidly evolving interchange mechanisms. Look for opportunities to exploit the IUE SpatialObject hierarchy in data exchange and browsing applications – the XV of spatial data. LMC Valley Forge has successfully demonstrated import of CMU building models using DEX. Perhaps this role can be extended within SAIP.

Year 3 - 1998 In addition to operational demonstrations, the IUE is extended to provide support for context-based IU. The representation of context is going to place increasing demands on the semantics of the IUE which will go beyond the current emphasis on geometry. The IUE hierarchy should also be extended to incorporate reasoning and planning mechanisms. In any case, even in current developments for RADIUS at CRD it has proven useful to have a general blackboard architecture for supporting the integration and control of IU algorithms.

Year 4 - 1999 Complete the development of advanced representations in the IUE. Extend the dissemination of the IUE to more sites in terms of new IU contracts. Perhaps at this point funding could be defined to extend the IUE to distributed and networking applications centered around collaborative algorithm development among large numbers of contractors. This theme will become increasingly important as the US military collaborations become more international.

Year 5 - 2000 Transition to self-support. At this stage a number of major contractors will be using the IUE and it should be in place at many universities with web-based access and interoperability. The model here is perhaps the one exhibited by LINUX. After a critical mass of users was achieved, the further development of the environment has become a world-wide shared enterprise. It will require about $1-2M/year to fund AAI to maintain the IUE and incorporate enhancements to the core. This amount will still have to be provided by a combination of application contracts and direct financial support from users wanting rapid bug fixes and enhancements. This latter model is currently one mechanism which is funding maintenance support for gcc.

7.2 Detailed Plan for Year 1

Define the target applications A key set of applications much be determined, in conjunction with DARPA. The basic libraries needed to support these applications should be defined, implemented. tested and deployed. This will further serve to support the next topic.

Complete and strengthen IUE core In its current state of implementation the IUE core is ready to evaluate by implementing IU library algorithms. It is already apparent that additional effort is required to provide more efficiency and to streamline the interaction between classes. For example, the initial implementation of the graph class to support coordinate transforms is too complex and general. A simpler graph implementation is required, perhaps to support chain and star networks. As another example, Ross Beveridge, Chris Graves and Keith Price have asked for a more efficient image region representation, based on run length encoding, based on initial experience in coding image segmentation algorithms. Continue this evolution and hardening of the core by implementing the initial IUE algorithm library.

Improve code compilation and linking efficiency It is recognized that improvements in GCC are required to produce effective code for the IUE, particularly in the case of templated classes involving STL. Rapid progress is underway in reducing code bloat and run time computational speed. However, currently the libraries are too large and too interconnected to permit small applications to be built on the IUE without linking a large unwieldy executable. The IUE must be more finely partitioned and dependencies between libraries reduced. Such an evolution will likely require some class design and implementation changes.

Implement the IUE GUI Initial experience with FRESCO is promising and there seems to be considerable interest in FRESCO as a portable object-oriented structured graphics library. The plan is to integrate FRESCO with the IUE image classes to provide a 2D image and geometry GUI. The display of 3D will be provided by an overlay plane supporting a public domain version of open-gl called mesa-graphics. The plan is to share the development of FRESCO between GE-CRD and AAI and to produce a common base GUI between TargetJr and the IUE. TargetJr will migrate to use the IUE image classes.

Implement the MSTAR Matcher Library AAI will host a student of Bob Hummel this summer to design and begin the implementation of the MSTAR geometric hashing matcher algorithm. The goal is to realize the algorithm in terms of basic IUE classes and to extend the class hierarchy if necessary to support the needs of a generic interpretation of the algorithm. It is likely that this work will emphasize the SpatialIndex portion of the IUE hierarchy. In this area, Chris Brown has indicated he may be able to supply a student this summer to help implement the basics of SpatialIndex with the goal of implementing the HoughTransform class. Also work with MIT, Gil Ettinger and Eric Grimson to integrate the requirements of their matching algorithm.

Integration with KHOROS Work with KRI to define the data-level integration of the IUE with KHOROS. Use the requirements of the MSTAR Matcher algorithms to define an initial level of data exchange between the IUE-Matcher and other modules in MSTAR.

Integrate UK and European algorithms A likely opportunity here is to work closely with Manchester to integrate the image processing algorithms from RADIAL into the IUE. Manchester aims to do this anyway and a closer collaboration will insure successful re-use of these image processing libraries. A first task is to finalize the agreement on the design of image processing classes initially reached at the Martha's Vineyard meeting.

Complete IUE Contract Commitments Complete the other commitments for the final year of the initial IUE contract, e.g. documentation, new code releases and libraries.

8 1997 Funding

In order to ensure the maintainence of a critical team at AAI, and insuring Rick Lerner as project leader it will be necessary to insure a base-level funding of $500K during 1997. This funding would support evolution and maintenance of the core IUE and provide resources to fix bugs and follow the changes in compilers and C++ language features. The demands of library implemenatations will also require a basic level of support for enhancements by AAI. Additional resources at AAI would be be applied to the IUE in the context of IUE applications, for example further tasks in MSTAR or new projects funded by new IU BAA's.

9 The Future of the IUE Technical Committee

The initial purpose of the IUE Technical Committee (IUE TC) has been largely completed, namely the definition of basic classes and functions required for IU algorithm development. In the next phases of the evolution of the IUE, the focus should shift to a governing body which is concerned with the application of the IUE class hierarchy and the implementation of IU libraries. To reflect this change in emphasis the following new IUE Users Group Committee is proposed. The new membership would take place after the completion of existing DARPA-funded TC support contracts. Members would no longer be indivdually funded, but there still should be a small amount of annual committee funding, say $50K to support IUE UGC meetings and workshops. Membership would involve individuals working with the IUE in the context of DARPA-funded projects or other national funding. New members will be added as dissemination and use of the IUE expands.