International Workshop On Software Reuse
Brussels - 8 June 94

REBOOT (ESPRIT P7808)
REuse Based on Object Oriented Techniques

The REBOOT Approach to Software Reuse

Jean-Marc MOREL / BULL S.A.
Contents

- Introduction to Software Reuse
- The REBOOT Project
- Organizing Reuse
- Development FOR and WITH Reuse
- Classification
- Product and Process Metrics
- The REBOOT Environment
- The REBOOT Offer
Why not Reuse Software?

One of the biggest breakthroughs in the industrialization of society came when people started to put together standardized components.

People effortlessly combine thousands of predefined pieces to put together totally new objects.

Therefore one of our key jobs is to devise and promote standards and mechanisms to make standardized software componentry possible.

Graham Samuel  ESF Project Director, Sema Group UK, Berlin
Some Pioneering Software Projects

- **Common Ada Missile Packages (CAMP)**
  - Conducted by McDonnell Douglas under contract to the Dod STARS program.
  - Example of an early domain-specific software reuse library.

- **Magnavox's AFATDS project (1987)**
  - 700 KLS of Ada (100 KLS reused).  
  - OO techniques and incentives.

- **NASA Goddard Space Flight Center**
  - 32 percent of software reused or modified in 25 software systems for unmanned spacecraft control in Fortran.
  - Subsequent efforts at GSFC with Ada code indicate higher reuse averages.

- **Raytheon Missile Systems Division**
  - Experience in business applications: 60% reuse, productivity up by 50%.
  - Improvement in maintenance due to a consistent style for all software.
A mature reuse organisation

- Example: TASKON A/S Norway (office automation, tailored end-user system)
  - Strong management commitment and knowledge about software reuse.
  - Entire product development based on reuse philosophy.
  - Reuse penetrates the entire organisation.
  - Development process supports object-orientation and reuse.
  - Applications engineered from reuse repository of about 1500 Smalltalk classes.
  - Re-engineering of class hierarchies occurs regularly (“back-room” activity).
  - The company incrementally increases its reuse assets.

Their need: A reuse repository with support for classification to manage the ever growing number of components.
Classical Software Development Process
Reuse oriented Software Development Process

1. Identify / Import
2. Re-engineer & Qualify
3. Classify & Store
4. Browse / Manage
5. Query / Retrieve
6. Evaluate / Compare
7. Adapt

Requirements
Software Development Process
Deliverables

Reuse Repository

International Workshop On Software Reuse - Brussels - 8 June 94

φ Hall S.A., Cop Current Innovation, B2F-Frameworks, Q Labs, Sema-Group see, Siemens AG, Simief and TIX Ingenieria Informatica S.P.A., 1994

8 58
Reuse Advantages

- Reduced development time (time-to-market).
- Improved software quality (functionality, reliability, usability, efficiency).
- Reduce complexity (maintainability, portability).
- Increased productivity.
- Risk reduction.
- Spread and sharing of knowledge across organizations and projects.
- Facilitates education in good system architecture.
- Psychological reuse satisfaction: subjective measure.
Reuse Impediments

- Problems in locating suitable components.
- Problems in understanding such components.
- Lack of confidence in such components. (Not Invented Here).
- Problems in adapting such components.
- Problems with over-generalized components, inefficiency?
- Conversion cost to a “reuse situation” (tools, training, culture).
- Problems with licencing rights and responsibilities (wrt. errors).

==> The objectives of the REBOOT project (REuse Based on Object-Oriented Techniques) are to help overcome these problems.
The REBOOT Project

To enhance Productivity & Quality in Software Development by promoting and assisting Reuse
The REBOOT Consortium

SINTEF/NTH (Trondheim)

Bull S.A. (Paris - Coordinator)

Cap Gemini Innovation (Grenoble)
IMAG (Grenoble)
Sema Group sae (Madrid)
E-P Frameworks (Ronneby)
Q-Labs (Lund)
Siemens AG (Munich)
Txt Ingenieria Informatica (Milano)
The REBOOT Program: An Holistic Approach

- Define a Methodology
  - How to adapt the organization?
  - How to manage a reuse project?
  - How to develop reusable components?
  - How to adapt the development process to reuse?
  - How to manage a library of reusable components?
  - How to measure the cost and benefit?
  - How to introduce reuse in a software organization?

