**LEGO PLUS Architecture guidelines**

***Software Architecture Design – System – Addendum***

Rev 1.0

Sommario

[Revision History 3](#_Toc533073013)

[References 3](#_Toc533073014)

[Terms and Definitions 3](#_Toc533073015)

[Notes 4](#_Toc533073016)

[1. Scope 4](#_Toc533073017)

[2. General guidelines 4](#_Toc533073018)

[1.1 Architecture 4](#_Toc533073019)

[1.2 Coding and usage 5](#_Toc533073020)

[1.3 Project archiving 6](#_Toc533073021)

[1.3.1 Filename and directories 6](#_Toc533073022)

# Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Rev** | **Date** | **Author** | **Description** |
| 0.0 | 19/12/2018 | E.L.Minthe | First issue |
| 1.0 | 20/12/2018 | E.L.Minthe | Minor changes to ID format; added [COD\_22] and [COD\_23]. |

# References

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Title** | **Rev** | **Date** |
|  | FW\_API\_Lego Plus Project.docx | X3 | 22/02/2013 |

# Terms and Definitions

|  |  |
| --- | --- |
| **Terms** | **Definition** |
| GRx | Golden Rule number X |
| [DIR\_x] | Directories and Filename rules |
| [ARC\_x] | Architecture rules |
| [COD\_x] | Coding standard rules |
| Cluster  or  Subsystem | Network of software components that communicates each other (see [1], par. 5) |
| Manager | Communication interface between the subsystem that it manages and other subsystems, via a software component called *Director*. |
| Director | Manages the routing of notifications between Managers. |
| Manager/Director notification system | It is a software strategy to implement the subsystem decoupling (called also *Mediator-Collegues* design pattern). |
| HAL | Hardware Abstraction Layer |
| Driver Interface | A *driver* is a reusable part of software that allows the access to a physical device.  A *driver* has the task to guarantee access to a physical device but does not make any decision about its usage. |

# Notes

Not covered in this revision:

|  |  |
| --- | --- |
| **From** [1] | **Chapters and paragraphs** |
| Chapters that need more insights | § 5.2, 5.5, 5.6, 5.7, 5.8, 5.9, 5.10, 6.2, 6.3, 6.4, 6.5, 6.6, 7.1.1, 7.1.5 |
| Void chapters | § 7.1.2, 7.1.3 |

# Scope

This document is intended to provide a set of rules to implement an effective and robust LEGO+ component.

All the rules derive from the architecture specification *LEGO-PLUS PROJECT Reusable Software Components for Hand Held Scanners*, 2013, Rev X3 (see [1]).

# General guidelines

## Architecture

|  |  |  |
| --- | --- | --- |
| **Rule ID** | **Rule** | **Reference** |
| [ARC\_01] | Keep the legacy code separated from the newly designed class by using adapters and wrappers. | [1], par 4.1 |
| [ARC\_02] | Multiple inheritance is prohibited except for interfaces | [1], par 4.2 |
| [ARC\_03] | The keyword “new” and “delete” shall be overloaded in order to allow an adequate control of the memory usage. | [1], par 4.2 |
| [ARC\_04] | Architecture topology shall be a Star (Point-MultiPoint (P-MP)) network of smaller Star(P-MP) Networks (called *Clusters*), every P-MP network has a central connection point and all the central connection points of the smaller networks (called Managers) are connected to a further central connection point (a component call Director). | [1], par 5 |
| [ARC\_05] | A *subsystem* shall realize an autonomous slice of the whole system in terms of features and functionalities.  So the subsystem does not rely on the presence of the other subsystems. | [1], par 5, 5.3 |
| [ARC\_06] | A *subsystem* shall be connected to the rest of the system through a special component called *Manager* through the mechanism called *Manager/Director notification system*. | [1], par 5 |
| [ARC\_07] | “Static” components shall be used in the connection between Manager and components in his subsystem and between different components in the same subsystem. | [1], par 5 |
| [ARC\_08] | To “decouple” Managers shall be used the Mediator Pattern where the *Colleague* is the *Manager* and the *Mediator* is the *Director*. | [1], par 5.3 |
| [ARC\_09] | *Director* is the only responsible to dispatch the information to the rest of the system. | [1], par 5.3 |
| [ARC\_10] | The Manager must manage only the functionalities for which he is responsible. | [1], par 5.3 |
| [ARC\_11] | Are allowed functionalities implemented by “non-lego” code. | [1], par 5.3 |
| [ARC\_12] | *Director* is a singleton pattern. | [1], par 7 |
| [ARC\_13] | Manage of Input notification - Each input to the Manager is pushed in a queue and the transitions are executed popping the input from the queue. | [1], par 7.1.1 |
| [ARC\_14] | Manage of Input notification – It is allowed the use of a semaphore instead of a queue. | [1], par 7.1.1 |
| [ARC\_15] | The Manager communicates with the director using the Notify() method | [1], par 7.1.4 |
| [ARC\_16] | The Manager who send a notification to the Director should not be aware if the destination Manager is present or not. | [1], par 7.1.4 |
| [ARC\_17] | Components that are not *managers* can send notifications to the *Director* and/or requests to the *Managers*. | [1], par 7.1.4 |

