Information systems modeling

Tomasz Kubik

OMG specifications adopted by ISO

Name	Acronym	Version	ISO documents
Business Process Model And Notation	BPMN™	2.0.1	19510:2013
Common Object Request Broker Architecture	CORBA®	3.1.1	Interfaces 19500-1:2012 Interoperability 19500-2:2012 Components 19500-3:2012
Knowledge Discovery Metamodel	KDM	1.3	19506:2012
Meta Object Facility	MOF™	1.4	19502:2005
Meta Object Facility	MOF™	2.4.2	19508:2014
Object Constraint Language	OCL™	2.3.1	19507:2012
OMG System Modeling Language	SysML®	1.4	19514:2017
Unified Modeling Language	UML®	1.4	19501:2005
Unified Modeling Language	UML®	2.4.1	Infrastructure 19505-1:2012 Superstructure 19505-2:2012
XML Metadata Interchange	XMI®	2.0	19503:2005
XML Metadata Interchange	XMI®	2.4.2	19509:2014

https://www.omg.org/spec/

UML and SysML specifications

Name	Acronym	Version	Status	Adoption date
Unified Modeling Language	UML®	2.5.1	formal	Dec. 2017
OMG System Modeling Language	SysML®	1.5	formal	May 2017

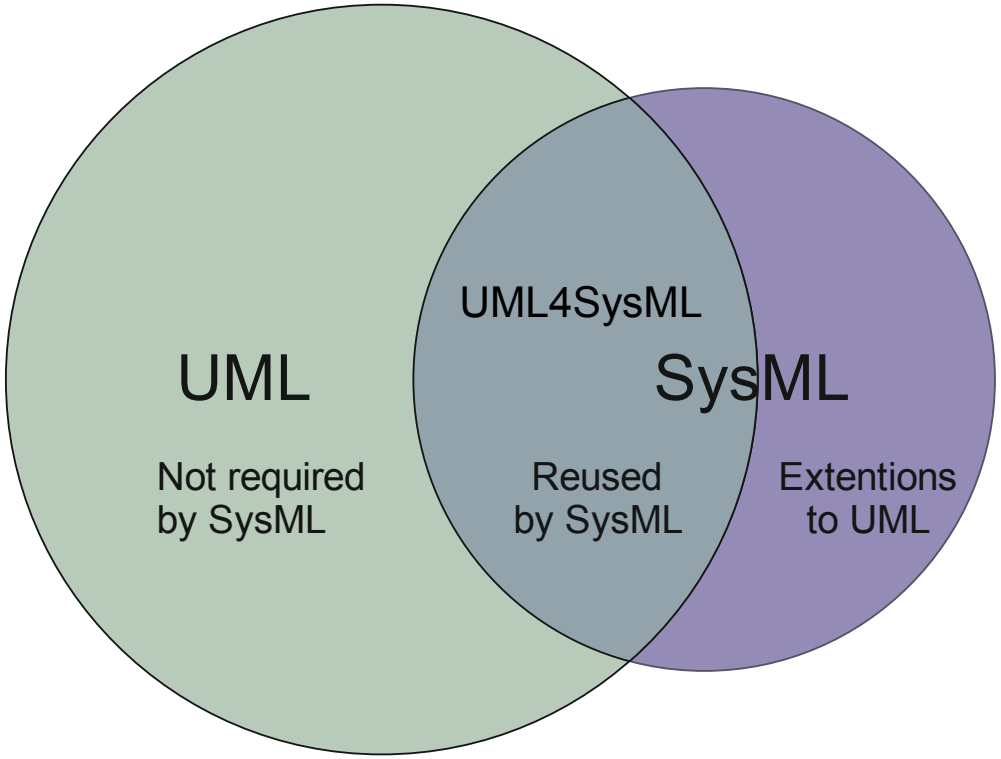
ISO/IEC 19505-1:2012, Information technology -- Object Management Group Unified Modeling Language (OMG UML) -- Part 1: Infrastructure
ISO/IEC 19505-2:2012, Information technology -- Object Management Group Unified Modeling Language (OMG UML) -- Part 2: Superstructure

ISO/IEC 19514:2017, Information technology -- Object management group systems modeling language (OMG SysML)

https://www.omg.org/spec/

OMG SysML

- The OMG Systems Modeling Language™ **(OMG SysML®)** is a general-purpose graphical modeling language for specifying, analyzing, designing, and verifying complex systems that may include hardware, software, information, personnel, procedures, and facilities.



see also:
https://www.omg.org/spec/SysML/
http://www.omgsysml.org/INCOSE-OMGSysML-Tutorial-Final-090901.pdf
http://www.jhuapl.edu/ott/Technologies/Docs/ModelingwithSysMLTutorial.pdf
http://sysmlforum.com/sysml-faq/
https://re-magazine.ireb.org/issues/2015-2-bridging-the-impossible/modeling-requirements-with-sysml/
http://www.pld.ttu.ee/~helena_k/sysml/slides/index.html

OMG Document Number: formal/2015-06-03

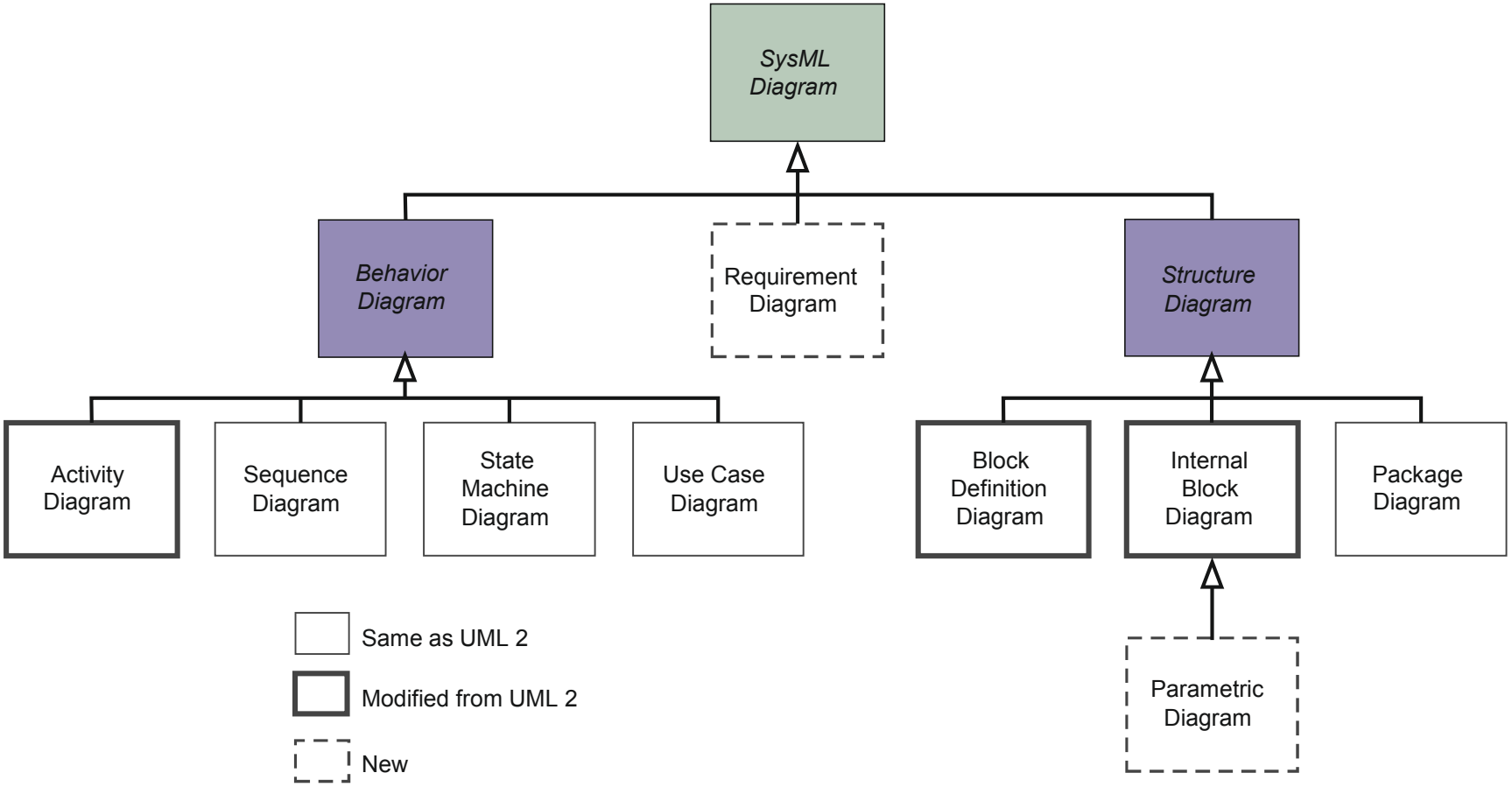
Latest OMG SysML version

- Version 1.5 (OMG document formal/2017-56-01)
  - available at: http://www.omg.org/spec/SysML/1.5/.
  - introduces an Abstract Requirement (primary change comparing to the previous version)
    - to support other kinds of requirements such as property-based requirements by extention
    - enables more precise expressions of requirements beyond purely text-based requirements.

