The Eli system, a knowledge-based reusable software synthesis system, is being developed for NASA Langley under a Phase II SBIR contract. Named after Eli Whitney, the inventor of interchangeable parts, Eli assists engineers of large-scale software systems in reusing components while they are composing their software specifications or designs. Eli will identify reuse potential, search for components, select component variants, and synthesize components into the developer's specifications. The Eli project began as a Phase I SBIR to define a reusable software synthesis methodology that integrates reusability into the top-down development process and to develop an approach for an expert system to promote and accomplish reuse. The objectives of the Eli Phase II work are to integrate advanced technologies to automate the development of reusable components and the use of reusable components within the context of large system developments, to integrate with user development methodologies without significant changes in method or learning of special languages, and to make reuse the easiest operation to perform. Eli will try to address a number of reuse problems including developing software with reusable components, managing reusable components, identifying reusable components, and transitioning reuse technology. Eli is both a library facility for classifying, storing, and retrieving reusable components and a design environment that emphasizes, encourages, and supports reuse. Eli is being developed incrementally and will be released in a series of builds with progressively more functionality. A related issue, not being addressed by the Eli project, is how to implement reuse within an organization.
Outline of Presentation

- Eli Project Background
- Problems that Eli Will Solve
- Overview of Eli Build Plan
- Some Eli Operational Issues

Eli Project Background

- Phase I completed in Fall 1987, objectives were to:
  - Define reusable software synthesis methodology that integrates reusability into the top-down development process
  - Investigate formal languages for specifying reusable component interfaces, operations and requirements
  - Investigate knowledge and database representations for organizing and storing both components and knowledge of the application domain and development process
  - Develop approach for expert system to promote and accomplish reuse
Eli Project Background (Conc)

- Phase II started in July 1988; objectives are to:
  - Integrate advanced technologies to automate the development of reusable components and the use of reusable components within the context of large system developments
  - Integrate with user development methodologies without significant changes in method or learning of special languages
  - Make reuse the easiest operation to perform
Problems That Eli Will Solve

What Reuse Problems Must Eli Address?

- Developing software with reusable components
- Managing reusable components
- Identifying reusable components
- Transitioning reuse technology

What is Eli?

- Library facilities for classifying, storing and retrieving reusable components
- Design environment that emphasizes, encourages and supports reuse
User Roles

• Eli will support the following user roles:
  – Classifier
  – Searcher
  – Promoter
  – System Administrator

Key Qualities of Eli

• Adaptability
• Performance
• Ease of Use

• 'Make reuse the easiest operation to perform' • •
How Will Eli Solve Reuse Problems?

Identifying Reusable Components

- Flexible component classification facilities
- Flexible browsing and querying facilities

Managing Reusable Components

- Efficient storage and retrieval of large component inventories
- Open architecture to support integration with user environment
- Facilities for tracking and promoting reuse activities
Developing Software With Reusable Components

- Direct support for Ada components, including adaptation and integration
- Support for object-oriented design and programming
- Integration of design surface with library facilities

Transitioning Reuse Technology

- Support for defining new types of components, new component characteristics and new component relationships
- Loose and tight integration capabilities to transition existing tools and information
Overview of Eli Build Plan

Build Plan

<table>
<thead>
<tr>
<th>Build 1</th>
<th>Build 1.5</th>
<th>Build 2</th>
<th>Build 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prototype of query and browsing functions</td>
<td>Basic reuse library system</td>
<td>Complete reuse library system integrated with Ada compiler and providing basic support for Ada Component Adaptation</td>
<td>Basic design and programming environment integrated with reuse library browsing and querying functions, full support for Ada component adaptation and integration with compiler</td>
</tr>
<tr>
<td>Prototype of advanced classification and query strategies</td>
<td>Prototype of advanced adaptation function for Ada components and integration with Ada compiler</td>
<td>Prototype of advanced adaptation mechanisms and integration of browse/query functions with design surface</td>
<td>Prototype of knowledge-augmented, user-transparent reuse assistance</td>
</tr>
</tbody>
</table>

Build 1.5

• This build will provide basic library capabilities:
  – Creation and maintenance of libraries
  – Creation and maintenance of classification schemes for library components
  – Classification and storage of components
  – Browsing of libraries to find/identify components
  – Querying on libraries to find/identify components
  – Extraction of classification schemes, components and component information
  – Integration of component classification, storage, query and extraction functions through a program interface
Build 1.5/2

User Application Space

Components

External Storage Facility

Open Architecture

Components

(classification, certification, measurement tools)

