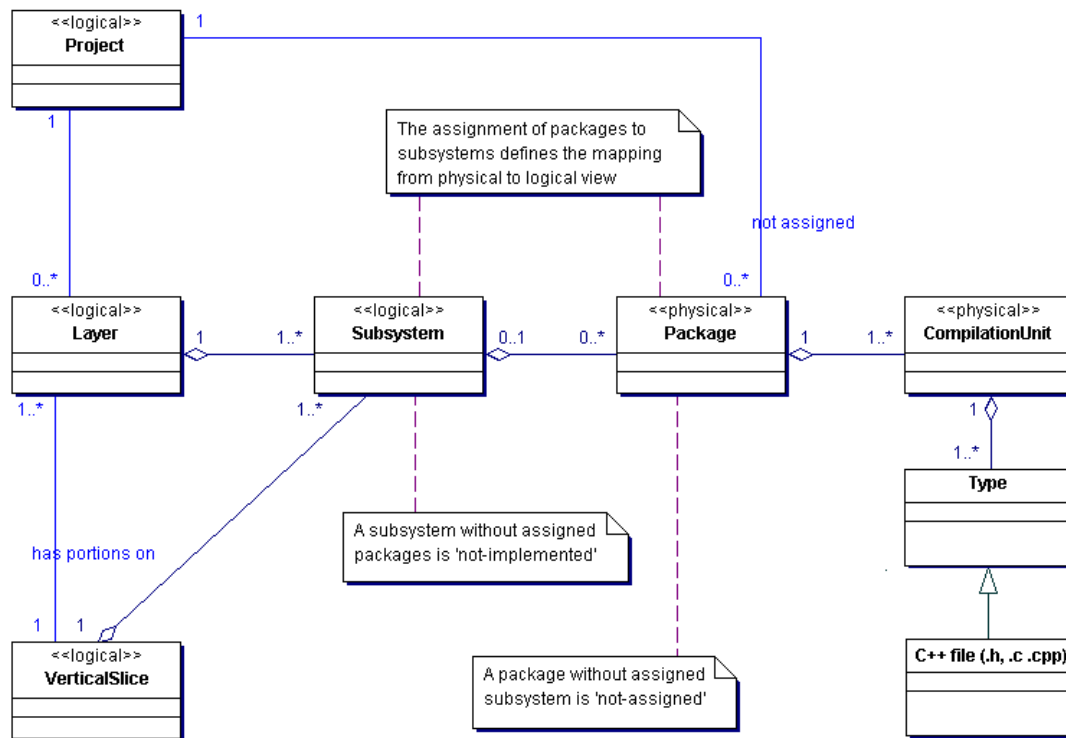


1 Structural elements

The *Dependometer* uses the following structural elements:



- Types from standard-library like ostream, string, etc. are recognized as "external types".
- The anonymous namespace is recognized as pseudopacket
- Global variables, free function, etc. are assigned to pseudopackets
- Use of Namespace-Aliases is supported.

NOTE:

The term 'Package' is used to describe either C++ namespaces provided directly in the source code, or 'namespaces' that are built by using the physical folder structure of your C++ project.

1.1 Project

Provides high-level metrics and general configuration information. The metrics are:

- average component dependency (ACD)
- cumulative component dependency (CCD)
- cumulative component dependency for balanced binary tree
- cumulative component dependency for cyclically dependent graph
- normalized cumulative component dependency (NCCD)
- number of compilation units (SIZE)
- percentage of packages with relational cohesion ≥ 1.0
- total number of assertions
- total number of efferent package dependencies
- total number of forbidden efferent package dependencies
- total number of project internal efferent package dependencies

1.2 Layer

Layers are technically driven horizontal divisions. The metrics are:

- contained subsystems and their inner layer dependencies
- afferent (incoming) and efferent (outgoing) dependencies - subsystems causing these dependencies

1.3 Subsystem

A subsystem is the result of vertically dividing the layers. The portion on each layer is called subsystem. Subsystems are considered part of the same vertical-slice (see vertical-slice) if their names are equal. For example the subsystems 'facade::customer', 'domain::customer' and 'persistence::customer' are part of a vertical slice with name 'customer'. The metrics are:

- containing layer
- containing vertical-slice
- contained packages and their inner subsystem dependencies
- afferent (incoming) and efferent (outgoing) dependencies - packages causing these dependencies
- Not implemented subsystems

Vertical-Slice

A vertical-slice is indirectly produced by defining subsystems. The metrics are:

- contained subsystems and their inner vertical-slice dependencies
- afferent (incoming) and efferent (outgoing) dependencies - subsystems causing these dependencies