Version	Adoption Date	URL
1.5	May 2017	https://www.omg.org/spec/SysML/1.5
1.4	August 2015	https://www.omg.org/spec/SysML/1.4
1.3	June 2012	https://www.omg.org/spec/SysML/1.3
1.2	June 2010	https://www.omg.org/spec/SysML/1.2
1.1	November 2008	https://www.omg.org/spec/SysML/1.1
1.0	September 2007	https://www.omg.org/spec/SysML/1.0

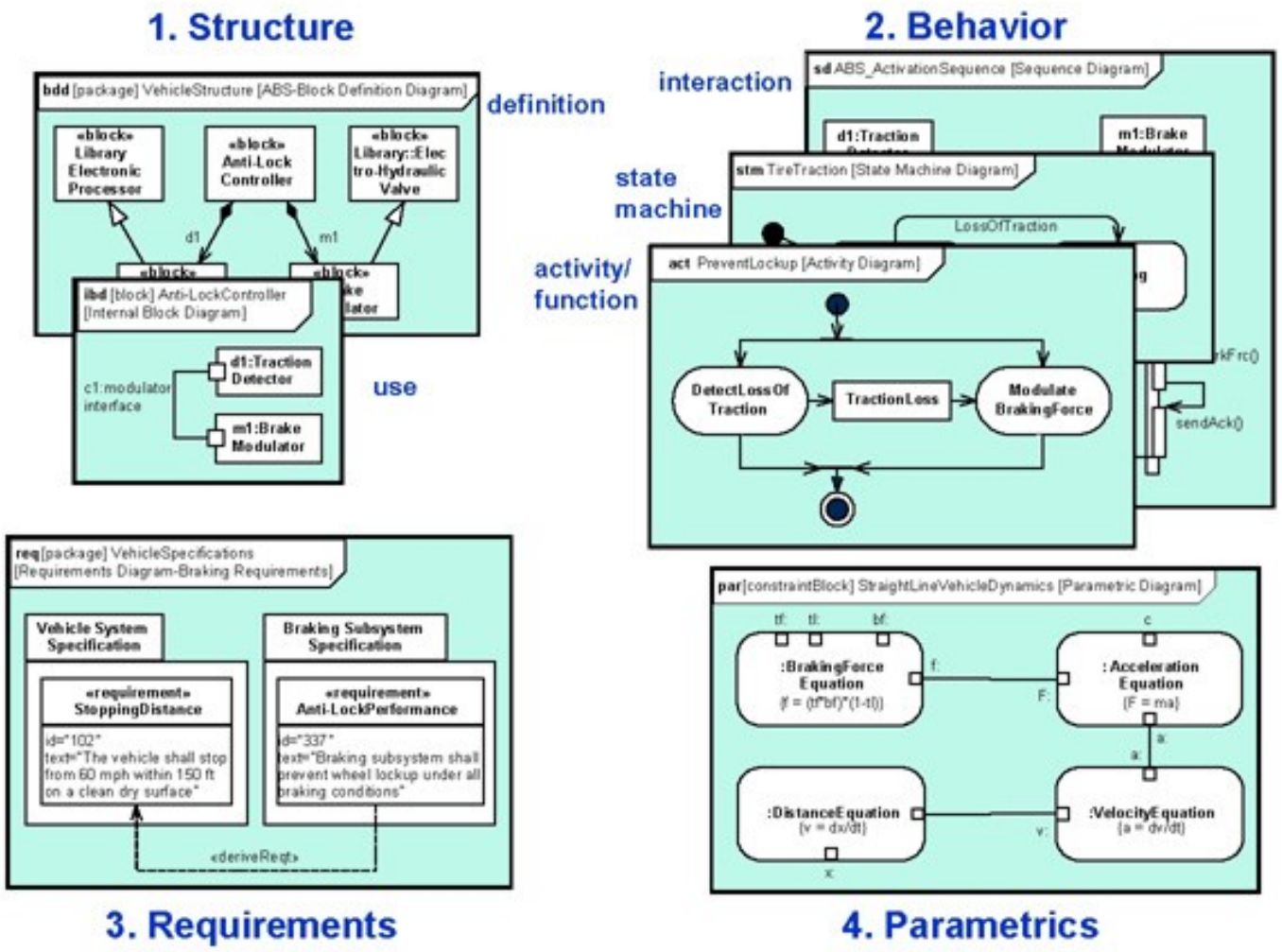
SysML

- SysML Diagram Taxonomy



OMG Document Number: formal/2015-06-03

The Four Pillars of SysML



Note that the Package and Use Case diagrams are not shown in this example, but are respectively part of the structure and behavior pillars

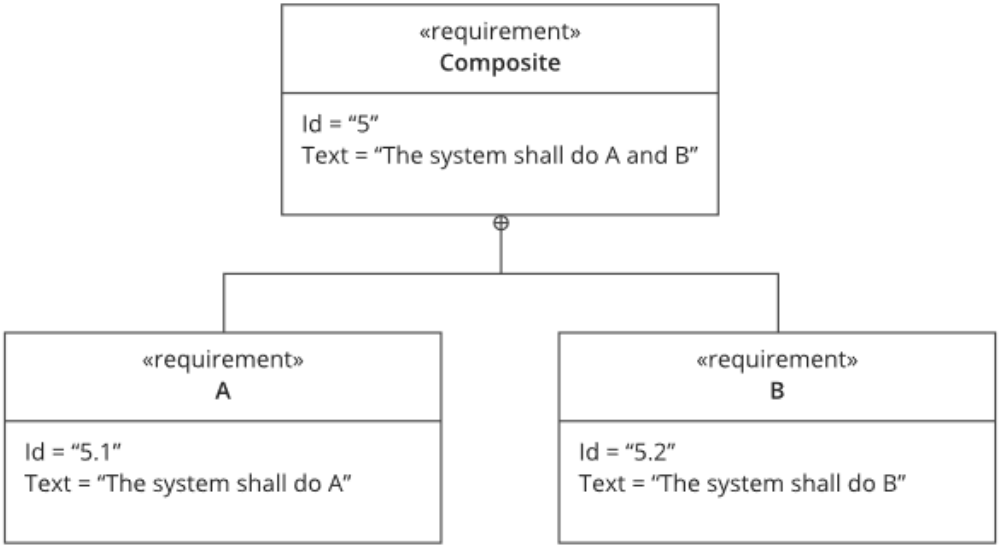
http://www.omgsysml.org/what-is-sysml.htm

SysML Requirements Diagram

- A **requirement** specifies a capability or condition that must be satisfied. A requirement can define a function that a system must perform, or a performance condition a system must achieve. A requirement can appear in other diagrams to show its relationships to other model elements.
- **Requirements** have properties and links to other elements (requirements or model elements).

https://www.microtool.de/en/what-is-a-requirements-diagram/
https://www.modeliosoft.com/en/resources/diagram-examples/requirement-diagrams.html