- Develop a Reuse Repository & Reuse tools

- Experiment with Industrial Applications.
  - Both to better understand the problems and assess proposed methods and tools.

- Transfer the Technology.
## Some Applications

<table>
<thead>
<tr>
<th>Partner</th>
<th>Application</th>
<th>Focus</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bull</td>
<td>Workflow</td>
<td>Development WITH reuse in the large</td>
</tr>
<tr>
<td>Cap</td>
<td>Resource allocation</td>
<td>Development of reusable components</td>
</tr>
<tr>
<td>Q-Labs</td>
<td>Telecommunication</td>
<td>Adaptation of existing methodology to reuse</td>
</tr>
<tr>
<td>E-P Frameworks</td>
<td>Real time inspection</td>
<td>Development FOR reuse</td>
</tr>
<tr>
<td></td>
<td>Fire alarm system</td>
<td>Development FOR reuse: OO framework</td>
</tr>
<tr>
<td></td>
<td>Gateway system for telecom.</td>
<td>Development FOR reuse: OO framework</td>
</tr>
<tr>
<td>Siemens</td>
<td>Material flow control</td>
<td>Development of reusable components</td>
</tr>
<tr>
<td></td>
<td>Telecommunication</td>
<td>Re-engineering for reuse</td>
</tr>
<tr>
<td>Txt</td>
<td>Factory planning</td>
<td>Development FOR &amp; WITH reuse</td>
</tr>
<tr>
<td></td>
<td>Embedded systems for aeronautic</td>
<td>Introduction of Reuse in Ada real-time applications</td>
</tr>
<tr>
<td>Garax (Sintef)</td>
<td>Voice communication systems</td>
<td>Reuse of role model (OORAM) &amp; Finite State Machine components</td>
</tr>
</tbody>
</table>

*International Workshop On Software Reuse - Brussels - 8 June 94*

Organizing Reuse

- What advantages are there with reuse?
- Is Reuse profitable for us?
- How to determine our reuse potential?
- How to adapt our organization for reuse?
- How to proceed?
- How to motivate people to follow the new strategy?
- How to measure the progress and success of reuse?
Various reuse approaches

- **General purpose reuse**
  - Small components (e.g. Abstract Data Types, GUI functions, mathlibraries)

- **Domain reuse**
  - Domain-specific components or sub-systems (e.g. financial service libraries, Syntax edit.)

- **Product line reuse**
  - Components common to several products of the same product line
Different reuse organizations

- Project-oriented

This requires:
  - Strong management
  - Strict programming norms
  - Efficient inter-team communication
  - Support for configuration
Different reuse organizations (cont'd)

- **Component-oriented**

  ![Diagram of component-oriented reuse organizations]

  - Project A Team
  - Project B Team
  - Project C Team
  - Component production Team
  - Component Repository

  This requires:
  - Efficient domain analysis

International Workshop On Software Reuse - Brussels - 8 June 94

Different reuse organizations (cont’d)

- Domain-oriented

Project A Team

Project B Team

Project C Team

With reuse

NetWork production team

DataBase production team

IA production team

User-Interface production team

For reuse

Component Repository

This requires:

- Group planning
Different reuse organizations (summary)

- Integration projects
- Component development and adaptation
- Component oriented organization
- Component development
- Projects adapting and integrating
- With reuse adaptation
- With reuse integration
- For reuse

Component Repository
Key factors for success

- Managerial involvement
- Strong technical leadership
- Efficient communication between teams
- Explicit integrations of reuse activities in development process
- Effective standardizations of product and process
- Repository and tools to support reuse
- Accurate training and motivation of people
Management issues

- Extended project interface: more negotiations
- Increased complexity of planning
- Cost-benefit analysis
- Extended and new roles: domain expert, component developers, librarians
- Development process
- Communication inter teams
- Additional documentation
- Maintenance of reusable components
Various Reuse Approaches

- Object-oriented techniques
  - Reusability is one of the most enticing "promises" of object-oriented techniques
  - Abstract data types, inheritance, information hiding, classes.

