Amélioration d'un système de pilotage d'essais thermiques automatisés

Anthony Simonigh

CONSERVATOIRE NATIONAL DES ARTS ET METIERS

CENTRE REGIONAL ASSOCIE DE MIDI-PYRENEES

MEMOIRE

présenté en vue d'obtenir le

DIPLOME d'INGENIEUR CNAM

SPECIALITE : INFORMATIQUE

OPTION : Architecture et Ingénierie des Systèmes et Logiciels (AISL)

par Anthony SIMONIGH

Amélioration d'un système de pilotage d'essais thermiques automatisés

Soutenu le 22 octobre 2015

JURY

PRESIDENT : Yann POLLET – Professeur du CNAM Paris

MEMBRES :

Hadj BATATIA – Maître de Conférences à l'INP Toulouse

Thierry MILAN – Maître de Conférences à l'Université Paul Sabatier

Luc FERRIE – Responsable VTH à THALES ALENIA SPACE

Frédéric VIDONI – Responsable Maintenance VTH à THALES ALENIA SPACE

Remerciements

La réalisation de ce mémoire concerne les travaux réalisés au sein de la société Thalès Alenia Space située à Toulouse. Le tutorat pédagogique a été suivi par M. Hadj BATATIA.

Tout d'abord, je tiens à remercier Pascale SEGALIE et Jean-Luc FORMOSA pour m'avoir accordé leur confiance en m'intégrant à leur équipe.
Je remercie mon tuteur Luc FERRIE pour sa disponibilité et ses conseils qui ont contribués à alimenter ma réflexion.

Je désire également adresser mes remerciements à tous les professeurs, collègues et toutes les personnes de mon entourage professionnel ou personnel, qui par leurs paroles, leurs conseils et leurs critiques ont guidé mes réflexions.

Je présente mon respect et ma gratitude à tous ces intervenants.

Liste des abréviations

TAS : Thalès Alenia Space

EMC : Electro-Magnetic Compatibility

VIB : Vibration

VTH : Vide Thermique

BPMN : Business Process Model and Notation

UML : Unified Modeling Language

SysML : Systems Modeling Language

TC : ThermoCouple

MVC : Modèle-Vue-Contrôleur

WPF : Windows Presentation Foundation

MVVM : Modèle-Vue-VueModèle

DSI: Direction des Systèmes d'Information

CU : Cas d'utilisation

GPIB : General Purpose Interface Bus

API : Application Programming Interface

Sommaire

Introduction

Ce mémoire a pour objet l'amélioration des outils des moyens d'essai du laboratoire vide thermique, par la mise en œuvre d'un système d'aide à l'organisation et à la réalisation des essais.

La réalisation des essais nécessite l'utilisation de différents instruments permettant le contrôle et la surveillance de l'environnement d'essais. Ces instruments constituent un ensemble de type et d'interface de communication hétérogène, notamment due à l'amélioration continue des moyens du laboratoire. Ces interfaces utilisent des liaisons Serie, GPIB, USB et Ethernet et se basent sur les protocoles RS232, Modbus et Visa.

Dans le cadre de l'amélioration des outils du laboratoire il est nécessaire de réaliser un système qui répond aux besoins et contraintes liés aux activités des essais vide thermiques, tout en garantissant la compatibilité avec l'ensemble des instruments existants et futurs. Ce système doit également permettre l'évolutivité des algorithmes de contrôle de pression et de régulation thermique par la mise en œuvre d'une architecture adaptée. Dans cette optique, nous avons réalisés ; l'analyse des outils existants, la spécification du système à produire, les choix techniques, le développement et la validation.

L'application des concepts de l'ingénierie système et les différents langages de modélisation tels que BPMN, UML ou SysML sont utilisés dans notre analyse et conception. Le développement de la solution a été réalisé avec les technologies C# .NET.

Le mémoire s'articule de la façon suivante :

Le chapitre 1 présente le contexte et l'environnement du projet ainsi que le fonctionnement du laboratoire pour la réalisation des essais.

Le chapitre 2 traite de l'analyse des outils existants et expose les problématiques et exigences générales du département.

Le chapitre 3 détaille la conception du nouveau système par une approche basée sur les concepts de l'ingénierie système.

Le chapitre 4 concerne l'étude des solutions adaptées qui répondent aux exigences et aux différentes contraintes imposées.

Le chapitre 5 présente la phase de réalisation du système en mettant en œuvre l'agilité et détaille les composants de communication et de pilotage des moyens.

Enfin une dernière partie conclura ce mémoire par un bilan général du projet, les limitations et les perspectives du système.

Chapitre 1: Contexte et problématique

1. Contexte

Le service moyen d'essais

Au sein de Thalès Alenia Space (TAS) sur le site de Toulouse, dont l'activité principale repose sur la conception et la réalisation de systèmes satellitaire, le service moyen d'essais, sous la responsabilité de Mme Pascale SEGALIE, participe à la recette des équipements et sous-systèmes spatiaux. Il est constitué de trois laboratoires:

- Le laboratoire EMC, qui teste la compatibilité électromagnétique de l'équipement en mesurant les parasites émis par celui-ci et en lui injectant des perturbations électriques. Ces tests sont réalisés en chambre anéchoïque voir photo Figure 1 ci-dessous). Une chambre anéchoïque est une salle d'expérimentation dont les parois absorbent les ondes électromagnétiques ou sonores. Cela permet de reproduire les conditions de champ libre qu'on retrouve dans l'espace.

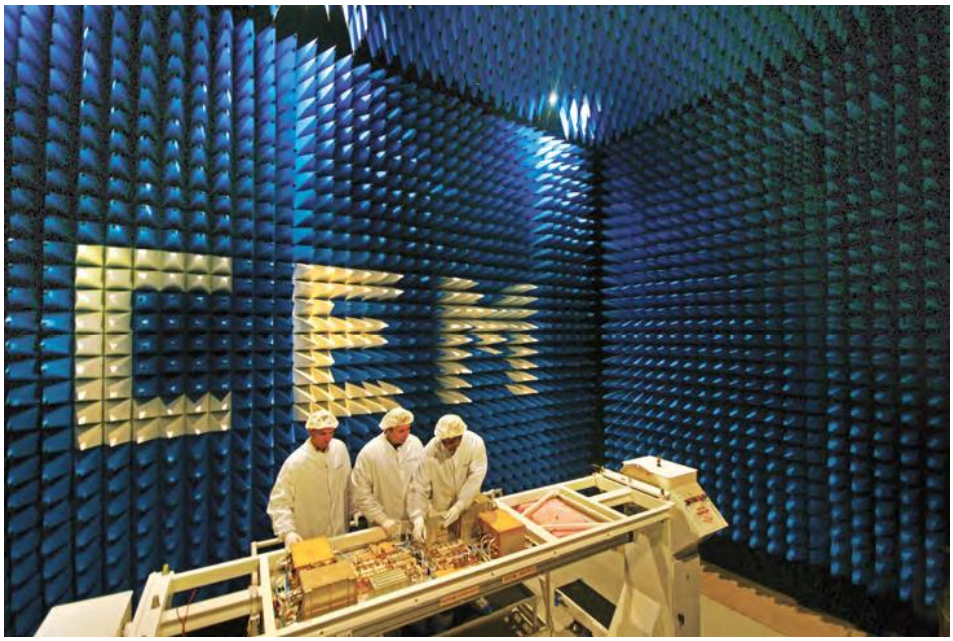


Figure 1 Test en chambre anéchoïque

- Le laboratoire VIB valide les propriétés mécaniques de l'équipement par la production de vibrations et de chocs. Ceux-ci simulent les différentes contraintes mécaniques subit par un équipement satellitaire tout au long de sa vie: du décollage du lanceur à la mise en orbite du satellite. La Figure 2, ci-dessous montre un banc de test de vibration.

Figure 2 Banc de test vibration

- Le laboratoire VTH valide le fonctionnement de l'équipement dans les conditions orbitales. Pour simuler ces conditions des chambres hermétiques sous vides appliquent des cycles de températures allant de -200 °C à +200°C pendant des périodes de quelques jours à plusieurs mois. La Figure 3, ci-dessous, montre une chambre hermétique appelée couramment caisson.

Figure 3 Caisson d'essais thermiques sous-vides

Les activités de ces laboratoires sont les suivantes :

- la rédaction des procédures d'essais, en accord avec le client (interne ou externe à TAS)
- la préparation des équipements de test
- le pilotage et la supervision des essais
- l'analyse des résultats
- la génération du procès-verbal (PV) à livrer au client
- l'étude et le développement de moyens spécifiques
- la maintenance des moyens d'essais

En parallèle de ces activités les laboratoires améliorent continuellement les outils et les processus mis en œuvre dans leur laboratoire.

Le laboratoire d'essais vide thermique

Le laboratoire d'essais vide thermique, auquel je suis rattaché, est composé de six personnes, qui réalisent l'ensemble des activités, sous la responsabilité de Mr Luc FERRIE.

Les moyens

Les moyens disponibles constituent un ensemble de 20 caissons thermiques sous vide et 5 chambres climatiques. Chaque caisson thermique est constitué d'une enceinte de test et d'une baie de pilotage, elle-même composée d'instruments de mesure et de contrôle ainsi que d'un PC de pilotage permettant la supervision de l'essai et la commande des instruments. La Figure 4 suivante illustre la composition d'un moyen.

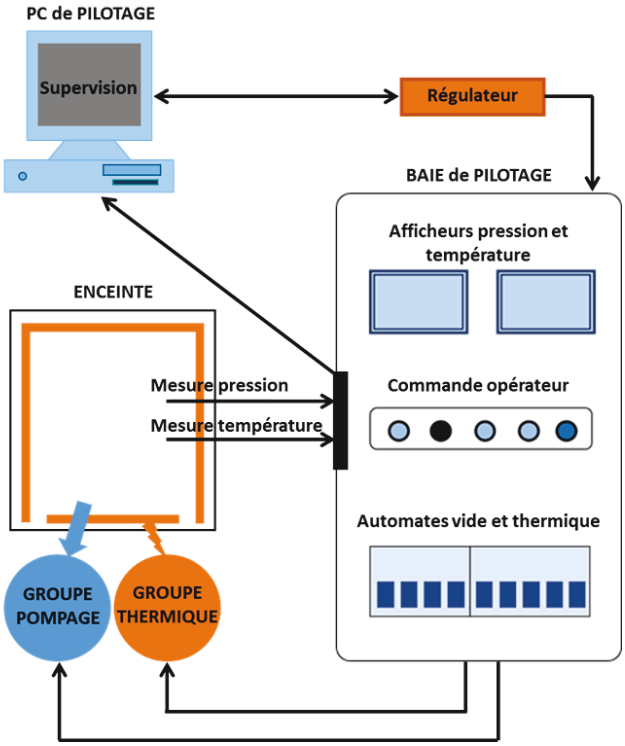


Figure 4 Schéma d'illustration d'un moyen d'essais thermique

Le schéma d'illustration du moyen présente les différents composants matériels qui permettent de réaliser un essai. On identifie 3 éléments principaux :

- L'enceinte, qui permet de simuler l'environnement spatial et où l'équipement à tester est installé. Cette enceinte est composée d'un groupe de pompage pour créer un environnement sous-vide et d'un groupe thermique qui applique des températures sur l'équipement de façon conduite via la plaque (le support) sur laquelle il est installé et de manière rayonné via l'écran.
- La baie de pilotage, qui permet de contrôler l'enceinte grâce à des automates contrôlant le groupe de pompage, le groupe thermique et des instruments de mesure de température et de pression, qui permettent eux même le monitoring de l'environnement. Cette baie fournie une interface de commande opérateur, une interface avec la baie client et des interfaces avec l'ordinateur de pilotage. Ces interfaces offrent la possibilité de communiquer avec les instruments de mesures et les appareils de régulation thermique. La Figure 5, ci-dessous, présente la baie de pilotage en question.

Figure 5 Baie de pilotage d'un moyen VTH

- L'ordinateur de pilotage, qui permet de contrôler l'environnement de test à l'aide d'un outil logiciel de pilotage s'adressant directement à la baie de pilotage par les interfaces disponibles.

Le personnel du laboratoire étant uniquement responsable de la mise en œuvre de l'environnement de test, les tests fonctionnels sur l'équipement sont réalisés par le client à l'aide de sa propre baie de test. Celui-ci peut interagir avec l'équipement testé par l'intermédiaire d'une traversée hermétique connectée à sa baie.

Cependant, la baie de pilotage du moyen d'essai thermique peut s'interfacer avec la baie client et permet l'envoi et la réception d'évènements. Ces évènements permettent, par exemple, de synchroniser le cycle de températures avec les routines de test du client.

Afin d'assister le personnel du laboratoire dans ses activités, des outils logiciels ont été développés au fil du temps au sein du laboratoire. Ces logiciels ont été réalisés à l'aide de technologies qui sont aujourd'hui obsolètes.

Pour assurer l'amélioration continue des outils il est donc nécessaire d'identifier les problèmes de l'existant et de proposer une solution adéquate en tenant compte des contraintes de la direction.

Besoins et enjeux

La solution à mettre en œuvre doit permettre de répondre à l'ensemble des besoins liés aux activités du personnel du laboratoire. Cette solution doit mettre en œuvre une architecture adaptée qui garantit sa maintenabilité et son évolutivité.

Il est impératif d'identifier l'ensemble des besoins satisfaits par les outils existants, puis ceux qui doivent être pris en compte dans la nouvelle solution. Par conséquent, il est nécessaire d'étudier le système existant et de comprendre le fonctionnement du laboratoire.

L'enjeu est important car la mise à niveau du parc informatique de TAS vers Windows 7 et la mise en place de nouvelles règles de sécurité relatives à l'utilisation des postes de travail et à l'utilisation des réseaux de communication par la Direction du Système d'Information (DSI), posent un ensemble de questions. Ces questions concernent essentiellement la compatibilité du système existant avec ces nouvelles contraintes définies pas DSI.

Les laboratoires des moyens d'essais, et dans notre cas celui des essais thermiques sous-vides, jouent un rôle capital dans la recette des équipements satellitaires et par conséquent dans la livraison de ceux-ci aux clients dans les temps impartis.

Enfin, au vu de la situation actuelle des filiales européennes du spatial, qui subissent une crise due à un marché commercial stagnant et à la concurrence d'acteurs émergeants d'Inde, du Brésil ou de la Chine[1], TAS a mis en place une stratégie afin de s'aligner sur les offres des concurrents tout en conservant son image de qualité. Cette stratégie implique, notamment, une amélioration de la productivité, celle-ci étant étroitement liée à l'amélioration continue des processus et outils de l'ensemble du groupe.

Planning

Afin d'atteindre cet objectif le laboratoire a ouvert un poste d'ingénieur informatique en contrat de professionnalisation, que j'occupe, sur une période de 24 mois.

Un planning prévisionnel a été mis en place et est découpé de la manière suivante :

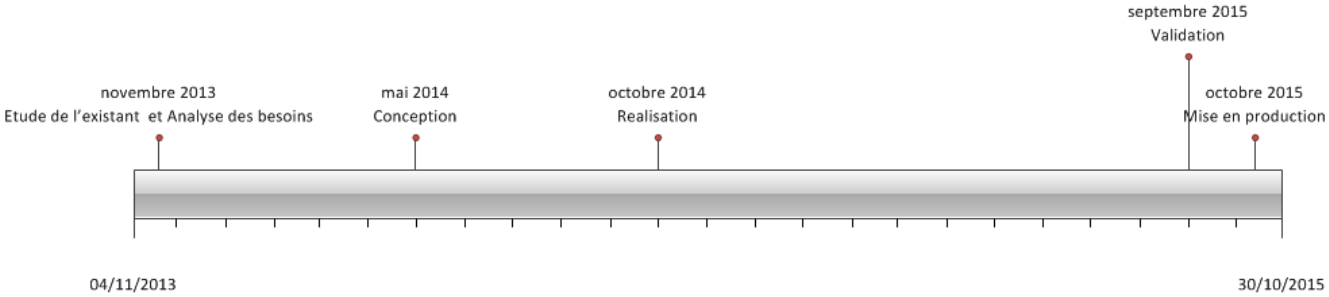


Figure 6 Planning prévisionnel

- **Etude de l'existant et analyse des besoins** : de novembre 2013 à mai 2014

Cette phase permet l'écriture d'un premier cahier des charges fonctionnelles. Ce document recueil les besoins capturés après m'être familiarisé avec l'environnement et les activités du laboratoire.

- **Conception** : mai 2014 à octobre 2014

L'objectif de cette phase est de spécifier le système et de proposer les solutions adaptées en tenant compte des contraintes imposées.

- **Réalisation** : octobre 2014 à septembre 2015

Cette période permet la réalisation de la solution. A l'issu de cette phase une version fonctionnelle sera livrée au laboratoire.

- **Validation :** septembre 2015 à octobre 2015
- La phase de validation permet de tester l'ensemble du système en condition réelles.

Les problèmes restant seront corrigés afin de livrer une version finale et de réaliser la recette.

- **Mise en production : octobre 2015**

Pendant la mise en production la solution sera déployée sur l'ensemble du laboratoire.

2. Problématiques

Dans un premier temps il est nécessaire d'étudier les outils existants afin d'en dégager les fonctions satisfaites par le système. Ces outils ont été développés au fil du temps par le personnel et par conséquent des personnes dont le développement logiciel n'est pas leur spécialité. Ainsi, les aspects de qualité interne n'ont pas été pris en compte. De plus, la documentation concernant leurs implémentations est pauvre et souvent incomplète. Ces manques de structuration et de documentation rendent complexe l'étude fonctionnelle de l'existant.

Dans un second temps, la réalisation de la nouvelle solution rencontre une complexité due à l'hétérogénéité des interfaces de communication avec la baie de pilotage. Cette hétérogénéité est la conséquence de la diversité des modèles des instruments utilisés. De plus, il est nécessaire de tenir compte de la potentielle évolution des moyens dans un contexte d'amélioration continue.

Par conséquent, l'amélioration du système pose deux problématiques :

1. **Comment réaliser l'étude d'un système logiciel existant dépourvu de documentation détaillée ?**
2. **Comment garantir l'interchangeabilité d'un ensemble hétérogène d'instruments par la mise en place d'une architecture adaptée ?**

Ce mémoire présentera les méthodes permettant de répondre à ces problématiques puis, leur application dans le contexte du laboratoire d'essais thermiques sous-vides.

Chapitre 2: Etude de l'existant

Comme vu dans le chapitre précédant, l'étude du système existant présente des difficultés compte-tenu de sa faible documentation et de sa structure qui ne permet pas d'être facilement maintenue. Ce chapitre présentera dans un premier temps la méthode qui a été utilisé pour dégager les fonctions de service du système existant puis, dans un second temps l'application de cette méthode sur les outils logiciels du laboratoire vide-thermique.

