Model Transformations and Code Generation

Ecole IN2P3 Temps Réel

Ansgar.Radermacher@cea.fr

École d'été, 26.11

- **15h00 – 16h30: TP S1 - Connecteurs**
  - Modélisation (inclus implémentation) d'un connecteur Socket à partir un squelette fourni
  - Test avec un exemple simple sur linux multi-processus (loopback sur la même machine). L'exemple contient la modélisation des composants et leur interconnexions (connecteurs), spécification du plateforme et spécification du déploiement [squelette fourni]
  - [Création d'un connecteur composite Accord distribue] (par encore sur, si ca tourne bien)
- **17h00 – 18h30: TP S2 – Déploiement sur cible**
  - Modélisation d'exemple déjà fait pour la cible [à synchroniser avec Shebli] à base des composants, i.e. modélisation des composants et leur interconnexions (connecteurs), spécification du plateforme et spécification du déploiement.
  - Génération modèle cible, cross-compilation, exécution sur cible

- **Project references: select Socket**
- **C/C++ Build => Settings => Tool Settings**
  - GCC C++ Compiler => Directories
    - Select “Add” => workspace, <own project directory>
    - Select “Add” => workspace, <accord project>
  - GCC C++ Linker => Libraries
    - Libraries: add socket
    - Lib. search path: add workspace/<own project directory>
    - Lib. search path: add workspace/<socket project>
- **C/C++ Build => Settings =>Build Artifact**
  - Name clash with directory => change name, e.g. add .bin

TP

- **Objective: Show that the use of declarative deployment information enables us to create**
  1. A local variant of the system (easier for testing/debugging)
  2. A distributed variant of the system with data acquisition running on PC/ARM7 target

- **Tasks**
  - Complete application implementation
    - Use of services available via required interfaces
  - Specify connector properties
  - Create deployment plans and provide declarative allocation specification
  - Test local application, load binary to target

