# xtUML / BridgePoint Training

Ericsson has expressed interest in a tailored approach to training engineers in the xtUML method and the use of the BridgePoint tool suite. The audience and goals, along with an outline of each component of this training regimen are described below.

## Audience and Goals

Programmers of varying levels of experience with little or no exposure to model-based development represent the core of the audience. The goal is to produce graduates who can build a working model, given only a functional specification as input.

## Background

In the 1980s and 1990s, Project Technology was very successful teaching the Shlaer-Mellor predecessor to the xtUML method. During this period the method was taught in three separate courses, each nominally four days in length. The exercises were done with paper and pencil, so syntactic correctness and the ability to execute and automatically translate the resulting models were unimportant. Instead, the focus was on learning proper use of the various modeling constructs and creating good abstractions. The three four-day courses in this series were (translating S-M nomenclature to xtUML):

* Components, Interfaces, and Classes
* State Models and Action Language
* Model Compiler Design and Construction

However, this was an era in which managers valued training. The following decade saw a shift away from knowledge and skill acquisition toward tooling. Rather than educate their engineers, managers sought to arm them with tools with the hope that doing so would increase productivity. Over time, market pressure drove the compression of the first two modeling classes into a single, five-day affair. Several modeling constructs and idioms were eliminated from the new one-week modeling class for lack of time to cover them.

Tool usage was taught during a separate three-day course, with the modeling course as a prerequisite. “Eight days of training?!!!” “We simply cannot afford to have our engineers take off eight full days for training!!” “The release is due out next month!!”

Eventually, market forces demanded a combined course that covered modeling and tool usage in less than a week. The result is the current four-day course that attempts to cover what was in the past eleven days of material.

The current audience for these training courses has a quite different educational background and experience base from that of the students in the 1980s and 1990s. However, these two generations of engineers share an important trait. Nearly all of them have been trained to write procedural code rendered in imperative languages like C, C++, and Java before having been introduced to the notion of first creating abstractions around which the code will eventually be written. The proposal below leverages the much broader range of technology available while recognizing that the fundamental challenge of teaching object-oriented, model-based development remains much the same as it was in 1990.

## Proposal

This training regimen includes several components, all of which share the following characteristics:

* May be delivered in an instructor-led or self-paced format
* Modularized, allowing the component to be delivered in two or more cohesive segments

Each of these components is briefly described in a sub-section below. The outline for the program is:

1. Motivational Discussion (half day)
2. Tool Introduction (half day)
3. Requirements Clarification (two days)
4. Basic xtUML Modeling (four days)
5. Tool Training (TBD)
6. Completion of Case Study Model (TBD)
7. Team Modeling Exercise (TBD)
8. Advanced xtUML Modeling (four days)

### Modeling and Tool Usage

Modeling is mostly about creating abstractions, and this process is quite different from using any particular tool to capture, maintain, and test these abstractions. Learning to create good abstractions is often challenging, especially for programmers. To avoid the distractions associated with learning a new tool, the exercises in all the modeling courses are completed using tools with which engineers are already familiar: pencil and paper.

### Case Study Subject Matter

Learning by example is a useful approach for nearly all forms knowledge acquisition, and modeling is no exception. Subject matter for examples employed by modeling courses must meet the following criteria:

* Easy to learn. Since the focus of the course is modeling, the case study must be easy to learn for a broad spectrum of students.
* Outside students’ expertise. To avoid distracting discussions over the accuracy of the case study materials, it’s best to use an application that is outside the students’ area of expertise.
* Tractable, but not simple. So that something approximating a complete application can be understood by the students within the time constraints of the course, the case study must be tractable. However, it must also be rich enough to support the teaching points for the course.

### Motivational Discussion

Before embarking on a new approach to any significant effort it is useful first to explore the need for a change and the rationale behind the new way of working. Such an exploration is best done in full duplex, preferably in person. This session leverages presentations, but it must not be a monologue in which the presenter does nothing more than dictate to the audience how things will be. Instead, the presentation must be interactive, seeking to engage the audience, encouraging questions, and facilitating discussions of the technical aspects as well as the inevitable fear, uncertainty and doubt that accompanies change of this sort.

Topics covered include:

* Motivation for change: issues with traditional development approaches
* Overview of model-based development and testing methods
* Automation of these methods through tooling

### Tool Introduction

* Prerequisite: Motivational Discussion

After the motivational discussion most engineers crave a look at the tooling. Therefore, the next step in the training sequence is a brief overview of the BridgePoint tool suite to expose the student to the look and feel of both the tool and xtUML models. After completing this component the student understands:

* The collection of UML notation used when constructing executable models
* That xtUML models can be executed, tested, and debugged
* That xtUML models can be compiled automatically into executable code

### Requirements Clarification

* Prerequisites:
  + Motivational Discussion
  + Tool Introduction

This modeling course arms the student with the skills necessary to clarify requirements expressed in a natural-language specification and then bridge the gap between that functional-decomposition of the system and an object-oriented model of it.