1. Méthodologie

La méthode proposée pour cette étude se décompose en deux parties principales :

- L'analyse du domaine, qui permet la compréhension de l'organisation du service par l'identification des acteurs et des processus métiers.
- L'analyse technique, qui permet l'étude des aspects technologiques, matériels et logiciels qui satisfont les activités du domaine d'étude.

Enfin, à partir de ces analyses nous serons capables d'apporter une critique sur l'existant et d'en tirer les éventuels axes d'améliorations autour des outils logiciels.

Analyse du domaine

Cette analyse permet la description de l'environnement par l'identification des acteurs et la modélisation des processus métiers.

Contexte

Dans un premier temps, il est important de comprendre le contexte dans lequel le système existant est déployé. Pour cela il est nécessaire de s'en imprégner, ainsi, accompagner le personnel et, dans la mesure du possible, participer à la réalisation de leurs activités est un bon moyen de comprendre leur rôle au sein de l'organisation.

La réalisation d'interviews permet à la fois de comprendre l'activité du personnel et de capturer les besoins et exigences des utilisateurs du système. Ces besoins permettront de proposer des axes d'améliorations pour le système futur.

A l'issu de cette première approche, nous pouvons modéliser l'environnement pour identifier les différents acteurs du domaine d'étude.

Processus métiers

Dans un second temps il est nécessaire d'étudier les processus métiers relatifs au domaine. La capture de ces processus est essentiellement réalisée par l'interview des acteurs concernés, dans le cas où il y a peu de documentation.

La modélisation de ces processus peut être réalisée avec un diagramme d'activité SysML/UML ou un diagramme de collaboration BPMN (Business Process Model and Notation). BPMN permet de modéliser le savoir-faire métier par une approche processus, bien qu'il ressemble fortement aux diagrammes d'activités, de plus BPMN apporte une sémantique plus riche qui permet par exemple d'identifier le type des processus (manuel, automatisé, etc..). Il fournit également la sémantique pour modéliser les données d'entrée et de sortie de chacune des activités.

A l'issu de cette analyse, on est capable de comprendre les besoins du domaine étudié et de fournir la liste des exigences (fonctionnelles ou non). Le diagramme d'exigences SysML est approprié car il permet de structurer les exigences et d'établir des relations entre celles-ci. Ensuite, il est nécessaire d'isoler les exigences fonctionnelles afin de modéliser l'organisation fonctionnelle.

Analyse technique

L'analyse technique de l'existant est l'étude des composants matériels et logiciels qui satisfont un ensemble de besoins du domaine d'étude.

Composants materiels

L'étude des composants matériels permet de capturer l'ensemble des ressources qui participent à l'exécution des activités du domaine. Le diagramme de block SysML permet de modéliser ces composants. Cette étude doit fournir les caractéristiques de chacun des blocs telles que le type, le modèle, la mémoire disponible, le processeur, le système d'exploitation et les interfaces disponibles. Evidemment les caractéristiques à fournir dépendent du composant étudier.

Composants logiciels

L'étude des composants logiciels permet de capturer les composants qui sont exécutés sur le matériel et qui réalisent les activités identifiés précédemment. Le diagramme de block SysML permet également cette description et doit préciser les fonctions réalisées ainsi que les données d'entrées et de sortie en identifiant les types d'interfaces. Ces flux de données doivent également être détaillés (format, volume, etc..)

A l'issu de cette phase on est capable de fournir l'architecture technique de l'existant en représentant la répartition des composants logiciels sur le matériel. A nouveau le diagramme de block SysML peut être utilisé pour représenter ces relations. Par rapport à UML, SysML est plus riche et plus flexible car moins centré sur le logiciel. Par conséquent, il convient parfaitement pour représenter un environnement hétérogène constitué de composants logiciels et matériels.

Enfin revoir les outils logiciels en s'appuyant sur les écrans de l'existant est une façon efficace de relever les besoins en termes de fonctionnalités. Les questions-types qui peuvent être posées pendant les réunions de revue avec les utilisateurs sont :

- Quel est le but de cet écran ? Que faites-vous sur cet écran ?
- Quels sont les éléments qui manquent ? Quels sont les éléments superflus ou qui sont rarement utilisés ?
- Quels sont les éléments à déplacer ?

Ces questions permettent d'affinés les exigences et apporte une base à la critique de l'existant.

Critique de l'existant

Enfin une fois ces analyses réalisées, il est nécessaire d'apporter une critique sur le système étudié. Pour ce faire on peut représenter un diagramme transverse qui montre la manière dont les besoins et exigences identifiés dans la première phase sont satisfaits par les composants vus dans la deuxième. Ainsi on est capable d'identifier les besoins non-satisfaits et fournir les axes d'améliorations et les éventuelles opportunités.

2. Application de la méthode au sein du laboratoire

Dans cette partie nous appliquerons la méthode proposée, sur le laboratoire VTH

Domaine

Le schéma de la Figure 7, ci-dessous, modélise le système logiciel VTH dans son environnement. Dans l'environnement direct on distingue les éléments suivants :

- Système VTH : C'est le système à l'étude, il représente l'ensemble des outils du laboratoire, ces outils fournissent des interfaces utilisateur à l'Opérateur VTH
- L'opérateur VTH, qui représente les utilisateurs du système au sein de laboratoire.
- La baie de pilotage, qui permet de contrôler le moyen d'essai et qui fournit des interfaces de communication avec le système.
- Le serveur de fichier, qui permet le stockage des données de manière centralisée et qui s'interface également avec le système VTH.

L'environnement indirect concerne les contraintes qui influencent le système. Celles-ci sont le client qui est à l'initiative de l'essai à réaliser et la Direction du Système d'Information qui impose des règles de sécurité au système VTH.

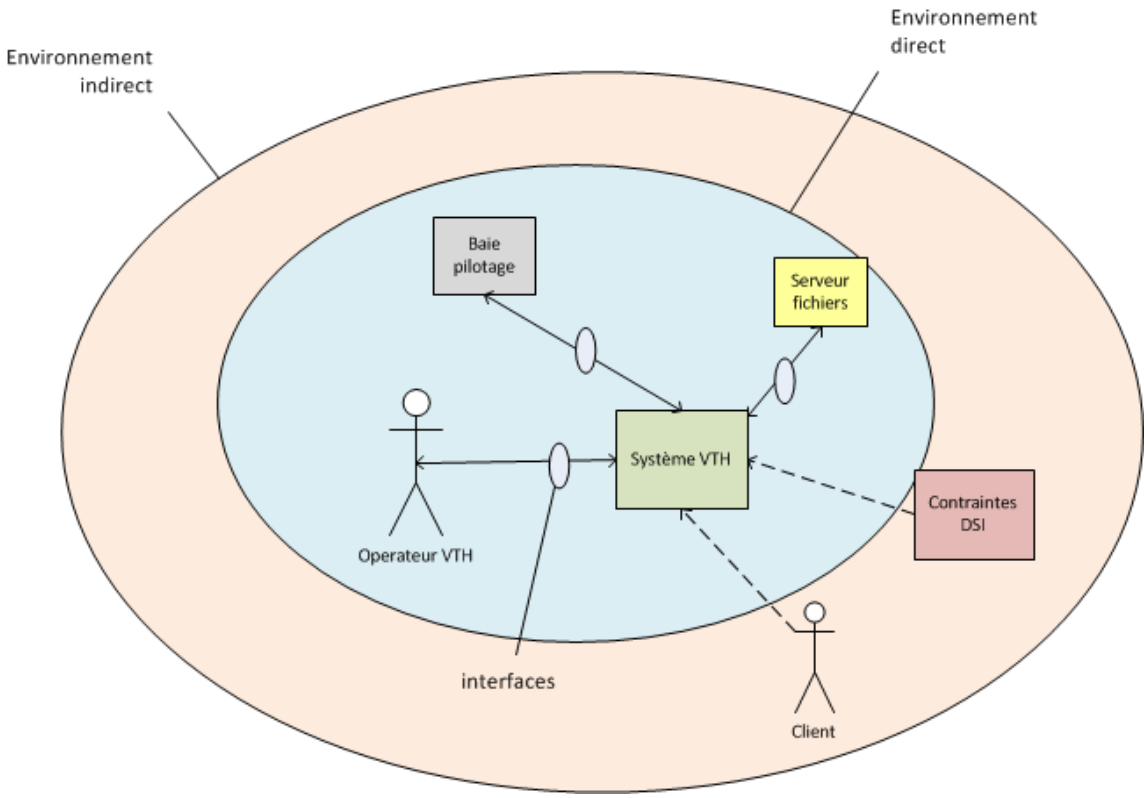


Figure 7 Contexte système existant

Les processus métiers

Dans cette partie nous allons détailler les processus métiers qui permettent la création d'une procédure d'essai et le lancement d'un essai. La syntaxe BPMN est mise en œuvre pour modéliser ces processus

Demande de réalisation d'essai

Le département VTH traite des demandes d'essais à l'initiative du client. Le client peut être une personne ou une entité externe à TAS. Si le client souhaite tester un nouveau type d'équipement, il envoie une demande de procédure accompagnée de tous les documents nécessaires à sa rédaction (cahier des charges, numéro d'affaire, type d'équipement, etc..). Une fois rédigée, la procédure part dans un circuit de validation.

Celle-ci validée, le client a la possibilité de planifier la date de l'essai en consultant le planning de disponibilité des moyens. Le diagramme de collaboration de la Figure 8, ci-dessous, présente la procédure de demande d'essai.

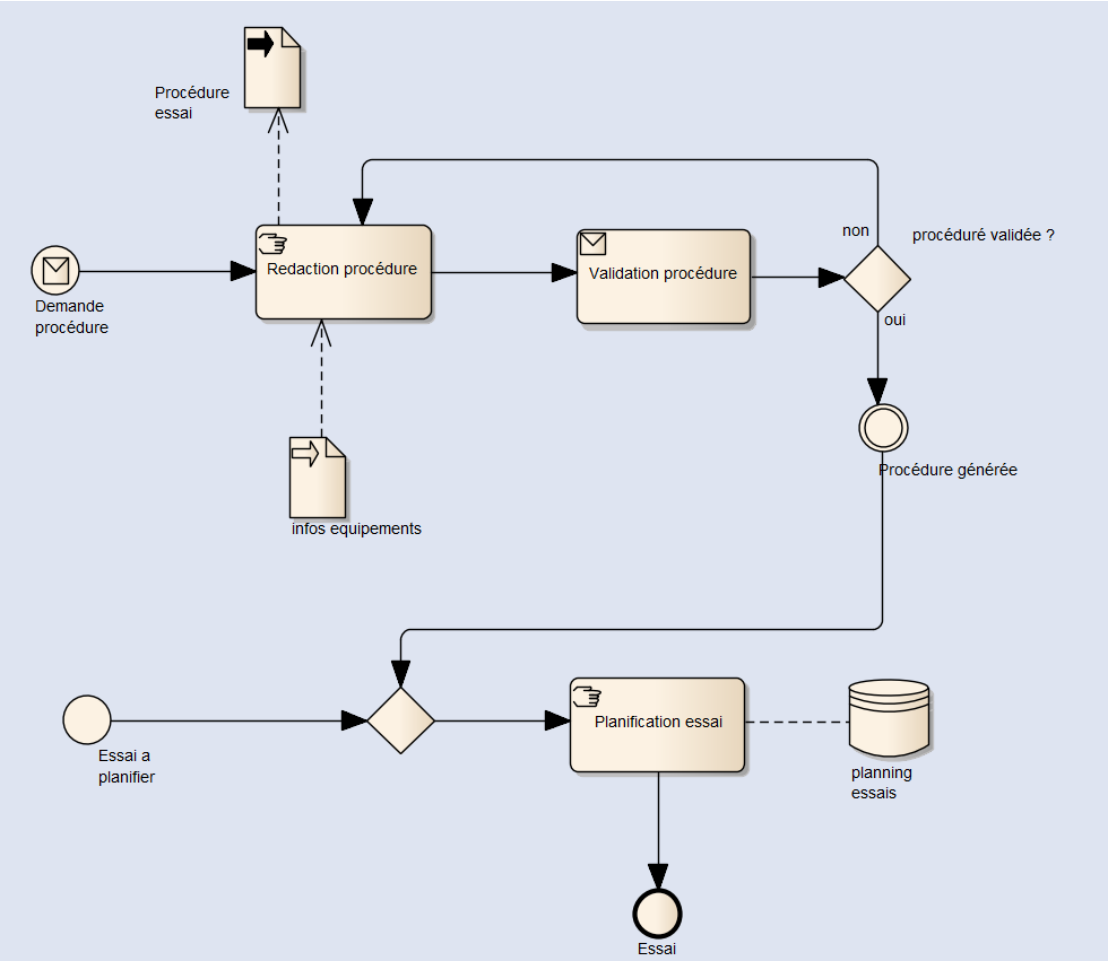


Figure 8 Processus demande d'essai

Lancement d'un essai

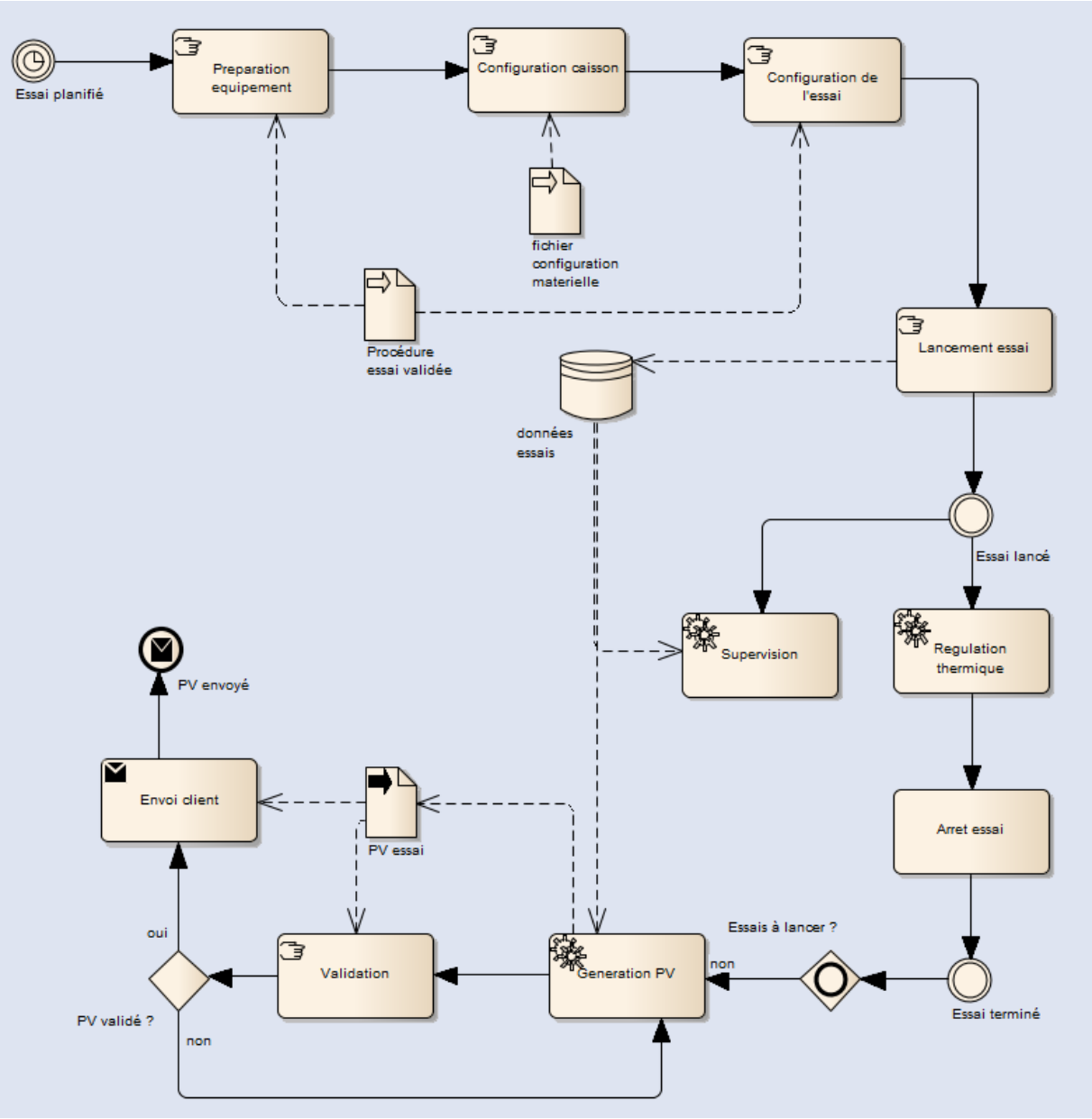


Figure 9 Processus lancement essai

Quand un essai est planifié, le technicien du laboratoire VTH récupère la procédure validée et prépare l'équipement. La préparation de l'équipement consiste à fixer le ou les équipements dans l'enceinte de test (le caisson thermique) et à placer les sondes de mesures telles qu'elles sont définies dans la procédure. Ensuite, le caisson est configuré avec les instruments nécessaires au bon déroulement de l'essai, cette configuration est réalisée via un logiciel qui offre une interface utilisateur et permet le chargement d'un fichier de configuration.

La configuration de l'essai est faite manuellement et consiste à fournir les informations de l'équipement à tester (numéro d'affaire, type, numéro de série), le cycle des températures à appliquer et les paramètres de pilotage tels qu'ils sont définis dans la procédure.
A ce stade l'essai est prêt à être lancé par l'opérateur. Une fois lancé, le logiciel de pilotage régule les températures en respectant la configuration saisie et enregistre les données (mesures, log de pilotage) dans un serveur de fichier afin d'être superviser à distance de manière automatique.
Lorsque le cycle de test est complété et si l'équipement a passé l'ensemble des tests nécessaires, un PV est généré et envoyé au client après validation de celui-ci par le responsable du département. Le diagramme de collaboration à la Figure 9 présente le processus de lancement d'un essai.

Analyse technique

Composants matériels

Les composants matériels qui concernent notre étude sont ceux sur lesquelles les composants logiciels sont déployés.