SysML requirements



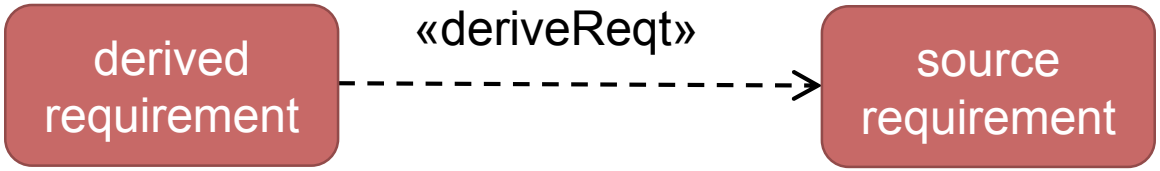
SysML Requirements Relationships	stereotype
Containment	«contain»
Trace	«trace»
Copy	«copy»
Derive	«deriveReqt»
Verify	«verify»
Refine	«refine»
Satisfy	«satisfy»

SparxSystem SysML Requirements Extensions
Extended Requirement
Functional Requirement
Interface Requirement
Performance Requirement
Phisical Requirement
Design Requirement

https://re-magazine.ireb.org/issues/2015-2-bridging-the-impossible/modeling-requirements-with-sysml/
http://sparxsystems.com/enterprise_architect_user_guide/12.1/systems_engineering/sysml_requirements.html

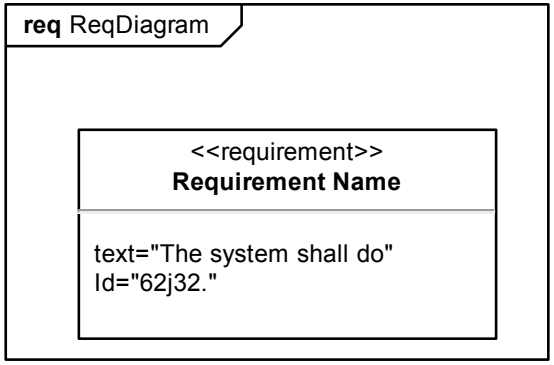
SysML requirements

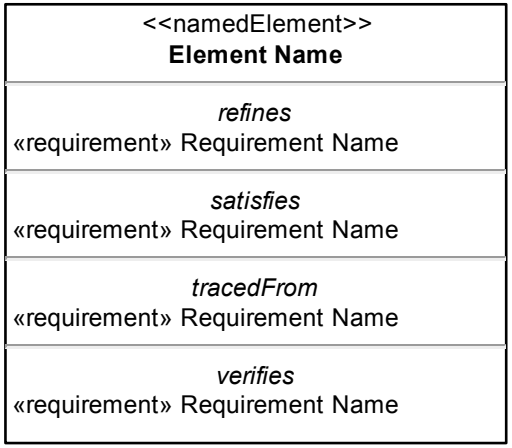
- A **derive requirement relationship** between a derived requirement and a source requirement

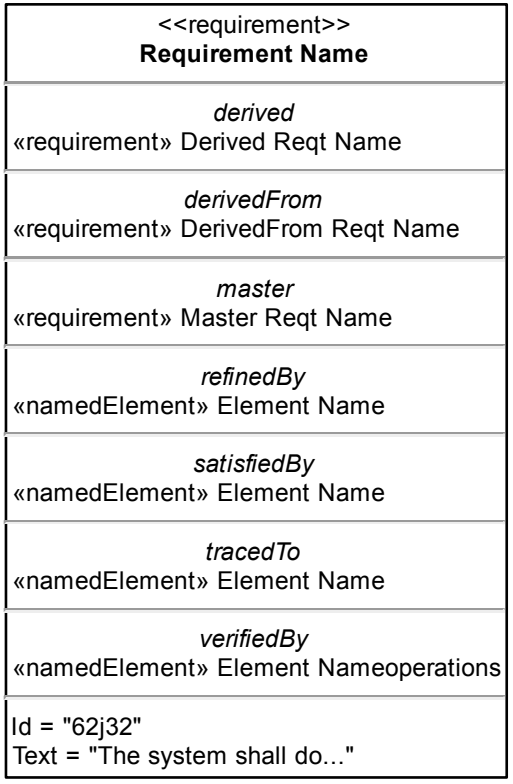


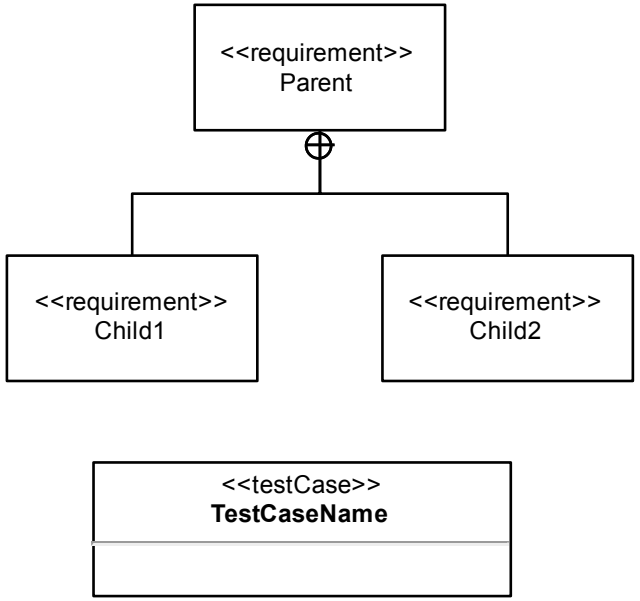
https://www.sciencedirect.com/topics/computer-science/containment-relationship

