Reusable Software Components

Trudy Levine
Fairleigh Dickinson University
levine@alpha.fdu.edu
http://alpha.fdu.edu/~levine

Once again, we are listing some reuse courses that are being provided to the software engineering community. The course syllabi, themselves, may be helpful in understanding the issues involved in software reuse. In addition, we see from our first entry that there is interest in reuse in the HDL market. As our previous columns stressed (see alpha.fdu.edu/~levine/reuse_course/columns and alpha.fdu.edu/~levine/wong), Ada could be ideal for this market.

~~~~~~

Qualis, an electronic system design consulting firm, offers two courses

Designing with Reuse in Mind 3 days

Productive Knowledge Series in Design Reuse

Course overview.

This three-day interactive course teaches critical skills to design reliable, reusable virtual components, and the complex system-on-chip (SoC) designs that use them. Drawing upon the proven knowledge of some of the best component and SoC design engineers and companies, this course shows you how to set up a reuse-friendly design environment, how to architect, implement, and verify a reusable component and SoC design, and how to package it for future reuse.

1.How reuse impacts component and SoC design, its economic foundations, how it effects engineer creativity, and how best-in-class design methodologies facilitate successful design for reuse. An overview of a proven SoC/virtual component design process that ensures component reusability, from architectural design to implementation to packaging for release.

2.How to manage and structure information for a complex SoC/component design to facilitate integration. How to use revision control and issue tracking to manage complex SoC/component design databases and ensure that design problems don't go unresolved. How to extract and interpret system state metrics to assess design stability, quality, and release readiness.

3.How to architect a SoC/component design using custom or VSIA-compliant on-chip busses. How to evaluate reusable components for suitability and trustworthiness. How to clearly articulate the SoC/component architecture through a written specification.

4.How to code a design for reusability, including general, HDL-specific, RTL-specific coding guidelines.

5.How to design for reusability, including clocking, partitioning, and synthesis guidelines.

6.How to verify SoC/component designs for first turn success. How to create a test plan, test harnesses, bus functional and behavioral models, and high-coverage test cases.

7.How to package a component for reuse. An understanding of the critical and the optional elements of all reuse packages. How to manage component evolution and issue resolution.

8.Sources of additional reuse knowledge, including text books, reuse field guides and website libraries.

You should have knowledge of the fundamentals of Verilog or VHDL prior to attending this course.

This is an interactive course with numerous labs that cement the concepts taught in the lecture portion of each session. Lab sessions offer access to the critical tools needed for successful SoC and virtual component design, including:

Design management: DesignSync, ProjectSync, IP Gear
Reuse process: Reuse Methodology Field Guide
Synthesis: Design Compiler, BuildGates, Leonardo
Simulation: NC-Verilog, VCS, VSS, ModelSim

Additional point tools are introduced throughout the course. For private on-site sessions, you may select the toolset to be used in the course.

If you wish to purchase a textbook to use as a reference after or during the class, we recommend:

Michael Keating and Pierre Bricaud " Reuse Methodology Manual for System-on-a-Chip Design",
Kluwer Academic Publishers, ISBN 0-792-38175-0

This course is structured as eight sessions taught over three days. Interactive labs and group exercises are distributed throughout the course. For more information about course content and structure, call us at +1.503.670.7200, or contact us via e-mail.
~~~~~~

Section 1: Understanding Reuse and its Processes

Understanding the foundations of reuse that are ushering in change.

Understanding the widening productivity gap, the role of time-to-market, and the basic underpinnings of reuse.

Applying the concept of value-add and how engineers add value in the product development process.

Recognizing the new shift in engineer creativity, and the role of creativity in complex SoC and component verification and architectural design.

How to enable rapid, reusable design development by adhering to best-in-class design flows and practices.

Exploring the role of reuse field guides and methodology documents for building SoC and component trustworthiness.

Interactive group exercise

Overview of a proven SoC and Virtual Component design process.

First things first: the role of the functional specification for component and SoC design. Understanding interface-based design and on-chip bus issues.

Exploring the test plan specification for components and SoC, and the role of behavioral modeling.

Implementing virtual components and SoC, recognizing the implementation differences between hard, soft and firm macros.

Understanding which models are required for implementation, including black-box synthesis, RTL, behavioral, static timing analysis, and layout/floor planning models.

The importance of the release package, including the required and optional components for VSIA compliance.

Identifying the tools used in a state-of-the-art design process involving reuse.

