MODEL DRIVEN SOFTWARE DEVELOPMENT

LECTURE : 8

MODEL INTERPRETATION

- *Model interpretation* does not generate code from a model to create a working software application.
- Instead, a generic engine is implemented, which parses and executes the model on the fly, with an interpretation approach
- exactly as interpreters do for interpreted programming languages

MODEL INTERPRETATION- ADVANTAGES

- It enables faster changes in the model because it does not require any explicit code-generation step. This can lead to a significant shortening of the turnaround time in incremental development approaches, as the model can be run and modified on the fly.
- It even allows changing the model at runtime without stopping the running application, as the interpreter would continue the execution by parsing the new version of the model
- In case of model interpretation you don't need (and you cannot) delve into the source code of the application anymore, simply because such a concept does not exist anymore (though some tools still allow you to add custom behavior as black-box components to be "interpreted" with the model depending on certain conditions).
- It makes possible the empowerment or modification of the behavior of the running application by updating the interpreter and still keeping the same models (the same can be achieved in case of code-generation by updating the generator but then the code needs to be regenerated, compiled, and deployed again)
- It allows easy debugging of models at runtime, because interpretation can proceed step by step.
- No deployment phase is needed, because the model is already the running version of the application.

COMBINING CODEGENERATION ANDMODEL INTERPRETATION

- Both code-generation and model interpretation are used in practice, and not necessarily as mutually exclusive alternatives
- Hybrid approaches are often used, either intertwined in the development process based on the experience of the developers or within a development platform as a combined solution.
- For instance, within the MDSE process a developer may choose code-generation to take an application to production
- but at the same time the developer can adopt model interpretation during development time, e.g., to speed up the design of functional prototypes of the system
  - thanks to the use of model simulations that help him to get a better understanding

POSSIBILITIES OF HYBRID

- To provide a model interpretation approach based on an internal code-generation strategy.
  - This means that the tools actually generate, compile, and execute the code. However, this is hidden from designer and he feels like interpretation
- To provide a code-generation oriented strategy that relies on predefined runtime components or frameworks to drastically reduce the amount of code to be generated.
  - The predefined components might be flexible enough to carry out complex tasks, which need only simple pieces of code to be generated on purpose.

REVERSE ENGINEERING

- These ***legacy systems*** are often large applications playing a critical role in the company's overall **information system;**
- They have been in use for a **long time**, they have been **developed** with now **obsolete technology, and sometimes they are not completely documented.**
- Therefore, the first problem to be solved when dealing with the evolution and/or modernization of legacy systems is to really understand
- what their **architecture, provided functionalities, handled data, and enforced business rules** and processes actually are.
- This **process** of obtaining useful **higher-level representations** of **legacy systems** is commonly
- called ***reverse engineering***

REVERSE ENGINEERING

- The main goal of MDRE is to offer a better support for the comprehension of existing systems.
- Taking as input the set of artifacts associated to the various components of the legacy system (spanning from source code, configuration files, databases, partial documentation, and so on),
- MDRE aims to create a set of models that represent the system
- These models can then be used for many different purposes, e.g., metrics and quality assurance computation, documentation generation, and tailored system viewpoints

MODEL-DRIVEN REVERSE ENGINEERING PROCESS

A MDRE process includes three main phases.

Model Discovery

- In MDRE, the idea is to switch as soon as possible from the heterogeneous real world (with many legacy artifacts of different nature) to the homogeneous world of models,
- where all artifacts are represented as a set of interrelated models.
- This is what we call the *model discovery* phase.
- A good approach for creating these models is to first focus on quickly creating a set of initial models that represent the legacy system at the same (low) abstraction level

MODEL-DRIVEN REVERSE ENGINEERING PROCESS

Model Understanding

- Most MDRE applications will require the processing of the raw models discovered in the previous phase in order to obtain higher-level views of the legacy systems that
- facilitate their analysis, comprehension, and later regeneration.
- Thus, the second phase is the *model understanding* phase where chains of model manipulation techniques are employed to query and transform the raw models into more manageable representations

Model (Re)Generation

- The processed models obtained at the end of the model understanding phase are finally used to generate and/or display the expected outcome of the reverse engineered process (e.g., the code of a refactored version of the system)

SYSTEM INTEROPERABILITY

- The ability of computer systems or software to exchange information and to use the information that has been exchanged
- MDI (Model Driven Interoperability) bridging process is actually composed of three main consecutive parts:
  - injection (text-to-model);
  - transformation (model-to-model); and
  - Extraction (model-to-text).
- Both projections and transformations can be either atomic or composite (i.e., chains of transformations).
- In most cases, such transformation chains are actually required in order to be able to split the overall problem into simpler ones, and so provide better extensibility, reusability, and maintainability for the bridge.