## Coding and usage

|  |  |  |
| --- | --- | --- |
| **Rule ID** | **Rule** | **Reference** |
| [COD\_01] | Each component should have one and only one interface in the API folder to be included when the component is used. | [1], par 3.1 |
| [COD\_02] | Each include file must not contain any other include file that is not necessary for its own compilation | [1], par 3.1 |
| [COD\_03] | If you want to use a C file with its own defining header that was intended for C compilers, you can include the header in extern "C" brackets:  extern "C" {  #include "header.h"  } | [1], par 3.2.1 |
| [COD\_04] | Use the \_\_cplusplus automatic definition in order to discriminate the part of the include file visible to C++ and C.  ... // section visible to both C and C++  #ifdef \_\_cplusplus  ... // section visible to C++ only  #endif  #ifndef \_\_cplusplus  ... // section visible to C only  #endif | [1], par 3.2.3 |
| [COD\_05] | Exclude standard “exceptions”. | [1], par 4.2 |
| [COD\_06] | Typing Run Time (RTTI) is to be excluded. | [1], par 4.2 |
| [COD\_07] | Templates are deprecated. | [1], par 4.2 |
| [COD\_08] | Encapsulation - Static variables declaration | [1], par 4.2 |
| [COD\_09] | Encapsulation - Keep everything private by default | [1], par 4.2 |
| [COD\_10] | Encapsulation - PUBLIC and PROTECTED should be only for the exposed interface. | [1], par 4.2 |
| [COD\_11] | Encapsulation - “friend” class are forbidden | [1], par 4.2, GR4 |
| [COD\_12] | Encapsulation - “global” variables are not allowed | [1], par 4.2 |
| [COD\_13] | Encapsulation - For data access only “get/set” methods are allowed. | [1], par 4.2 |
| [COD\_14] | Encapsulation - all member variables must be private or protected. Access to variables must be made through public methods | [1], par 4.2, GR1 |
| [COD\_15] | Interfaces - At least one of the methods of the ‘interface’ class should be a pure virtual function, so the ‘interface’ class cannot be instantiated | [1], par 4.2, GR2 |
| [COD\_16] | Inheritance – Multiple inheritance is forbidden. It is permitted in case of inheritance from multiple interfaces. | [1], par 4.2, GR3 |
| [COD\_17] | Include deadlock – Forbidden.  Use the *forward declaration* to prevent deadlock. | [1], par 4.2, GR5 |
| [COD\_18] | Use prefix and a valid name in the use of *enums* and *defines*. | [1], par 4.2, GR6 |
| [COD\_19] | *Device Interface* shall use Hardware Abstraction Layer to access the physical device. | [1], par 6 |
| [COD\_20] | A *Device Interface* shall expose a standard interface to the higher software levels. | [1], par 6 |
| [COD\_21] | A component (client) must access to the physical device using the *Driver Interface.* | [1], par 6 |
| [COD\_22] | A class shall be documented. Brief description of its behavior. | - |
| [COD\_23] | A class method or a function shall be documented with: brief description, in parameters, out parameters, return value, errors. | - |

## Project archiving

### Filename and directories

|  |  |  |
| --- | --- | --- |
| **Rule ID** | **Rule** | **Reference** |
| [DIR\_01] | Given a Lego component named “ComponentName” the interface is described by a single include file shall be <ComponentName>\_API.h. | [1], par 2.1 |
| [DIR\_02] | Each include file must be protected against multiple inclusions using:  #ifndef COMPONENTNAME\_API\_\_H\_INCLUDED  #define COMPONENTNAME\_API\_\_H\_INCLUDED  ...  #endif // COMPONENTNAME | [1], par 2.1 |
| [DIR\_03] | The compile-time options of a component are described in a single include file with name <ComponentName>\_Target.h.  Each time that the component is customized for a specific target this file must be used. | [1], par 2.1 |
| [DIR\_04] | The component may be distributed on one or more source files. ComponentName.c/cpp or if they are in multiple files: ComponentName\_MainPart.c/cpp and ComponentName\_Subsystem.c/cpp  **EXCEPTION**: long filenames are allowed if they are meaningful. | [1], par 2.1 |