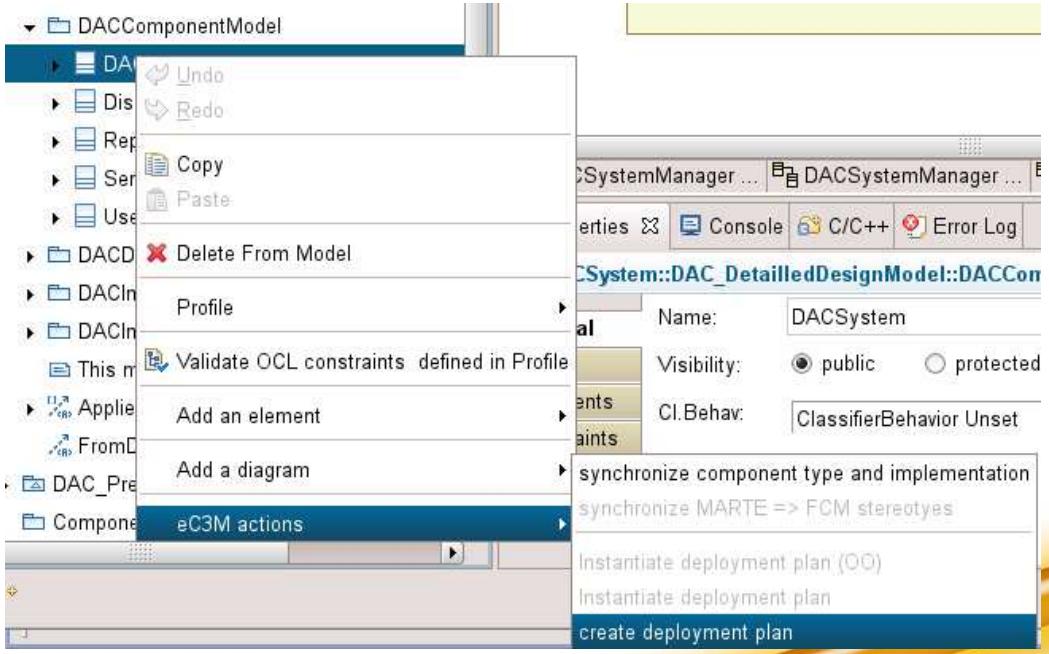
Complement implementation

Specify connector information

- **Specify that the connectors between User and DACSystem is an FCM connector with …**
  - Type is AccordCall, implementation is AccordCall_impl
  - Assure that the MarteCalls model library is imported (import package, library from repository, only select top-level class, deselect contained elements)
- **In case of distributed version, specify that the FCM connector between DAC system and Sensor has …**
  - Type is Async_Call, implementation is Socket_impl
  - Assure that the DistributeCalls model library is imported
- **Note: The explicit specification of an implementation is undesirable, since the use of an implementation is allocation dependent (will be supported later)**

Create an initial deployment plan

- **Create a deployment plan**
  - Select the main system class (either in a diagram or in outline), right-click on
  - Result should appear in DeploymentPlans/newDeploymentPlan
  - Assure that FCM is applied to whole model and that all components carry either the stereotype ComponentType or ComponentImpl