Baie de pilotage : La baie de pilotage embarque tout le matériel nécessaire au pilotage du moyen. Il y a par conséquent une baie de pilotage par moyen (20). Elle est composée des éléments suivants :

- **Mesure pression** : Permet l'acquisition des mesures de pression
- **Mesure température :** Permet l'acquisition des mesures de température
- **Commande automate :** Permet la commande de relais afin d'interagir avec les automates de la baie et envoie des événements à la baie client.
- **Régulateur thermique** : Permet de contrôler le groupe thermique afin d'appliquer les températures souhaitées

Poste de pilotage : C'est le PC sur lequel sont déployés les composants logiciels relatifs au pilotage. Il utilise le système d'exploitation Windows XP et il s'interface avec les différents instruments de la baie, par RS232 et GPIB, afin de lire les données mesurées. Le poste est également relié au réseau industriel afin d'accéder au serveur de fichier, notamment pour le partage des données de l'essai en cours.

Le serveur de fichier : C'est un disque réseau accessible à l'adresse *tlstrore06://ess-env-vth* par l'ensemble des postes du laboratoire. Il est utilisé pour le stockage de tous les documents utilisés dans le laboratoire et des données essais afin de permettre la supervision des moyens à distance.

Les postes de supervision : Ce sont des PC reliés au réseau du laboratoire, permettant l'affichage des données essais de l'ensemble des moyens en temps réel.

Les postes bureautiques : Ce sont les PC sur lesquels l'opérateur édite les procédures, les PV et gère la maintenance des moyens. Ces postes peuvent aussi avoir le rôle de poste de supervision, c'est le cas du personnel du laboratoire qui supervise les essais directement depuis le poste bureautique.

Le diagramme ci-dessous illustre les relations entre les différents composants matériels.

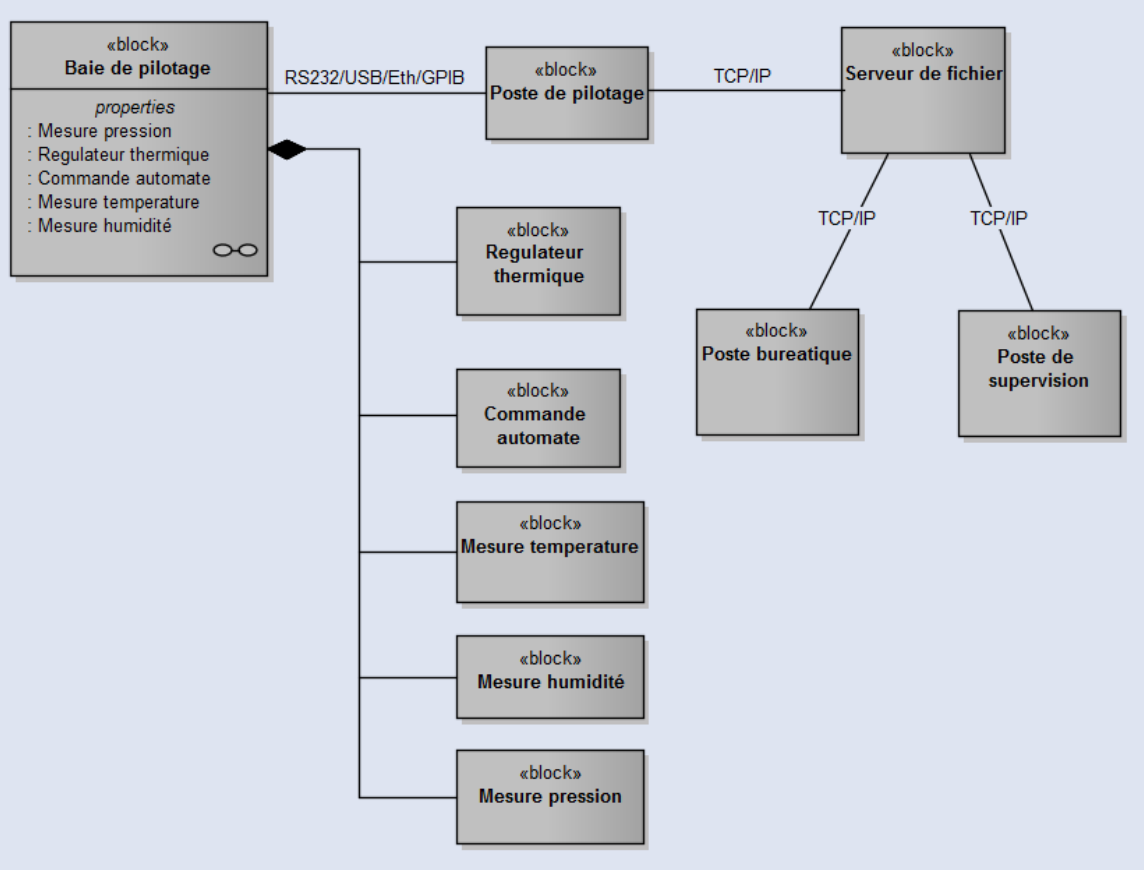


Figure 10 Diagramme de blocs composants matériels existants

Afin de communiquer avec le poste de pilotage la baie fournie les interfaces des instruments. Ces interfaces varient selon la marque et l'année du modèle :

- **RS232** : C'est une liaison de type série qui est largement répandu dans l'industrie. Cette interface est disponible sur l'ensemble des instruments du laboratoire.
- **GPIB** : (General Purpose Interface Bus) est un bus de communication à courte distance également connu sous le standard IEEE-488. Ce bus permet d'adresser 15 périphériques[2] .Cette interface est disponible sur les instruments de marque HP/Agilent.
- **USB** : C'est la liaison de type série la plus répandu aujourd'hui et qui a commencé à remplacer le RS232 sur la plupart des ordinateurs du marché. Cette interface est disponible sur les instruments HP/Agilent les plus récents.
- **Ethernet** : Ce protocole de réseau permet de relier les instruments directement sur un réseau grâce à un connecteur RJ45. Ce type d'interface est souvent présent sur les instruments récents, et offre la possibilité de communiquer avec un nombre (presque) illimité d'équipements.

Le tableau I, ci-dessous, liste l'ensemble des instruments présents dans le laboratoire, en précisant les fonctions remplies et les interfaces disponibles.

Remarque : Les régulateurs thermiques et les instruments de commande des automates sont classés dans le type Cmd (Commande).

Tableau I : Comparatif des instruments du laboratoire VTH

Modèles	Fonctions				Interfaces			
	Temp.	Pression	Humidité	Cmd	GPIB	USB	Eth.	RS232
Agilent 34970A	X			X	X			X
Agilent 34980A	X			X	X	X	X	X
Alcatel ACM 1000		X						X
Alcatel ACM 2000		X						X
Inficon VGC 403		X						X
Interface Spirale	X		X	X				X
Eurotherm 2404	X			X				X
Eurotherm 3504	X			X				X
Eurotherm T2550	X			X				X

Composants logiciels.

La recherche de la documentation disponible et la manipulation des outils du laboratoire a permis de recueillir l'ensemble des composants logiciels. Ci-dessous la liste des composants, du système actuel, identifiés en précisant leurs fonctions, leurs dépendances et les technologies utilisées.

Tableau II : Liste des composants logiciels existants

Fichier	Description/Fonctions	Dépendances	Techno. utilisées
Vide_thermique.exe	Cet exécutable est le point d'entrée pour accéder aux différents outils actuels, il propose une interface permettant d'appeler les différents exécutables qui composent la solution logicielle dédiée au laboratoire vide thermique.	test_des_tops.exe, configuration_caisson.exe, supervision.exe, pv.exe, pilvth.vxe et utilitaire_maintenance.exe.	VB6, Winform.
configuration_caisson.exe	Cet exécutable offre une interface pour éditer la configuration des instruments du moyen avec le poste de pilotage. Il permet également de tester la communication et de la sauvegarder sous forme de fichier éditable.	driver_vth.dll, fichiers de configuration	VB6, Winform
supervisation.exe	Cet exécutable permet de superviser en temps réel l'ensemble des essais en cours. Il accède aux données essais sur le disque réseau (tls-store06://ess-env-vth).	TeeChartV6.ocx, fichiers essais	VB6, Winform
pv.exe	Cet exécutable permet l'édition d'un PV à partir des données essais.	fichiers template, fichiers essais	VB6, Winform
utilitaire_maintenance.exe	Cet exécutable permet le contrôle des performances d'un caisson et la gestion des instruments et pièces détachées concernés par les opérations de maintenance.	TeeChartV6.ocx	VB6, Winform
pilvth.vxe	Cet exécutable permet la configuration, le pilotage du moyen, le lancement d'un essai et son suivi.	courbe_vth.ocx, gestion_objectif.ocx, regul_rechauf_pilvth.ocx, pilote_pil.ocx, programmation.exe, visu_cycle.exe et driver_vth.dll	HpVee 5.0
courbe_vth.ocx	Interface d'affichage des courbes de températures et de pressions pour génération du PV.	TeeChartV6.ocx	VB6, Winform
regul_rechauf_pilvth.ocx	Librairie permettant le pilotage des baies de réchauffage.		VB6

pilote_pil.ocx	: Librairie offrant une interface de contrôle au client de l'essai		VB6, Winform
programmation.exe	Permet la saisie du cycle de températures de l'essai	visu_cycles.exe	VB6, Winform
visu_cycle.exe	Affiche la représentation graphique d'un cycle à partir des données saisies		VB6, Winform
driver_vth.dll	Librairie permettant la communication avec l'ensemble des instruments du laboratoire.		VB6, Modbus, Serie

Le tableau II, ci-dessus, synthétise l'ensemble des composants répertoriés nous a permis de comprendre les relations entre eux. Celles-ci sont des relations de dépendances et de composition le diagramme de blocs de la Figure 11, ci-dessous, illustre ces relations entre les différents composants logiciels.

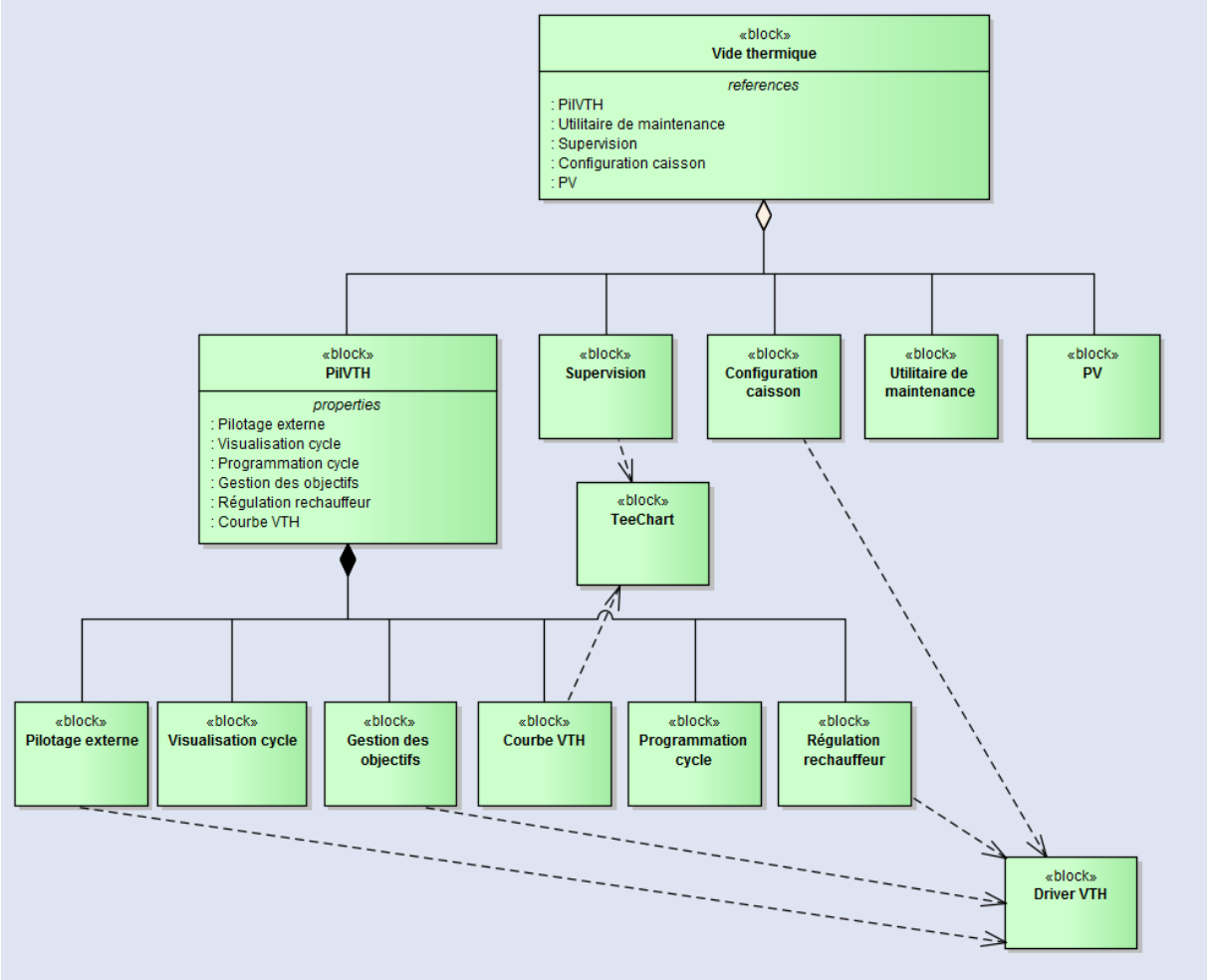


Figure 11 Diagramme de bloc composants logiciels existants

L'utilisation d'exécutables permet de bien découpler les différents composants. Cependant l'utilisation de différents langages de programmations au sein d'un même composant rend

sa maintenance difficile. L'étude du code source de chaque composant à montrer que de nombreuses fonctions sont dupliquées sur plusieurs composants, voir au sein du même, ce qui rend sa compréhension d'autant plus difficile.

Les données

Toutes les données manipulées et échangées par les composants sont stockées sous forme de fichiers texte. Cette étude se limite aux données de mesure et de log générées et aux fichiers nécessaires à la configuration de l'essai ou du caisson.

Fichier de données de mesure

C'est un fichier texte à l'extension .txt. Chaque ligne correspond à une acquisition. Elle commence par la date et l'heure de l'acquisition et contient les données mesurées.

Ci-dessous l'extrait d'un fichier de mesure.

```
date,heure,pression,T1,T2,T3,T4,pilote plaque,pilote ecran,n°enreg
2/7/2014,13:51:41,1000,23.365,23.405,23.313,23.31,25,25,0
2/7/2014,13:52:41,1000,23.376,23.405,23.321,23.31,23.405,23.313,1
2/7/2014,13:53:40,1000,23.181,23.074,23.042,22.874,23.405,23.321,2
2/7/2014,13:54:41,68.59,23.062,23.15,22.977,22.915,23.074,23.042,3
```

Fichier de suivi

C'est un fichier texte à l'extension .suivi qui contient tous les événements liés à l'essai (changement de palier de température, actions, utilisateurs, alarmes, etc). Ce fichier a donc un rôle de fichier de log.

Ci-dessous l'extrait d'un fichier de suivi.

```
2/7/2014   ;   13:51:36 ; Palier 0 ; T =25 ; Demarrage des mesures
2/7/2014   ;   14:6:21 ; Palier 0 ; T =23.102 ; Depart thermique (pression OK)
2/7/2014   ;   14:6:21 ; Palier 1 ; T =23.102 ; depart thermique
2/7/2014   ;   14:6:23 ; Palier 1 ; T =23.102 ; Retour en mode auto
2/7/2014   ;   14:18:39 ; Palier 1 ; T =24.473 ; Debut du palier 1
2/7/2014   ;   14:29:39 ; Palier 2 ; T =25.09 ; Passage au palier 2
2/7/2014   ;   14:34:38 ; Palier 2 ; T =25.006 ; Debut du palier 2
2/7/2014   ;   14:34:39 ; Palier 3 ; T =25.006 ; Passage au palier 3
2/7/2014   ;   14:34:39 ; Palier 3 ; T =25.006 ; Fin du palier de degazage
2/7/2014   ;   14:39:38 ; Palier 3 ; T =25.036 ; Debut du palier 3
2/7/2014   ;   15:41:36 ; Palier 4 ; T =25.136 ; Passage au palier 4
2/7/2014   ;   15:46:35 ; Palier 4 ; T =25.096 ; Debut du palier 4
3/7/2014   ;   6:59:35 ; Palier 4 ; T =25.128 ; Affichage des donnees du cycle
3/7/2014   ;   6:59:45 ; Palier 4 ; T =25.128 ; Arret de l'essai
```

Fichiers de configuration caisson

Ces fichiers texte permettent de configurer les interfaces de communication des caissons et de contrôler leur validité.