CM system, development library, project database

Build 2

• This build will provide a complete, sophisticated library system:
  – Import/export of libraries and classification schemes
  – Enhanced manipulation of classification schemes and component classifications
  – Semi-automated derivation of Ada component characteristics
  – Classification support for Classic-Ada components
  – Clustering of components and support for "like this" querying
  – Enhanced and additional forms of interactive browsing and querying on component characteristics
  – Storage, retrieval and modification of query sessions, including batch submittal of queries and query sessions
Build 2 (Conc)

– Version control on libraries, classification schemes, components and component information
– Access control to libraries, classification schemes, components and component information
– Adaptation and integration of reusable Ada components with user application
– Collection and reporting on library and classification scheme usage, and component submittal and extraction
– Customization and tailoring capabilities

Build 3

This build will provide an object-oriented design surface with the following capabilities:

– Integration with Eli library facilities for design-time reuse assistance
– More automated derivation of component characteristics and classification of components
– Ordered assessments of components identified as result of queries
– Advanced support for Ada component adaptation and integration
Some Eli Operational Issues

User Roles

Classifier

Searcher

System Administrator

Reuse Promoter
Eli "Black Box" View

Eli Interface Requirements

<table>
<thead>
<tr>
<th>Interface Area</th>
<th>Principal Eli Focus</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Host Operating System</td>
<td>Transportability</td>
</tr>
<tr>
<td>• User's Development Environment Framework</td>
<td>Interoperability</td>
</tr>
<tr>
<td>• User's Development Environment Tools</td>
<td>Interoperability</td>
</tr>
<tr>
<td>• User's Development Environment Policies, Procedures and Methods</td>
<td>Adaptability</td>
</tr>
</tbody>
</table>
Eli Host Operating System Interfaces

Approach: Establish localized internal interfaces and utilize industry standards (e.g. Unix, XWindows, TCP/IP, Postscript) for transportability

- Device management
- Process Management
- File Management
- Communications

Eli Interfaces to User's Development Environment Framework

Approach: Support many levels of interaction including an open architecture -- procedural access to internal Eli facilities, published information schemas/structures, and an ASCII import/export interchange mechanism.

- Eli invocation
- Import of environment roles, access rights, procedures, etc.
- Configuration management of components
- Ada library manager
- Environment information management facilities
- Invocation of other environment tools/facilities
Eli Interfaces to User's Development Environment Tools

Approach: Provide open architecture -- procedural access and ASCII import/export facilities to allow users to exchange information with other tools

- Ada compilation system
- Documentation tools
- Other CASE (i.e. design surface) tools
- Other reuse systems (e.g. libraries, domain analysis tools)
- Project management tools

Eli Interfaces to User's Development Environment Policies, Procedures and Methods

Approach: Make Eli facilities adaptable to accommodate a wide spectrum of usage

- User roles and access rights
- Usage scenarios/sequences/work flows
- Configuration management procedures
- Component certification procedures
- Custom component attributes/facets
- Custom classification schemes
- Site/library installations
# Eli Distribution Options

<table>
<thead>
<tr>
<th>Non-distributed library model</th>
<th>Classification Update (Library Control)</th>
<th>Component Classification &amp; Storage</th>
<th>Library Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interaction of remote, separately controlled libraries (e.g., interlibrary loan)</td>
<td>Local</td>
<td>Local</td>
<td>Local only</td>
</tr>
<tr>
<td>Master/branch library (e.g., bookmobile)</td>
<td>Local to master library</td>
<td>Local to master library</td>
<td>Accessible across affiliated branches</td>
</tr>
<tr>
<td>Partitioned library (e.g., library system)</td>
<td>Single point or negotiated</td>
<td>Partitioned</td>
<td>Accessible across library sites</td>
</tr>
<tr>
<td>Cooperating, distributed libraries</td>
<td>Distributed</td>
<td>Distributed</td>
<td>All libraries accessible transparently from any site</td>
</tr>
</tbody>
</table>

## Library Interaction Through Design Surface

The diagram illustrates the interaction of library classification, update, and classification through a design surface. It shows how different components interact, with focus on features such as search scope, traverse hierarchy, and query sessions. The diagram is complex and includes various symbols and labels, indicating the flow and interaction between different library systems and their components.

**Set Search Scope**
- Search by window
- Search by scheme
- Search by component
- Search by library

**Traverse Hierarchy**
- Query object = language
- Query object = language

**Query Session**
- Component set 1: 22 finds
- Component set 2: 2 finds
- Component set 3: 2 finds

**Results**
- Library: CORE
- Language: German
- Component: Language ownership

**Component Set**
- Function = filter
- Object = languages

**Automated Measurement System Filter**
- Function = filter
- Object = language
- Component set 2: 2 finds

**ORIG!NAL PAGE IS OF POOR QUALITY**