Section 2: Design Revision Control and Issue Tracking

Defining an intuitive file system structure.

How to delineate files pertaining to various abstraction levels, testbenches, or physical tools.

Separating application-specific and reusable components to facilitate updates or extraction for later submission to the repository.

Minimizing the impact of importing the file structure for a component into an SoC design, or an SoC design into a higher-level system or board-level design.

Lab session 1 : Establishing an effective revision control system for component and SoC design.

Description of the different revision control models, such as mirroring versus local copies, the lock model versus merge model, etc.

Understanding the implications of each model, and how to match revision models to the phases of the design process.

Identifying which design files to maintain. Understanding the difference between a source file and a derived file.

How to check files into and out of the revision control system. How to manage a product release from the revision system, and how to tag a collection of source files for release. How to 'backtrack' the design to earlier stable releases.

Exploring the role of system state metrics, including code change rates, design stability, etc. How to extract and interpret metrics from the revision control system.

Lab session 2: The importance and role of issue tracking in component and SoC design.

Understanding when issue tracking should begin in the design process.

Evaluation of traditional issue tracking systems such as the water cooler and 3M post-it methods. How to establish a modern issue tracking systems such as Synchronicity ProjectSync, gnats, and others.

Understanding the mechanics of issue reporting. How to write useful issue reports. How issue tracking and revision control are integrated.

Exploring the role of system state metrics, including bug find / fix rates, design stability and quality. How to extract and interpret metrics from the issue tracking system.

Section 3: SoC/Virtual Component Architecture (VSIA compliant)

A thorough introduction to the on-chip bus hierarchy and standard intercomponent interfaces, and the role of bus bridges.

Processor bus

System bus

Peripheral bus

How to architect a SoC or component.

The functional core of a component.

Defining and structuring bus interfaces.

How to manage component configuration. How to balance component flexibility versus simplicity.

How to select and evaluate suitable and trustworthy components.
How to select components from a centralized IP repository.
What to look for in a reusable component, and the role of the packaging checklist.
Evaluating a component for trustworthiness and conformance to standard reuse guidelines.
Balancing the tradeoffs between using a questionable component and creating a new one.

Section 4: Coding for Reusability

General coding guidelines for reusability, including capitalization, code structure and layout, comments, etc., for all languages (Verilog, VHDL, C, C++, Perl, Vera, etc.).
Naming conventions.

Lab session 1

VHDL-specific coding guidelines for reusability:

Using VHDL-93 features and esoteric constructs.
Labeling opportunities.
The proper use of subtypes.

Verilog-specific coding guidelines for reusability:

How to code defensively to avoid race conditions and ensure portability.
Minimizing the impact on the global name space.
Avoiding unspecified behaviors.

Section 5: Design for Reusability

Configuration management
How to manage VHDL libraries.
Specifying all required files for Verilog simulation in a portable and flexible fashion.
Separating project- and user-preferences in tool configurations.
How to use makefiles.
RTL-specific design guidelines for reusability:
How to balance tradeoffs between coding for speed and creating parameterizable/ programmable components.
How to code for functional and manufacturing testability.
How to create components that can be retargeted to diverse technologies.
Synthesis-specific coding guidelines for reusability:
The importance of simplified, realistic synthesis scripting.
Understanding the KISS principle.
How to create parameterizable synthesis scripts with useful default configurations.
How and when to create software-based synthesis script generators.
How software-based script generators encourage reusability.
How to code for complete bottom-up synthesis. How to manage synthesis time budgeting in scripts.

Section 6: SoC/Virtual Component Verification

The importance of creating trustworthy components. An overview of the component and SoC verification processes.
Understanding the appropriate levels for verifying components and SoC designs, including unit level, component level, and macro level verification.
How to ensure proper code, path, and statement coverage, and how to determine what level of coverage is appropriate.
How to articulate the test strategy through clear specifications:
What is a test plan? What is the appropriate level of test specification? When should a test plan be created?
The key components of a test plan, and the role of standardized test plan templates.
The role and process of test plan review.
Rules of thumb for scheduling test plan implementation work. How to ensure minimum test plan implementation time by maximizing parallel scheduling.
The role of behavioral models and how they accelerate verification time and improve component trustworthiness.
Understanding how behavioral models affect simulation performance, and how they compare to RTL models.
How to manage simulations that include behavioral, RTL, and non-HDL simulation models.
The key elements of component verification, including the test harness, bus functional models, and test cases.
How to test components for connectivity.
How to test for functional correctness.
The key elements of SoC verification, and its similarities to component verification (test harness, bus functional models, test cases).
Understanding the role of HW/SW co-verification, and how to determine when and how it is appropriate.

