

Transformation of class hierarchies during software development in UML

Authors: H. Wijekoon & V. Merunka

Presenter: Himesha Prabhakara Wijekoon, M.Sc.

PhD. Student, Czech University of Life Sciences Prague, Prague, Czechia
email: wijekoon@pef.czu.cz

DigitalWorld 2022 Congress

**The Sixteenth International Conference on Digital Society
ICDS 2022**

Porto, Portugal, June 26-30, 2022



Himesha Wijekoon M.Sc.



- Himesha Wijekoon is a lecturer in information technology at the Department of Industrial Management in University of Kelaniya, Sri Lanka.
- He holds a M.Sc. in Systems Engineering and Informatics from Czech University of Life Sciences Prague, Czech Republic.
- At present, he is a Ph.D. candidate at the Department of Information Engineering in the Czech University of Life Sciences Prague, Czech Republic.
- He also has more than 10 years of experience as a software engineer in multiple companies.
- His main research interests include *crowdsourcing, software engineering, software modelling and model driven engineering.*

Vojtěch Merunka, M.Sc., Ph.D.

- <http://vojtech.merunka.eu>
- Vojtěch Merunka is an Associate Professor in Information Management since 2005 at the Czech University of Life Sciences in Prague and the Czech University of Technology in Prague. He teaches software engineering and information management.
- He holds a Master's in Computer and Software Engineering and has obtained his Ph.D. in Data Processing and Mathematical Modelling in 1998.
- He also has 15 years of experience in the international management and consulting company Deloitte.
- He is a founding member of the MOBA (Modelling of Business Agility) workshop and the SIGMAS (Special Interest Group on Modelling and Simulation) at the AIS.
- Vojta is professionally interested in object-based programming languages and object-oriented methods and tools for modelling and simulation.



Introduction

- The objective of **Unified Modelling Language** (UML) has been and is to replace older methodologies of the aforesaid authors by one methodology that is a combination of the best of the older ones.
- **Model Driven Architecture** (MDA) philosophy is a synthesis of previous best experiences in the creation of large-scale software, where there is a semantic gap between programmers and people in the area of the modeled problem.
- This paper discusses the usage of the UML standard in the Model Driven Architecture approach for software development, how different types of **object hierarchies** can be expressed in UML class diagrams.

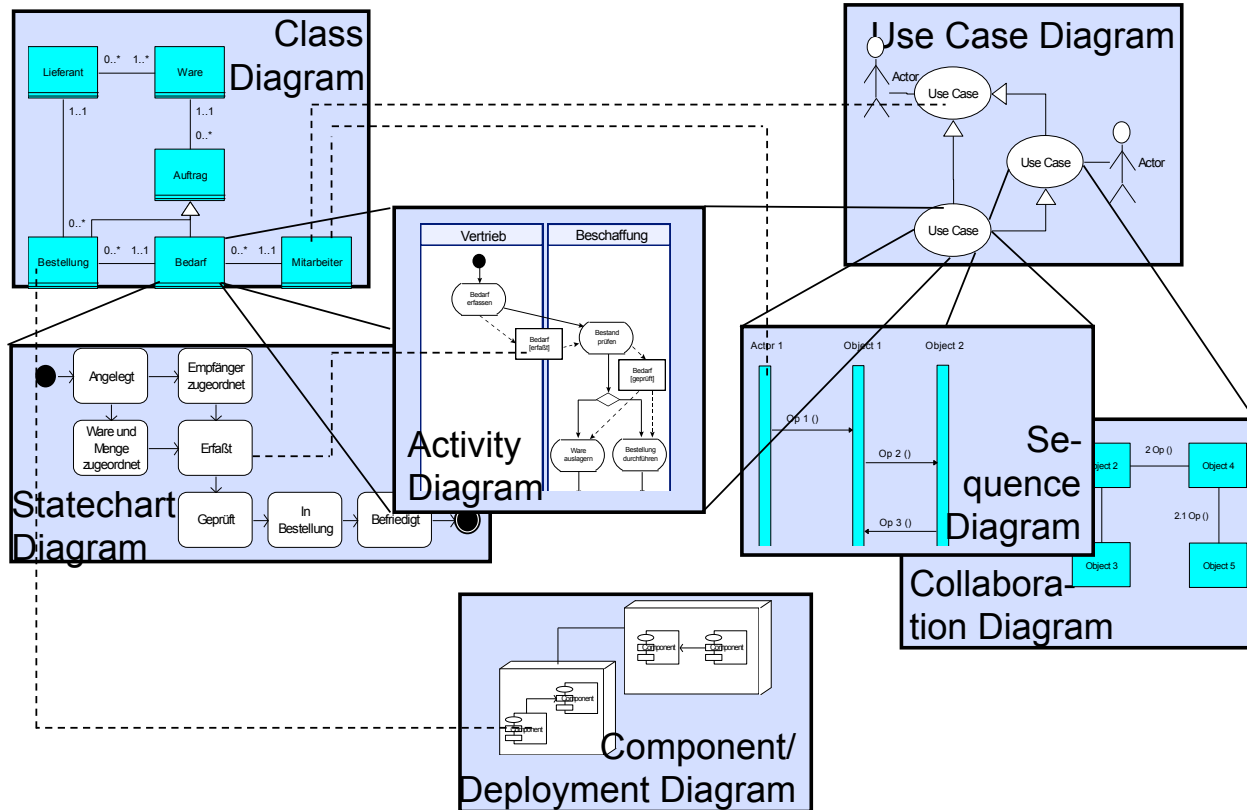
Unified Modeling Language (UML)

