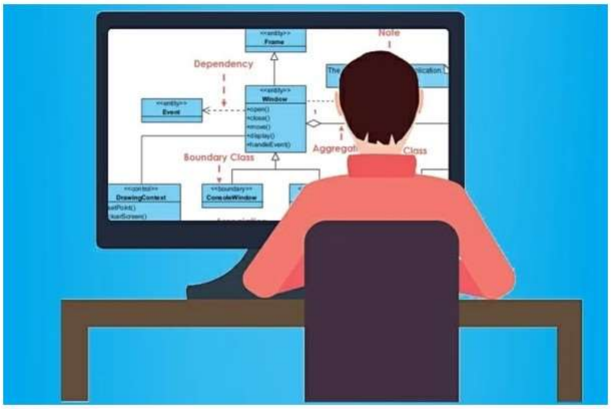
Software Design & Architecture
Spring 2022 - Week-15



مدرس: مہندس ماجد کلیم

جامعہ بحریہ، واقعگاہ کراچی

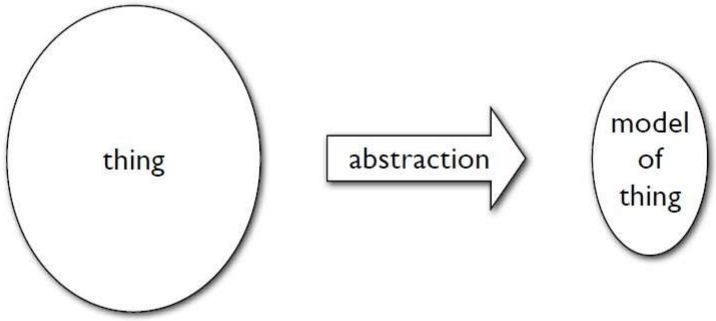
Engr. Majid Kaleem

WEEKLY AGENDA

TENTATIVE WEEKLY DATES		TENTATIVE TOPICS
1	Mar 7th – Mar 11th	INTRODUCTION TO THE COURSE; DEFINING SOFTWARE ARCHITECTURE & DESIGN CONCEPTS
2	Mar 14th – Mar 18th	DESIGN PRINCIPLES; OBJECT-ORIENTED DESIGN WITH UML
3	Mar 21st – Mar 25th	SYSTEM DESIGN & SOFTWARE ARCHITECTURE; OBJECT DESIGN, MAPPING DESIGN TO CODE
4	Mar 28th – Apr 1st	FUNCTIONAL DESIGN; UI DESIGN; WEB APPLICATIONS DESIGN **ASSIGNMENT & QUIZ #1**
5	Apr 4th – Apr 8th	MOBILE APPLICATION DESIGN; PERSISTENCE LAYER DESIGN
6	Apr 11th – Apr 15th	CREATIONAL DESIGN PATTERNS
7	Apr 18th – Apr 22nd	STRUCTURAL DESIGN PATTERNS **ASSIGNMENT & QUIZ #2**
8	Apr 25th – Apr 29th	BEHAVIORAL DESIGN PATTERNS
← MID TERM EXAMINATIONS →		
9	May 9th – May 13th	INTERACTIVE SYSTEMS WITH MVC ARCHITECTURE; SOFTWARE REUSE
10	May 16th – May 20th	ARCHITECTURAL DESIGN ISSUES; ARCHITECTURE DESCRIPTION LANGUAGES (ADLS)
11	May 23rd – May 27th	ARCHITECTURAL STYLES/PATTERNS & DESIGN QUALITIES
12	May 30th – Jun 3rd	ARCHITECTURAL STYLES/PATTERNS & DESIGN QUALITIES **ASSIGNMENT & QUIZ #3**
13	Jun 6th – Jun 10th	QUALITY TACTICS; ARCHITECTURE DOCUMENTATION
14	Jun 13th – Jun 17th	ARCHITECTURAL EVALUATION TECHNIQUES
15	Jun 20th – Jun 24th	MODEL DRIVEN DEVELOPMENT **ASSIGNMENT (PRESENTATIONS) & QUIZ #4**
16	Jun 27th – Jul 1st	REVISION WEEK
← FINAL TERM EXAMINATIONS →		

WHAT IS A MODEL?

