



Nerys



Mesulog

Nerys Group companies

Code modulaire sous LabVIEW

Utilisation des Iib, Ivlib, Ivclass, Ivlibp, dll

Le présentateur



- Amaury LAURENT
- Développeur LabVIEW depuis 2014
- Certifications : CLED et CLA
- Responsable projets systèmes embarqués

ARCHITECT



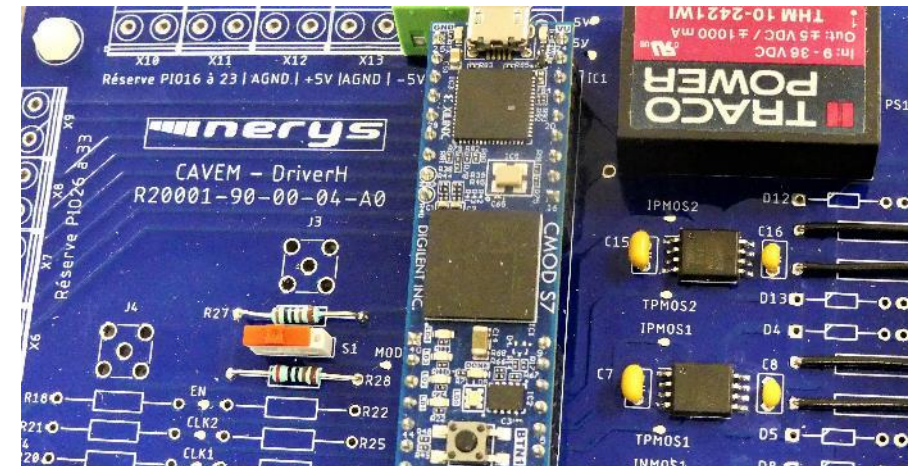
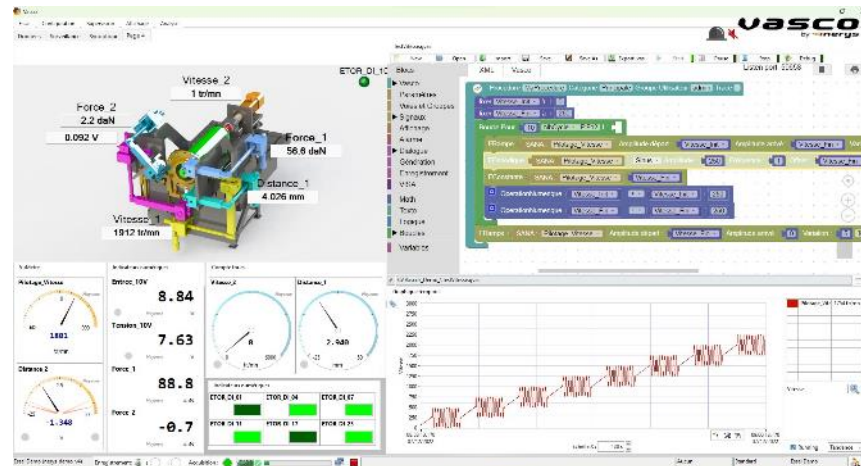
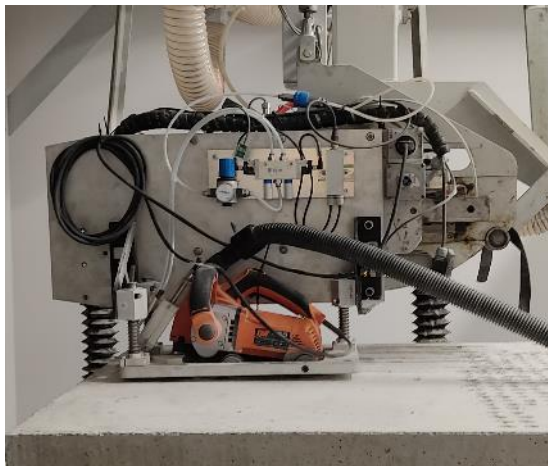
EMBEDDED
DEVELOPER



NERYS en bref



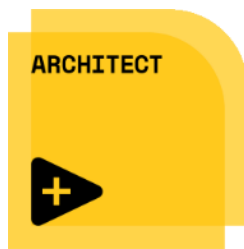
- Fondée en 2007 à Gardanne (Bouches du Rhône), une équipe de 15 personnes
- Concepteur de bancs de tests, de caractérisation et de systèmes de mesure :
 - Bureaux d'études spécialisés en électrotechnique, électronique et mécanique
 - Pôle développement logiciels, avec une expertise LabVIEW.
- NERYS propose une solution flexible de pilotage de moyens d'essais appelée "Vasco"
 - Configurable
 - Historisation longue durée
 - Personnalisable sous LabVIEW
 - Scénarios
 - Compatible avec cibles temps réelles NI
 - Gestion de la calibration, traçabilité