- Before the arrival of UML, in early 1990s, we had several competing object-oriented methodologies with mutually different notations. These were so called first generation object-oriented methodologies. Many software companies used a combination of several methodologies instead of just one methodology. **UML has brought along a unification of the previous notations.**
- **Obviously, the UML is not a method. UML is “only” a modelling language. UML can be combined with several software development methods.**

Some UML issues

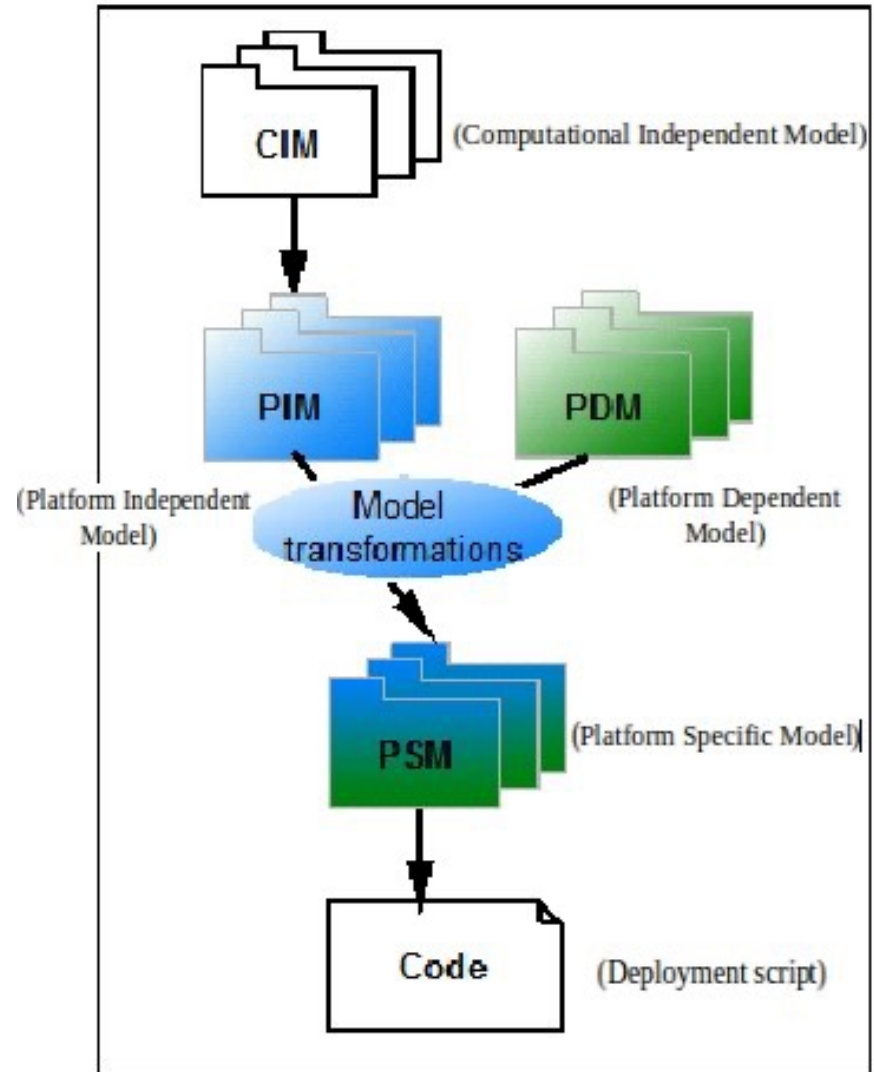
Most criticism at UML is directed at its **complexity** and **inconsistency**.