- A simplified representation *(usually mathematical or graphical)* used to explain the workings of a real world system or event



abstraction = forgetting details

3

WHAT IS A MODEL? VARIOUS DEFINITIONS

"A model is a simplification of a system built with an intended goal in mind. The model should be able to answer questions in place of the actual system." Jean Bézivin

*"A **model** is an **abstraction** of a (real or language based) **system** allowing predictions or inferences to be made."* Kuehne

*"**Models** help in developing **artefacts** by providing **information about the consequences of building those artefacts** before they are actually made."* Ludewig

*"A **model** of a system is a **description** or **specification** of that system and its environment for some certain purpose."* OMG

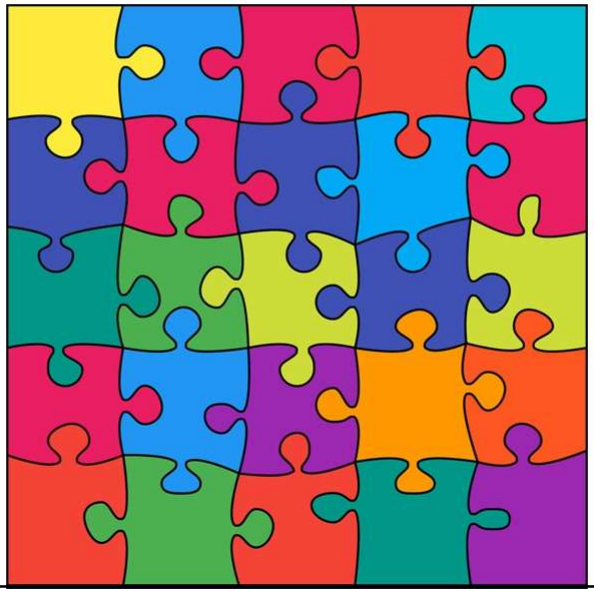
4

WHAT IS MODEL DRIVEN DEVELOPMENT?

- *Model-driven development (MDD) uses graphical models and pre-built application components so that users can visually construct complex applications.*
- Model-driven development (MDD) is a format to write and implement software quickly, effectively and at minimum cost.
- The rapid solutioning of complex systems from Models which are smaller and more abstract than are those synthesized systems
- Model-driven Development *(MDD)* is an approach that represents the SDLC as a *modeling* and *model transformation* activities.

5

WHAT IS MODEL DRIVEN DEVELOPMENT?



6

WHAT IS MODEL DRIVEN DEVELOPMENT?

- Domain Specific Modeling *(DSM)* provides more expressiveness through visually expressing domain elements using Domain Specific Languages *(DSL)* that are typically captured by *domain experts*.
- The methodology is also known as *model-driven software development (MDSD)*, *model-driven engineering (MDE)* and *model-driven architecture (MDA)*.
- The MDD approach focuses on the construction of a software model.
- *The model is a diagram that specifies how the software system should work before the code is generated.*
- Once the software is created, it can be tested using model-based testing *(MBT)* and then deployed.

7

SOME CONCEPTS!

- **Model based testing** ***(MBT)*** is a software **testing** technique where run time behavior of software under test is checked against predictions made by a **model**.
- A **model** is a description of a system's behavior.
- *Behavior can be described in terms of input sequences, actions, conditions, output and flow of data from input to output.*
- Domain-specific modeling ***(DSM)*** is a software engineering methodology for designing and developing systems, such as computer software.
-
- DSM involves systematic use of a domain-specific language to represent the various facets of a system.

8

SOME CONCEPTS!

- ***DSL*** is a programming language or specification language dedicated to a *particular problem domain*, a particular problem representation technique, and/or a particular solution technique.
- This is in contrast to a general-purpose language (GPL), which is broadly applicable across domains.
- ***Search google to find examples of DSL.***

9

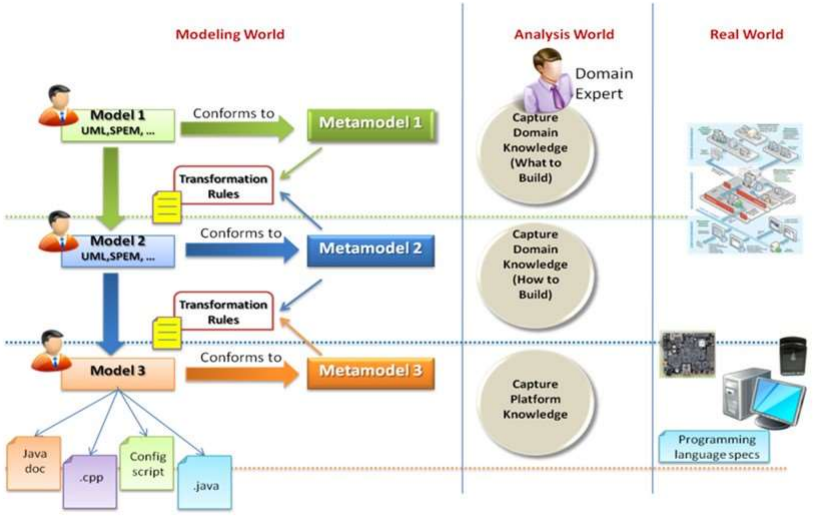
WHY USE MODEL DRIVEN DEVELOPMENT?