NERYS GROUP en bref



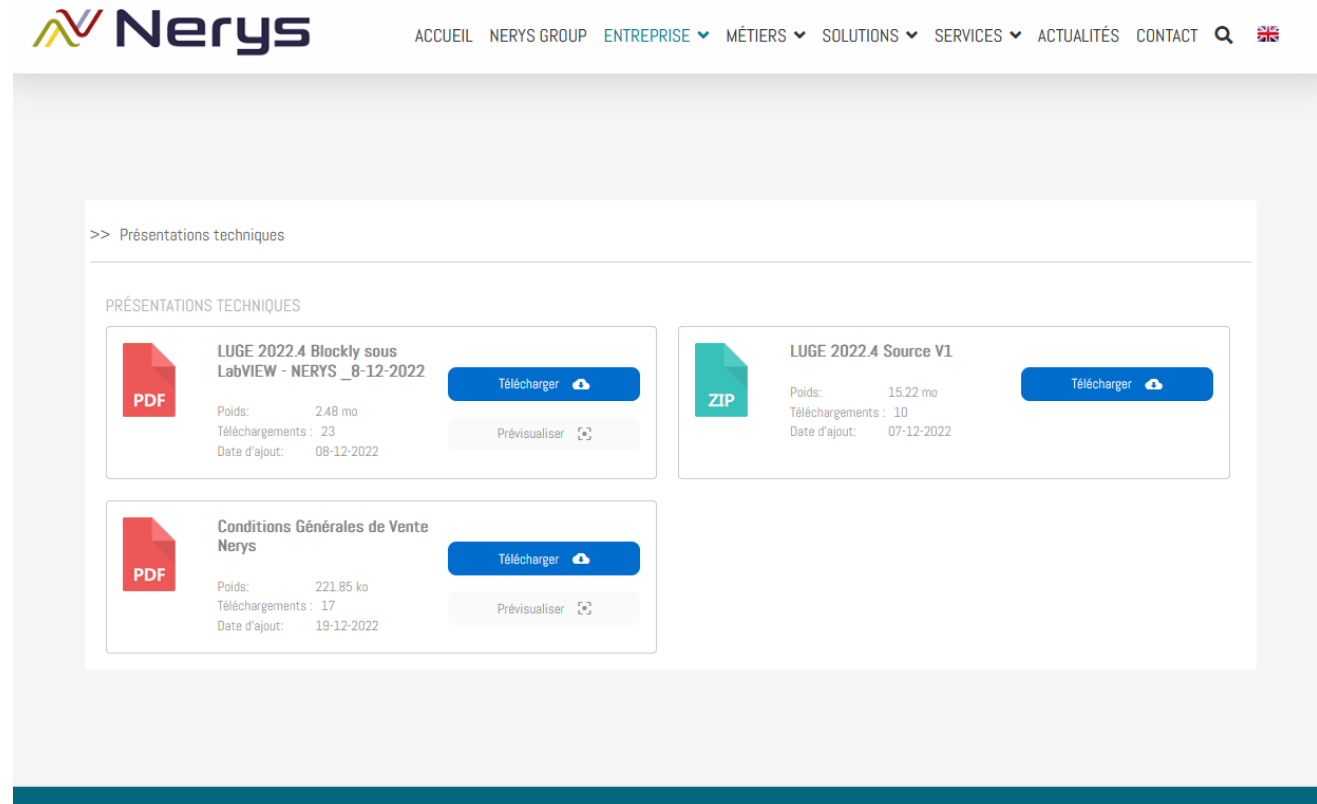
- Depuis mai 2022 : NERYS + MESULOG → NERYS GROUP
 - Gardanne (13), Moirans (38), La Rochette (73)
- Compétences logicielles NI :
 - NI LabVIEW® (Windows, RT, FPGA)
 - NI TestStand
 - NI VeriStand
- Partenaire National Instruments
- Développeurs certifiés LabVIEW et TestStand
- Sur le web :
 - <https://nerysgroup.com/fr/>
 - <https://www.mesulog.fr/>
- Nous contacter :
 - contact@nerysgroup.com
 - contact@mesulog.fr





- Les sources des exemples ainsi que cette présentation sont disponibles sur le site :

<https://nerysgroup.com/fr/entreprise/documentation>



The screenshot shows the Nerys website's documentation page. At the top, the Nerys logo is on the left, and a navigation menu includes ACCUEIL, NERYS GROUP, ENTREPRISE (with a dropdown arrow), MÉTIERS (with a dropdown arrow), SOLUTIONS (with a dropdown arrow), SERVICES (with a dropdown arrow), ACTUALITÉS, CONTACT, a search icon, and a language selector (UK flag). Below the navigation, the page title is '>> Présentations techniques'. The main content area is titled 'PRÉSENTATIONS TECHNIQUES' and contains three download cards. Each card has a file icon (PDF or ZIP), a title, a 'Télécharger' button with a download icon, and a 'Prévisualiser' button with a refresh icon. The first card is for 'LUGE 2022.4 Blockly sous LabVIEW - NERYS _8-12-2022' (PDF, 2.48 mo, 23 downloads, added 08-12-2022). The second card is for 'LUGE 2022.4 Source V1' (ZIP, 15.22 mo, 10 downloads, added 07-12-2022). The third card is for 'Conditions Générales de Vente Nerys' (PDF, 221.85 ko, 17 downloads, added 19-12-2022).