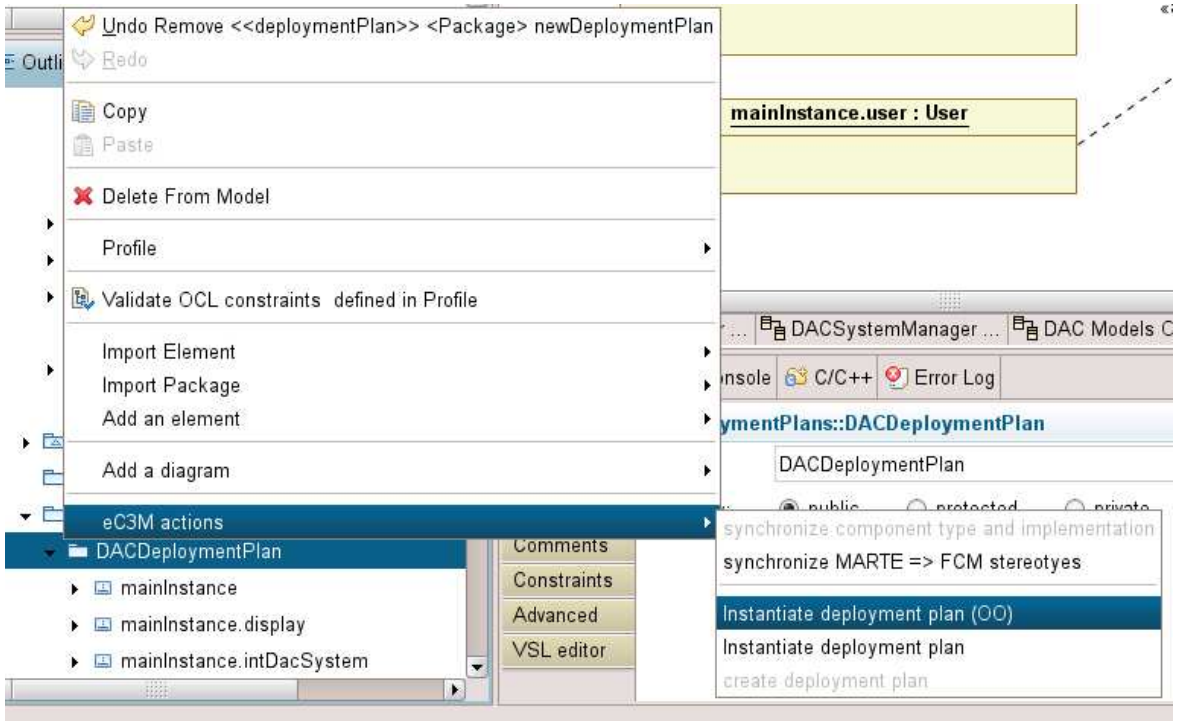


Define two allocation

- **Define an allocation for the initial deployment plan**
- **Create a new Deployment diagram**
- **Drag instances into the diagram**
- **Create two different plans:**
  1. "Trivial" allocation: all instances are on a local node. It is sufficient to allocate only the main instance
  2. Target allocation: allocate all instances except the Sensor on the master station, the sensor on the data capture station
- **Allocations are specified by means of an abstraction, stereotyped with Allocate (from MARTE::MARTE_Foundations::Alloc)**
- **Assure that the MARTE stereotype is applied**

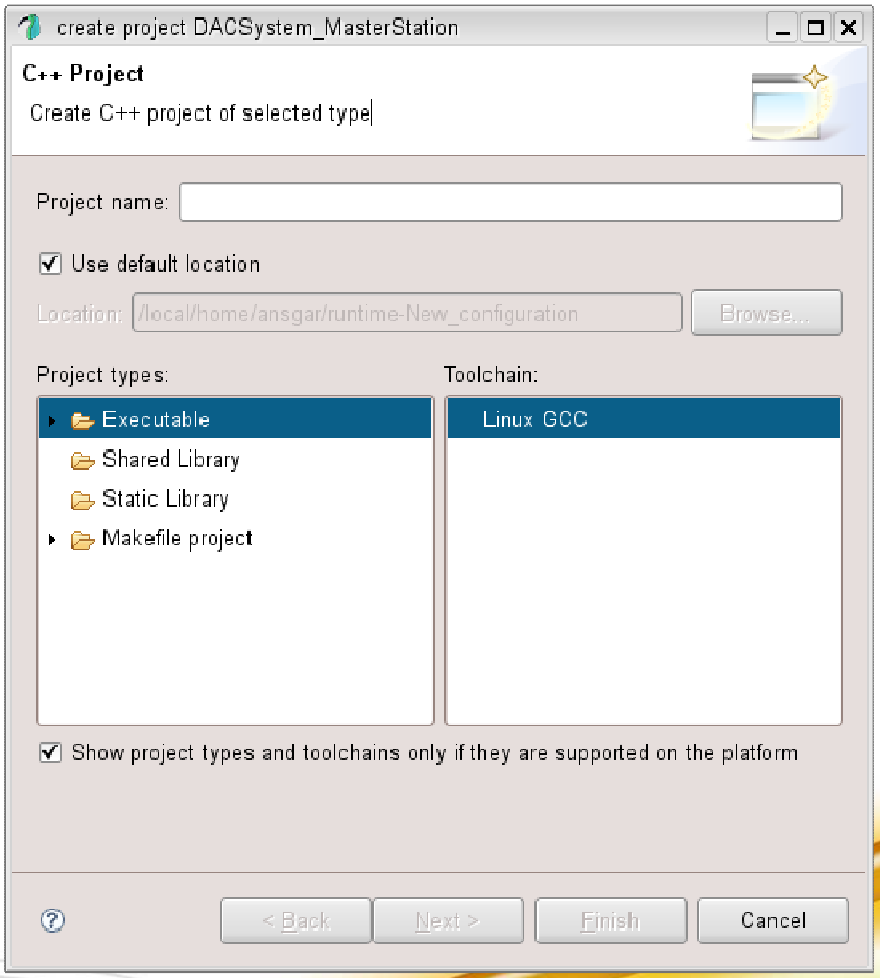
Create code from deployment plan

- **Right click on deployment plan, choose Instantiate deployment plan (OO)**



Create code from deployment plan

- **When prompted, create a new CDT project (be sure to use the name indicated in the dialog title)**
- **Select suitable options for your target (can be done later as well)**
  - Add the project itself to the directories (settings dialog) [be sure to modify C++ part]



Build the project

- Select “Build project” in the Eclipse dialog, choose right compiler chain
- First:Upload binary with TFTP to