Le fichier à l'extension .pil contient la liste des instruments utilisés, la configuration de l'interface de communication et la liste des voies à utiliser pour l'essai. Par conséquent, il est nécessaire de créer un nouveau fichier dès lors qu'un essai n'utilise pas le même nombre de voies que le précédent. Ci-dessous l'extrait d'un fichier .pil :

```
Centrale_acquisition_1
10
hp34970a
COM1
mesure thermocouple k
101,1
102,0.9
103,0.8
104,0.7
105,0.7
106,0.6
107,0.6
108,0.6
109,0.6
110,0.6
mesureur_Pression
acm1000
pression(1 voie)
COM4
2
```

Ce fichier contient la liste des instruments utilisés sur le caisson d'essai, leur temps d'utilisation et leur date de calibration. Ces données sont utilisées pour la génération du PV ainsi que pour la gestion de la maintenance, et sont formatées de la façon suivante :

Contenu fichier.conf	Format des données
Centrale acquisition	Type instrument 1
AGILENT_HP / 34970A	Modèle
59239	N°Immo
US37021889	N°Série
05/09/2015	Date de calibration
cryogénérateur	Type instrument 2
LEYBOLD / COOLPAK 4000	.
20080337	.
20900458141	.
20/09/2012	
Mesureur de pression	
ALCATEL / ACM1000	
112674	
460-2520	
17/09/2014	
Pc	
/	

131571 Null pompe_cryo LEYBOLD / COOLVAC 5000 CL-V - 31000538893 Null pompe_palettes LEYBOLD / SV200	

Fichier de données PV

C'est un fichier texte à l'extension .donnees_pv qui contient les données relatives aux paliers réalisés pendant l'essai. Chaque palier précise la liste des sondes de référence mentionnées dans la procédure d'essai. Ce fichier permet la génération d'un tableau, dit de compliance, qui apparaitra dans le PV. Ce tableau permet de récapitulé l'ensemble des paliers réalisés ou non tels qu'ils ont été définis dans la procédure. Ci-dessous l'extrait d'un fichier de données PV.

```
N° palier,Heure début palier,Heure fin palier,Heure debut de palier num,Heure fin palier num,Consigne plaque finale,Temperature specifiee,Duree specifiee,TMax1,TMin1,TMax2,TMin2,TMax3,TMin3,TMax4,TMin4,TMax5,TMin5,TMax6,TMin6
1,2/7/2014 ; 14:18:39,2/7/2014 ; 14:29:39,63540073120.808,63540073779.551,25.2635,25,10,25.172,24.445,25.09,24.473,25.126,24.504,25.128,24.544,25.071,24.433,25.099,24.471
3,2/7/2014 ; 14:34:38,2/7/2014 ; 14:34:38,63540074079.321,63540074079.321,25.2635,25,60,24.991,24.991,25.006,25.006,25.023,25.023,25.039,25.039,25.06,25.06,25.07,25.07
3,2/7/2014 ; 14:39:38,2/7/2014 ;
```

Architecture physique de l'existant

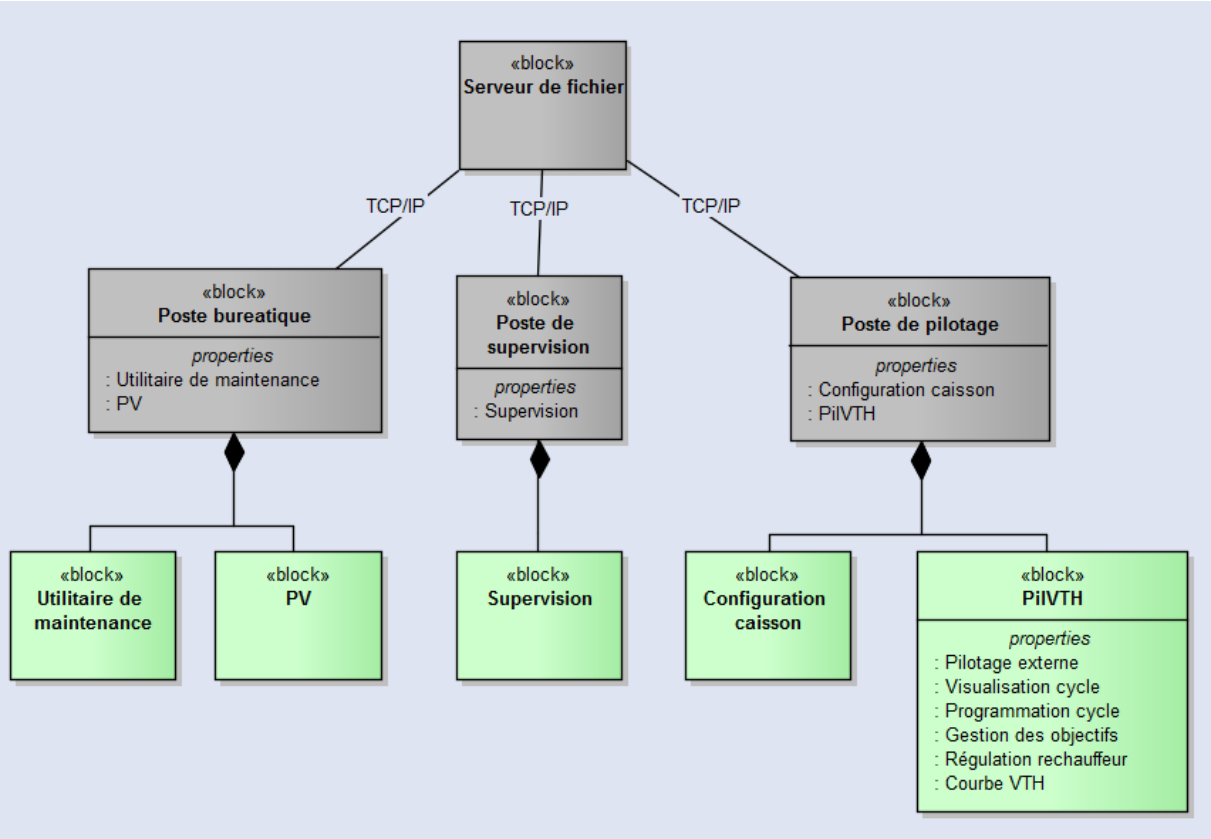


Figure 12 Architecture physique de l'existant

La Figure 12, ci-dessus, présente le déploiement des composants logiciels sur les composants matériels identifiés.

Architecture fonctionnelle globale

Après avoir identifié et étudié l'ensemble des outils du laboratoire, nous avons pu déterminer une architecture fonctionnelle qui récapitule l'ensemble des fonctions réalisés par les outils. Cette architecture est indépendante de l'architecture physique et technique de l'existant car elle ne tient pas compte des technologies mises en œuvre pour réaliser ces fonctions. Afin de représenter cette architecture nous avons utilisé un diagramme de blocs SysML, disponible ci-dessous.

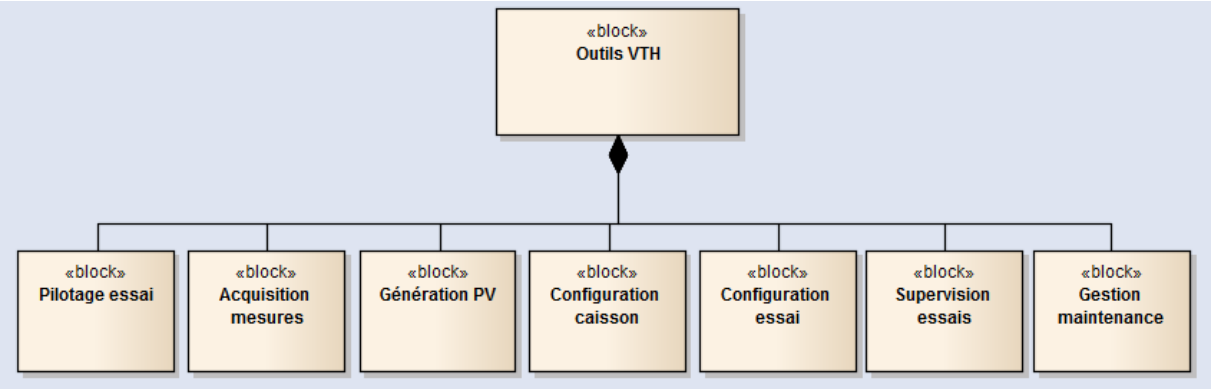


Figure 13 Architecture fonctionnelle globale

L'architecture fonctionnelle est composée de 6 « blocs » fonctionnels qui sont les suivants :

- **Pilotage essai :** Réalise le contrôle et la commande du moyen d'essai en respectant la procédure. Il contient les algorithmes de régulation thermique.
- **Acquisition mesures :** Permet l'acquisition et l'enregistrement des données mesurées pendant l'essai à partir des différentes sondes (pression, température, etc..)
- **Génération PV :** Permet la génération du PV correspondant à un essai donné à partir des données mesurées et la procédure associée.
- **Configuration caisson :** Permet de configurer et de tester les instruments utilisés sur le moyen qui est utilisé pour l'essai à lancer.
- **Configuration essai :** Permet la configuration de l'essai en respectant la procédure associée. Il permet notamment la configuration des voies de mesures à utiliser, la définition du cycle de température à appliquer et la stratégie de pilotage.
- **Supervision essais :** Permet le suivi en temps réel du déroulement des essais du laboratoire, par l'affichage de courbes de mesures et d'alarmes.
- **Gestion maintenance :** Permet de gérer les dates de calibrations et l'inventaire de tout le matériel utilisé dans le laboratoire (instruments, pièces détachées, régulateurs, etc…)

Bilan de l'existant

Cette étude a également permis de capturer les besoins formulés par les utilisateurs durant les interviews et en menant des revues sur les principales interfaces utilisateurs du système

existant (voir exemple annexe 9). La liste des exigences capturées est disponible à l'annexe 3. Le diagramme d'exigences dans la Figure 14, ci-dessous, synthétise ces exigences.

Une exigence peut spécifier une fonction que le système devra réaliser (exigence fonctionnelle) ou une condition de performance, de fiabilité, de sécurité, etc… (exigence non-fonctionnelle). Ces exigences servent à établir un contrat entre le client et ceux qui réalisent le système futur[4].

Afin de modéliser ces exigences, nous avons utilisé le diagramme d'exigences SysML. Ce diagramme capture les hiérarchies d'exigences, ainsi que leurs relations. Ci-dessous les deux diagrammes d'exigences créés : Le diagramme des exigences fonctionnelles, puis le diagramme des exigences non-fonctionnelles.

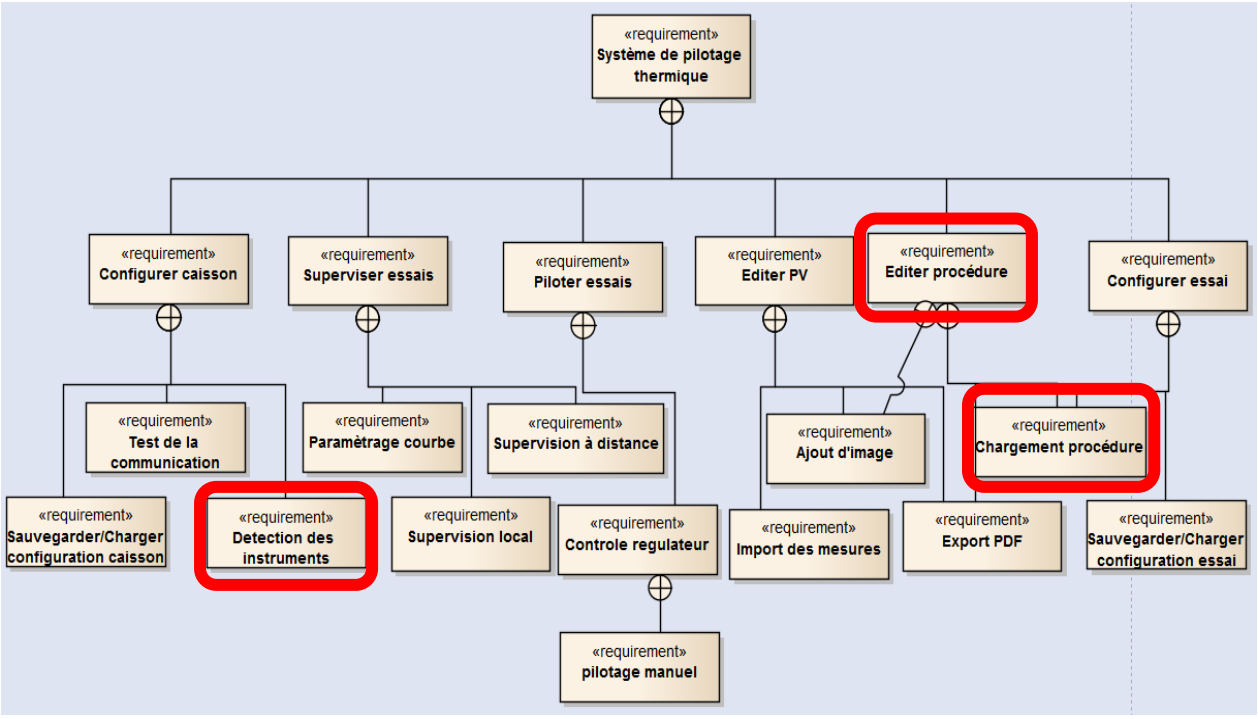


Figure 14 Diagramme d'exigences fonctionnelles

Les exigences encadrées en rouge représentent les besoins du laboratoire qui ne sont pas satisfaits aujourd'hui. Le besoin de gestion de la maintenance n'apparait pas car l'amélioration de cet outil sera réalisée dans le cadre d'un projet ultérieur.

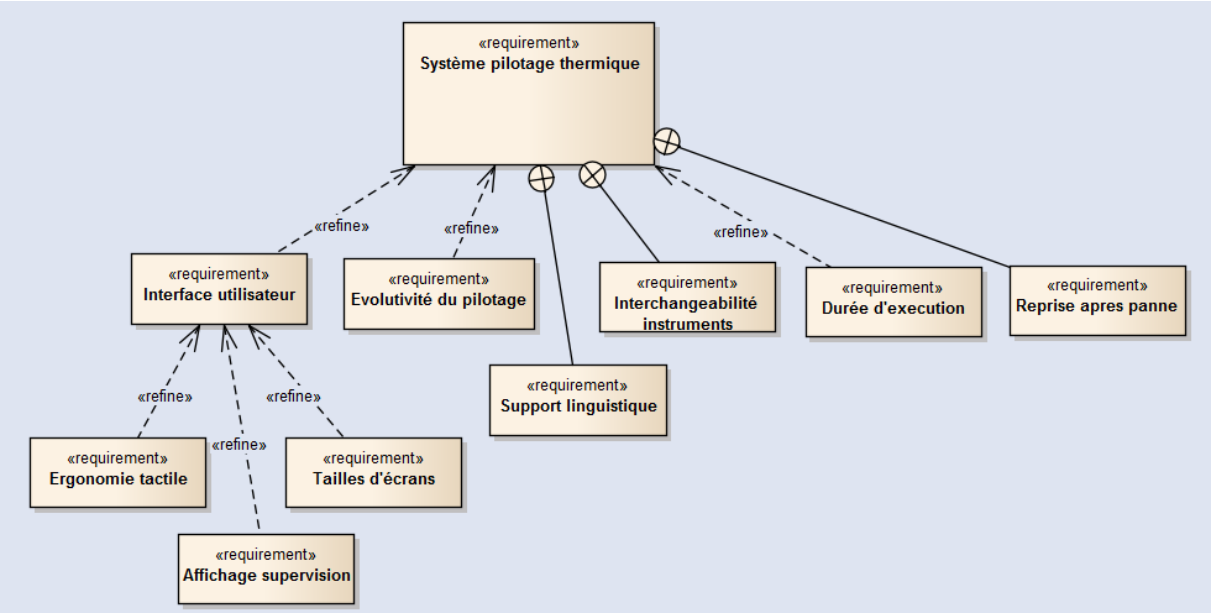


Figure 15 Diagramme d'exigences non fonctionnelles

L'étude technique des outils existants réalisée met en évidence les difficultés à maintenir et à faire évoluer les outils logiciels actuels. Ces problèmes s'expliquent, principalement, par l'utilisation de technologies différentes et obsolètes (VB6, HpVEE) et une architecture logicielle inadaptée aux besoins d'évolutivité et de maintenabilité. L'étude détaillée de ces outils a montré la duplication de fonctions sur différents composants et, dans certains cas, au sein du même composant. Ce qui complique la compréhension du code et potentiellement ses performances. De plus, l'utilisation de fichiers texte pour le stockage des données ne permet pas de garantir l'intégrité des données.

En parallèle, l'arrêt du support de Windows XP depuis le 8 avril 2014 oblige le laboratoire à mettre à jour son parc informatique vers Windows 7. Ceci ajoute des problèmes de compatibilités des outils existants.

Par conséquent, il y a un risque d'obsolescence de ces outils et un besoin urgent de refondre l'ensemble des solutions logicielles du laboratoire, en utilisant des technologies récentes qui garantissent leur compatibilité avec le nouveau parc informatique et garantis sa maintenabilité et son évolutivité par le choix d'une architecture adaptée.

L'étude organisationnelle par la modélisation des processus du laboratoire a montré que certaines tâches manuelles pouvaient être automatisées, telles que la rédaction de procédure et la configuration de l'essai. Le fait de recopier manuellement les données de la

procédure écrite en amont lors de la configuration de l'essai est une source d'introduction d'erreurs, pouvant compromettre sa bonne application.

Objectifs

A partir de l'étude de l'existant, des observations faites sur le terrain, des interviews et réunions menés, nous avons pu dégager les besoins et exigences auxquels la nouvelle solution devra répondre.

Fonctions de service du système

A partir de l'ensemble de ces données nous avons établis un cahier des charges fonctionnelles. Ce cahier des charges nous a permis d'identifier les fonctions de service du système à réaliser, qui sont :

- Réalisation d'essais thermiques :
    - Sous vide
    - Avec baie de réchauffage
- Supervision des essais :
    - En local (sur poste de pilotage)
    - A distance
- Edition de procédures d'essais
- Edition de PV d'essais

Les apports de la nouvelle solution par rapport à l'existant sont les suivants :

- Création et édition des PV essais. Simplification de la procédure actuelle.
- Extraction des données de procédure afin de préconfigurer l'essai à réaliser (ex : N° affaire, cycles à appliquer, nombre de voies de mesures, ..)
- Configuration des instruments du poste caisson simplifiée :
- Détection automatique des instruments
- Prise en charge de nouveaux instruments (interchangeabilité)
- Architecture permettant de garantir l'évolutivité afin de:
    - Modifier facilement l'algorithme de pilotage
    - Ajouter des fonctionnalités

L'ensemble des données sera stocké dans une base de données afin de garantir leur intégrité, celle-ci devra être accessible depuis l'ensemble des postes du laboratoire.

Chapitre 3 : Conception du système d'essais thermique

Dans ce chapitre nous traiterons de la conception du système à produire. Dans un premier temps nous présenterons la méthode qui a été appliquée pour réaliser cette conception, puis nous détaillerons chacune des étapes réalisées.

1. Méthode

Concernant la conception du système nous nous sommes appuyés sur les concepts de l'ingénierie système qui préconise une approche descendante. Cette approche nous permet, à partir des fonctions de services préalablement définies, de décomposer le système jusqu'à un niveau de détail permettant d'identifier les composants à réaliser.

L'étude de l'existant a permis de dégager les fonctions de services du système à produire. A partir de celle-ci nous pouvons modéliser les diagrammes de cas d'utilisation (UC) qui devront être détaillés de façon textuel. Ces cas d'utilisation doivent être validés avec les utilisateurs. Des prototypes d'IHM correspondant aux cas d'utilisation principaux ont été réalisés afin de procéder à ces validations.

Ensuite, il est nécessaire de proposer un découpage fonctionnel du système en détaillant le rôle de chaque composant de premier niveau identifié.
A partir de cette architecture fonctionnelle, nous réaliserons une recherche des solutions techniques afin de choisir les solutions adaptées aux contraintes imposées par l'environnement (OS, sécurité, qualité interne, etc..).
Une fois les choix techniques fait nous modéliserons l'architecture technique du système à réaliser ce qui achèvera la phase de conception.