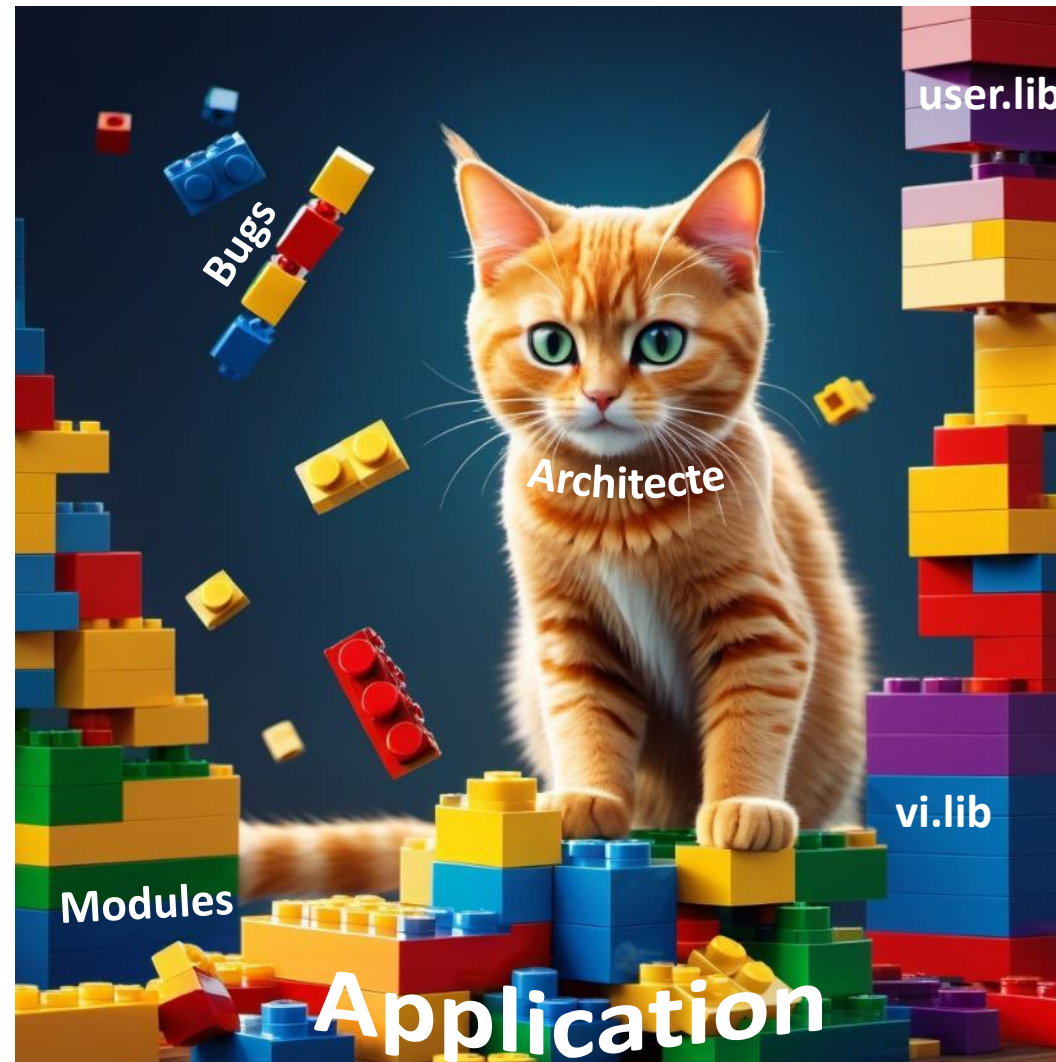


Pour commencer, un peu de théorie

Demandons à Chat GPT...



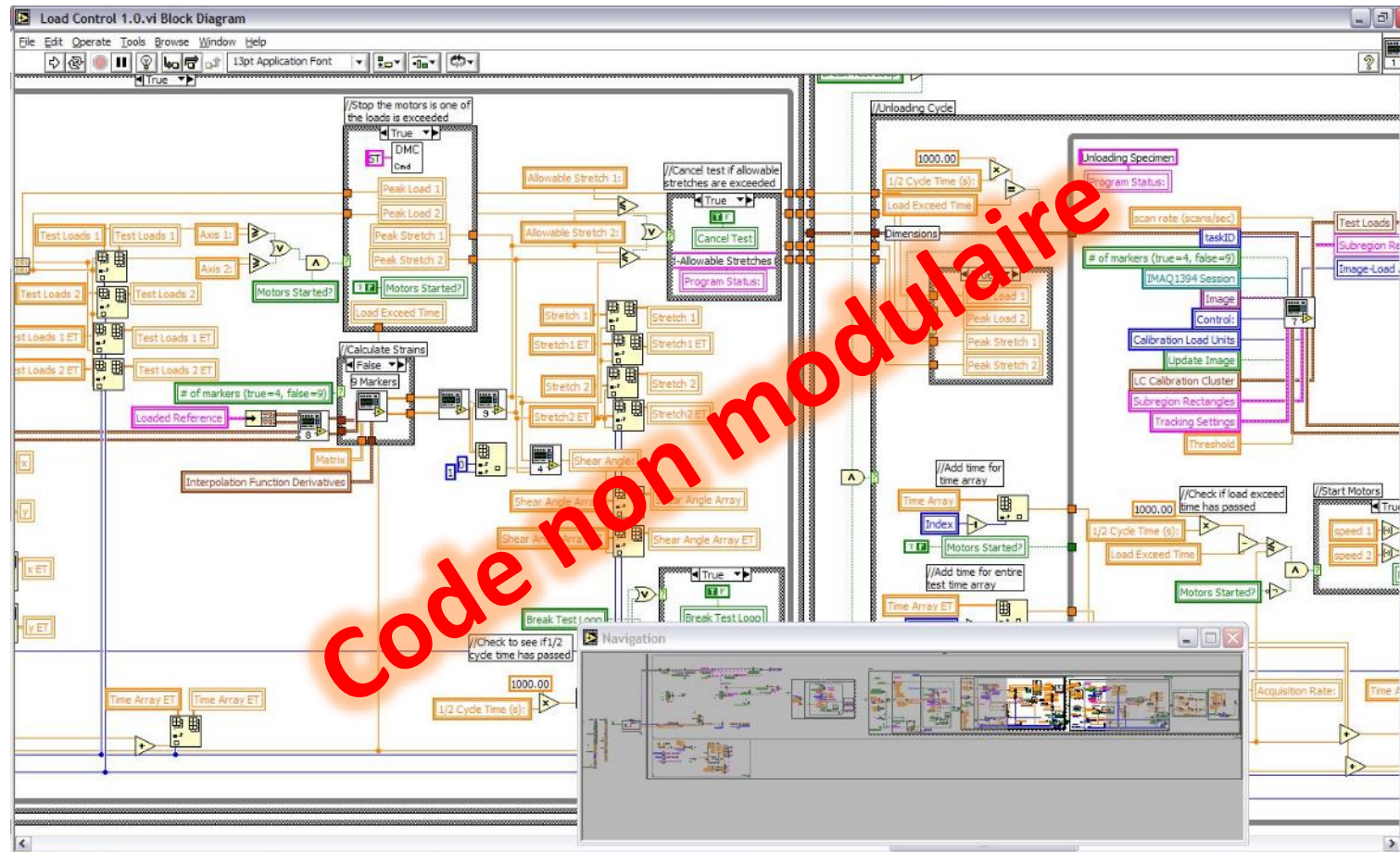
Qu'est-ce qu'une application ?



Qu'est-ce qu'un module ?



<https://lavag.org/topic/2202-how-not-to-code-large-applications/>



Qu'est-ce qu'un module ?



https://fr.wikipedia.org/wiki/Programmation_modulaire

Programmation
procédurale

- Contient simplement une série d'étapes à réaliser
- N'importe quelle procédure peut être appelée n'importe quand (branchements)

Programmation
modulaire

- Décompose une grosse application en modules cohérents
- Permet de développer et d'améliorer les modules indépendamment les uns des autres
- Permet de réutiliser les modules dans d'autres applications (capitalisation)

Programmation
orientée objet

- Définition des interactions entre des objets logiciels
- Les fonctionnalités de l'application sont réalisées à travers les interactions entre les objets