- Domain Specific Software Architectures or Frameworks

- Application generators
  - Report generators
  - Expert-system generators
  - Parser and Compiler generators
  - Graphical User Interface generators
Reusable Software is more than source code

- To maximize the benefit, reuse must be considered from the beginning of, and throughout, the software life-cycle.
  - Requirements
  - Analysis
  - Design
  - Coding
  - Testing & Integration
  - Maintenance

- The entire product line should be engineered with reuse in mind.

- Policies, procedures, guidelines, and standards must be ‘reuse friendly’.
Overview of the Reusable Component model

- Component
  - Analysis
  - Doc.
  - Tests
  - Design
  - Code
  - Interface
  - Body
  - Classification terms
  - Qualification data
  - Reuse History

Dependencies on other components

Includes Component breakdown Realizes
Reusable component life cycle

- Development FOR reuse is the planned activity of constructing a component for reuse in contexts (systems) other than the one for which it was initially intended.

- Development WITH reuse is the search for, evaluation, adaptation and integration of existing components in a new context (system).
Development FOR reuse strategies

- A priori Development FOR Reuse

- A posteriori Development FOR Reuse
  = RE-ENGINEERING
  = Reverse + Forward Engineering

- Development FOR Reuse while developing an application
Development FOR reuse activities

- Identify potential generality
- Identify potential reuse customers
- Collect additional requirements with benefit estimates
- Analyze requirements for specificity and conflicts
- Identify subsystems which can be developed for reuse
- Determine design alternatives and their costs
- Decide on design alternative and development for reuse timing
Overall development FOR reuse process

Reuse Process and Guidelines

Analysis

Architectural Design

Detailed Design

Implementation

Testing

Application A

Application B

Future Applications

Reusable components

International Workshop On Software Reuse - Brussels - 6 June 94

Why Domain Analysis?

- To discover and define models and architectures common to a family of applications with the purpose of making them reusable.

Group dynamic modeling process

- Models are developed jointly with people (experts, users, managers, designers, etc.) who otherwise would have been interviewed separately.

==> Models completed in shorter time
Inconsistencies can be dealt with immediately
Domain Analysis method [Arango 93]

1. Domain characterization
   - Business analysis
   - Risk analysis
   - Domain description
   - Data identification
   - Inventory preparation

2. Data collection
   - Abstraction recovery
   - Knowledge elicitation
   - Literature review
   - Analysis of context and scenarios

3. Data Analysis
   - Ident. of entities, operations & rels.
   - Identification of decisions
   - Modularization
   - Analysis of similarity
   - Analysis of variation
   - Analysis of combinations
   - Trade-off analysis

4. Taxonomic classification
   - Clustering
   - Abstraction
   - Classification; generalization
   - Vocabulary construction

5. Validation - Evaluation
Re-engineering for reuse method

Analysis of Sourcecode

"obvious" Object

Clustering of Data-Items

Clustering of Functions

Detect Meaning

Functional closure

Isolation

Wrap it up

Reusable Object

No candidate

Determine Reuse Potential

International Workshop On Software Reuse - Brussels - 8 June 94


REBOOT
Development WITH reuse strategies

- Reuse in the large
  - Identify large reusable components early in the software life cycle

- Reuse in the small

- Instantiate Object-Oriented Framework
Development WITH Reuse Activities

- Identify components and subcomponents to be developed
- Search for potential reusable components
- Evaluate identified reusable components
- Negotiate customer requirements on the basis of candidate components
- Adapt and integrate selected components in design
Overall development WITH reuse process
Development WITH reuse process: The Analysis phase

- **Requirements:**
  - Identification of the domain
  - Knowledge acquisition
  - Search for domain knowledge
  - Search and evaluation of standard components
  - Static analysis of the system
  - Adaptation of the static analysis
  - Dynamic analysis of the system
  - Adaptation of the dynamic analysis

- **System Specifications:**
  - Ready to design?