Section 7: Component Productization and Release
How to package a virtual component for reuse by others.
The importance of ensuring a component is trustworthy.
Identifying and collecting the standard package elements, including specifications, source code, synthesis scripts, etc., and optional elements.
How to properly package highly configurable components.
Performing the final review and submission to a component repository. The importance of ensuring submission compliance.
How to deal with component evolution. Identifying who is responsible for maintaining components. How to submit derivative and enhanced components.
Section 8: Sources of Reuse Knowledge
Overall review of the course topics covered.
Where to find sources for additional information on design for reuse, including textbooks, reuse field guides, and website libraries.
How to contact your instructor when you have future questions.
http://www.qualis.com/syllabi/dr-100.html

Making Reuse Happen 1 day Management Series in Design Reuse
Course overview.
This one-day seminar presents you with the critical skills you need to carry-out the implementation of a successful hardware design reuse methodology. Drawing upon the proven experience of some of the best methodology consultants and companies, this seminar shows you what the critical success factors are, both technical and non-technical to a corporate reuse program. With the help and guidance of our experienced reuse methodology consultants, you will learn how to:
1.Define the various levels of reuse.
This section describes different levels of reuse and how each level contributes to the benefits and costs of reuse. It also classifies organizations according to the level of reuse being practiced and their commitment to reuse. From this, you may find out that your organization already practices reuse, and all that is required is to move to a higher level of reuse practices.
2.Evaluate your organization's potential and aptitude for reuse. For reuse to be successful, there must be redundancies and opportunities where reuse can be applied. Also, your organization must be able to take advantage of this potential for reuse through business and technical mechanisms. In this section, you will learn the traits of an organization with high reuse potential and aptitude. You will be able to evaluate how your organization, whether it is a simple department or an entire company, measures up against these traits. Once you understand strengths and weaknesses of your organization, you will know where the greatest challenges in implementing reuse will be.
3.Understand the benefits and costs of reuse. The business case for reuse is easy to make. Everyone sees the economic benefits of reuse. What few managers see (or are willing to see) are the costs, both economic and cultural. This section does not present yet another reuse cost model. Instead, it focuses on the organizational commitment that must be made toward these costs, and presents systems to help transfer these costs to the beneficiaries. Furthermore, the benefits of reuse are not limited to lines of code that did not need to be written. This section also shows where the greatest benefits of reuse can come from and how to maximize them. Pilot projects are a valuable technique to demonstrate the value of reuse. These projects can prototype the design process that is modified to make reuse happen. Choosing the right project is the key.
4.Organize a business unit to make reuse happen. Even if reuse is economically beneficial to an organization, it will not just happen on its own. Processes and mechanisms must be purposefully put in place to promote reuse. This section shows how to tune a reuse initiative to an organization's existing structure and culture. It also shows some project-level management structures that promote reuse of existing assets, and shows the creation of new reusable assets.
5.Implement the mechanisms to support reuse. Reusable assets have a life-cycle. Various mechanisms are needed to support them throughout their progress. This section presents several such mechanisms, from simple checklists to complex database management software, helping support the technical and business issues.
6.Identify and overcome the obstacles to reuse. If reuse were easy to implement, everyone would be doing it. The obstacles are numerous but predictable. This section presents then dispels myths on reuse, and shows common failure modes for reuse programs. The biggest obstacle, the people that make up the organization, can have as many faces as the number of individuals involved. This section also presents common personality traits and their reaction to the introduction of a reuse initiative and how to leverage (or deal with) them.

7.Know the external legal issues and industry agencies. Most reuse will happen within an organization that has few or no legal obstacles. However, it is very likely that your organization will call upon external providers of reusable assets. The number of business models and legal issues equals the number of providers. This section presents common business models and strategies for dealing with licensing legalities. It also presents industry agencies and consortia that have been created to support the exchange and interoperability of reusable assets.

This seminar is designed for anyone who is faced with making reuse happen in any organization. This responsibility falls onto high-level managers, as well as project managers and system architects. This seminar may also be of interest to lead design engineers who wish to gain an appreciation for the business issues and challenges in making reuse happen. Those also involved in the day-to-day creation and integration of reusable assets will be interested in pursuing our hands-on Designing with Reuse in Mind class.

Recommended textbooks.