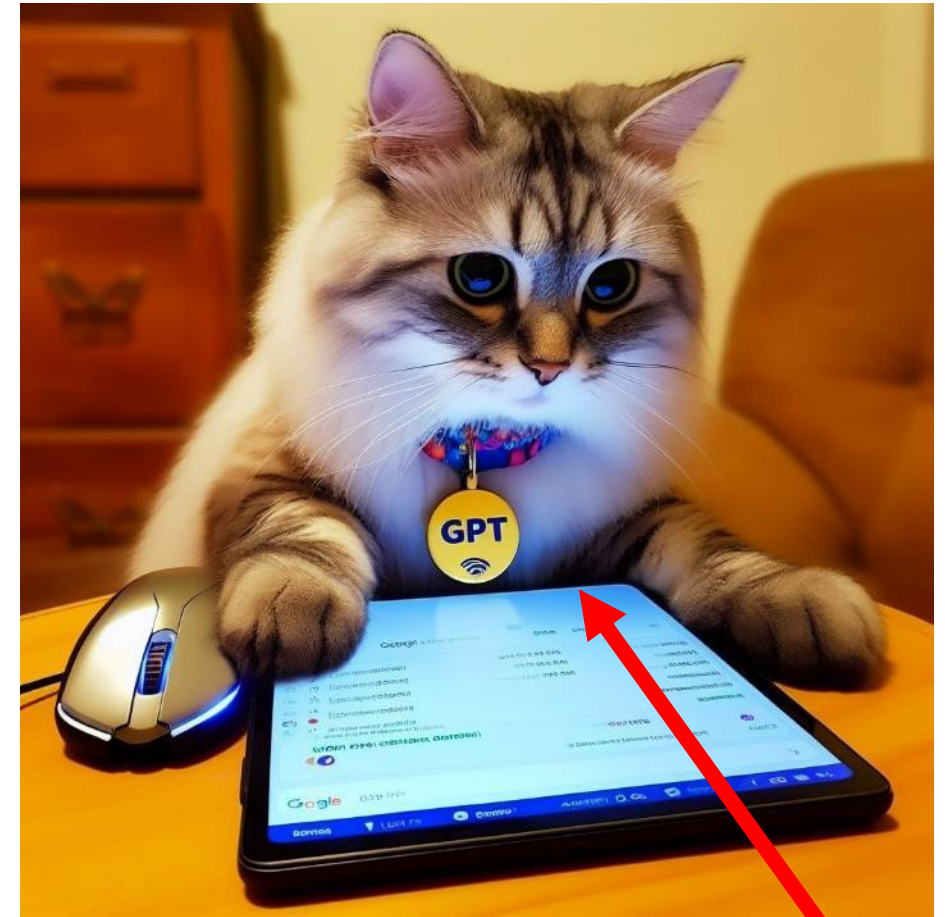
Par abus de langage, une application orientée objet pourra être considérée modulaire

- Objectifs:
 - Découper l'application en modules fonctionnels (architecture)
 - Simplifier les développements en équipe
 - Améliorer la planification
 - Renforcer la testabilité (tests fonctionnels par modules)
 - Capitaliser le savoir-faire en réutilisant les modules
- Les gains sont palpables tant par le client (qualité) que par nous (efficacité)

Les préceptes de la modularité



- Dépendances limitées entre les modules
 - Pas de globales communes à plusieurs modules
- Interfaces explicites
 - Utilisation d'accesses
 - Limite les accès injustifiés à des données
- Information publique et privée
 - Exposer uniquement les données pertinentes
 - Réfléchir aux fonctions de l'API



<https://rmdiscala.developpez.com/cours/LesChapitres.html/Cours4/Chap4.3.htm>

ChatGPT



- Les modules sont indépendants (voir autonomes)
 - Pas de variables globales
 - Pas d'appels croisés
- La communication entre modules gagne à être standard
 - Sinon : développer des wrappers
 - Tous les échanges entre modules passent normalement par le noyau applicatif
- Documenter son module:
 - API
 - Fonctionnement
- Tester son module



En plat de résistance, un peu de LabVIEW

Demandons encore à Chat GPT...



Les outils disponibles sous LabVIEW



LabVIEW offre plusieurs solutions pour diviser son application en modules.

- La llb
- La dll
- La lvlib
- La lvclass

Ces solutions ne sont pas une fin en soi, mais des moyens techniques de structurer et distribuer des modules

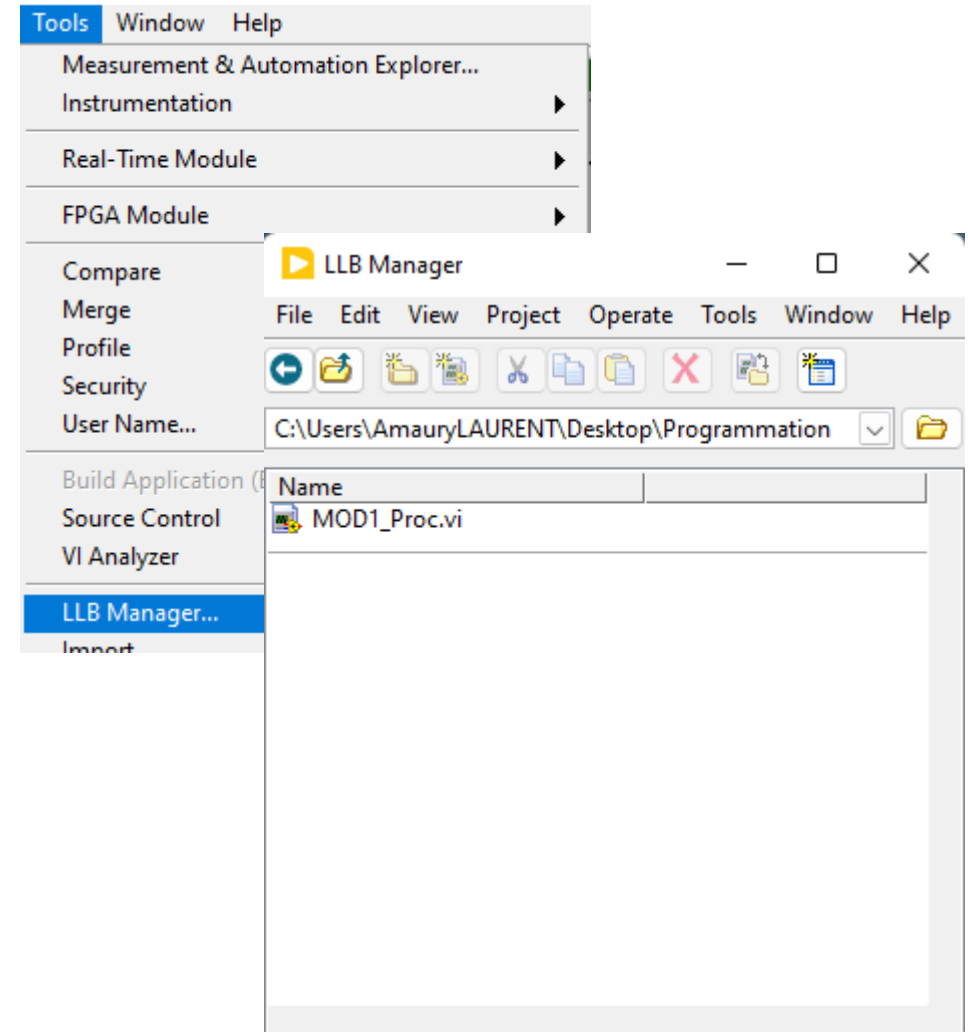
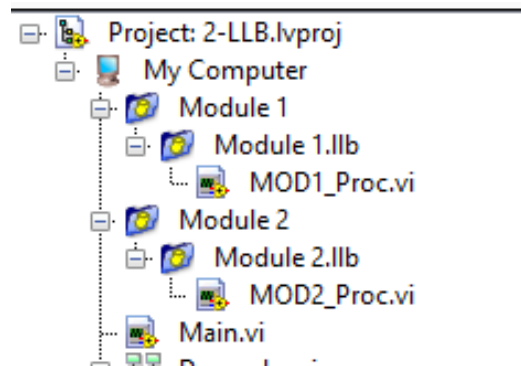
ChatGPT