Upon completion of this component the student is capable of:

* Uncovering and resolving deficiencies, inconsistencies, and ambiguities within a natural-language requirements specification
* Specifying use cases with pre/post-conditions
* Building an activity diagram for each use case
* Building a sequence diagram for each use case

### Basic xtUML Modeling

* Prerequisites:
  + Motivational Discussion
  + Tool Introduction
  + Requirements Clarification

With a focus on modeling, this course covers the xtUML method, teaching both the language of modeling and the techniques of object-oriented abstraction.

Upon completion of this component the student is capable of building an xtUML model, given only a functional specification as input. This includes:

* Defining component structure and interfaces
* Building class and state models
* Writing action language
* Connecting models to externally-produced code
* Creating modelled test cases

Given limited time (four days) and the student’s finite capacity to absorb new concepts, this component necessarily omits a number of modeling constructs, idioms, and techniques. However, the material covered in this component is sufficient for building working models.

### Tool Training

* Prerequisites:
  + Motivational Discussion
  + Tool Introduction
  + Requirements Clarification
  + Basic xtUML Modeling

With a solid foundation in xtUML modeling the next step in the training sequence is comprehensive tool training. While this component provides a guided tutorial presenting all the tooling capabilities in a logical sequence along with a quiz for each section, it can also be used as an on-demand reference manual.

Upon completion of this component the student understands the mechanics of:

* Creating workspaces and projects
* Creating every type of model element and diagram supported by the xtUML editor
* Connecting models to externally produced code
* Leveraging version control and configuration management
* Comparing and merging branches
* Searching for model elements within a model
* Executing and debugging a model with Verifier
* Compiling a model and compiling and executing the resulting generated code

### Completion of Case Study Model

* Prerequisites:
  + Motivational Discussion
  + Tool Introduction
  + Requirements Clarification
  + Basic xtUML Modeling
* Co-requisite: Tool Training

This component includes no instruction or guidance but instead is an assignment. The student is provided with a set of requirements for a particular application, most likely the same application used as a case study for the basic xtUML modeling course. The student is then expected to do the following with BridgePoint:

* Build requirements clarification models (use case specifications and activity diagrams) for the application
* Build a complete executable model of the application
* Build a modeled test suite to test and verify the application model
* Debug the model using Verifier
* Generate code for the application and test suite using a specified model compiler
* Test and verify the generated code

### Team Modeling Exercise

* Prerequisites:
  + Motivational Discussion
  + Tool Introduction
  + Requirements Clarification
  + Basic xtUML Modeling
  + Tool Training
  + Completion of Case Study Model

This component emphasizes the value of modeling in small teams of two to four engineers each. Using an industry-specific example (telecom for Ericsson), the students complete the following activities:

1. Divide the system into components and subsystems
2. Assign portions of the model to teams for development
3. Negotiate interfaces among teams
4. Build class, state, and action models within small teams
5. Unit testing within small teams
6. Integration testing among teams

### Advanced xtUML Modeling

* Prerequisites:
  + Motivational Discussion
  + Tool Introduction
  + Requirements Clarification
  + Basic xtUML Modeling
  + Tool Training
  + Completion of Case Study Model

This course covers all the modeling constructs, and several idioms and techniques omitted from the basic modeling course while focusing on the creation of good abstractions. With a goal of teaching students to build models that are easy to understand, extend, and maintain, upon completion of this component each student:

* Knows all of the modeling constructs available in the xtUML method
* Has been exposed to the most useful idioms and patterns
* Understands the range of abstraction levels available
* Can evaluate models along several dimensions including:
  + Accuracy of the abstraction
  + Understandability
  + Level of abstraction
  + Degree of invariance
  + Coupling and cohesion

## Summary

Note that this training regimen cannot promise to produce engineers capable of building high-quality models. For some, producing excellent abstractions is a gift while others must work years to acquire a reasonable level of competency. However, even poorly abstracted models are more accessible and hence more valuable than poorly abstracted code.

Accordingly, this training program produces engineers competent in the xtUML method. How well the method is then employed will vary from one engineer to the next in much the same way as code quality varies across an organization. Though, unlike code, models expose abstractions, encouraging and facilitating improved quality along all the dimensions enumerated above.