Because of the recent introduction of reuse methodology in the area of hardware design, no books currently exist on managing hardware design reuse initiatives. However, many of the lessons learned by the software community on reuse over the past 20 years are directly applicable to hardware design. We have found the following books to be very useful:

Wayne C. Lim " Managing Software Reuse", Prentice-Hall ISBN 0-13-552373-7

Will Tracz " Confessions of a Used Program Salesman", Addison-Wesley Publishing Company ISBN 0-201-63369-8

CONTACT:

URL: http://www.qualis.com/syllabi/dr-200.html

Qualis Design Corporation
Three Centerpointe Drive, Suite 250
Lake Oswego, Oregon 97035 USA
Phone: 503.670.7200
Fax: 503.670.0809
Email: hot@qualis.com

Janick Bergeron - VP of Technology
Qualis Design Corporation
janick@qualis.com

~~~~~~

DOD DATA & ANALYSIS CENTER FOR SOFTWARE (DACS) COURSE ANNOUNCEMENTS

Topic: Reuse

WHAT: "Software Measurement: Implementation and Practice."

WHEN: None Scheduled; On Site Option Available

WHERE: ITT Industries, Alexandria, VA

COURSE: This course is designed for the software professional involved in project management, oversight for software intensive projects, software acquisition management, or software development and engineering who has experience with software and software development but is not familiar with measurement or measurement practice.

WHAT: "Achieving High Return on Software Process Investment with Cleanroom Software Engineering"

WHEN: None Scheduled; On Site Option Available

WHERE: ITT Systems, Alexandria, VA

COURSE: This course is intended for software managers, software engineers, SEPG members, and anyone interested in software process improvement and return on investment.

For additional information on these courses, contact:

DoD DACS Customer Liaison
TEL: 315/334-4905
mailto:cust-liasn@dacs.dtic.mil

Or register on-line at: <http://www.dacs.dtic.mil/forms/regform.shtml>.

Access to the Software Technology Topic Areas is available at <http://www.dacs.dtic.mil>.

"Business Case For Software Process Improvement Report & Spreadsheet"
~~~~~~

The purpose of this State of the Art Report (SOAR) is to provide the details necessary to rationalize, from a business perspective, investing in and performing software process improvement. Software process improvement has received much attention in the last few years. However, it has been very difficult to translate benefits achieved in one organization to another organization. The intent of this SOAR is to generalize and model the cost benefits one can achieve from software process improvement.
AVAILABILITY =>Paper Report ONLY -- FREE;
visit the DACS WWW Site to read the report at <http://www.dacs.dtic.mil/techs/roi.soar/soar.html>.
Nancy L. Sunderhaft, Analyst
ITT Industries, Advanced Engineering & Sciences
DoD Data & Analysis Center for Software
775 Daedalian Drive
Rome, NY 13441-4909
nsunderhaft@dacs.dtic.mil & nsunderhaft@rome.ittssc.com
URL: http://www.dacs.dtic.mil
Voice: (315) 334-4949 or (800) 214-7921
Fax: (315) 334-4964
~~~~~~

The **Ohio State University** (OSU) offers an integrated year-long sequence of software component engineering courses within its undergraduate computer science curriculum. The sequence begins with the first course for Computer Science majors (but for which some prior programming experience at the level of a high school course is a prerequisite). The sequence builds on the work of the OSU Reusable Software Research Group.
Several years of experience and evaluation of this approach using RESOLVE/C++ has demonstrated that even freshmen are able to understand RESOLVE-style formal specifications and are able to use them to reason about the behavior of component-based systems.

A Software Composition Workbench, consisting of an integrated set of software tools, supports the RESOLVE approach to component-based software development and is under active development. The workbench guides students in their use and application of RESOLVE methodology with C++ as the delivery vehicle.

Materials used in the courses in RESOLVE/C++ notation are available for dissemination. Tools of the workbench are expected to be released in the future.
CONTACT:
Bruce W. Weide, E-mail: weide@cis.ohio-state.edu;
RSRG Web page: http://www.cis.ohio-state.edu/rsrg/
Course Sequence Web page: http://www.cis.ohio-state.edu/~weide/sce/now

~~~~~~

REUSE EDUCATION IN FRANCE

We teach a 2-day reuse course which can be customized for in-house training on reuse. An example follows.

Software REUSE a Holistic Approach

A two-day course embracing business, organizational, engineering, and resource management across the entire software process. The course is intended for managers and developers from an organization that aim for competitive advantages in software development.