Requirements diagram

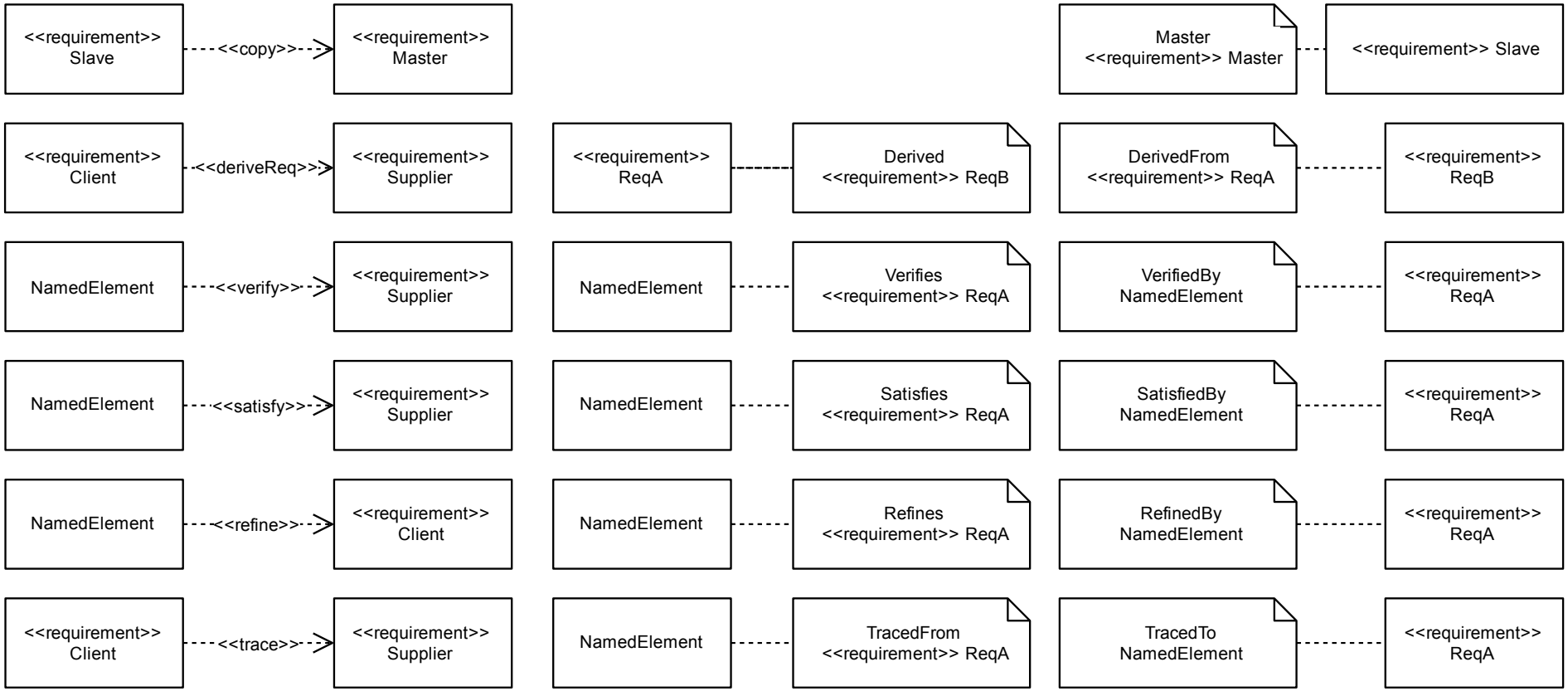








Requirements diagram



Example

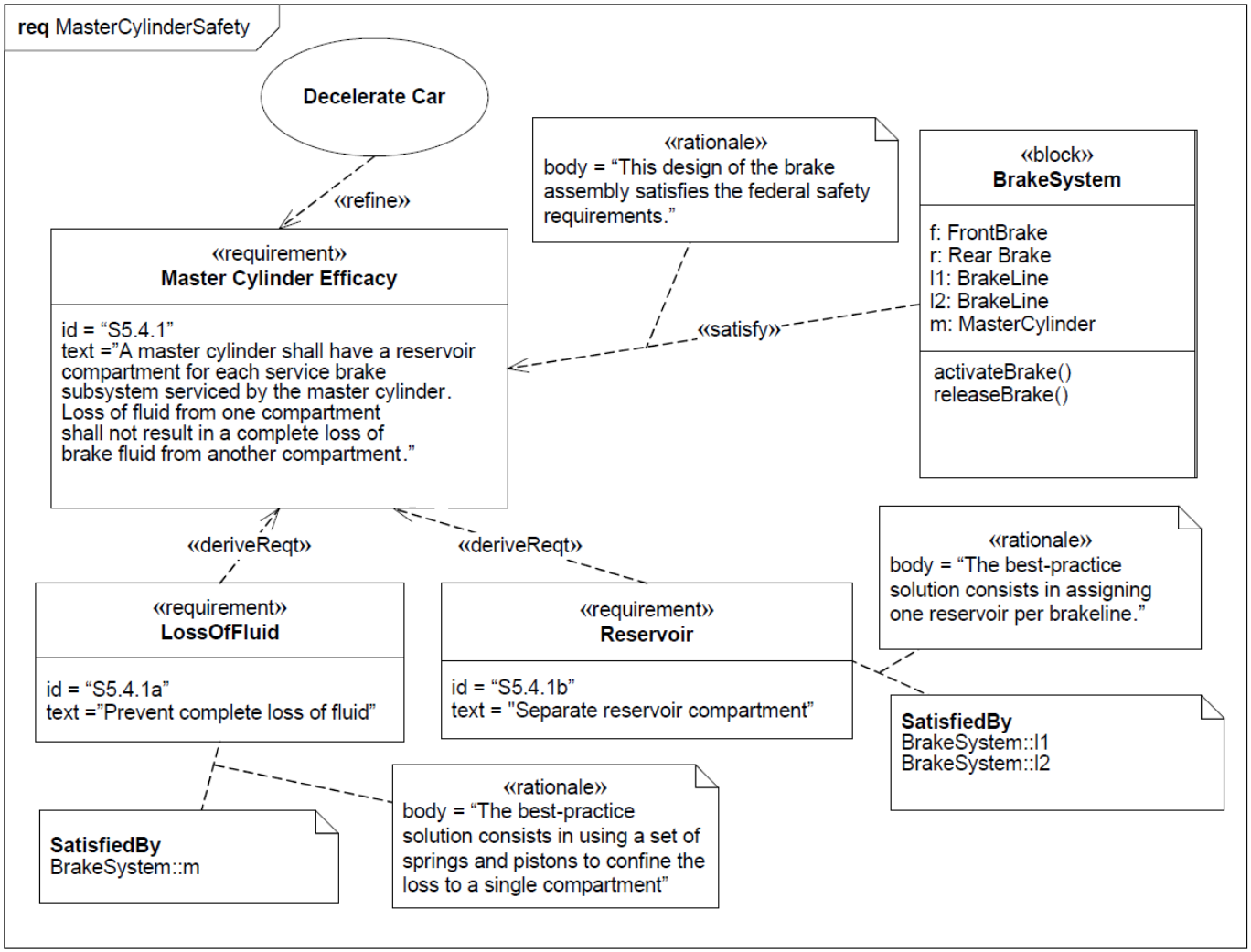


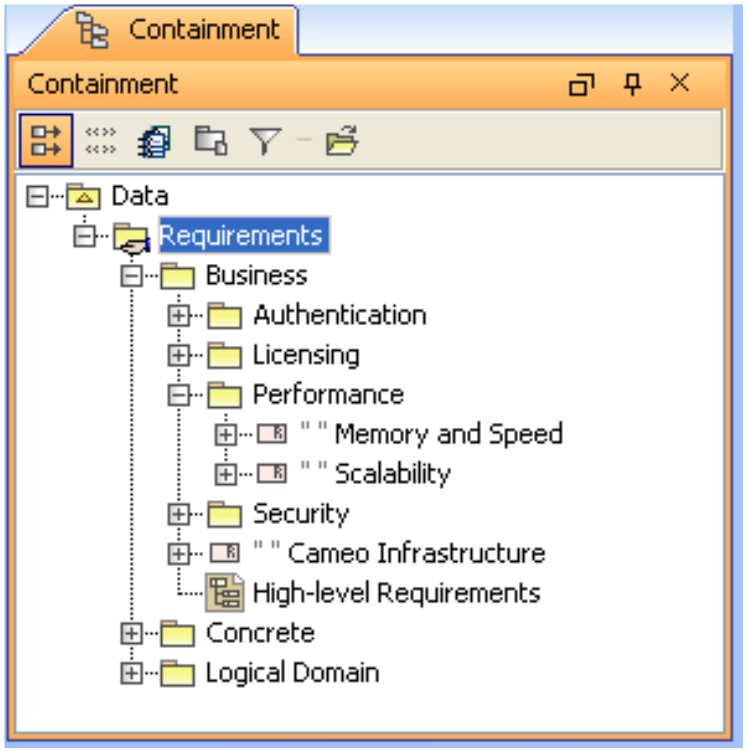
Figure 16.3 - Links between requirements and design

A Rationale documents the justification for decisions and the requirements, design, and other decisions. A Rationale can be attached to any model element including relationships. It allows the user, for example, to specify a rationale that may reference more detailed documentation such as a trade study or analysis report. Rationale is a stereotype of comment and may be attached to any other model element in the same manner as a comment.

https://www.omg.org/spec/SysML/1.5/PDF

Requirements Layers

- How to organize them?
  - User requirements
  - System requirements
  - Specification of software project
  - Functional requirements
  - Non-functional requirements
  - Domain requirements