2. Cas d'utilisations

Diagramme de cas d'utilisations

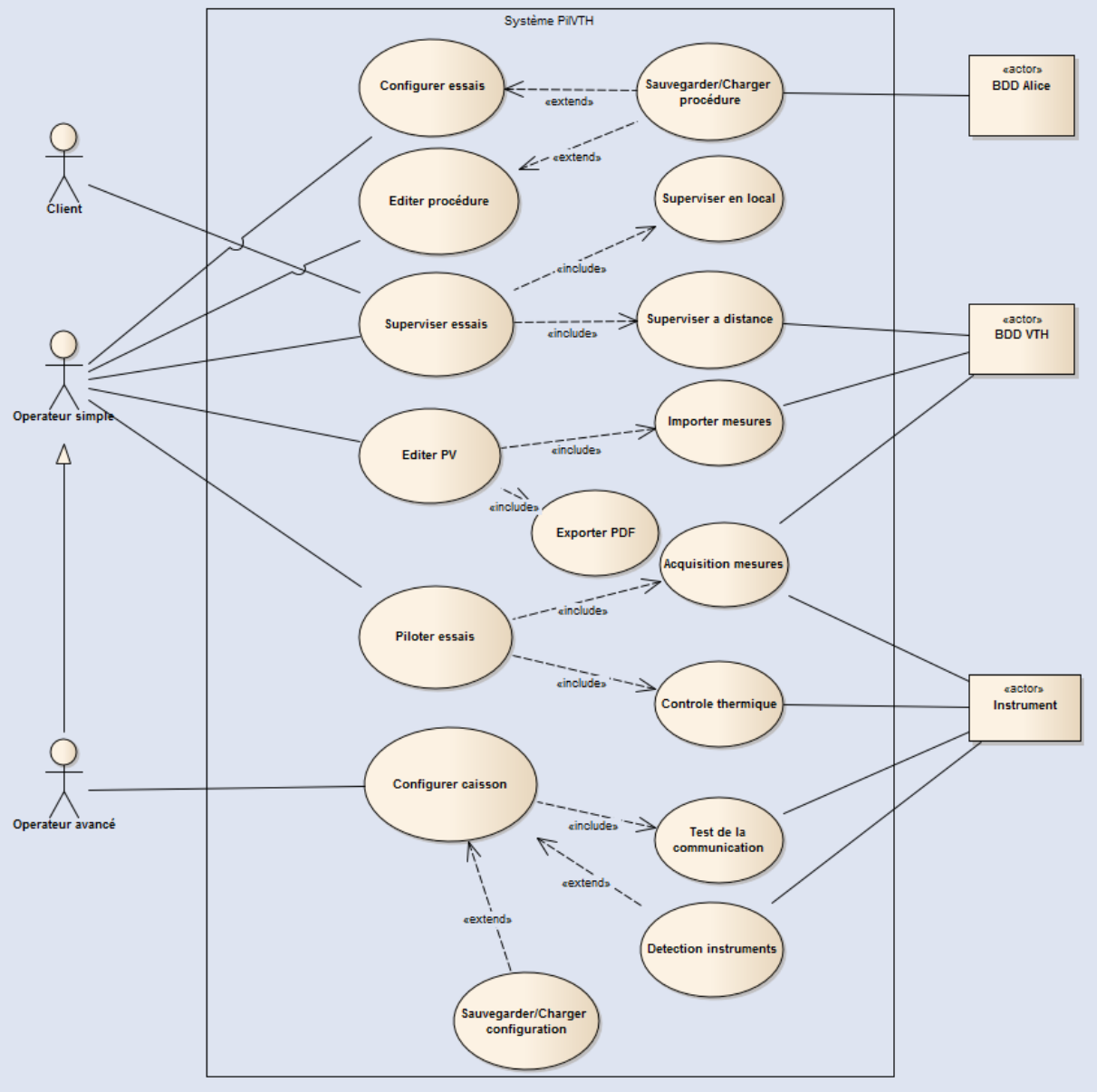


Figure 16 Diagramme de cas d'utilisation

Le diagramme de cas d'utilisation ci-dessus identifie les interactions fonctionnelles entre les acteurs et les fonctions du système à définir.

Deux types de relations sont mises en œuvre dans le diagramme : « include » lorsque que l'UC source appelle systématiquement l'UC destination pour être réalisé et « extends » lorsque l'UC source appelle optionnellement l'UC destination.

Aussi, deux types d'acteurs sont identifiés :

- Les acteurs principaux, ceux pour qui le cas d'utilisation produit un résultat observable [3].Dans notre cas ce sont les utilisateurs du système et sont:
    - Client : C'est la personne qui est responsable de l'équipement testé, son interaction avec le système se limite à la supervision de son essai en local ou à distance.
    - Operateur simple : C'est le personnel du vide thermique, il peut interagir avec toutes les fonctions du système sauf la configuration du caisson.
    - Operateur avancé : Il hérite des pouvoirs de l'opérateur simple et est capable de configurer le caisson de test.
- Les acteurs secondaires sont les acteurs qui fournissent des informations complémentaires, ils peuvent uniquement consulter ou informer le système lors de l'exécution du cas d'utilisation. Dans notre cas il s'agit d'acteurs non-humains et représentent les systèmes connectés au notre. Nous avons identifié les acteurs secondaires suivant :
    - BDD Alice : C'est une base de données qui permet de récupérer l'ensemble des fichiers associés à un équipement donné.
    - BDD VTH : C'est une base de données qui stocke l'ensemble des données de mesures des essais.
    - Instrument : Représente les instruments qui composent la baie de pilotage du caisson d'essai.

Description des cas d'utilisations

Une fois les cas d'utilisation identifiés nous avons pu les décrire textuellement afin d'identifier les scenarios possibles sur chaque cas. Ces scenarios représentent un enchainement d'actions. Ils peuvent être de deux types; nominal ou alternatif.

Le plan-type qui a été utilisé pour décrire les scenarios est le suivant :

- Numéro du scenario
- Description
- Acteurs concernés
- Préconditions
- Scenario nominal

- Scenarios alternatifs
- Exceptions
- Règles métier
- Post-condition
- Commentaires

A titre d'exemple, voici la description du cas d'utilisation configurer caisson :

ID	2
Description	Les opérateurs avancés pourront configurer les caissons de test thermiques. Le cas d'utilisation comprend la création des fichiers de configuration, la détection automatique du matériel, la configuration des appareils de mesure (sondes thermiques et de pression), des appareils de contrôle (régulateurs) et la configuration des baies de réchauffage.
Acteurs	Les opérateurs avancés pourront invoquer ce cas d'utilisation.
Préconditions	• L'opérateur est identifié. • L'opérateur connait la configuration de communication des instruments à installer. • Les drivers des instruments à utiliser sont disponibles. • L'IHM de configuration du caisson est ouverte. • L'opérateur est présent sur le poste caisson
Scenario nominal	• L'opérateur sélectionne quel type d'instrument il souhaite configurer (mesure température/mesure pression/régulateur plaque/régulateur écran/Commande et alarme/baies de réchauffage) en choisissant « Nouveau » dans la catégorie correspondante. • L'opérateur sélectionne l'instrument en le choisissant dans la liste des drivers présents ou entrant le n° d'immobilisation. • *L'opérateur entre la configuration de communication (exemple : RS232, COM2, 9600, 8,1, none).* • *Remarque : Le choix du driver (ex IVI) pourrait rendre facultative cette étape* • L'opérateur teste la communication et valide la configuration. • Le système ajoute l'instrument dans la liste des périphériques configurés sous la catégorie correspondante et met à jour le fichier de configuration caisson qui est sauvegardé localement au format XML sur les postes caissons
Scenarios alternatifs	**Modification de la configuration d'un instrument** • L'opérateur sélectionne l'instrument à éditer en le

	sélectionnant dans la liste des périphériques configurés ou en entrant le n° d'immobilisation • L'opérateur modifie la configuration de communication • L'opérateur teste la communication et valide la configuration • Le système met à jour l'instrument dans la liste des périphériques configurés sous la catégorie correspondante et met à jour le fichier de configuration caisson qui est sauvegardé localement au format XML sur les postes caissons. **Modification de la catégorie de l'instrument** • L'opérateur sélectionne l'instrument à éditer en le sélectionnant dans la liste des périphériques configurés • L'opérateur fait « glisser » l'instrument sélectionné dans la catégorie voulue par « drag and drop » • Le système met à jour l'instrument dans la liste des périphériques configurés sous la catégorie correspondante et met à jour le fichier de configuration caisson qui est sauvegardé localement au format XML sur les postes caissons. **Configuration des instruments par fichier de configuration** • L'opérateur sélectionne « charger configuration » • Le système affiche un « browser » sur le disque local pour trouver le fichier de configuration XML à charger. • Le système charge le fichier XML et affiche les instruments configurés présents • L'opérateur valide la configuration • Le système affiche un message de confirmation du type « Etes-vous sur de vouloir écraser la configuration actuelle ? » • L'opérateur choisi « oui » • Le système met à jour le fichier de configuration caisson qui est sauvegardé localement au format XML sur les postes caissons.
Exceptions	**La configuration de la communication est mauvaise** • l'opérateur teste la communication • Le système envoie un message avec le type d'erreur rencontrée **Un ou plusieurs champs n'ont pas été saisi** • l'opérateur teste la communication • Le système envoie un message d'erreur et indique le champ à compléter
Règles métier	La configuration ne sera modifiée que lorsque le matériel de

	mesure et de contrôle du caisson sera changé
Post-condition	La configuration est sauvegardée sur le disque local pour chaque caisson de simulation
Commentaires	Possibilité d'évolution en cours du projet suivant les retours des utilisateurs.

Les descriptions textuelles de l'ensemble des cas d'utilisation sont fournies à l'annexe 1.

Prototypage IHM

Au cours de cette phase nous avons réalisé le prototypage des IHM en nous basant sur les cas d'utilisations définis, l'existant et les besoins capturés lors de son l'analyse. Afin de présenter le comportement souhaité du système pour chacun des cas d'utilisation je me suis appuyé sur les prototypes réalisés pour échanger et valider les cas d'utilisations avec les futurs utilisateurs.

Ci-dessous le prototypage d'un des écrans de configuration du caisson réalisé avec l'outil open-source Pencil[5] :

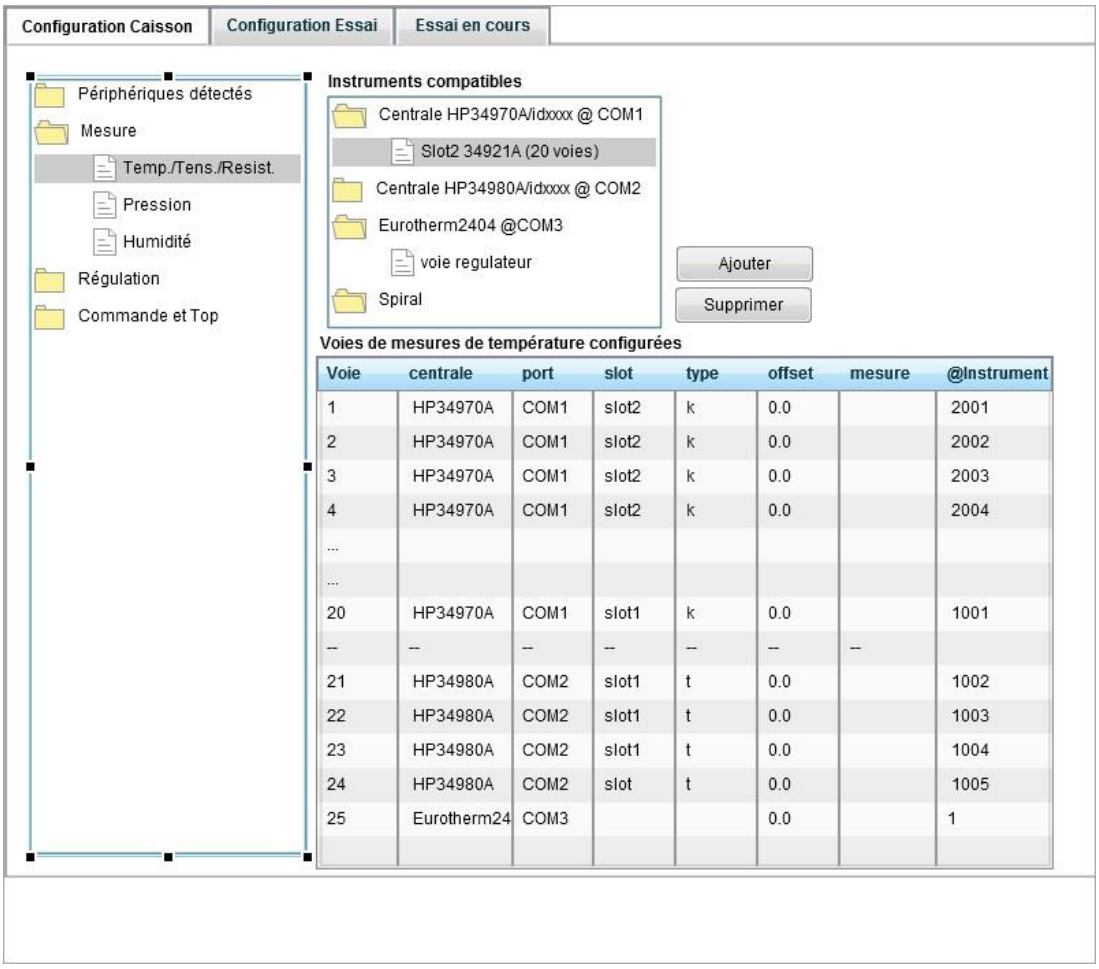


Figure 17 Prototype IHM: configuration caisson

L'utilisation de prototypes d'IHM pendant les réunions avec les futurs utilisateurs est, d'après mon expérience, un bon moyen d'impliquer davantage l'audience. En effet, les utilisateurs se projettent plus facilement dans la manipulation du futur système. Par conséquent, il est plus facile d'échanger sur les aspects fonctionnels en partant d'un support visuel.

L'ensemble des prototypes réalisés est disponible à l'annexe 2.

3. Architecture fonctionnelle

L'identification et la description des cas d'utilisation a permis d'identifier tous les services attendus du système à réaliser. En se basant sur ces services nous avons définis l'architecture fonctionnelle du système et identifier trois composants principaux qui sont :

- L'outil caisson : C'est le composant qui est dédié au pilotage des moyens.
- L'outil supervision : C'est le composant qui permet de superviser les essais à distance.
- L'outil bureautique : C'est le composant qui permet de créer les procédures et les PV d'essai.

La représentation de l'architecture fonctionnelle globale du système est disponible ci-dessous :

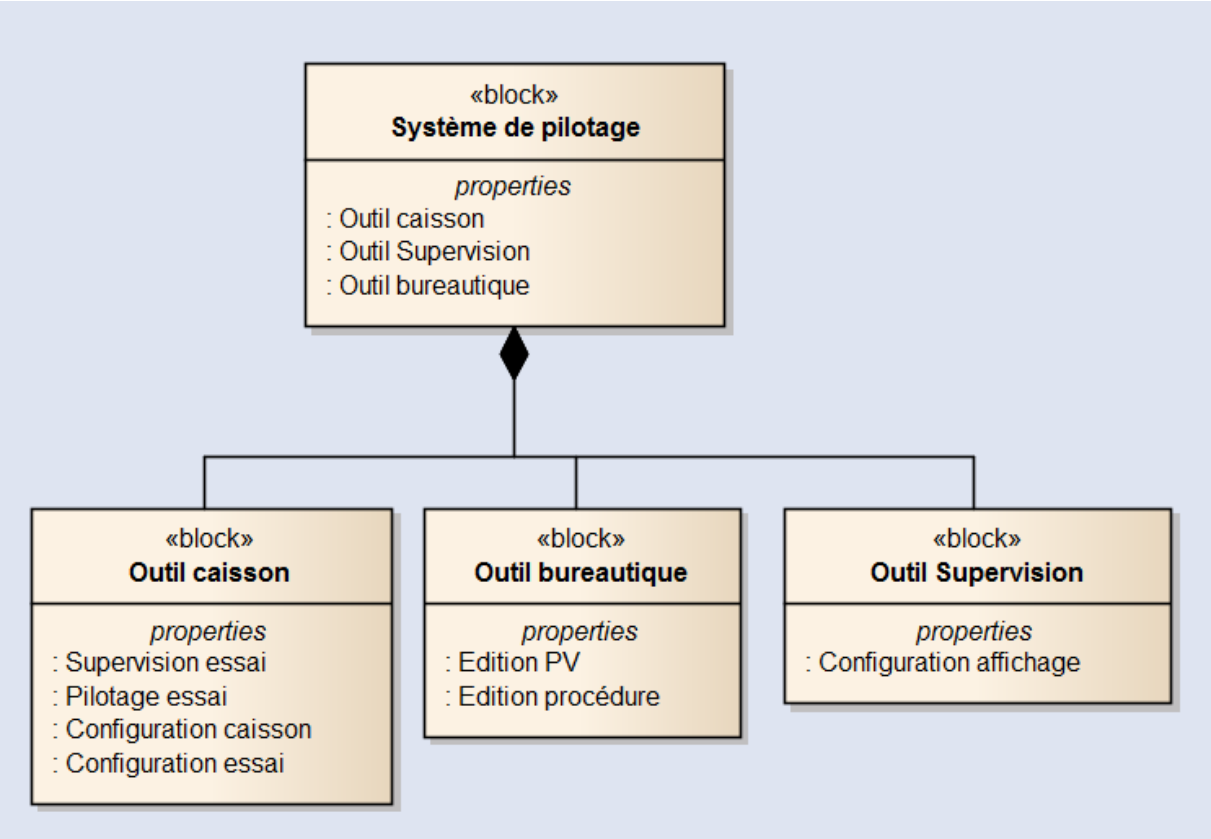


Figure 18 Architecture fonctionnelle globale du système

Une fois l'architecture globale définie nous pouvons fournir un premier niveau de détail des composants identifiés.