# Ericsson Deployment Sequence

The first deployment of the training program described above is targeted for the CSCF organization, which is organized into over a dozen small teams of 3-8 engineers each. Typically, each team is assigned a work package with a nominal duration of 2-8 months.

Ericsson must therefore fit the training program into the context of the CSCF organization. This organization (and perhaps most within Ericsson) has a preference for executing tasks in one-week blocks. Consequently, we should organize the training sequence into one-week blocks, adjusting the content of each segment accordingly. Two weeks of consecutive training is acceptable, but we will likely see resistance to sequences composed of more than two consecutive weeks of training.

The first CSCF model-based project commenced in July, 2013 and is scheduled to complete in February, 2014. This project developed an Emergency Access Transfer Function (EATF), a new function for the CSCF node. The next project will focus on modeling the core of the existing functions within the node, so unlike EATF, it is expected that the requirements for this next project are well understood and accurately documented.

The following sequence begins after a team completes a work package using traditional methods and is embarking on the development of its first model-driven work package.

1. Motivational Discussion and Tool Introduction
   1. Format: Delivered in person by the MDW team.
   2. Length: (1) day
   3. Contents: Overview of the entire model-driven workflow including xtUML modeling for design and model-based test development.
2. Requirements Clarification
   1. Format: Instructor-led
   2. Length: (2) days
   3. Contents: Material sufficient to enable the students to clarify requirements for a work-package by specifying use cases and building activity and sequence diagrams.
3. Tool Training for Requirements Clarification
   1. Format: Self-paced, instructor-supported
   2. Length: (1) day
   3. Contents: Material required to build analysis models for requirements clarification, including:
      1. Creation of workspaces and projects
      2. Ericsson conventions for structure and packaging of analysis models
      3. Specification of use cases, activity diagrams, and sequence charts
4. Work Package Analysis Modeling Sprint(s). This segment is composed of one or more sprints for the purpose of clarifying the requirements for the work package.
   1. Entry criteria: Functional requirements for the work package
   2. Exit criteria: Complete set of analysis models ready for review by the systems engineering team.
5. Basic xtUML Modeling
   1. Format: Instructor-led
   2. Length: (3-4) days
   3. Contents: Material sufficient to enable the students to construct fully operational executable models and modeled test cases.
6. Tool Training for Basic xtUML Modeling
   1. Format: Self-paced, instructor-supported
   2. Length: (2-3) days
   3. Contents: Material required to build executable models
7. Completion of Case-study Model
   1. Format: Self-paced, instructor-supported
   2. Length: (3) days?
   3. Contents: Each student, individually, completes a simplified, but operational version of the case-study model from the basic xtUML modeling course. The goal of this segment is to ensure that every student is capable of producing a working model from a set of functional requirements. The requirements in this case are necessarily constrained to allow the model to be completed and tested within the time allotted for the segment.
8. Team Modeling Exercise
   1. Format: Instructor-led
   2. Length: (2) days?
   3. Contents: With the class divided into small teams of 2-3 students each, this segment begins with a set of functional requirements and analysis models for a telecom example. Each team is assigned a portion of the model to complete. Ultimately, the teams work together to integrate and test the entire model.
9. Work Package Executable Modeling Sprint(s). This segment is composed of one or more sprints as required to construct and unit test executable models for the work package.
   1. Entry Criteria: Reviewed and approved functional specification and analysis models specifying and clarifying (respectively) the requirements for the work package.
   2. Exit Criteria: Tested and verified executable models satisfying the requirements for the work package.
10. Advanced xtUML Modeling.
    1. Format: Instructor-led
    2. Length: (3-4) days
    3. Contents: Remaining constructs, additional idioms, evaluation of model quality along several dimensions.
    4. Sequencing: This component may be delivered whenever the prerequisites have been satisfied. However, it is recommended that the students first complete at least one production executable modeling sprint before attending this course to increase retention of the material.

## Deployment Segments by Week

The training program described above is intended to be delivered during the execution of a model-driven work package as described below.

1. Week (1)
   1. Motivational Discussion
   2. Tool Introduction
   3. Requirements Clarification
   4. Tool Training for Requirements Clarification
2. Weeks (2) – (M)
   1. Requirements clarification through analysis modeling for a work package
3. Week (M+1)
   1. Basic xtUML Modeling
   2. Tool Training for Basic xtUML Modeling
4. Week (M+2)
   1. Case-study Model Completion
   2. Team Modeling Exercise
5. Weeks (M+2) – (N)
   1. Executable modeling sprints
6. Week (N+1)
   1. Advanced xtUML Modeling
7. Weeks (N+2) – (X)
   1. Executable modeling sprints