UML has too many concepts and is **not orthogonal**. There are several ways in the UML diagrams to show the same certain details.



MDA Approach

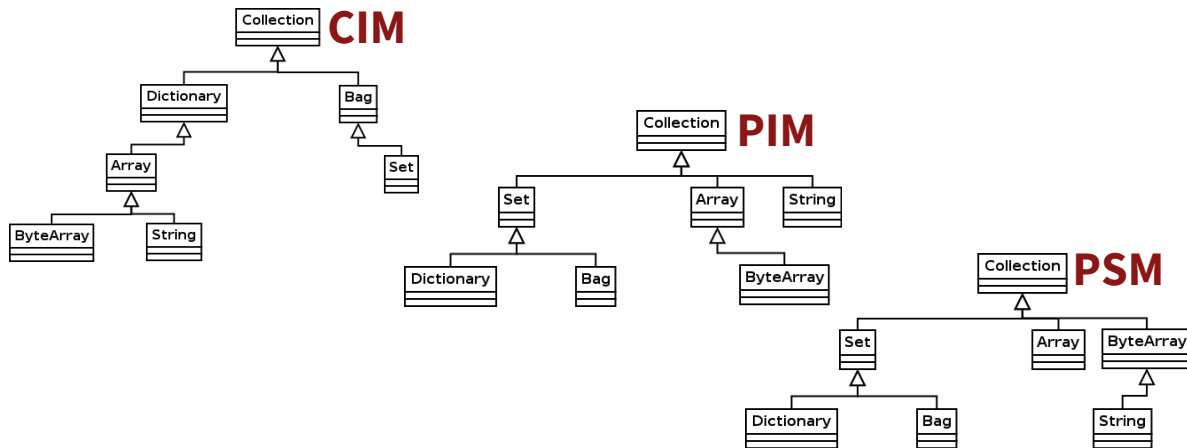
- **Computer-Independent Model (CIM)** reflects customer's **business requirements**.
- **Platform Independent Model (PIM)** deals with the system specification that does not change according to the particular type of computer platform chosen.
- **Platform-specific model (PSM)** combines PIM with a specific technology solution.



Three different types of class (object) hierarchies

Conceptual hierarchy of classes (of CIM stage of MDA), hierarchy of data types (of PIM stage of MDA), and hierarchy of inheritance (in PSM stage of MDA) do not necessarily mean the same thing regardless all three hierarchies are drawn **in the same way in UML**, and all we can only use some stereotypes to distinguish among them in detail.