Outil caisson

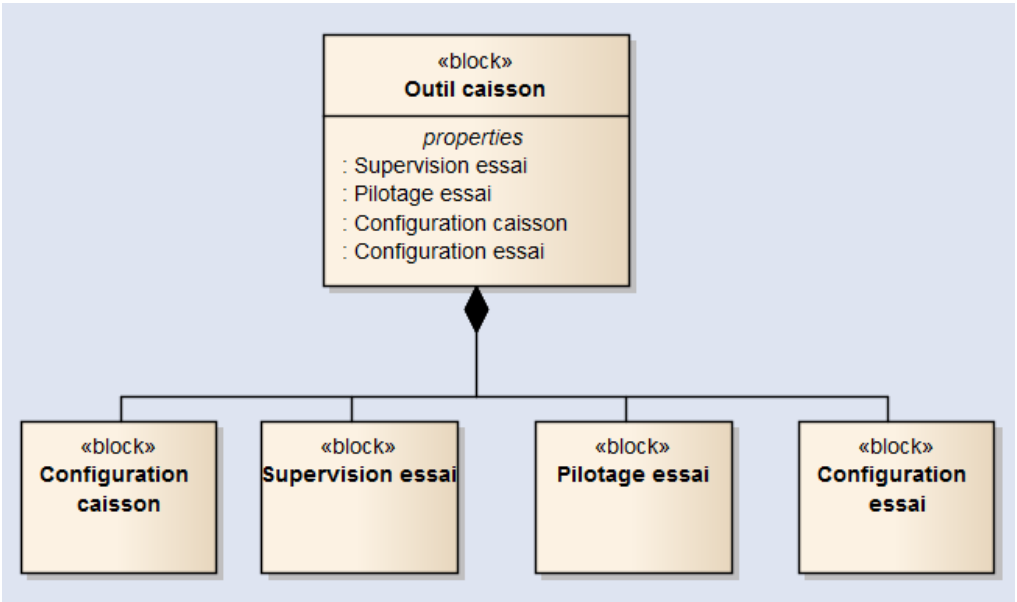


Figure 19 Architecture fonctionnelle: outil caisson

L'outil caisson offre toutes les fonctionnalités nécessaires au lancement d'un essai sur un moyen de test (caisson). Comme nous pouvons le constater dans la Figure 19 ci-dessus, les fonctionnalités sont les suivantes :

- Configuration caisson : Ce composant permet à l'utilisateur de sélectionner les instruments qui sont utilisés sur le moyen. Il permet également d'enregistrer la configuration actuelle ou d'en charger une existante.
- Supervision essai : Ce composant permet d'afficher à l'utilisateur les données mesurées en temps réel sous forme de courbe.
- Pilotage essai : Ce composant permet le contrôles des moyens afin d'exécuter l'essai qui a été configuré.

  Configuration essai : Ce composant permet de saisir les informations de l'essai à lancer, de configurer les voies et la stratégie de pilotage à utiliser.

Outil bureautique

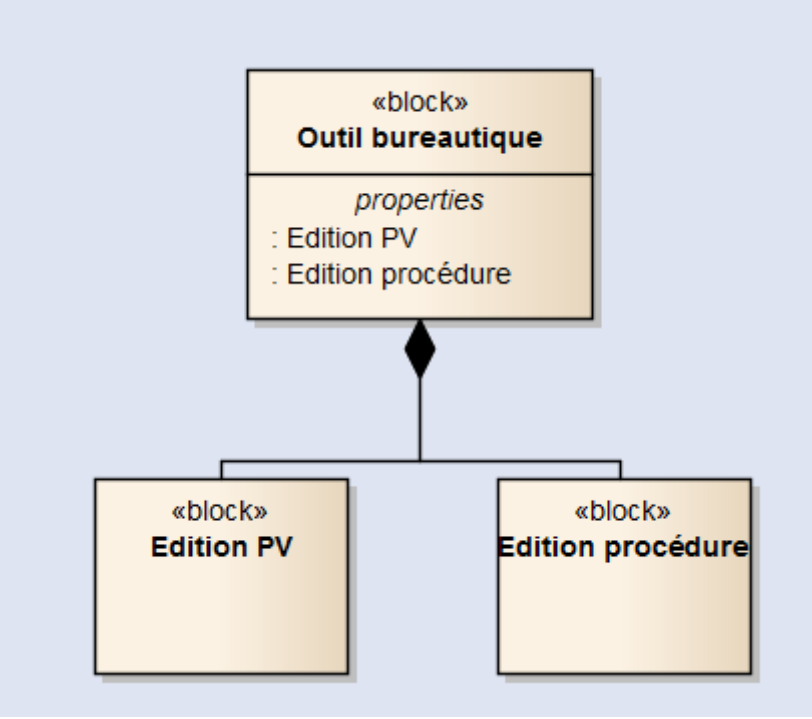


Figure 20 Architecture fonctionnelle: outil bureautique

L'outil bureautique permet à l'opérateur de rédiger les procédures d'essais et les PV essais à fournir au client. Comme nous pouvons le constater dans la Figure20, ci-dessus, les fonctionnalités sont les suivantes :

- Edition procédure : Ce composant permet de rédiger la procédure qui doit être validée par le client. Cette procédure est exportée dans un format imprimable et dans un format qui permet son exploitation par l'outil caisson afin de préparer l'essai.
- Edition PV : Ce composant permet de rédiger le PV qui contient les essais réalisés pour un équipement. Il est généré à partir des données essais et est à destination du client.

Outil de supervision

L'outil de supervision permet à l'opérateur de surveiller l'ensemble des caissons en temps réel. L'outil permet la configuration de l'affichage afin de permettre à l'opérateur de sélectionner les caissons qu'il souhaite surveiller.

4. Choix techniques

Une fois l'architecture fonctionnelle définie, nous nous sommes intéressés aux solutions qui permettront de la réaliser. Le projet n'impose pas les technologies à mettre en œuvre, par conséquent, nous avons étudié toutes les solutions techniques qui permettent la réalisation du système tout en respectant les exigences.
Cette partie présente la synthèse de cette recherche en classant les solutions dans deux catégories : les solutions sur mesures et les solutions dites « clés en main ».

Critères de sélection

Les solutions envisageables permettant la réalisation du système sont nombreuses. Ainsi, il est nécessaire d'effectuer une recherche en tenant compte des contraintes afin de trouver le meilleur compromis possible.

Afin de sélectionner et d'évaluer les différentes solutions nous avons établis une liste de critères. Ils cadrent le choix des solutions sur les aspects de planning, de coût, de maintenabilité et de technologies mises en œuvre. Aussi, le langage de développement choisi dans le cas d'un développement complet du système doit répondre aux critères suivant :

- Etre un langage répandu, afin d'assurer la compréhension du code par un grand nombre de développeurs.
- Etre un langage « haut-niveau » et orienté objet de préférence, pour une lecture facile du code par toute personne ayant quelques bases de développement logiciel.
- Etre compatible avec les interfaces de communications utilisées par les instruments.

Les solutions sur mesure

Pour estimer la faisabilité des différentes solutions permettant de créer un système sur mesure, c'est-à-dire un développement complet du système, nous avons choisi d'étudier deux langages de développement ; Java et C# .NET .

Java, dans un premier temps, car c'est un langage répandu qui est maitrisé par la majorité des développeurs. Puis, dans un dans un second temps, C# .NET qui est un langage très proche de Java en reprenant sa syntaxe générale et ses concepts.

Nous avons réalisé des prototypes permettant de valider les fonctionnalités suivantes :

- communication instrument par RS232, GPIB, USB, LAN
- affichage des mesures sous forme de courbes
- stockage des données de mesures en XML

Solution Java

Afin de valider les critères définis précédemment, nous avons commencé par rechercher les librairies permettant la communication avec les instruments du laboratoire. Ces librairies sont les suivantes :

- **RxTx** : C'est une librairie open-source Java qui utilise du code natif (via JNI) et qui permet la mise en œuvre des communications via ports série et parallèle[6].
- **Java Simple Serial Connector (JSSC)** : C'est une librairie open-source Java qui permet la mise en œuvre de communication via ports série[7].
- **JPIB** : C'est une librairie open source Java qui permet la communication via les ports GPIB spécifique aux instruments Hp/Agilent[8].

Ensuite, nous avons fait une recherche des librairies disponibles pour l'affichage de données sous forme de graphique :

- **jFreeChart** : C'est une librairie open source Java qui permet l'affichage de données sous forme de graphiques[9].
- **TeeChart** : C'est une librairie disponible en plusieurs langages (.NET, Java, ActiveX, Php, HTML5). C'est la même librairie qui est utilisée sur les logiciels de supervision et de pilotage existants[10].

Et enfin nous avons comparé les librairies de traitement de fichier XML ; jDOM, SAX et StAX.
Pour réaliser cette étude nous avons développé des prototypes d'affichage des données de communication avec les instruments, ainsi qu'un utilitaire de configuration des instruments de mesure pression.

Les recherches autour des systèmes de contrôle et d'acquisition de données couramment appelé SCADA (Supervisory Control And Data Acquisition) m'ont orienté vers un projet open-source, iControl [11].

iControl

iControl est un projet Open Source réalisé par Kurt Pernstich (National Institute of Standards and Technology) permettant l'automatisation d'acquisitions de données depuis des instruments de mesures. Celui-ci permet de communiquer via RS232 et GPIB et utilise les librairies jFreeChart et RxTx présentées précédemment.
Cette solution offre de nombreux avantages : le projet est très bien documenté, il est possible d'ajouter de nouveaux instruments en héritant d'une classe instrument générique ou en utilisant des fichiers de définition d'instrument, ce qui permet l'ajout de drivers sans recompilation de la solution et enfin l'auteur du projet s'est montré disponible pour répondre à l'ensemble de mes questions.

Pour évaluer cette solution, nous avons implémenté un driver RS232 permettant la mesure de température avec l'Agilent34970A. Puis, nous avons réalisé des tests pour évaluer les performances.
Le premier test a consisté à mesurer 3 voies de mesure à une fréquence de 10 secondes sur une durée d'environ 10 minutes. Ce test s'est déroulé sans problèmes mais nous avons

remarqué que la fréquence dérivait dans le temps et que le rafraichissement du graphique prenait environ 1 seconde. La Figure 21, ci-dessous montre l'aperçu du graphique.

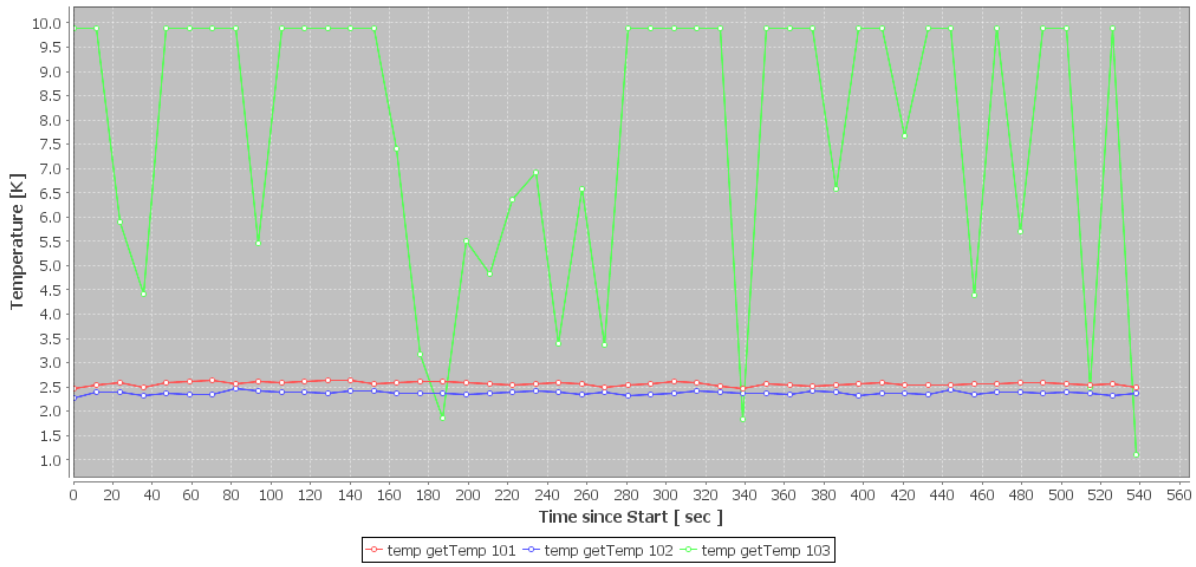


Figure 21 Affichage courbes iControl

Le deuxième test a consisté à mesurer 12 voies de mesures sur une durée d'environ 60 heures. Celui-ci a montré que la librairie d'affichage des courbes était plantée mais que l'acquisition de donnée continuait à fonctionner.

Enfin, nous avons réalisé un troisième test similaire au deuxième en remplaçant la librairie jFreeChart par TeeChart. Ce test a montré qu'au bout de trois jours de mesures TeeChart ne présentait aucun ralentissement. Ainsi, nous en avons conclu que l'utilisation de la librairie RxTx, pour la communication série, et TeeChart, pour l'affichage des courbes, répond à nos exigences de performance. Par conséquent, il est envisageable de faire un fork du projet iControl afin de créer l'outil de pilotage du laboratoire.

Solution C# .NET

Le langage C# .NET est un langage créé par Microsoft afin que la plateforme Microsoft .NET soit dotée d'un langage permettant d'utiliser toutes ses capacités. Le framework Microsoft .NET fourni une librairie permettant d'exploiter les capacités de Windows, qui est le système d'exploitation utilisé sur l'ensemble des postes du laboratoire.

Comme pour l'étude du langage Java, nous avons recherché les librairies permettant de valider la communication, la validation et le stockage des mesures.

Les librairies de communication qui ont été validées sont les suivantes :

- **VISA** : Visual Instrument Software Architecture est un standard de communication utilisé par de nombreux constructeurs et dans notre cas pour les instruments de mesure Hp/Agilent. L'utilisation de l'API VISA permet de s'abstraire de type de liaison utilisé par l'instrument tel que GPIB, Serie, LAN, USB[12].
- **NModbus 4** : C'est une librairie qui implémente le protocole Modbus sur des liaisons séries et TCP. Elle permet la communication des instruments compatibles, dans notre cas les régulateurs thermiques de type Eurotherm.

Pour la communication avec les autres instruments basés sur la liaison Série, la classe SerialPort est fourni par le framework .NET

Enfin, compte tenu des tests réalisés avec la librairie TeeChart Java nous avons choisis de réutiliser cette même librairie pour l'affichage de courbe avec C# .NET.

Enfin, au vu du durcissement des règles de sécurités au sein du groupe Thalès, qui concernent notamment l'installation d'application sur les postes bureautique, C# est un bon « candidat » étant donné que les mises à jour du framework .NET sont gérés par DSI.

Les solutions clés en main

Les solutions dites, « clés en main » regroupe les solutions standardisées adaptables, c'est-à-dire les solutions qui répondent aux fonctions principales de notre système et que l'on peut adapter pour répondre aux besoins du laboratoire.

Intespace Dynaworks/Dynatherma

Dans un premier temps nous avons contacté la société Intespace, basée à Toulouse et qui fournis un ensemble de moyens d'essais de simulation d'environnement dans le domaine du spatial, de l'aéronautique et de la défense. Ils proposent des services d'essais mécaniques, acoustiques, thermiques et électriques tout comme les moyens d'essais de TAS. Intespace a déjà collaboré avec les moyens d'essais de TAS afin de mettre à disposition ses moyens d'essais. Afin d'assurer l'ensemble de ces activités Intespace a développé et met à

disposition des clients un logiciel de gestion et de traitement des données, Dynaworks. De plus, cette solution est déjà utilisé au sein de TAS pour la simulation de l'environnement et pour la gestion des essais au centre d'essai de basé à Canne.

Cette solution est complète et couvre l'ensemble des activités du laboratoire de la préparation de l'essai jusqu'à la génération du PV. A partir de notre cahier des charges une proposition technique nous a été présentée. Elle nécessite la mise en place d'un sous réseau par salle d'enceintes afin de sécuriser le fonctionnement sur chaque bâtiment. Chaque sous réseau est composé :

- D'un serveur de BDD et d'un redondant qui permet le traitement et l'analyse des données ;
- D'un poste d'acquisition et de supervision ainsi qu'un redondant par baie de test ;
- D'un serveur d'historisation des données qui permet la supervision des essais à distance grâce à la mise en place d'une application graphique distribuée.

La Figure 22, ci-dessous, présente l'architecture physique proposée.

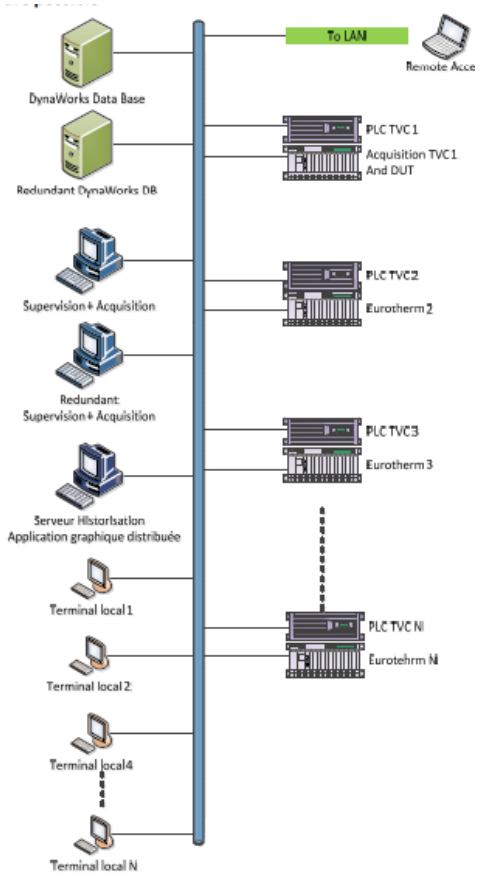


Figure 22 Architecture physique Dynaworks

D'un point de vue fonctionnel cette solution répond à tous nos besoins et un support est prévu en cas de problème dans l'offre. Cependant ses interfaces utilisateurs ne sont pas modifiables dans la version présentée (v6) et la configuration de l'outil pour réaliser l'ensemble des activités du laboratoire est plus complexe que l'existant car davantage de possibilités sont disponibles. Il y a, par conséquent, une formation (lourde ?) à faire aux utilisateurs. Autres inconvénients, la solution est compatible uniquement avec les instruments de type Hp/Agilent supportés par la librairie Visa, pour la prise en charge de nos instruments il est nécessaire d'adapter la solution ce qui entraine un surcoût non-négligeable. Ensuite, la solution présentée est compatible uniquement avec les distributions Linux pour la partie pilotage thermique, ce qui nécessitera le déploiement d'un nouvel OS sur tous les postes de pilotage.

Enfin, cette solution est d'un point de vue fonctionnel et financier « trop luxueuse » pour les besoins du laboratoire et le coût de cette solution dépasse largement le budget alloué au projet. Enfin, cette solution nécessite des changements important au niveau de l'architecture physique, c'est pourquoi cette solution ne sera pas retenue.

Wonderware

Wonderware est une société proposant des solutions logicielles pour la réalisation d'application industrielle. Cette société a distribué environ un million de licences à travers le monde en une vingtaine d'années, ce qui lui permet de prétendre à une position de leader mondial sur le marché des applications industrielles.

A notre demande, une présentation des possibilités offertes par leurs solutions nous a été faite par Mr Jean-Phillipe JOLY, responsable des ventes Sud-Ouest Wonderware.

La société propose des solutions de supervision industrielle en offrant une bibliothèque d'objets métiers et d'outils graphique accessibles depuis un environnement de développement.