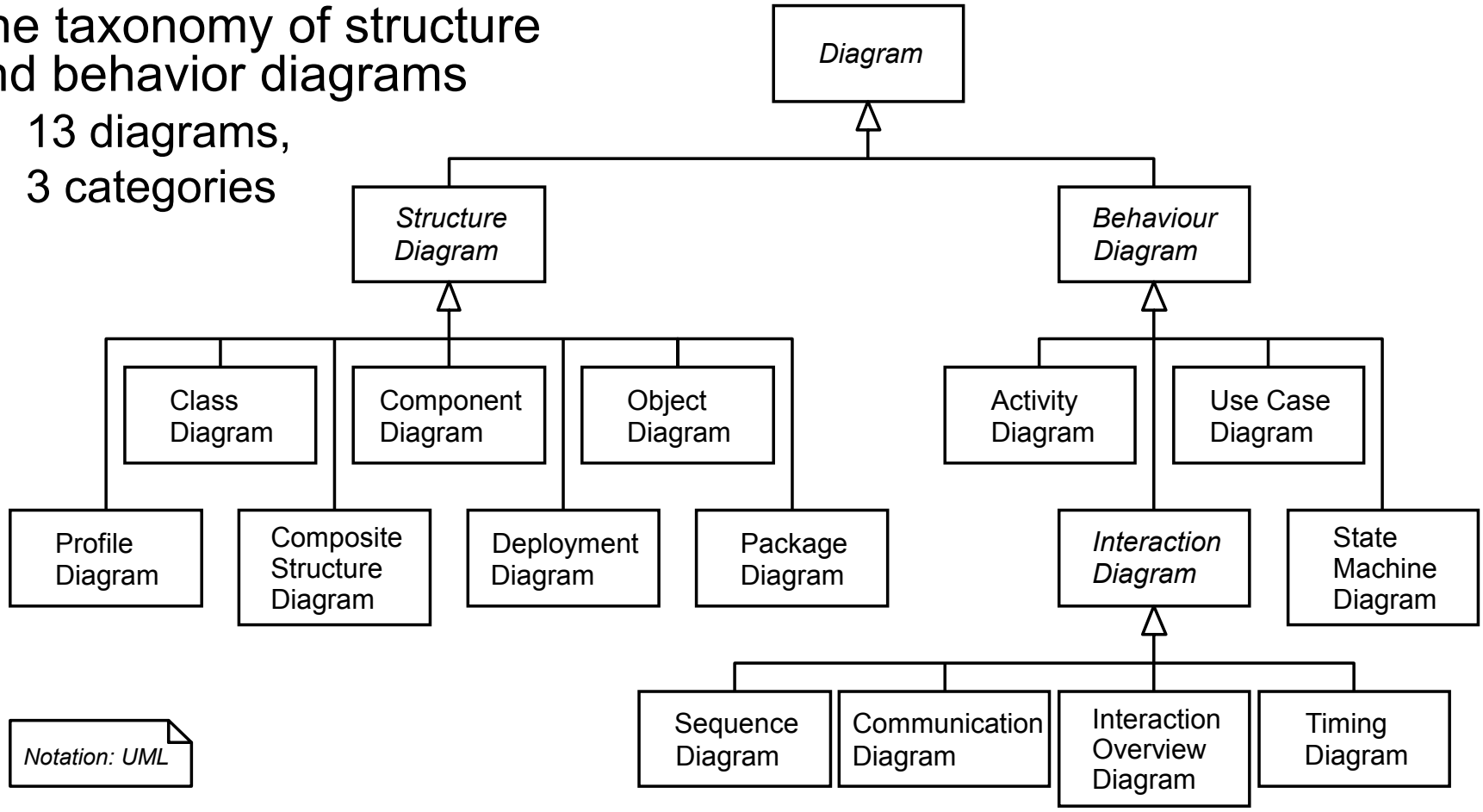
https://www.nomagic.com/mbse/images/whitepapers/Requirements_Writing_in_SysML.pdf

UML

- Helps you to specify, visualize, and document models of software systems, including their structure and design, in a way that meets all of the requirements.

UML Diagrams

- The taxonomy of structure and behavior diagrams
  - 13 diagrams,
  - 3 categories

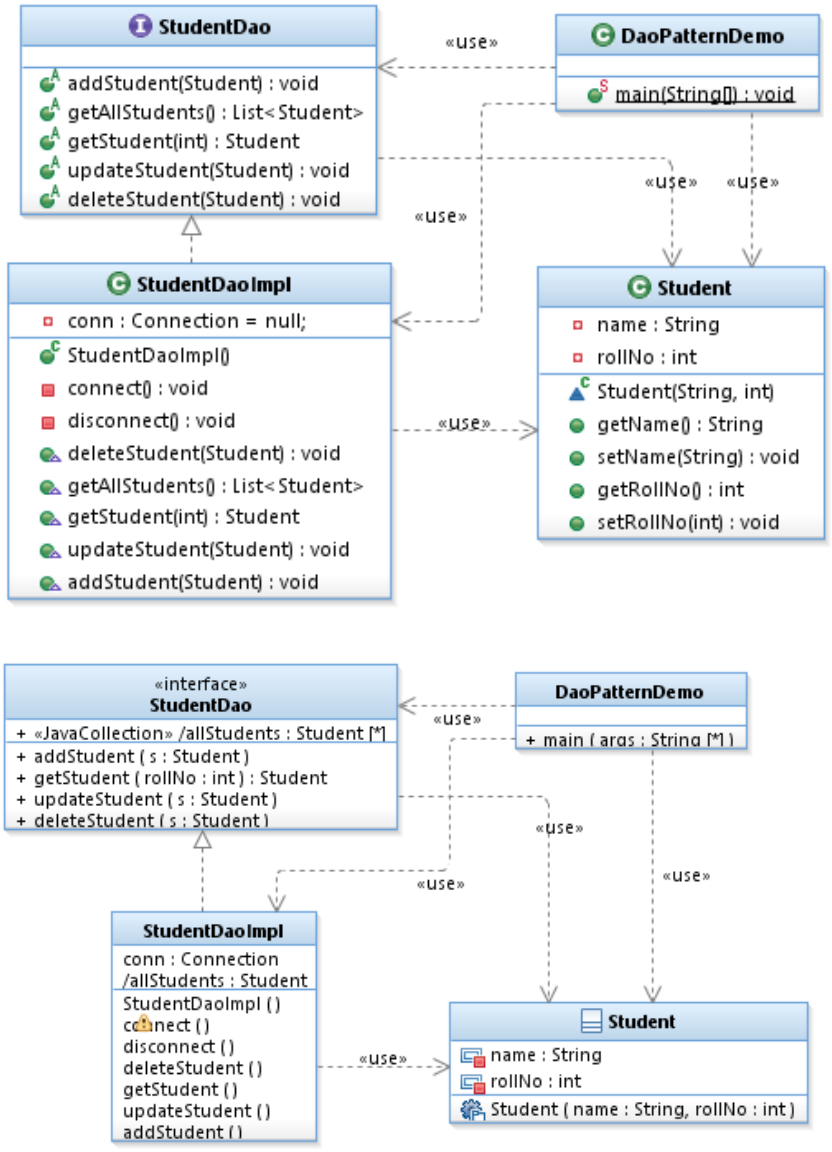


OMG Document Number: formal/15-03-01

see also:
https://www.uml-diagrams.org/uml-25-diagrams.html

Class diagram

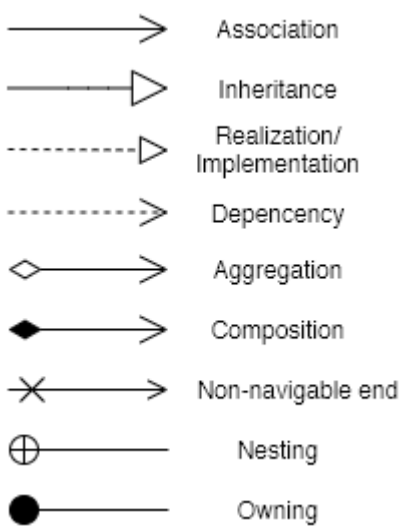
- depicts a **static view** of the model, or part of the model
- shows named **classes**, their attributes (fields), operations (or methods), relationships and associations
- each class is represented by a rectangle with three compartments: name, attributes and operation
- classes may have definitions of constraints, tagged values and be **stereotyped**
  - *Note, that UML 2.5 standard hierarchy of diagrams (see UML 2.5 diagrams overview), shows class diagrams and object diagrams as completely unrelated. Some other authoritative UML sources state that component diagrams and deployment diagrams containing only instance specifications are also special kinds of object diagrams.*
- depending on tools and chosen profiles the class diagrams can be styled in various ways (but should be consistent)

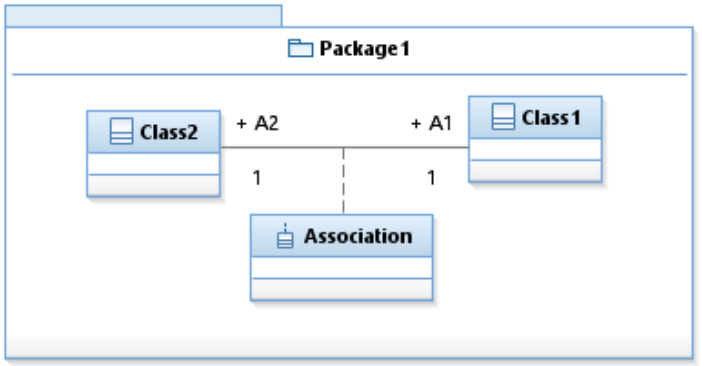


see Figure 11.29, Figure 11.26 in *Unified Modeling Language 2.5.1* (https://www.omg.org/spec/UML/2.5.1/PDF)
see *UML Class and Object Diagrams Overview* (https://www.uml-diagrams.org/class-diagrams-overview.html)
see *UML 2 Tutorial - Class Diagram* (https://sparxsystems.com/resources/tutorials/uml2/class-diagram.html)