IS-A hierarchy \neq supertype/subtype hierarchy \neq inheritance hierarchy



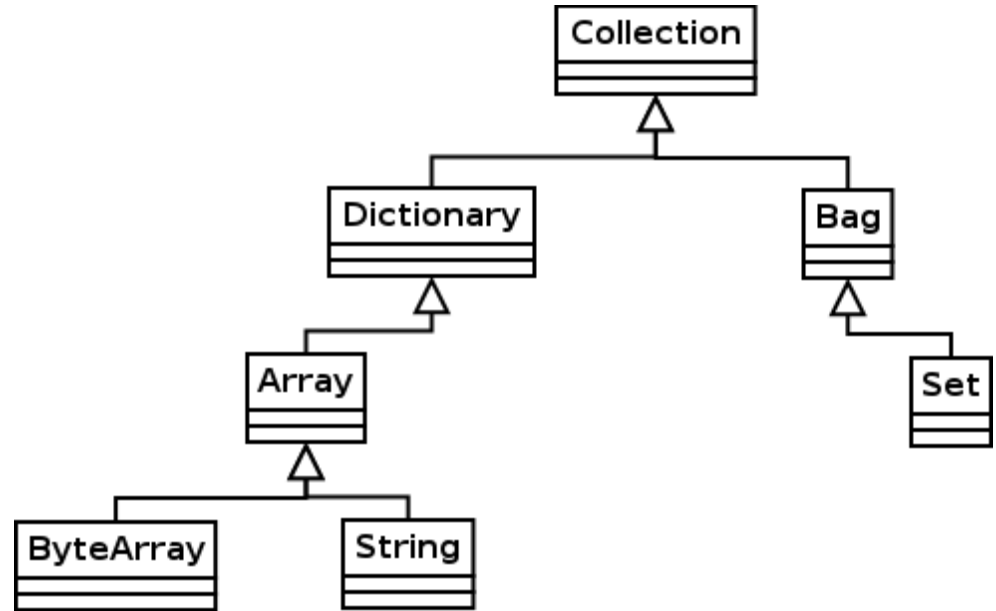
CIM – ISA hierarchy of objects (e.g. class taxonomy)

The perspective of the user or the business domain analyst – the instances of lower-level classes then must be elements of the same domain that also includes the instances of the classes of the superior class. It means that a lower-level domain is a subset of a higher-level domain.

$A \supset B$

where A is a superclass
and B is a subclass means

$\text{extent}(A) \supset \text{extent}(B)$



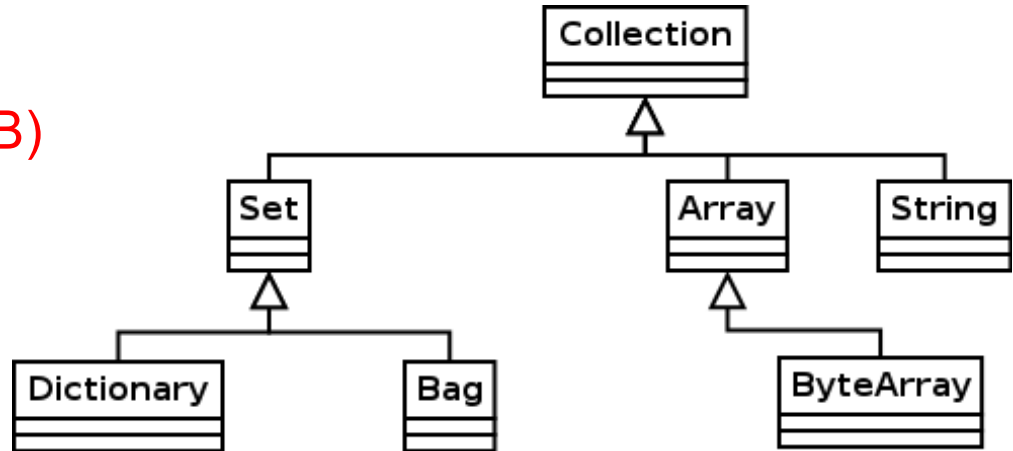
PIM – supertype/subtype hierarchy (also interface hier.)

This is a view of an application programmer who needs to know how to use the objects (typically from libraries or development frameworks) in the system but does not program them, just use them.