1-La llb



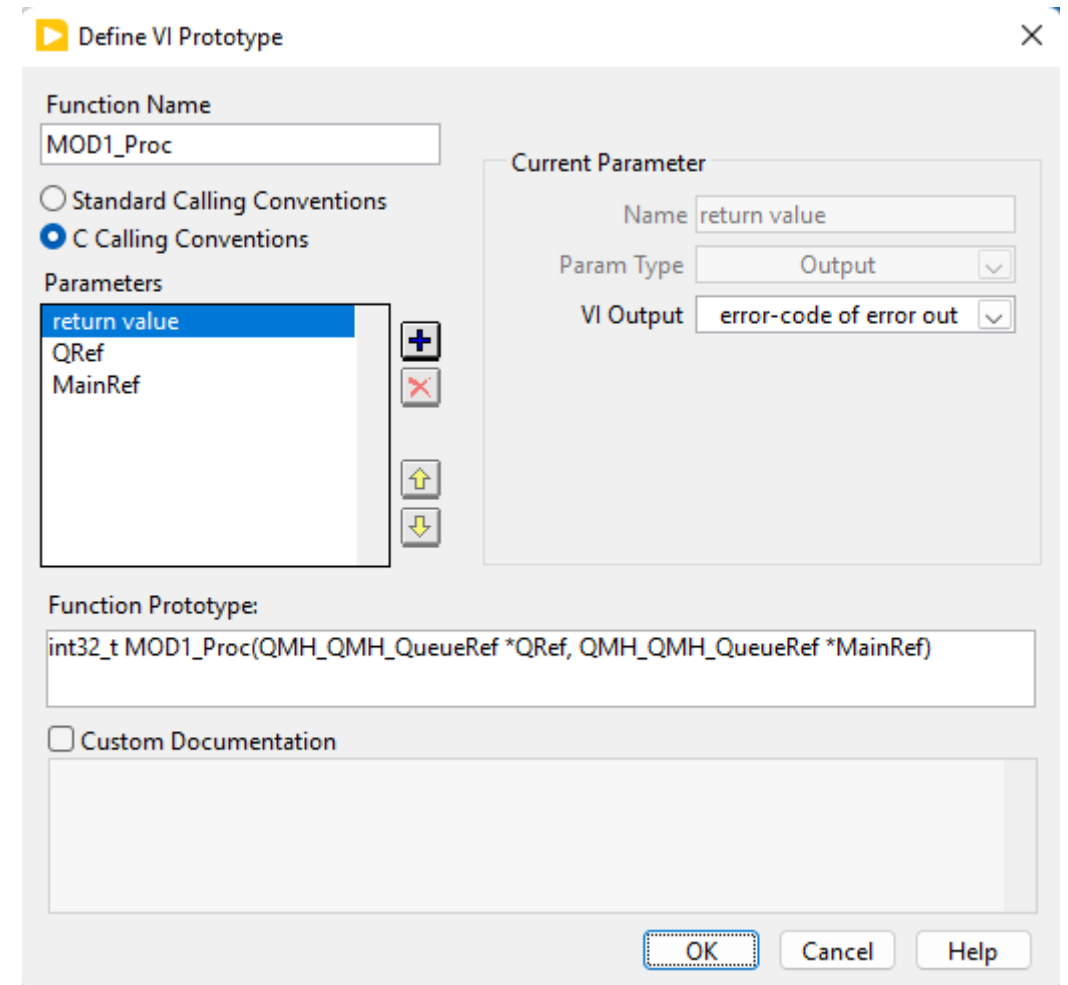
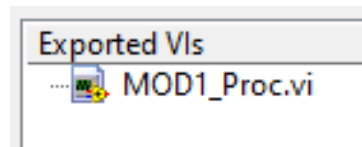
- Technologie vieillissante
- Une llb est un zip qui contient des VI, CTL, ...
- Un fichier sur disque par module
- Pour récupérer un module : copier la llb
- Possibilité de définir un VI Top Level
- Utilisation très similaire aux dossiers virtuels