Relationships in class diagram

- Associations are represented by lines with ends
  - *Arrows denote association end **navigability***
  - *All **class-owned** association ends are navigable (by definition)*
  - *All **association-owned** ends are not navigable (by convention)*
  - *An association with neither end marked by navigability arrows means that the association is navigable in both directions*
  - *However if the **dot notation** is used, the absence of the dot signifies ownership by the association (but this explicit end-ownership notation is not mandatory and often not supported by the modeling tools)*
  - *ends can be **named** and have declared **multiplicity***
- Association can have operations and attributes – if so, it is modelled by an **association class.**



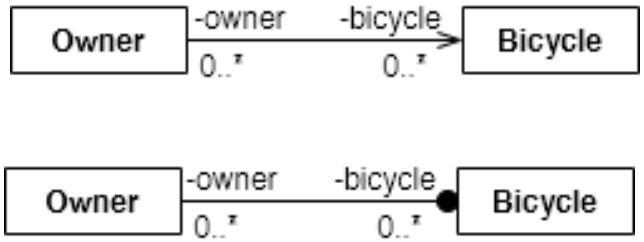


- Association is **owned** by a class if the definition of that class has a feature that is typed by the class at the opposite end.
- Association is **navigable** if there is a pass between association ends.

see: https://www.omg.org/ocup-2/documents/getting_it_right_on_the_dot.pdf

Navigation vs owning

- **Navigability** is *a run-time concept*. It identifies whether or not it is possible to *efficiently* navigate from an instance of one class to an instance of an associated class at run time. This requirement can be met by declaring class attributes referring to the objects of other classes (tables, list and other constructs are possible). A bidirectional association between two classes means that both objects on both sides know about each other.
- **Navigability** concept **differs** from **owning** concept. *Navigability in UML is rather vague and has lost much of its value for modeling. In a sense, the concept has been deprecated in UML 2 and even more so in later revisions, such as UML 2.5…… there is little real need for using those arrowheads in models.*



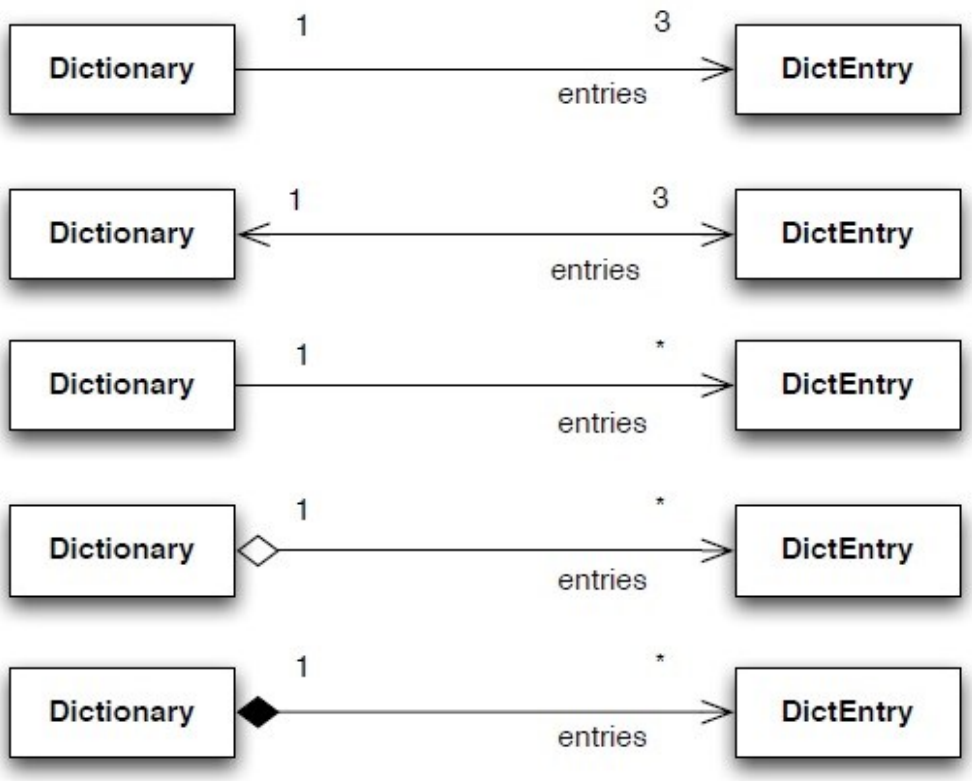
Equivalent UML 1 (top) and UML 2 (bottom) diagrams

https://www.omg.org/ocup-2/documents/getting_it_right_on_the_dot.pdf

Access modifiers:

modifier	symbol	description
public	+	any object can access such field or method anywhere in the program
private	-	accessible only from within the owning class
protected	#	accessible from within the owning class or a subclass of that class
package	~	accessible for the objects of other classes within the same package

Examples



One directional

```
public class Dictionary {
  private DictEntry d0,d1,d2; }
public class DictEntry{}
```

Bi-directional

```
public class Dictionary {
  private DictEntry d0, d1, d2;}
public class DictEntry {
  private Dictionary owner;}
```

One directional infinite multiplicity

- when the `Dictionary` would have (n) references from `DicEntry` classes, but the size is unknown,
- it's useful when `Dictionary` is kinda abstract definition

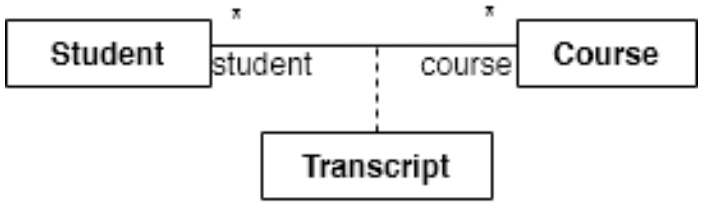
Aggregation

- is used when `Dictionary` is kinda container for `DicEntry` but with `Dictionary` object finalization (kill) the hosted `DicEntry` objects would continue their lives (keep persistent)

```
public class Dictionary {
// when dics are injected and kept alive
  private List<DicEntry> dics; }
public class DictEntry{}
```

Composition differs from Aggregation in that way, that finalization of `Dictionary` causes finalization of the hosted objects `DicEntry`

Examples



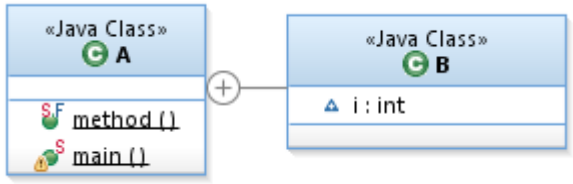

```
class Student {
    public Set<Transcript> transcripts;
}

class Transcript {
    Student student;
    Course course;
    Date subscriptionDate;
}

class Course {
    Set<Transcript> transcripts;
}
```

```
class Student {
    public Map<Student,Transcript> transcripts;
}

class Transcript {
    Student student;
    Course course;
    Date subscriptionDate;
}

class Course {
    Map<Student,Transcript> transcripts;
}
```

- UML-to-Java mapping (**forward engineering**) and Java-to-UML mapping (**backward engineering**) might be not unique. All depends on how these mappings are defined.
- In the example presented the `Set` class can be substituded by the `List` or `Map` classes without breaking the model.