- **Base of reusable components:**
  - Inquiry
  - Consultation
  - Search
  - Evaluation
  - Adaptation

- **General analysis steps:**
- **Reuse specific analysis steps:**
Classification

"The manner in which the assets are organized for ease of search and extraction within a reuse library"

"... Only the librarian knows from the collocation of the volume, from its degree of inaccessibility, what secrets, what truths or falsehoods, the volume contains. Only he decides how, when, and whether to give it to the monk who requests it."

"The Name of the Rose" by Humberto Ecco
Classification and Retrieval

+ Classification is essential for meaningful organization of the collection

+ Search and Retrieval of components must be effective (Mapping of wanted versus existing functionality)

+ Selection of similar components depends on how collection is organized

==> We have chosen a faceted classification scheme
The REBOOT Classification scheme

- **Facet-based:** Ideas of Ranganathan and Prieto-Diaz adapted to the classification of object-oriented components.
  
  One term-space per facet.

- **Four facets:**

  - **Abstraction:** Noun characterizing a component (stack, flight reservation)
  
  - **Operations:** Operations of the component (push etc., reserve etc.)
  
  - **OperatesOn:** Entities the component acts on (integers, seats)
  
  - **Dependencies:** Non-functional characteristics (Unix-based, C++)
An example of classification
(with the specialize / generalize hierarchy of the termspace)
Reuse Product Metrics

To measure

- Reusability
- Quality

of potential reusable components

and to compare different components
The Reusability Model

- Factors -
  - Portability
  - Flexibility

- Criteria -
  - Environment independence
  - Modularity
  - Generality
  - Code complexity
  - Self descriptiveness
  - Documentation quality
  - Component complexity
  - Error tolerance
  - Observed reliability

- Metrics -
  - % system dep. code
  - % machine dep. code
  - Cyclomatic complexity
  - LOC / # methods
  - Fan-in, fan-out
  - Module self descriptiveness
  - % Comments
  - % Documentation
  - #Tests / #errors
The Quality Model

- Reliability
  - M12: MTTF observed during integration and acceptance testing
- Maturity
- Consistency
  - M14: Name consistency
  - M15: Procedure consistency
- Testability
  - M13: Traceability
  - Mn: Requirements quality
- Maintainability
  - M7: Relative number of system dependent LOC
  - M16: Average boolean expression complexity
  - M17: Maximum static nesting level
  - M18: Internal variable ratio
  - Me: Protocol complexity
  - Md: Polymorphism complexity
- Simplicity
- Modularity
  - M5: Average member function size
  - Mb: External class complexity
  - Mh: External class group complexity
  - Mg: Internal class group complexity
- Self-descriptiveness
  - M3: Comments density
  - M4: Comments descriptiveness
A posteriori factor estimates

\[ Reliability(t) = e^{-t/maturity} \]

Maintainability = \( \frac{\text{Maintenance productivity}}{\text{Development productivity}} \)

\[ Portability = 1 - \frac{\text{Cost to port}}{\text{Cost to develop}} \]

Flexibility = \( \frac{\text{Change productivity}}{\text{Development productivity}} \)

\[ Understandability = 1 - \frac{\text{Cost to understand}}{\text{Cost to develop}} \]

Confidence = Reliability + \( 1 - \text{Reliability} \) Fault tolerance
Reuse Process Metrics

- To measure:
  a. The effort spent in reuse-specific activities (costs)
     - additional effort to develop reusable component
     - effort to document for reuse, to qualify and classify
     - effort to search, evaluate, retrieve and incorporate.
     - ...
  
  b. The improvements due to reuse (savings)
     - Saved work by reuse of components
     - Reduced number of errors
     - Reduced maintenance costs
     - Reduced rework because of better analysis and design
     - Increased standardization
     - ...