Un devis nous a été fait sur une solution qui couvre nos besoins en terme de stockage de données et les outils nécessaires au développement du système en interne. Bien que cette solution semble répondre à nos besoins, et notamment à nos contraintes de maintenabilité et d'évolutivité, nous ne l'avons pas retenu, le coût de la licence par poste étant jugé trop

élevé. De plus, l'ensemble du développement étant à réaliser en interne il y a un risque de ne pas respecter le planning prévu initialement.

Produit TAS : JAMES

JaMeS est un outil développé en Java au sein de Thalès Alenia Space. Cet outil a été réalisé afin de répondre aux besoins d'automatisation et de supervision des essais sur charge utile de plusieurs laboratoires. L'architecture de ce logiciel permet d'être adapté à différents contexte d'essais.

Une phase de spécification de la solution basée sur cet outil a été initialisée avec l'équipe de Mr Follet, responsable du produit JaMeS. Le cahier des charges et la spécification fonctionnelle du système à réaliser leur a été transmis dans le but de fournir une proposition technique et commerciale.

Bien que cette solution convienne sur les aspects fonctionnels, de planning et de maintenance, son coût dépassant le budget alloué fait que nous n'avons pas nous engager pour la phase de réalisation.

Solution choisie

Les solutions telles que Dynaworks et Wonderware, bien qu'elles semblent adaptées à nos besoins et contraintes, ont des tarifs qui sont proportionnels au nombre de caissons mis en œuvre par nos moyens. Après calcul, le coût dépasse le budget prévu et l'éventuelle distribution de la solution dans d'autres centres d'essais n'est pas envisageable pour cette même raison. La solution JaMes qui a retenu notre attention n'a pas été choisie car la tarification finale de la solution, réalisée après une première phase de spécification, dépassait également le budget prévu. Le développement sur mesure du système avec les technologies C# .NET ou JAVA est plus adaptées car cela permet d'avoir une meilleure maitrise du coût et de fournir une solution en adéquation avec les besoins du laboratoire. Cependant la compatibilité des librairies VISA avec C# .NET et les mises à jour de la librairie .NET prises en charge par DSI sont des atouts qui nous ont permis de sélectionner cette technologie.

Par conséquent, nous nous sommes orientés vers une solution qui implique le développement complet du système en se basant sur les technologies C# .NET. Ne pouvant

pas assumer la charge de développement seul au vu du temps restant nous avons fait appel à la société Altran basée à Blagnac afin de sous-traiter la réalisation de la solution. Afin de réduire au maximum le coût de réalisation de la solution nous avons demandé à ce qu'une partie du développement soit à ma charge.

Le chapitre suivant présente l'organisation de la réalisation de la solution avec Altran et les développements à ma charge.

Chapitre 4 : Réalisation du système

Cette dernière partie portera sur la réalisation du système et sur l'organisation de cette phase conduite en utilisant les aspects de la méthode agile. Ces aspects d'agilité seront abordés dans une première partie ou nous traiterons les problématiques liées à la conception architecturale « agile ». Ensuite, nous présenterons l'organisation de l'équipe projet en termes de responsabilités et de rôles. Puis, nous détaillerons comment nous avons résolu la problématique d'interchangeabilité des instruments que nous avons identifiés et comment elle a été implémenté par l'implémentation des composants de communication. Enfin, nous détaillerons la conception du composant de pilotage et la façon dont celui-ci met en œuvre les qualités d'évolutivité et de maintenabilité.

1. Conduite du projet

Dans cette partie nous traiterons de la gestion de la réalisation du projet avec Altran basé sur Scrum. Je présenterai rapidement les concepts généraux de l'agilité et la place de l'architecture dans les méthodes agiles.

Méthodes agiles

Les méthodes agiles sont des ensembles de pratiques adaptées aux projets de développement en informatique. Elles ont un référentiel commun, appelé le manifeste agile, qui définit quatre valeurs fondamentales à partir desquelles douze principes généraux ont été déclinés, ces valeurs sont les suivantes[13] :

- **« Les individus et leurs interactions plus que les processus et les outils »**

La communication et les relations entre les personnes réalisant le développement est un aspect essentiel.

- **« Des logiciels opérationnels plus qu'une documentation exhaustive »**

L'agilité met l'accent sur le logiciel en lui-même plutôt que sa documentation, qui peut être une charge importante. Une pratique en agilité et de fournir un code abondamment commenté.

- **« La collaboration avec les clients plus que la négociation contractuelle »**

Il est important que le client intervienne tout au long du projet plutôt que de négocier un contrat en début du projet. Cela dans l'objectif de valider l'adéquation des développements réalisés avec ses attentes.

- **« L'adaptation au changement plus que le suivi d'un plan »**

Il est nécessaire que la planification et la structure du système à réaliser soient flexibles afin de permettre des demandes de modifications du client tout au long du projet.

La méthode agile choisie par Altran pour le projet se base sur l'organisation définie par Scrum. Celui-ci défini un cadre de travail qui découpe le développement en plusieurs itérations appelées « sprints ».
Dans le cadre de notre projet la durée d'un sprint est fixée à deux semaines. Chaque sprint commence par une réunion durant laquelle le client et le responsable du développement en définissent le but, c'est-à-dire les fonctionnalités à fournir durant le sprint. Ces fonctionnalités sont « tirées » du backlog qui contient l'ensemble des fonctionnalités priorisées du produit à fournir.
A l'issu d'un sprint, une réunion de validation, en présence de l'équipe scrum et des parties prenantes, permet de valider l'incrément réalisé durant cette itération.

L'agilité dans le développement d'un projet met l'accent sur la qualité externe du produit, c'est-à-dire ce que le client va percevoir. Ainsi, la notion d'architecture logicielle qui concerne davantage l'aspect interne du produit n'est pas toujours évidente à mettre en place en agilité.

Architecture et agilité

L'architecture, dans le cadre de notre développement, est très importante car elle va permettre au système de garantir le respect des contraintes de maintenabilité et

d'évolutivité. Comme vu précédemment, du fait que l'agilité se concentre davantage sur la qualité externe, la notion d'architecture n'est pas facile à mettre en place.

A l'inverse des méthodes agiles, les méthodes de développement « classiques » définissent en amont du projet l'architecture complète du système à réaliser et, de ce fait, rend tout changement dans la solution compliqué car cela impose de revenir sur des phases de conception architectural et de modifier la documentation associée.

Dans le cadre d'un développement agile, une solution envisageable est de planifier dans les premiers sprints des activités de conception architecturale de haut-niveau, cette conception doit être réalisée par un architecte ou des parties prenantes ayant les connaissances techniques nécessaires ainsi qu'une vision globale du système à produire. Ces activités réalisées au cours du premier sprint permettent :

- De faire les principaux choix logiciels et matériels
- D'identifier les composants ou services réutilisables
- De produire de diagrammes de haut niveau

Dans le cadre du projet de réalisation du système de pilotage, une architecture fonctionnelle et des choix techniques ont été suggérés en amont. Ainsi, les activités dans le premier sprint furent de valider l'architecture fonctionnelle et les choix techniques proposés. Cela a permis d'établir une architecture logicielle de la solution à réaliser qui tient compte des exigences et qui permettent d'être adapté aux éventuelles demandes changement.

2. Organisation du projet

Cette partie présente les aspects organisationnels du projet. Ces aspects concernent la composition de l'équipe projet, le planning de réalisation et les outils utilisés.

Equipe projet

L'équipe projet est constituée de personne de TAS et Altran qui se répartissent les rôles définis dans Scrum :

- Le propriétaire du produit, ou product owner, qui est le représentant des clients et des utilisateurs.
- Le scrum master, qui est responsable de la compréhension et de la mise en œuvre de la méthode Scrum.
- L'équipe de développement, qui est responsable de la livraison d'une itération en adéquation avec ce qui a été défini en début de sprint.

Le tableau ci-dessous montre la répartition des rôles

Tableau III : Equipe projet scrum

Equipe projet	Rôles
Luc FERRIE (TAS)	Product Owner
Anthony SIMONIGH (TAS)	Assistant Product Owner/ Equipe dev.
Eric BABOIN (Altran)	Scrum master
Nicolas ROUS (Altran)	Equipe dev.

Planning

Le projet a été découpé en 10 sprints d'une durée de 2 semaines à 1 mois en fonction du contenu du sprint et de la disponibilité des membres de l'équipe projet. Les sprints planifiés ont été définis comme suit :

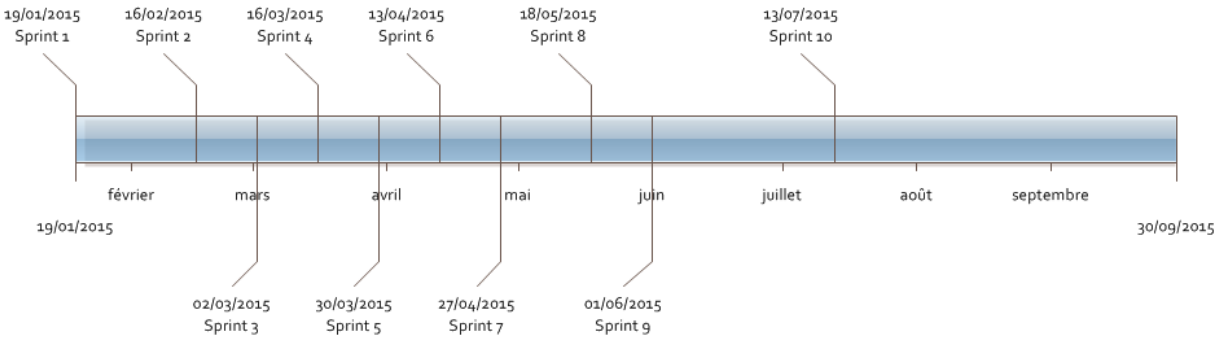


Figure 23 Planning sprints de développement

Outils

Pour répondre aux besoins de communication, de travail collaboratif et de développement, nous avons utilisé les outils mis en place par Altran :

- **Redmine** : C'est une application web libre de gestion de projets qui nous a permis de fournir les outils de gestion adaptés à l'application de la méthode Scrum. Cette application nous a permis de définir les différents sprints et leur contenu pendant les réunions, d'éditer les fonctionnalités et suivre l'avancement des tâches.
- **SVN** : C'est un logiciel de gestion de version qui a été utilisé pour la gestion du code source. Il fonctionne sur le mode client-serveur avec un dépôt centralisé sur un serveur et des logiciels clients installés sur les postes, ceux-ci recopient les fichiers depuis le serveur.

  Microsoft Visual Studio 2013 : C'est l'environnement de développement Microsoft qui permet de développer le système en C# .NET.

3. Architecture logicielle

L'architecture logicielle a été réalisée avec l'équipe de développement Altran. A partir de l'étude des solutions que nous avons faite en amont, nous avons pu valider le choix de la technologie C# .NET et des librairies testées. Afin de respecter l'exigence qui prévoit de réaliser une IHM adaptée à une utilisation tactile et clavier/souris, nous avons étudiés les différentes technologies d'interfaces. Ainsi, nous avons choisi d'utiliser la technologie WPF qui est plus puissante que son prédécesseur Winform, grâce à l'exploitation des capacités de la carte graphique et la proposition d'interfaces similaires à ce que l'on retrouve sur les dernières versions de Windows.

La mise en place de cette technologie permet également de bien découpler la vue de la partie fonctionnelle par l'utilisation d'un langage déclaratif pour la réalisation des vues (XAML) et la mise en place du pattern d'architecture adaptée qui est MVVM (Modèle-Vue-VueModèle).

Choix du pattern architectural MVVM

L'utilisation de la technologie WPF pour la réalisation des interfaces utilisateurs oriente le développement vers le modèle MVVM qui est le plus adapté au vu des technologies mises en œuvre.

Le modèle d'architecture MVVM est une variation du modèle MVC où la réalisation de la Vue est adaptée et découplée du reste de l'application afin d'être réalisée par un designer plutôt qu'un développeur.

Le designer étant davantage concerné par les aspects graphiques et ergonomiques que par l'écriture du code, l'implémentation de la vue est réalisée dans un langage déclaratif qui est XAML dans le cas des technologies WPF. Tout comme MVC, MVVM est un pattern adapté à la réalisation de système interactif et qui permet de bien séparer les responsabilités de chaque composant.

MVVM sépare le composant en trois composants ; Modèle, Vue et VueModèle, les responsabilités des composants dans le cas des technologies Microsoft sont les suivantes:

- Le modèle, comme dans le pattern MVC, correspond aux données et à la logique métier permettant d'accéder ou de produire ces données. Celui-ci est réalisé en C# et n'a pas connaissance de l'interface graphique qui lui est associée.
- La vue est constituée des éléments graphiques et gère les interactions avec les utilisateurs. Pour présenter les données à l'utilisateur, le databinding est utilisé et permet de lier (binder) les objets du modèle à la vue. Dans les cas simple la vue peut binder directement le modèle à condition que celui-ci présente les propriétés attendues par la vue. Les vues sont composées d'un fichier .xaml pour l'aspect graphique et d'un fichier .xaml.cs, aussi appelé code behind, qui permet d'instancier la classe vue-modèle et le datacontext (pour le databinding).
- VueModèle est l'adaptateur entre le modèle et la vue, il permet l'adaptation des types du modèle en types exploitables par la vue via le databinding et expose les commandes que la vue utilisera pour interagir avec le modèle. La classe vue-modèle expose des propriétés qui sont directement exploitables depuis le fichier XAML[14].

La Figure 24, ci-dessous, montre les relations entre les différents composants du pattern.

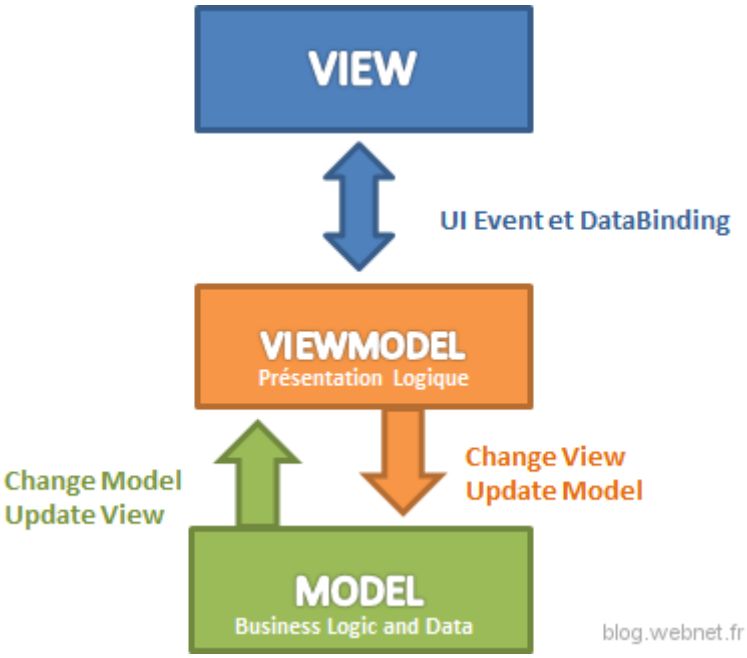


Figure 23 Modèle MVVM

Architecture technique

Une fois les choix techniques validés nous avons pu déterminer l'architecture technique du système à réaliser. Le système est constitué de trois composants principaux ; VTH_Caisson, VTH_Desk, VTH_Supervision et VTH_Lib et est structuré de la manière suivante :

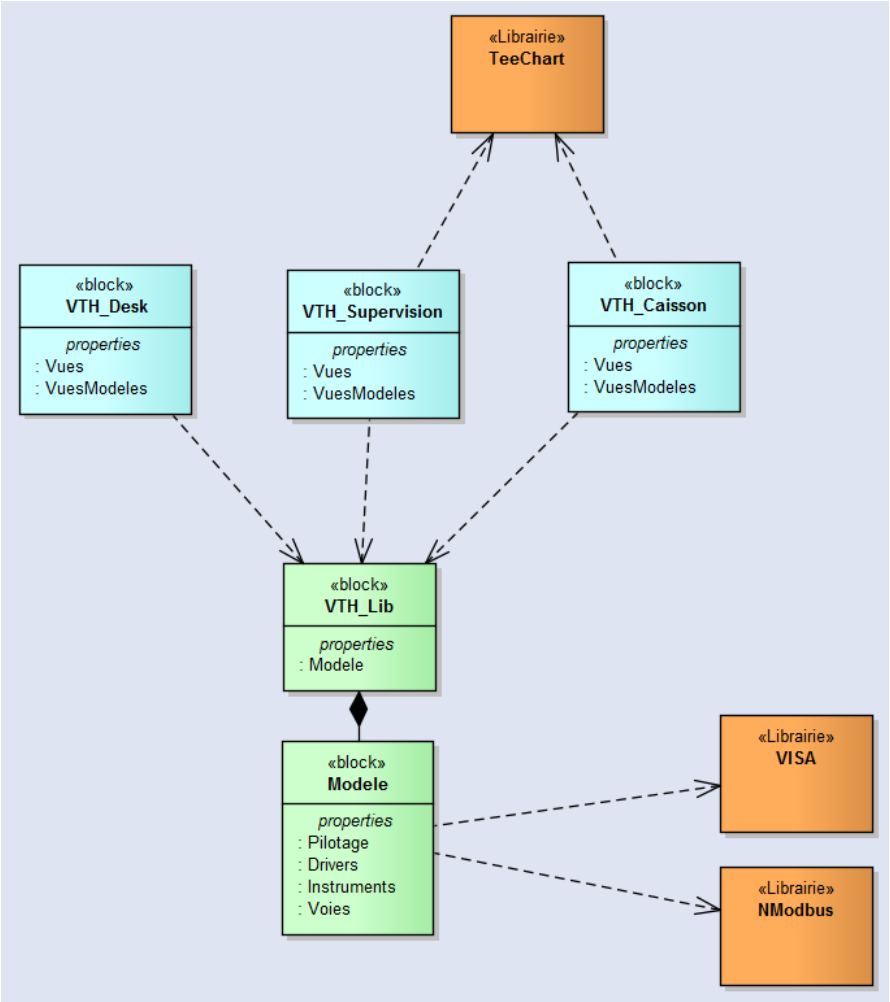


Figure 24 Architecture technique

La Figure 25, ci-dessus, montre l'architecture technique globale de la solution et fait apparaitre les dépendances avec les librairies testées en amont du projet.

Les blocs bleus correspondent aux trois outils qui composent le système précédemment définis dans l'architecture fonctionnelle :

- **VTH_Desk :** C'est l'outil qui permet à l'utilisateur de générer PV et Procédure.
- **VTH_Supervision** : C'est l'outil qui permet de superviser à distance l'ensemble des caissons en temps réels.
- **VTH_Caisson** : C'est l'outil installé sur les postes de pilotage et qui permet de réaliser les essais.