Nested class

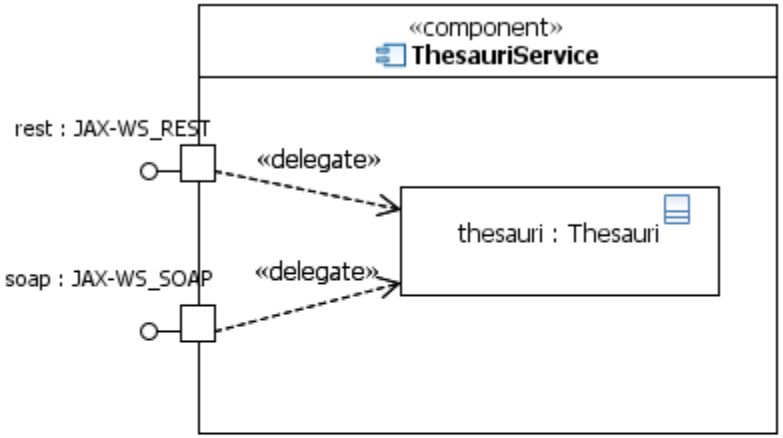
```
public class A {
  public class B {
          int i;
  }
 public final static void method(String a) {

 }
 public final static void method2(String a) {
 }
 public static void main(String ... args){
          int j;
 }
}
```

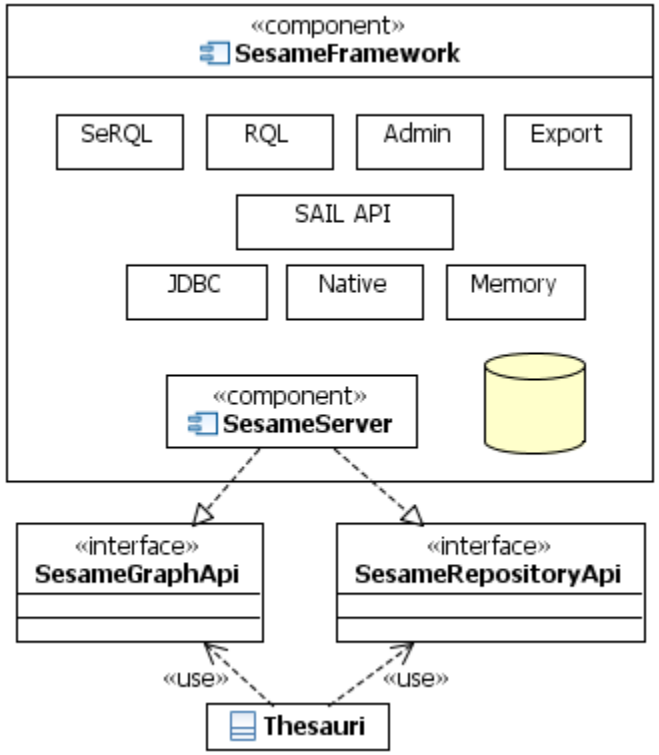


see also:

- *Essentials of modeling with Rational Software Architect - Self-paced training* (https://www.ibm.com/support/knowledgecenter/SS8PJ7_9.1.2/com.ibm.xtools.gs_using_rsx.doc/topics/c_gs_essentials_of_rsa.html) – this is the whole course on UML modeling in IBM RSA
- *UML Class Diagram Tutorial* (https://www.visual-paradigm.com/guide/uml-unified-modeling-language/uml-class-diagram-tutorial/)

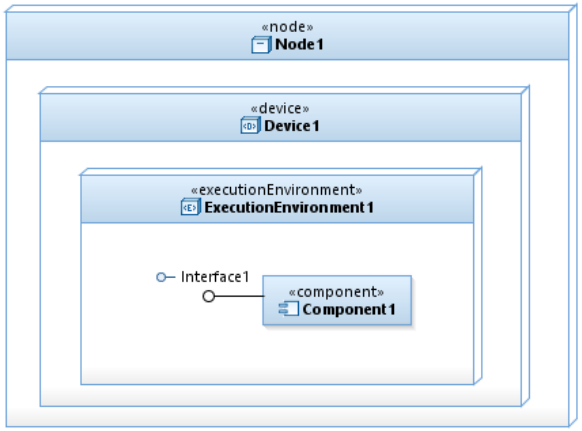
Component diagrams

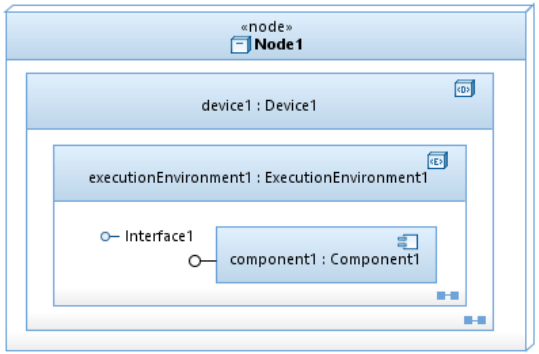


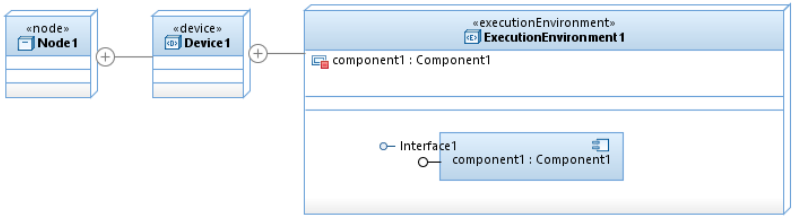


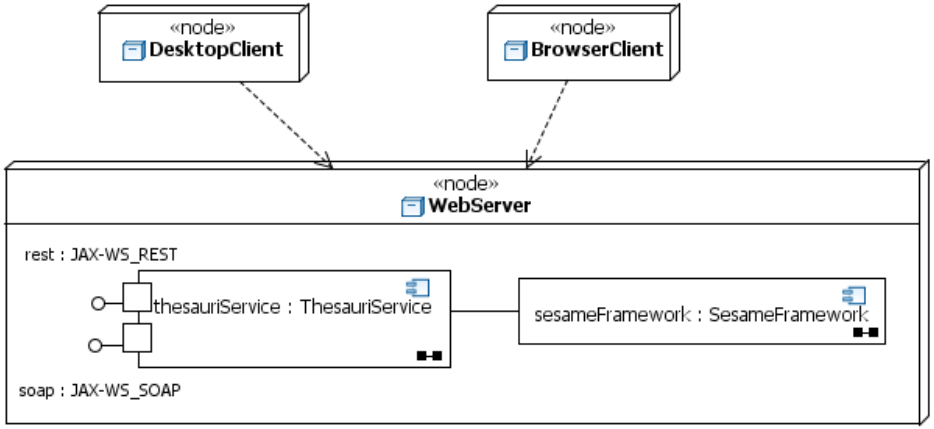
https://www.uml-diagrams.org/component-diagrams.html
https://www.uml-diagrams.org/deployment-diagrams-overview.html