1.4 Package

Represents directly C++ namespaces or the physical folder structure – to be considered a package there has to be at least one C++ file. Allowed dependencies are read from package descriptions (not from the configuration xml file). The metrics are:

- containing subsystem
- contained compilation units and their inner package dependencies
- afferent (incoming) and efferent (outgoing) dependencies - compilation units causing these dependencies
- depth of package hierarchy
- not to subsystems assigned packages

1.5 Metrics common for the elements layer, subsystem and package

- abstract types (Na)
- abstractness (A)
- accessible types
- afferent (incoming) dependencies
- afferent coupling (Ca)
- assertions
- average component dependency (ACD)
- cumulative component dependency (CCD)
- depends upon elements (incl. self)
- distance (D)
- efferent (outgoing) dependencies (internal and external)
- efferent coupling (Ce)
- external type relations
- forbidden efferent (outgoing) dependencies
- instability (I)
- internal type relations
- number of compilation units (SIZE)
- relational cohesion (Rc)
- types (Nc)
- no physical project internal dependencies detected
- project internal/external
- abstract/concrete
- accessible/not accessible
- check actual dependencies against explicitly allowed dependencies
- usage detection of explicitly allowed dependencies
- cycles and cumulated participation of dependencies
- levelization

1.6 Compilation Unit

Represents a C++ file. (e.g. a “.c” or “.cpp” file as well as an accompanying header “.h” file). The metrics are:

- containing package
- contained types and their inner compilation unit dependencies
- abstract types
- accessible types
- afferent (incoming) dependencies
- assertions
- concrete types
- depends upon elements (incl. self)
- efferent (outgoing) dependencies (internal and external)
- forbidden efferent (outgoing) dependencies
- more external than internal relations per package exist
- the most external relations exist with package
- total external package relations
- total internal package relations
- types
- no physical project internal dependencies detected
- abstract/concrete
- accessible/not accessible
- cycles and cumulated participation of dependencies
- levelization

1.7 Type

Represents a type definition (a C++ class). The metrics are:

- Containing compilation unit
- abstract/concrete
- accessible/not accessible
- afferent (incoming) dependencies
- efferent (outgoing) dependencies (internal and external)
- forbidden efferent (outgoing) dependencies
- interface
- no physical project internal dependencies detected
- number of childs (NOC)
- depth of class inheritance
- depth of interface inheritance
- cycles and cumulated participation of dependencies
- levelization

1.8 Thresholds

It is possible to define upper and lower thresholds. In case of a violation you receive a notification in the shell and these thresholds are explicitly marked in the reports.

Overview of (non-trivial) calculated metrics

<i>shortcut</i>	<i>Name</i>	<i>description</i>	<i>reference</i>
Ca	Afferent coupling	Number of classes outside that depend upon classes inside (i.e. incoming dependencies)	[PAP]
Ce	Efferent coupling	Number of classes outside that classes inside depend upon. (i.e. outgoing dependencies)	[PAP]
I	Instability	Ce/(Ca+Ce) This is a metric that has the range: [0,1]. If there are no outgoing dependencies, then I will be zero and the package is stable. If there are no incoming dependencies then I will be one and the package is instable.	[PAP]
Nc	Number of classes	Number of classes	[PAP]
Na	Number of abstract classes	Number of abstract classes (abstract class, interface)	[PAP]
A	Abstractness	Na/Nc The A metric has a range of [0,1]. A value of zero means that the package contains no abstract classes. A value of one means that the package contains nothing but abstract classes.	[PAP]
D	Distance	A+I-1 (as absolute value - modified!), it has a range of [-1,1]. Negative means „zone of pain“ and positive means „zone of uselessness“.	[PAP] modified
Rc	Relational coupling	Number of internal relations / Nc	[AUP]
SIZE	Number of compilation units	Number of compilation units	[LSD]
	Component dependency	Number of elements an element depends directly or indirectly (transient) upon.	[LSD]
CCD	Cumulative component dependency	Sum over all components Ci in a subsystem of the number of components needed in order to test each Ci incrementally.	[LSD]
ACD	Average component dependency	CCD/Number of components	[LSD]
CCD _{bbt}	CCD of a tree like system	CCD of a tree like system	[LSD]
NCCD	Normalized cumulative component dependency	CCD/CCD_{bbt}	[LSD]
CCD _{cdg}	CCD for a cyclically dependent graph	N*N (N=Number of components)	[LSD]
	Cumulated dependencies	Like the Component dependency, but calculated for other elements like Layer, Subsystem and Package	[LSD]
	Level	Levelization	[LSD]