Software reuse will, according to Barry Boehm at DARPA, be one of the major sources of saving in software development over the next 15-20 years. By reusing systems or system parts that already have been developed, an organization enhances its possibilities to both improve the productivity and the quality of the produced software.

The course is based on the work performed within REBOOT (Reuse Based on Object-Oriented Techniques) and SER (Software Evolution and Reuse), two major projects in the reuse-area in the ESPRIT programs. The REBOOT project has covered all the major aspects of reuse and industrial experience has been obtained. SER is a follow up project aimed at disseminating the knowledge from REBOOT and several other ESPRIT projects. The course is a complete reuse methodology (REBOOT) and the result from thirteen real life software development projects adapted for reuse of large and generic components with a focus on object-oriented frameworks and patterns.

This seminar will provide an overview of reuse showing different aspects that have to be considered in a mature organization.

Course Outline

Overview

Introduction to the important aspects of a mature reuse organization. Examples of successful reuse experiences from different domains are presented.

Organizing reuse

Strategic advantages and challenges in changing to organized reuse. Presentation of different mature reuse organizations based on different market and product scenarios. New and changed roles necessary in a reuse organization.

Managing reuse projects

Reuse from the project managers' viewpoint. Topics discussed:

- different types of reuse activities in a development project and how to manage them,
- new roles involved in a reuse project, their potential conflicting interests, and how to handle them
- changes in project planning, managing and following up when organized reuse is involved.

Development for reuse

An overview of the specific problems connected to the development of reusable components including domain analysis and guidelines for design and implementation.

Re-engineering for reuse

Re-engineering by using semi-automatic techniques for restructuring the existing software so it can be reused.

Component management

How to effectively store and retrieve reusable components in a mature reuse organization. Issues covered:

- additional information necessary to make a component reusable and to continuously improve its reusability,
- classification of components
- configuration management of components
- management of the component library

Reuse metrics

Description of how to measure if reuse is having the desired effect. Main focus on:

- Process metrics, i.e. how to measure quality, lead-time and productivity in the presence of reuse.
- Product metrics, i.e. how to measure the reusability and quality of reusable components.

Case studies and examples

Presentation of projects and organizations that have introduced software reuse. Covering both organization aspects as well as technical challenges.

Reuse introduction process

A detailed description of the Reuse Maturity Model which is used to determine the organizations current reuse maturity, as well as giving guidance on how to introduce reuse. Also the Reuse introduction plan which is a template for how to plan, execute and monitor a reuse introduction program.

Adapting existing development processes to reuse

Outline of how to adapt the existing development process of a mature organization to reuse, without changing entirely to a pure object-oriented development process.

REBOOT reuse tools

Description of prototype tools developed within the REBOOT project to support parts of the reuse process, and how they have been incorporated in commercial CASE tools. Mainly support for classification, product metrics and re-engineering.

State-of-the-art components - Frameworks and Patterns

Frameworks are generic components developed and reused for applications within a domain. Patterns are good design solutions solving a certain design problem which can be reused itself or used as a building block of a framework.

A practical approach for efficient large scale reuse

The REBOOT approach reduces many of the classical problems connected to traditional reuse. The approach requires reuse within a domain and a mature development organization but gives efficient large scale reuse.

The ten most common mistakes when introducing reuse. From the experiences when introducing reuse in several organizations we have been able to find the ten most common mistakes which are both technical, organizational and human mistakes.

One copy of the REBOOT methodology handbook, Software Reuse - a Holistic Approach, edited by Even-Andr` Karlsson, Q-Labs, will be included in the course. This handbook follows the course syllabus, to provide background reading and support for all techniques presented. Handouts of the slides will be delivered to all participants.

REBOOT and SER

REBOOT (REuse Based on Object-Oriented Technology) has been one of the major projects on software reuse within the European ESPRIT programme. The project studies, develops, evaluates, and disseminates advanced methodologies for reuse-driven and object-oriented software development with the emphasis on planned reuse. The REBOOT project has developed a comprehensive solution to capitalize on software assets, increase business and productivity, and reduce lead time through ORGANIZED software reuse.

SER (Software Evolution and Reuse) is an ESPRIT follow-up project to REBOOT in order to further promote and deploy software evolution and reuse solutions in European software producing organizations. The project includes activities such as: collecting feedback from SER experiences, leverage European SER projects, and disseminate SER solutions by organizing tutorials, courses, and workshops.