Deployment diagrams



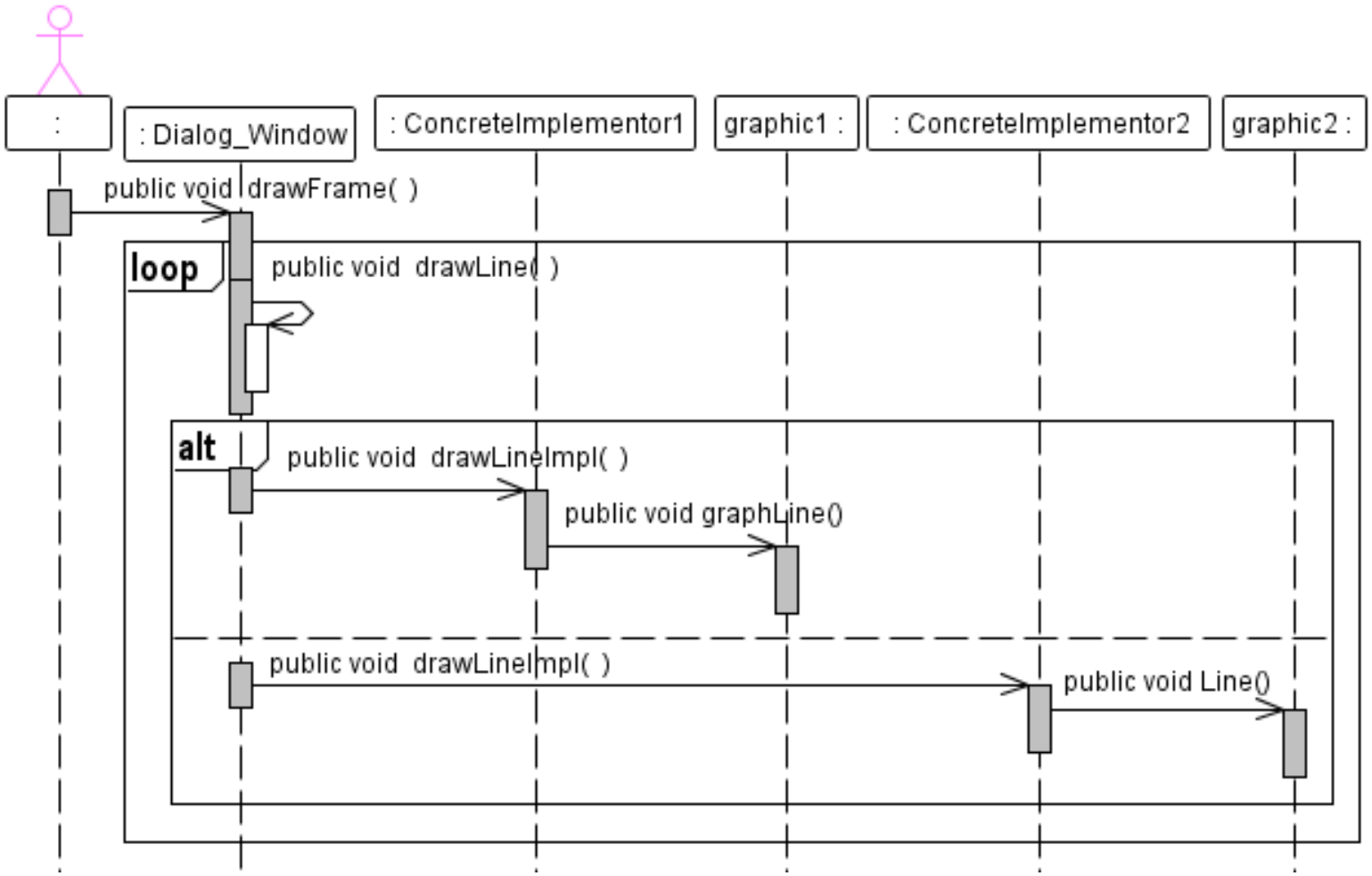






see also: https://www.uml-diagrams.org/deployment-diagrams.html

Sequence diagram

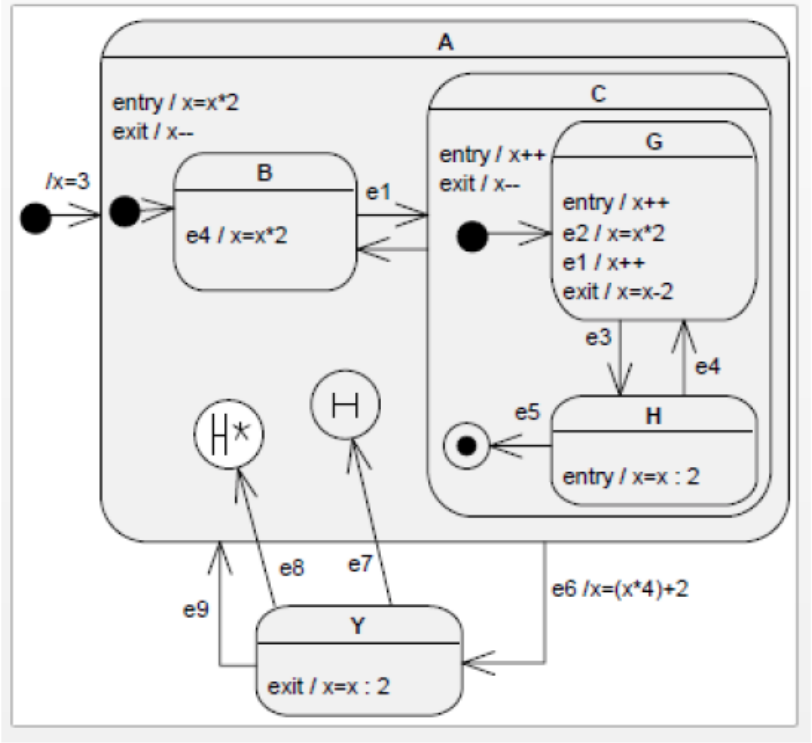


see also:
https://www.visual-paradigm.com/tutorials/how-to-draw-uml-sequence-diagram.jsp

State machine diagram

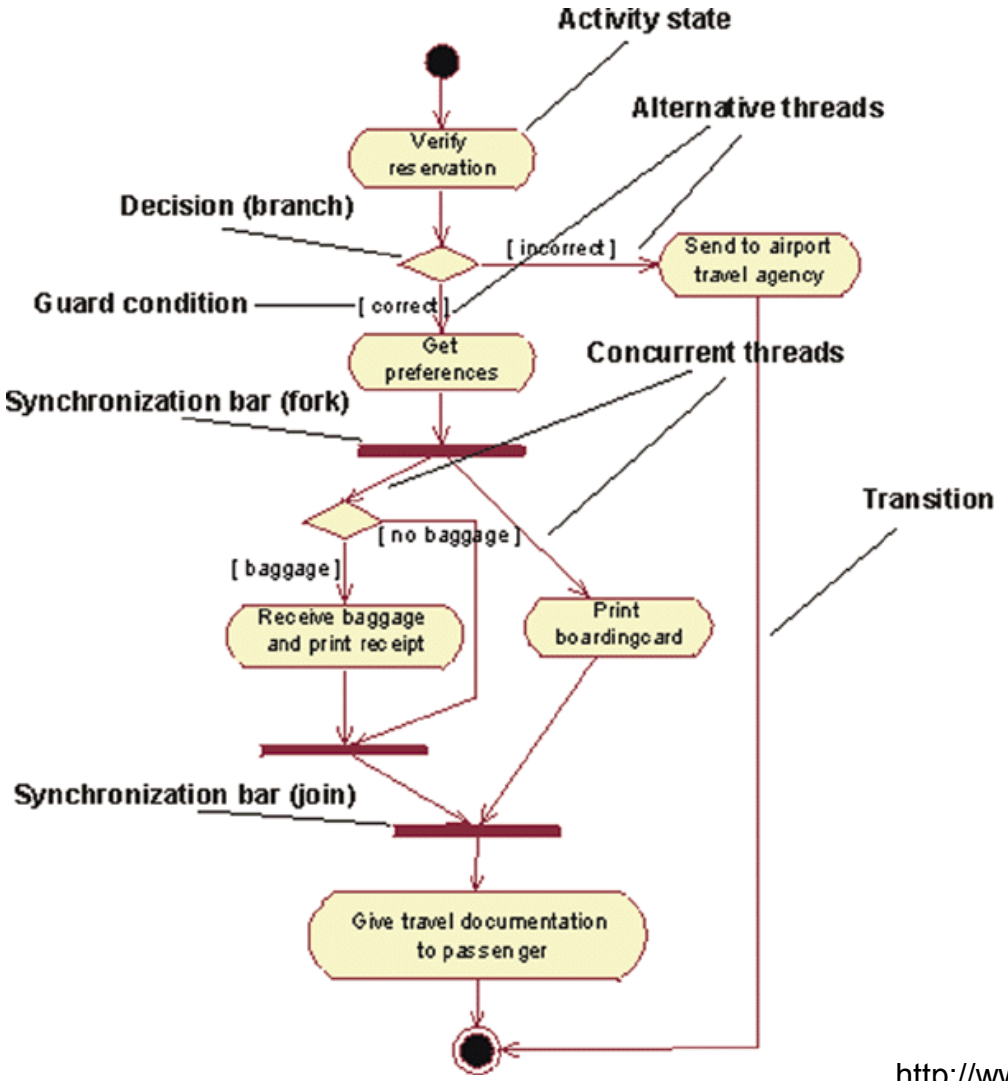
2. You are given the following state machine diagram. What is the value of x after the occurrence of the event chain

e1 e3 e4 e6 e7 e4 ?



https://stackoverflow.com/questions/34734278/what-is-an-order-of-transitions-in-state-diagram-how-to-use-history-pseudo-stat?rq=1

Activity diagram



http://www.ibm.com/developerworks/rational/library/2802.html