$A \supset B$

where A is a superclass
and B is a subclass means

$\text{interface}(A) \subseteq \text{interface}(B)$



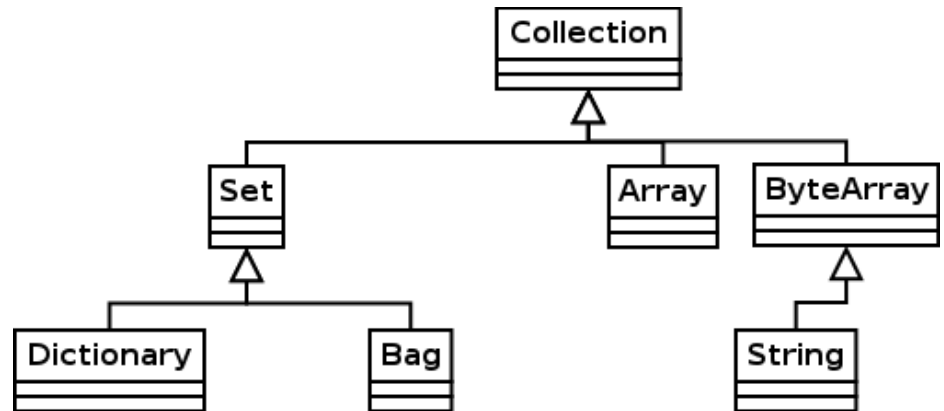
PSM – implementation hierarchy (e.g. class inheritance)

The view of a system programmer who needs to create these objects. This hierarchy is a hierarchy of inheritance because inheritance is a typical tool for the development of new classes.

$A \rightsquigarrow B$

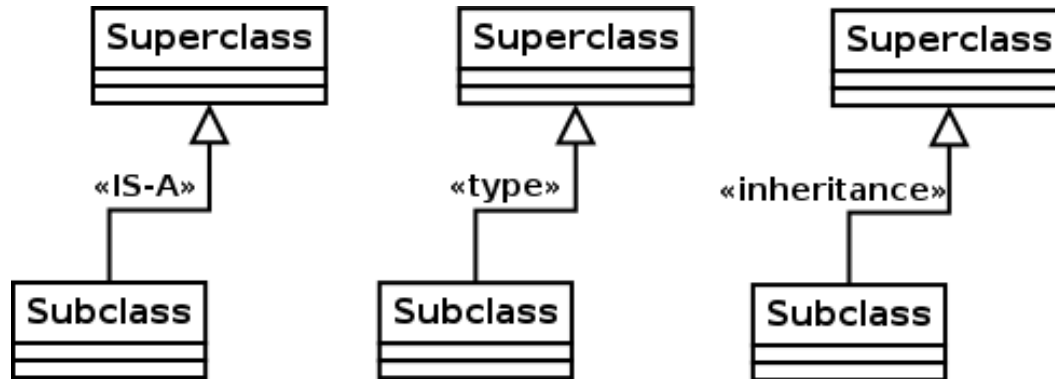
where A is a superclass
and B is a subclass means

$\text{methods}(A) \subseteq \text{methods}(B)$



Discussion

We have just explained the need for different class hierarchies. The problem remains to be resolved is how to express them in the UML class diagrams. Fortunately, the UML standard includes an extension mechanism that allows new concepts to be introduced in a standard way. Of course, if each UML class diagram clearly indicates what phase of the model it is (CIM, PIM, or PSM in the style of MDA, for example), then this stereotype is unnecessary.



The need of MDA way of thinking

During system development it is necessary to **gradually transform** the system model into a condition that is necessary for physical implementation of the system in the final program form in the specific programming language.

According to our experience, initial business objects cannot be viewed only as some kind of a simplification of the same future software objects. This is the common error of the software analysts who use UML. The initial business model must also contain concepts that are not directly supported by current programming languages.

The problem is even more complicated because the function of the major systems built has impact on the very organizational and management structure of a company or organization where the system is implemented – such as new or modified job positions, management changes, new positions, new departments, etc. Therefore, it is desirable to also address the change of these related structures during the work on information systems.

Conclusion

We demonstrated the need of precise interpretation of modelling concepts on an example of gradually transforming object class hierarchy. This approach is a practical realization of MDA ideas in UML.

- UML is not a method; it is “only” a standardized tool for recording ideas. UML needs some method, otherwise it doesn't help.
- Analysis in UML must not be just a graphical representation of the simplified future program code.
- UML itself does not accurately emphasize which concepts are to be used in the separate analysis phase. Unfortunately, many books on UML look at modelling through the eyes of implementation only and are written in a programmer's language.

The thoughts described in this article are a synthesis of our own experiences from object-oriented modelling at Deloitte, from own research activities and from university teaching the development of information systems.

Thank You!

QUESTIONS?