### Example of collected cost data

<table>
<thead>
<tr>
<th>Data carrier</th>
<th>Data value</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>The domain where the reused component was developed - title</td>
<td>Time management</td>
<td>Framework for time management</td>
</tr>
<tr>
<td>Type of component (method, class, framework, etc.)</td>
<td>framework</td>
<td></td>
</tr>
<tr>
<td>Size of reusable component - Lines Of Code (LOC)</td>
<td>3000</td>
<td></td>
</tr>
<tr>
<td>Cost of development for reusable component - person-days</td>
<td>120</td>
<td></td>
</tr>
<tr>
<td>Estimated costs for making non-reusable component - person-days</td>
<td>80</td>
<td>However this module has supported several changes in requirements of the general system. Most time would have been necessary if it were not reusable.</td>
</tr>
<tr>
<td>Effort spent on documentation - person-days</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>Development costs spent on design - person-days</td>
<td>30</td>
<td>The hierarchy has been changed several times.</td>
</tr>
<tr>
<td>Development costs spent on coding - person-days</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>Development costs spent on testing - person-days</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>Number of errors found during testing</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>
The REBOOT Environment

- Runs on UNIX (currently on Sun Sparcstations).
- Compliant with standards (OSF/MOTIF, ...).
- Designed to be easily integrated in Soft. Dev. Envt. (e.g. ISD).
- Written in C++ (more than 100.000 lines).
- Portable onto other database platforms (e.g. RDBMS, OODBMS).
- Available for experimentation under non-disclosure agreement.
The Reuse Coordinator

Role Session Activities Display

Role: Session

Activities: Display

Import

Current role:

Process Component

Browse/Edit OM

[Diagram showing various activities and roles]

Export History

Insert

Quality

Delete

Search by Terms

Search by Browsing

[Diagram showing various activities and roles]

Load Model

Problems & Recommendations

Comparative Evaluation

Extract

Reuse Report

Edit Temp Space

Edit Database

Edit Notes

Exit History
The Qualification tool: Display of Kiviat graph
The Classification-Retrieval service

Diagram showing the hierarchy of classification and retrieval services.
The Navigator (to search by browsing and maintain database)
The REBOOT Offer

The Methodology Handbook

The Training package

The REBOOT Environment

The Know-How
## The REBOOT Methodology Handbook

- **The audience**
  - **High level manager:** Model to evaluate costs and benefits of reuse in an organization, and outline of the ideal organization.
  - **Project manager:** Support to analyze reuse potential, to organize, to manage and to measure.
  - **Reuse champion:** Guide to incremental introduction of reuse.
  - **Methodologist:** Guidelines and examples to adapt the existing software development to reuse.
  - **Software engineer:** Guidelines to develop FOR and WITH reuse.
The REBOOT Methodology Handbook

- The contents

- Chapter 1: Organizing and managing reuse
  - Benefits of reuse - Success stories
  - Mature reuse organization - Keys reuse area

- Chapter 2: Managing a reuse project
  - How to organize - Roles and tasks
  - Adaptation of the software development process to reuse

- Chapter 3: Development FOR and WITH reuse
  - OO development FOR and WITH reuse
  - Re-engineering for reuse - Cleanroom adaptation

- Chapter 4: Library organization
  - Classification - Component library management

- Chapter 5: Metrics
  - Product metrics - Process metrics - Cost estimation

- Chapter 6: Introducing reuse into an organization
The REBOOT Methodology Handbook (cont'd)

- The version V1.5 (400 pages) has been delivered mid-April 94,

- It will be finalized (version V2) by the end of August 1994, and should be published by John Wiley & Son by the end of 1994.

- V2 will take into account Reviewers' comments and include:
  - additional examples
  - an extended version of the Re-engineering methodology
  - a more detailed questionnaire to assess the Reuse Maturity Level
The REBOOT Courses

- Based on the contents of the Methodology Handbook + Experience

- For various audiences:
  - Management: (2 hours) Non-technical aspects (i.e. Organization, Management, etc.).
  - Project Managers: (1 day) Non-technical aspects + High level technical aspects
  - Developers: (2 days) Technical aspects: How to develop FOR and WITH reuse
  - Library Manager: (1 day) The Reuse tools

- Generic courses to be adapted to the audience and context
Conclusion

We are ready to help you introduce REUSE in your organization