CONTACT:

Jean-Marc MOREL PC: CL F12D22
Software Development Methodology
Bull S.A Rue Jean-Jaures
F-78340 Les Clayes-Sous-Bois, FRANC
E-mail : Jean-Marc.Morel@bull.net
Phone: +33 (0)1 3080 7448
Fax: +33 (0)1 3080 7078

THE **ASE2 CDROM** contains a course on Software Reuse:

SE 508 - Software Reuse

The purpose of this course is to explore contemporary topics in systematic software reuse. This includes the impact of Object-Based and Object-Oriented Design and Programming with Ada83, Ada95, and C++ along with Domain Engineering on the software development process. The course concentrates on the practical aspects of applying architecture-centric, domain-specific, library-based reuse methodologies integrated with the software development process to create software systems in an efficient, cost-effective manner. The course illustrates how object-oriented and domain engineering techniques coupled with domain-specific libraries can be used to effectively develop significant software systems in a short period of time, frequently realizing reuse on the order of 70% or more. Libraries of object-based reusable software components will be used to design and implement solutions to problems.

Material presented in this course includes information from the Software Productivity Consortium (the Reuse-Driven Software Processes Guidebook), the Air Force's Comprehensive Approach to Reusable Defense Software program, ARPA's Software Technology for Adaptable Reliable Systems program, Europe's ESPRIT III Project #7808 (REBOOT - Reuse Based on Object-Oriented Techniques), and several other sources (including IBM and HP). All of the reading material can be found in the Public Ada Library.

The major sections of the course are:

Review of Software Engineering and How Reuse Fits In
Object-Based and Object-Oriented Analysis and Design (with emphasis on designing with reuse, comparing and contrasting Ada83, Ada95, C++)
Domain Engineering
Designing for Reuse
Selected Key Topics (including Non-Technical Issues Pertaining to Software Reuse)

The texts for the course are:

Conn, Richard, "Software Reuse - SE 508 - Course Notes, Version 4," 1997, Software Engineering Department, Monmouth University (included in this release) - Required text
Karlsson, Even-Andre (editor), "Software Reuse: A Holistic Approach," 1995, John Wiley and Sons, ISBN 0-471-95819-0 - Supplementary text
Tracz, Will, "Confessions of a Used Program Salesman: Institutionalizing Software Reuse," 1995, Addison Wesley Publishing Company, ISBN 0-201-63369-8 - Supplementary text

CONTACT:

Richard Conn R_Conn@msn.com
http://unicoi.kennesaw.edu/ase/support/cardcatx/reuse4.htm

Carnegie Mellon University Software Engineering Institute

Developing Reusable Software -01

The Software Engineering Institute of Carnegie Mellon offers a videotaped course titled Developing Reusable Software. The course material covers

1) the reuse process
2) characteristics of reusable software
3) domain analysis

Developing Reusable Software presents participants with both the technical aspects and engineering tradeoffs involved in creating reusable software, and in reengineering existing software to enhance its reusability. Reusability depends on reuse-driven development processes. The course describes these processes as well as analysis and design methods that promote reuse. The course uses comparisons and examples to present alternative approaches for domain analysis, domain design, component implementation, and components-based applications engineering.

By taking this course, participants will gain an understanding of what makes a software component more or less reusable, and what is necessary to reverse engineer existing software that has reuse potential but that is not entirely reusable as it stands. Those responsible for implementing a software reuse initiative will be able to define technical activities of design with reuse and design for reuse, and assess linguistic and methodological support. The course also provides knowledge necessary for populating a library with well-designed components that can be reused by other people and projects.

The course consists of lectures and problem-solving activities, with a strong emphasis on laboratory work, and can be used as direct delivery or as a supplement to existing related courses. Lecture material is mostly language independent; however, component development and modification are required to illustrate the notions of reusability and reengineering. **The Ada language** is designed to support the goals and principles of modern software engineering, particularly those of software reuse, and is used during the course as the primary standard notation for illustration and development purposes.

Topics covered in this videotaped course include:

- The Reuse Process
- Characteristics of Reusable Software
- Domain Analysis
- Object-oriented models
- Feature-oriented domain analysis
- Domain Design
- Domain-specific software architectures
- Object-oriented design
- Domain Implementation
- Component specification
- Design-by-composition
- Design-by-adaptation
- Design-by-extension
- Class-wide programming
- Concurrent and distributed components
- Reengineering components
- Software construction with reusable components

For information see
http://www.sei.cmu.edu/products/videos/dev.reusable.sw.html
or contact Customer relations
Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213-3890