2-La DLL



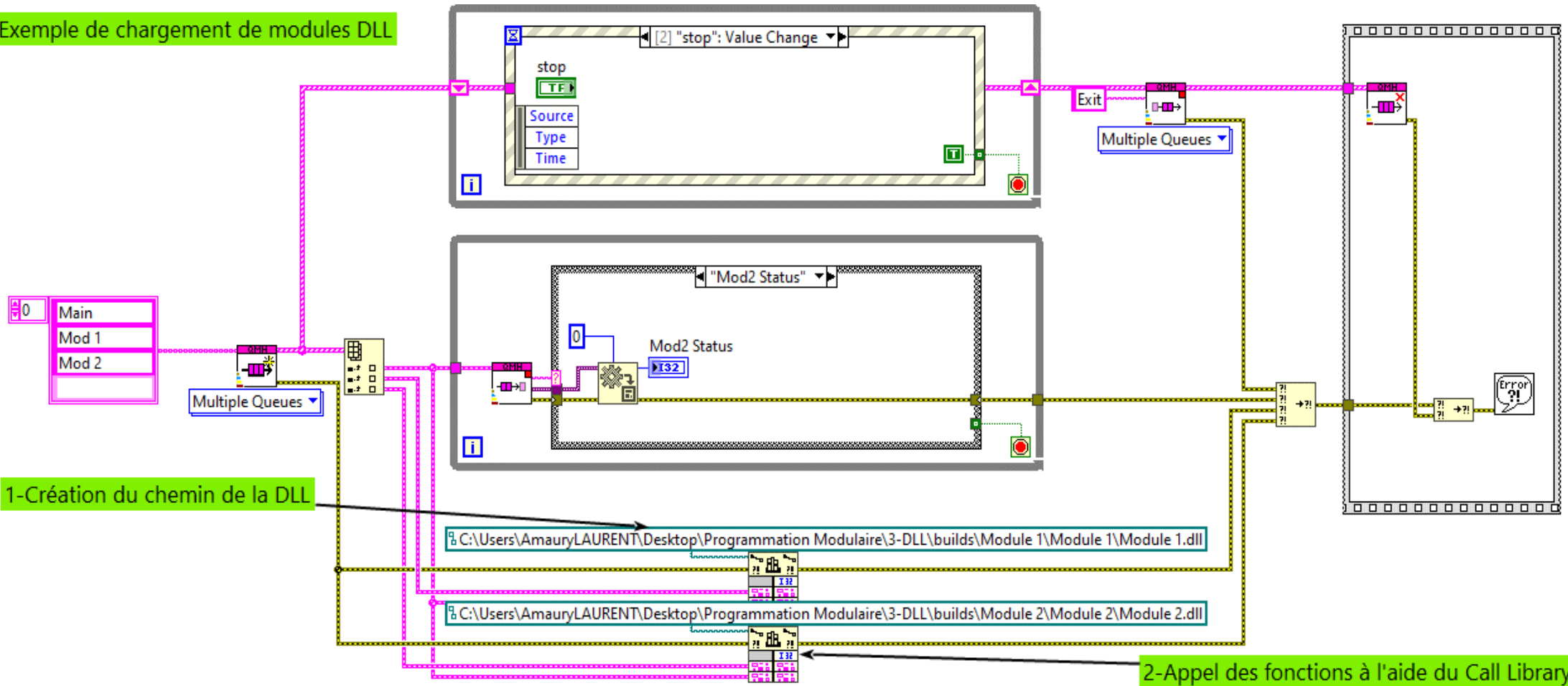
- DLL : Dynamic Linked Library
- Permet d'intégrer un module logiciel écrit dans n'importe quel langage (y compris LabVIEW)
- Ne distribue pas les sources
- Doit être incluse dans la distribution
- Gestion de version native
- Nécessite de recompiler pour changer de plateforme



2-DLL



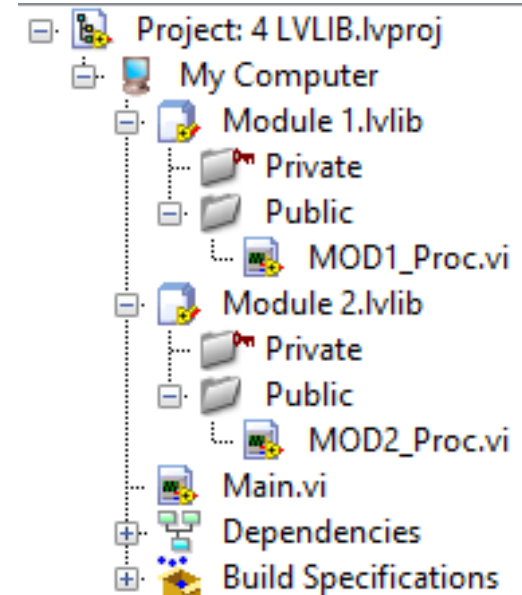
Exemple de chargement de modules DLL



3-La LVLIB



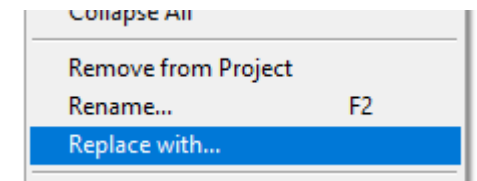
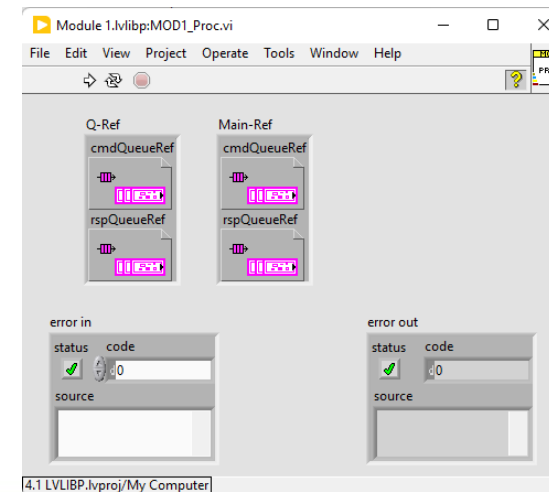
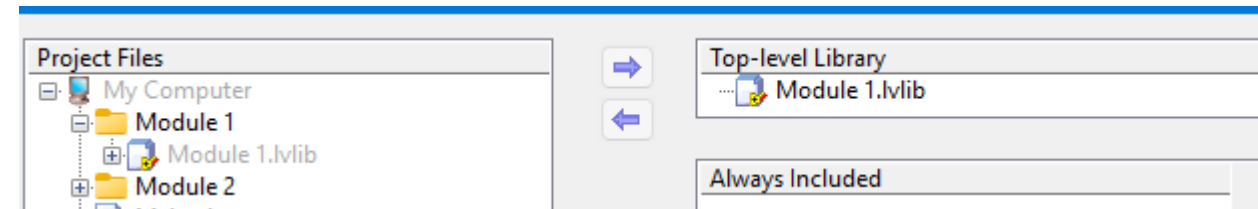
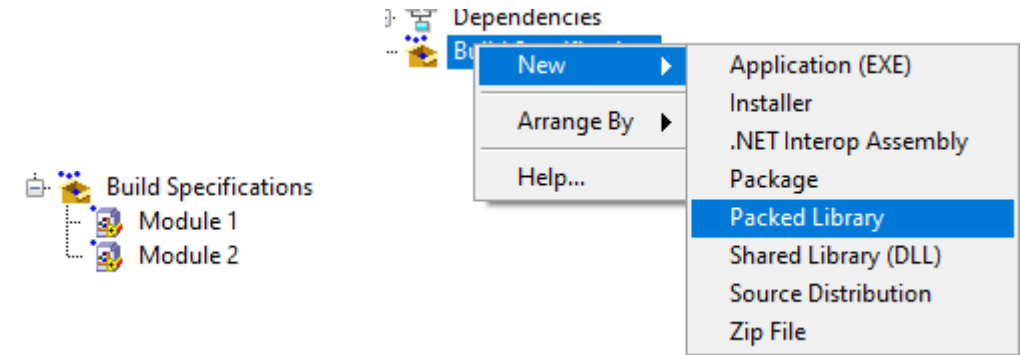
- Comme le lvproj : un fichier xml
- Sur le disque : la lvlib + les sources du module
- Pour récupérer un module : copier tout le dossier (lvlib + sources)
 - /!\ éviter de déplacer les fichiers à la main
 - Privilégier l'onglet Files de l'explorateur de projet
- Décrit le contenu du module
 - Autorisations d'accès (access scope)
 - Gestion de version
 - Protection par mot de passe
 - Documentation
- N'impacte pas le code