- Adopting MDD techniques moves the developers focus from coding to analysis reducing errors as well as time-to-market.
- This is because:
  - MDD and DSM raise the level of abstraction beyond the current programming languages by using the same concepts from the problem domain.
  - Model transformations is the mean through which models can be transformed from one level of abstraction and platform technical knowledge to the another, inserting new information each step of the way.

10

MODEL DRIVEN DEVELOPMENT SCHEME

*A **metamodel** or surrogate **model** is a **model** of a **model**, and metamodeling is the process of generating such metamodels.*



SPEM (The Software and Systems Process Engineering Metamodel) is a Profile of UML, which uses UML as a notation and takes an object-oriented approach

11

TWO CORE CONCEPTS

- There are two core concepts associated with model-driven development: ***abstraction*** and ***automation***.
- Abstraction means to organize complex software systems.
- In MDD, complex software gets abstracted, which then extracts easy-to-define code.
- Once developers transform the abstraction, a working version of the software model gets automated.

12

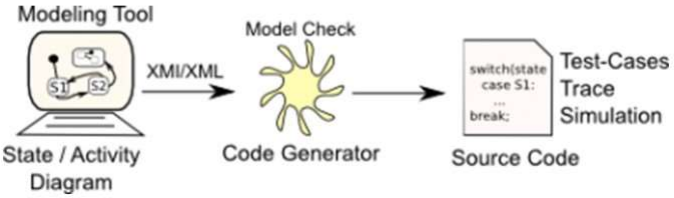
TWO CORE CONCEPTS

- This automation stage uses a domain-specific language (DSL), such as HTML, and scripting languages, like ColdFusion, which can integrate other programming languages and services -- .NET, C++, FTP and more -- for use in websites.
- DSL is a language specialized to an application domain. A model is written in a DSL language and is utilized for transformation in coding language from the model to working software.

13

TWO CORE CONCEPTS

- Because model-driven development uses *visual modeling* techniques to define data relationships, process logic, and build user interfaces, model-driven software development empowers both developers and business users to rapidly deliver applications *without the need for code*.
- Consequently, model-driven development is significantly faster than traditional programming languages like C# and Java.
- Agile software development methods are often paired with MDD.



14

BENEFITS OF MODEL DRIVEN DEVELOPMENT

- The MDD approach provides advantages in productivity over other development methods because the model simplifies the engineering process.
- It represents the intended behaviors or actions of a software product before coding begins.
- The individuals and teams that work on the software construct models collaboratively.
- Communication between developers and a product manager, for example, provides clear definitions of what the software is and how it works.
- Tests, rebuilds and redeployments can be faster when developing multiple applications with MDD than with traditional development.

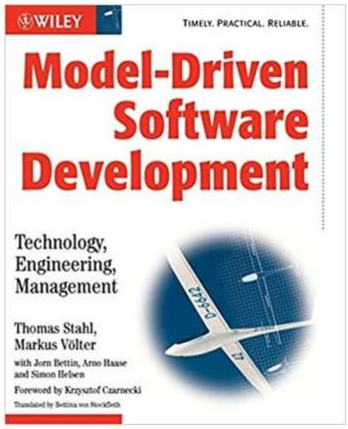
15

MDD TOOLS

- Software tools, such as Rational Software Architect, Simulink and Sirius, create models for an MDD approach to software design.
- IBM's *Rational Software Architect* is a modeling and development tool that uses Unified Modeling Language *(UML)* to design models for C++ and Java Platform, Enterprise Edition *(Java EE)* applications and web services.

16

Further Reading



17

```
If(anyQuestions)
{
 askNow();
}
else
{
 thankYou();
 submitAttendance();
 endClass();
}
```

REFERENCES

1. ***Software Architecture***, *Perspectives on an Emerging Discipline By Mary Shaw & David Garlan*
2. ***The Art of Software Architecture***, *Design Methods & Techniques By Stephen T. Albin*
3. ***Essential Software Architecture***, *By Ian Gorton*
4. ***Microsoft Application Architecture Guide***, *By Microsoft*
5. ***Design Patterns***, *Elements of Reusable Object-Oriented Software By by Erich Gamma, Richard Helm, Ralph Johnson & John Vlissides*
6. ***Refactoring, Improving the Design of Existing Code***, *By Martin Fowler & Kent Beck*