3.1-La LVLIBP (LVLIB Packaged)



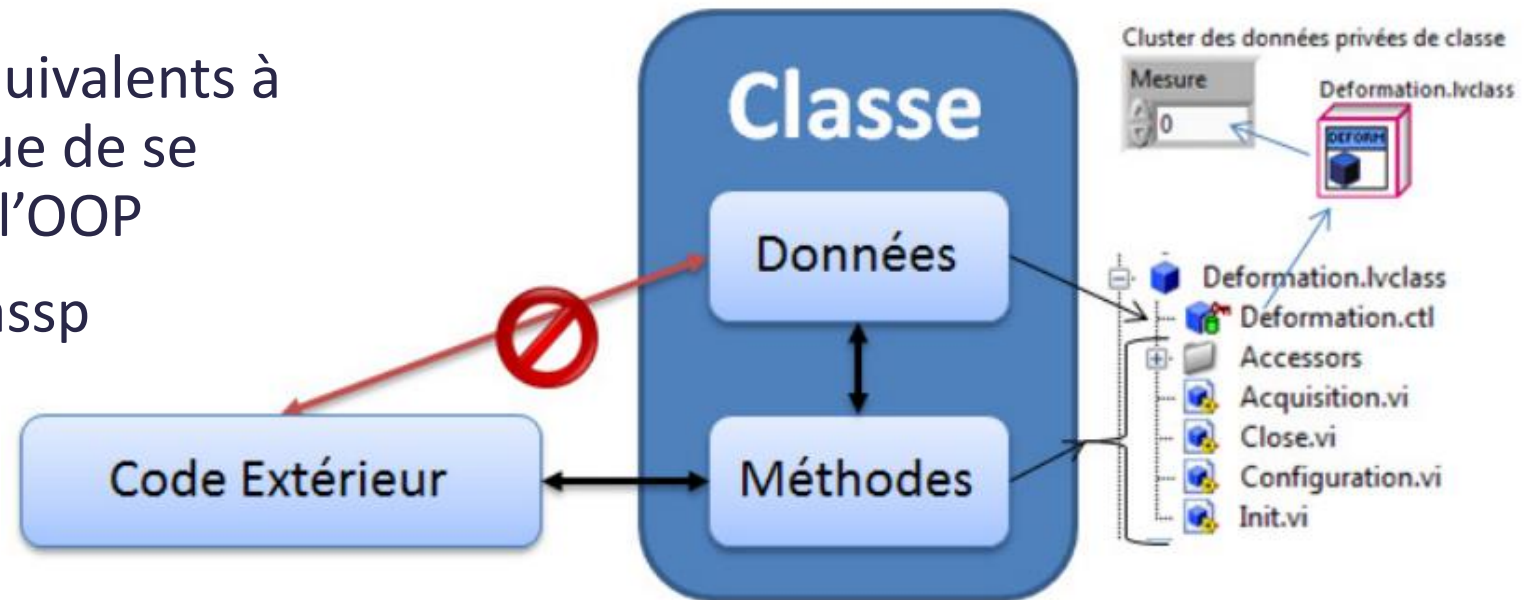
- Il s'agit de la distribution d'une lplib dont les sources ont été supprimées
- 1 seul fichier sur le disque
- Impossible d'accéder aux sources
- Doit être incluse dans la distribution de l'application
- Peut être chargée dynamiquement
- Utiliser le « Replace With... » pour passer rapidement de lplib à lplibp



4-La LVCLASS



- Extension de la programmation modulaire : la programmation objet (OOP)
- Lvclass = Lvlib, le format de fichier est identique (XML)
- Avantages/inconvénients équivalents à ceux de la Lvlib, mais implique de se conformer aux principes de l'OOP
- Impossible de faire une Lvclassp
- Ensemble de données (attribues) et d'actions (méthodes)
- Héritage, dispatch dynamic, encapsulation



Merci Luc pour l'illustration

4.1 Les Plugins : la modularité ultime



- Mix de l'OOP avec la lplib
 - Permet de charger dynamiquement un module dans une application
 - Permet de dissocier les cycles de vies du noyau et des modules
 - Permet de livrer uniquement les modules utilisés
- Obligation de respecter une séquence de construction
 1. Empaqueter l'interface des plugins (classe mère) dans une lplib
 2. Appeler l'interface empaquetée dans l'appli
 3. Faire hériter les plugins de la classe mère empaquetée
 4. Empaqueter les modules en activant l'option « Ne pas inclure les bibliothèques dépendances »

4.1 Les Plugins



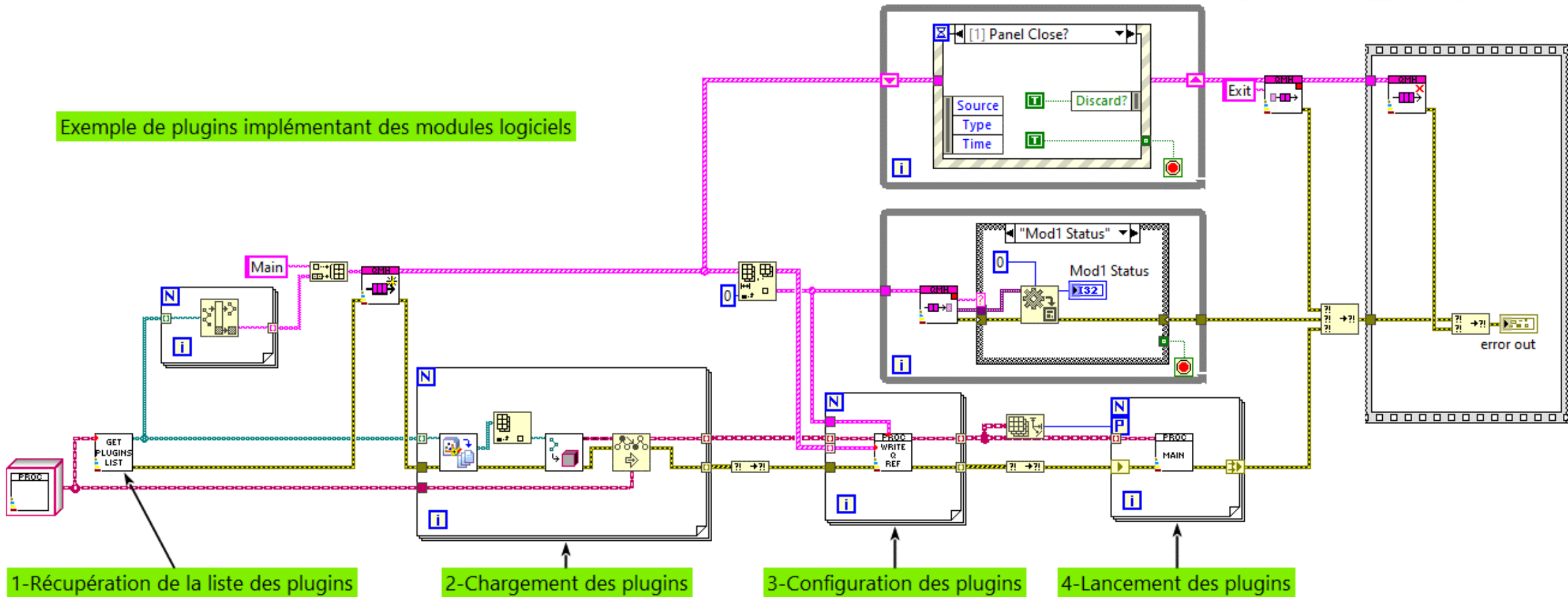
- Le projet principal ne connaît pas les modules
- Tout le monde connaît la classe abstraite (classe mère dans une lplib)
- Aucun couplage entre les modules
- Chargement 100% dynamique à partir des fichiers lplib

The screenshot displays the LabVIEW IDE interface. On the left, the Project Explorer shows a project named 'Project: 6 PLUGINS.lvproj' with a tree structure including 'My Computer', 'Main.vi', 'Module.lvlibp', 'Dependencies', 'Items in Memory', 'user.lib', 'vi.lib', 'QMH.lvlib', 'QMH_OneQueueRef.ctl', 'QMH_ResponseParam.ctl', and 'Build Specifications'. On the right, the 'Change Parent Class for "Module 1" Class' dialog is open, showing a tree of 'All Classes in Project' with 'LabVIEW Object' as the parent of 'Module.lvlibp:Module', which is the parent of 'Module 1:Module 1 (Current Class)' and 'Module:Module'. Below this, the 'Advanced' settings pane is visible, with 'Additional Exclusions' selected. The 'Exclude dependent packed libraries' checkbox is checked and highlighted with a red box.

4.1 Les Plugins



Exemple de plugins implémentant des modules logiciels



Petits rappels sur la construction

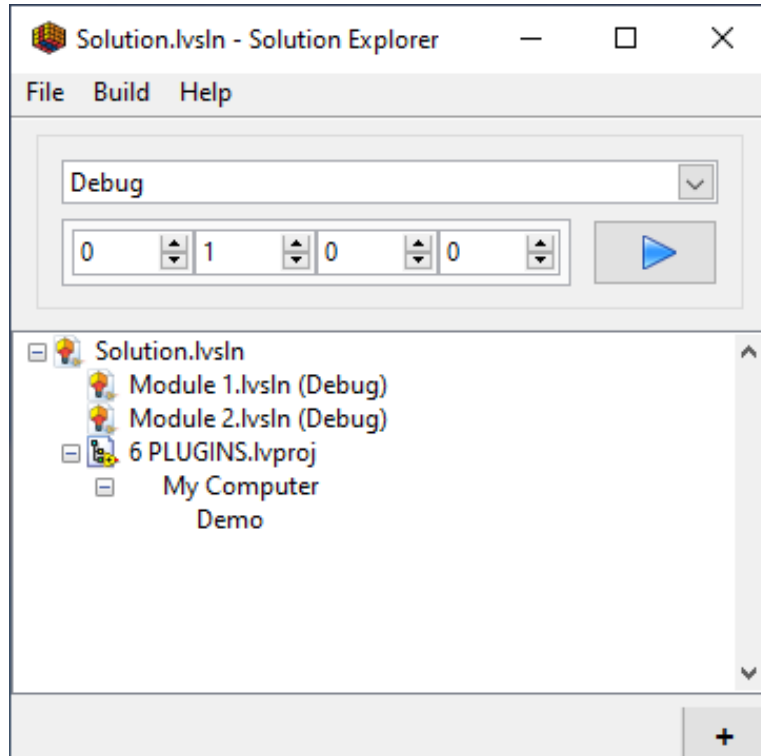


- LabVIEW génère du code compilé à la volée à partir du code source
 - Soit embarqué dans les VI (par défaut)
 - Soit dans le cache (separate compile code)
- Le Run-Time Engine exécute le code compilé
- La construction d'un exe ou d'une lplib est un processus de génération d'un fichier binaire dans lequel les sources sont supprimées pour distribuer une application ou un module

Aspects à prendre en compte lors du choix de la technologie employée pour le développement modulaire



ChatGPT



- Décrit la hiérarchie entre les composants d'un projet
- Automatise la construction de l'application
- Facilite le passage Debug/Release

<https://www.mooregoodideas.com/products/solution-explorer/index.html>



En dessert, une petite synthèse

Ne demandons pas à Chat GPT...



Synthèse



	DLL	LLB	LVLIB	LVCLASS	LVLIBP	Plugins
Multi-langage	oui			non		
Sources accessibles	non	oui	oui	oui		non
Cross-plateforme	non	oui	oui	oui		non
Mono fichier	oui	oui	non	non		oui
Simple à intégrer	non	oui	oui	oui	oui	non
Sources protégées	oui	non	oui	oui		oui
Binaire pour distribution	oui	non	non	non		oui
Gestion de version	oui	non	oui	oui		oui



**N'oubliez pas :
Mettez de l'ambition dans votre ingénierie !**

Merci pour votre
attention

Des questions ?