Ces trois exécutables contiennent uniquement les vues et les vues-modèles associées, c'est-à-dire les interfaces utilisateurs et la logique de contrôle qui est liée à l'IHM. Ces composants référencent VTH_Lib qui fournit les données à afficher.

VTH_Lib implémente toute la logique métier et les données affichées par les différentes interfaces. Il contient notamment ; l'implémentation des algorithmes de pilotage, les drivers de communication, la définition des instruments et les voies de mesures associées.
La librairie TeeChart est référencée par VTH_Supervision et VTH_Caisson afin d'afficher le suivi des mesures sous forme de courbe en temps réel.
VISA et NModbus sont les librairies utilisées pour la communication avec les instruments et sont référencées dans VTH_Lib.
Cette architecture technique a permis de structurer le développement et de pouvoir repartir la réalisation des composants ainsi identifiés entre Altran et moi-même. Altran étant responsable des composants liés à la Vue et au VueModèle et moi sur les composants du Modèle.

Architecture physique

L'architecture physique permet d'identifier la répartition des différents composants logiciels, vus dans l'architecture technique, sur les composants physique.

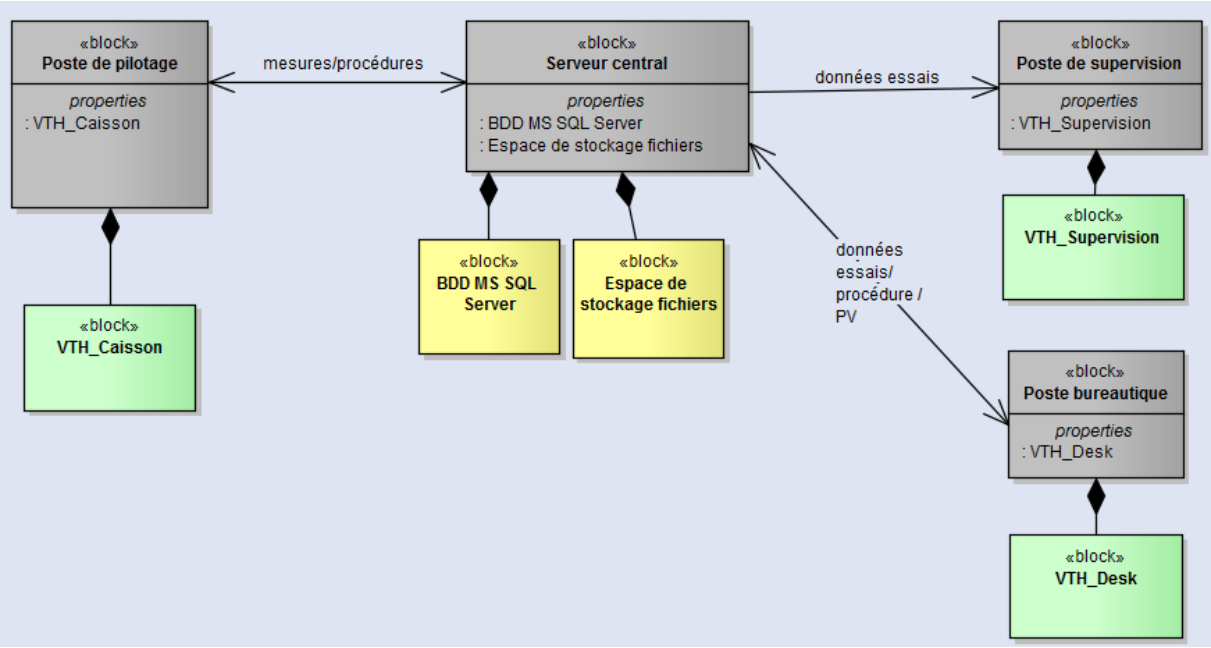


Figure 25 Architecture physique

La Figure 26, ci-dessus, identifie quatre composants physiques :

- Le poste de pilotage est l'ordinateur sur lequel est déployé le logiciel VTH_Caisson. Son interaction avec le serveur central consiste à envoyer les données de mesures et

à récupérer les données de procédures. Pour rappel, il y a un poste de pilotage par caisson soit vingt-cinq postes.

- Le poste de supervision affiche les données des essais en temps réel depuis le serveur central. VTH_Supervision est deployé sur ces postes.
- Le poste Bureautique utilise VTH_Desk, il interagit avec le serveur afin d'enregistrer les procédures créées et de lire les données essais pour la génération de PV.
- Le serveur centrale est composé d'une base de donnée MS SQL Server et contient l'ensemble des données du système, le diagramme de la BDD est disponible à l'annexe 8.

Il est important de remarquer que l'architecture physique proposée et le déploiement des composants se base sur l'existant. Seul le serveur de BDD a été ajouté à l'architecture physique existante. Nous avons fait le choix d'impacter au minimum l'existant afin de contrôler le coût de la solution et parce que l'évolution des règles de la DSI durant le projet nous ont obligés à reconsidérer certains éléments.

4. Réalisation du composant : communication instruments

Comme vu précédemment l'architecture globale du système est MVVM, ainsi, toute la partie donnée et la logique métier qui permet d'accéder à ces données, sont développés au sein du modèle. La Figure 27, ci-dessous, présente les composants principaux qui constituent le modèle.

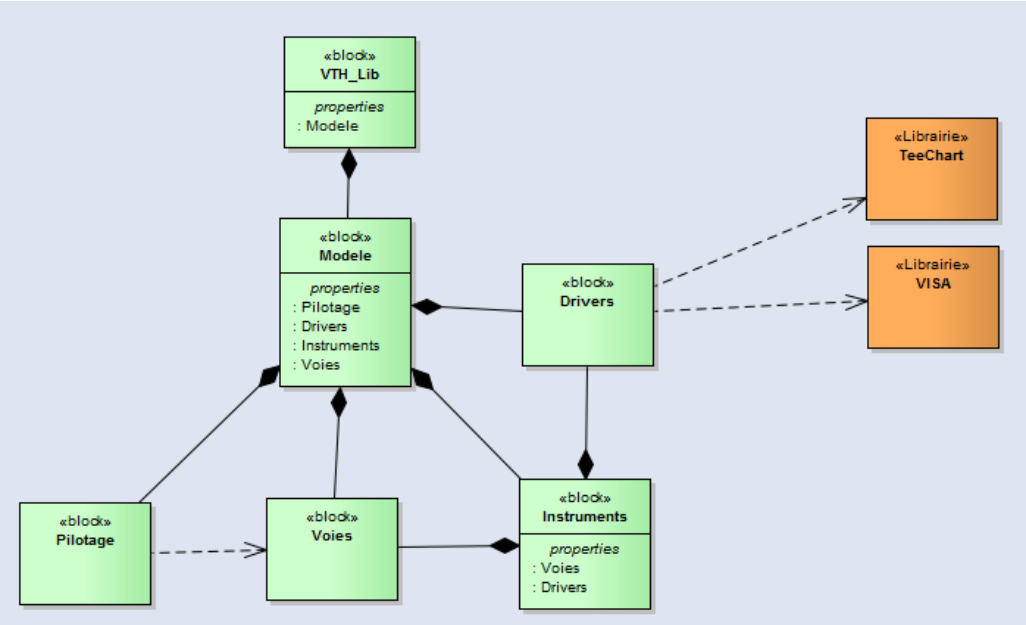


Figure 26 Compositions du modèle

Dans un premier temps je détaillerai la conception du composant en montrant comment le choix d'une architecture adaptée permet de répondre aux problématiques d'interchangeabilité des instruments, puis nous verrons la composition de chacun des éléments de l'architecture ainsi défini par les diagrammes de classe correspondant.

Architecture

Une des exigences du système est de permettre la prise en charge d'un nouvel instrument facilement. Il est donc nécessaire de mettre en œuvre une architecture permettant l'interchangeabilité des instruments sans affecter le reste du système.

Pour résoudre les problèmes d'interchangeabilité des instruments, nous avons choisi de mettre en œuvre le pattern d'architecture en couches. Ce pattern permet de décomposer un problème en regroupant dans une même couche les composants qui ont le même niveau d'abstraction et de responsabilités.

Comme vu dans le tableau I, les instruments sont classés en deux grandes familles, les instruments de mesures et les instruments de commandes. Pour réaliser leur fonction(s) ces instruments s'appuient sur différents drivers qui permettent de requêter ou d'envoyer des commandes via des ports de communication.

Par conséquent, il y a des composants de haut niveau qui s'appuient sur d'autre de bas niveau. En plus de cet aspect de composant de haut et bas niveau il est nécessaire de prendre en compte le fait que les technologies de communication sur les nouveaux instruments pourront être différentes de celles d'aujourd'hui, ou bien que les futurs essais nécessiteront le monitoring de nouveaux types de mesure. De ce fait, il est nécessaire de faire en sorte que les couches de communication et celles qui définissent les types d'instruments puissent être facilement remplaçables, sans affecter le reste du système.

Afin de résoudre cette problématique nous avons décomposé le problème en différentes couches. En partant de la couche la plus basse puis en remontant jusqu'au plus haut niveau de fonctionnalité. Chaque couche propose des services qui s'appuient sur ceux fournis par la couche directement inférieure, qui s'appuient eux-mêmes sur la couche inférieure, etc.

De plus, pour garantir l'évolutivité du module de communication il est nécessaire de découpler les couches qui sont amenées à évoluer par des interfaces. Ainsi, nous avons identifié les couches concernées : la couches drivers de communication et la couche types d'instruments.

L'architecture du module de communication est illustrée dans la Figure 28 ci-dessous.

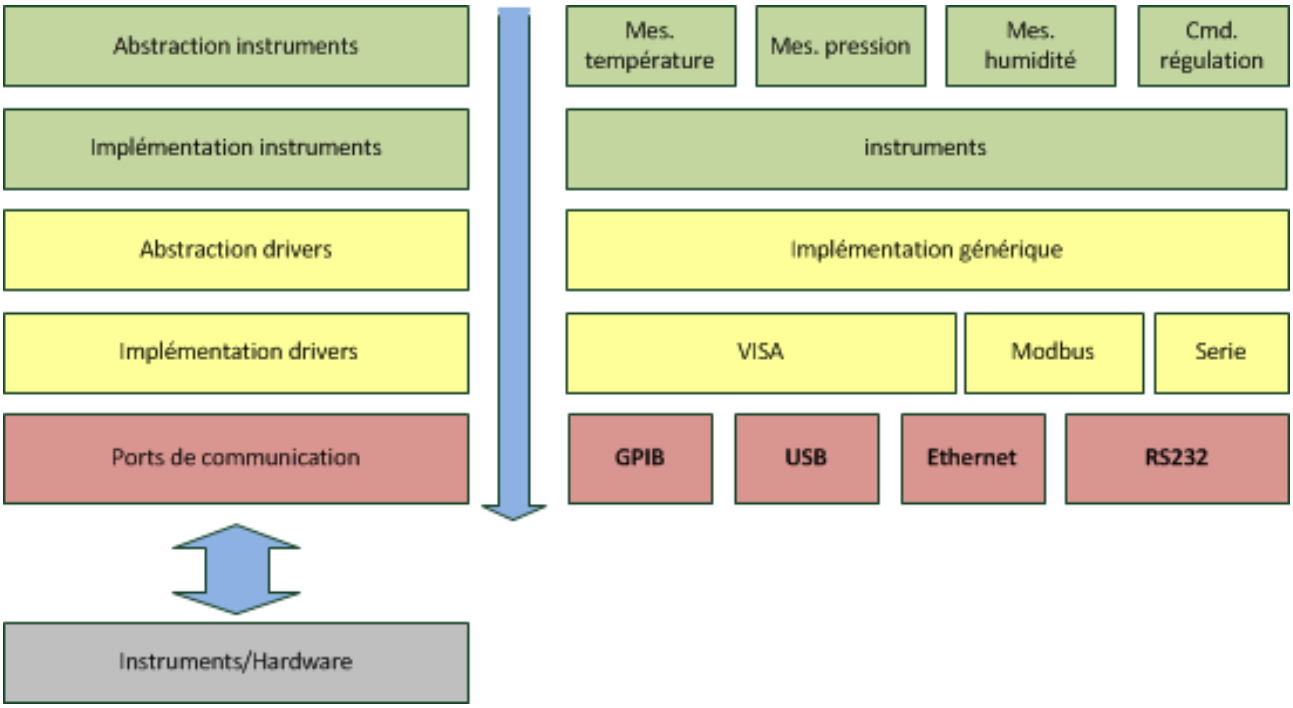


Figure 27 Architecture du composant instrument

Dans cette architecture on identifie cinq couches, voici la liste des couches en partant de la couche du plus bas niveau :

- **Ports de communication** : cette couche matérielle représente les différents ports de communication, disponibles sur les postes de pilotage.
- **Implémentation drivers** : cette couche contient l'implémentation des différentes librairies de communication (VISA, Modbus et Série) afin de réaliser les opérations de bas niveau d'envoi et réception de données. Chacun des composants spécialise la couche supérieure qui contient une définition générique des drivers.
- **Abstraction drivers** : Cette couche est l'interface avec les couches supérieures et permet à celles-ci de manipuler une classe générique afin de garantir l'évolutivité des drivers en découplant ceux-ci de l'implémentation des instruments.
- **Implémentation instruments** : Cette couche réalise l'implémentation des classes correspondant aux instruments du laboratoire en réalisant les classes définies dans la couche supérieure.
- **Abstraction instruments** : Cette couche est l'interface qui est manipulée par l'application utilisatrice, elle fournit le comportement générique et les signatures des méthodes que doivent implémentés les couches inférieures qui la réalise. Cette

couche fournie également la définition des familles d'instruments utilisés dans le laboratoire.

Diagrammes de classes

Une fois les couches définies nous avons réalisés les diagrammes de classes correspondant aux différentes couches identifiées.

Couches drivers

Les couches drivers, identifiées en jaune dans la Figure 28, concernent la couche « Implémentation drivers » qui contient les classes DriverModbus, DriverVISA et DriverSerial. Ces classes implémentent les protocoles de communication (Modbus, VISA et Série) en utilisant les librairies nécessaires. La couche « Abstraction drivers » contient la classe abstraite qui permet de faire l'interface avec les couches supérieures.

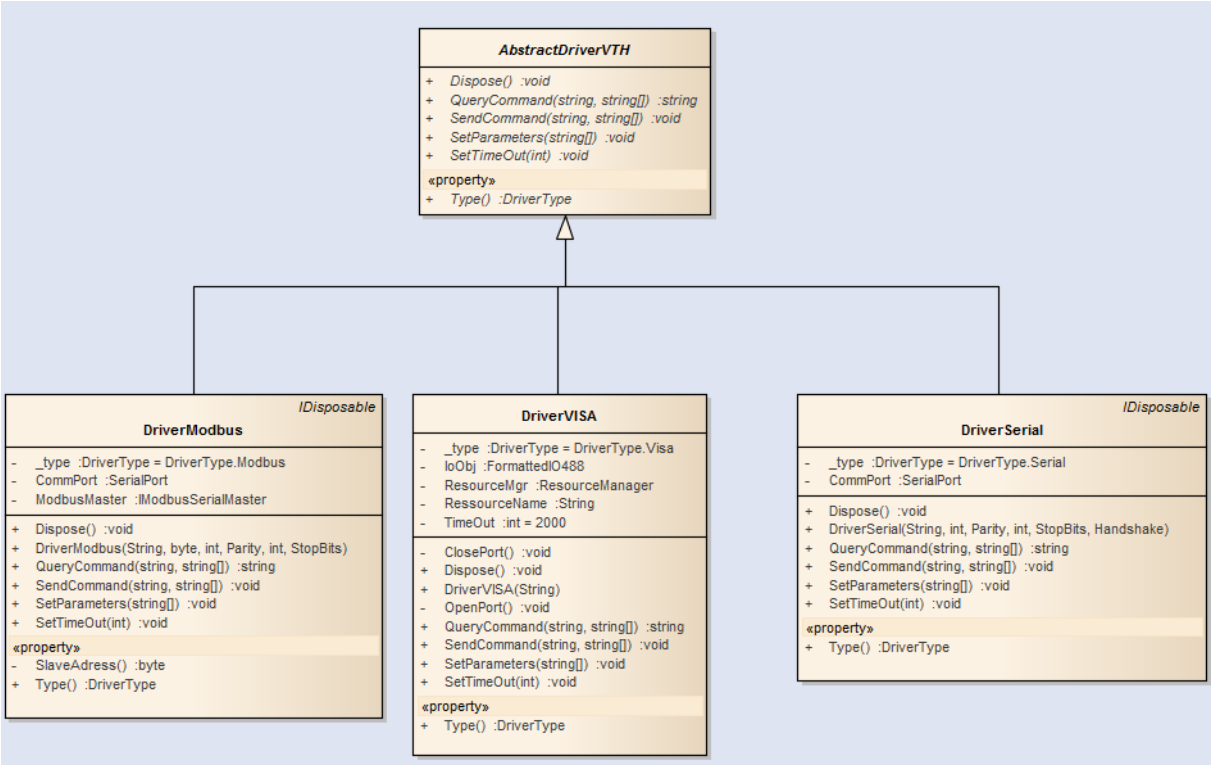


Figure 28 Classes couches drivers

La Figure 29, ci-dessus, montre les relations entre les composants de la couche implémentation drivers et abstraction drivers. La classe d'interface de la couche d'abstraction est de type « abstract », ce choix vient du fait que la classe d'interface contient un attribut, ce qui n'est pas possible avec le type « interface ». Cette classe fournie la

signature des méthodes qui sont utilisées par les couches supérieures et qui doivent être implémentées par les couches inférieures.

La classe **AbstractDriverVTH** définies les méthodes et propriétés qui sont communes à tous les types de communication utilisés dans le laboratoire. Ces méthodes sont de type abstract et doivent être implémentés par les classes qui en héritent. Les méthodes sont :

- **QueryCommand**(string cmd, params string[] parameters): string

Envoie une requête (cmd) et renvoie la réponse correspondante.

- **SendCommand**(string cmd, params string[] parameters): void

Envoie une commande passée en parameter (cmd).

- **SetParameters**(string[] parameters) : void

Configure le port de communication avec d'éventuels paramètres.

- **SetTimeOut**(int timeout) :void

Configure le délai d'attente de réponse du port de communication avant de générer une exception.

Remarque : le mot clé **params** en C# permet de spécifier un paramètre de méthode qui prend un nombre variable d'arguments. Il est utilisé par exemple dans le driver Modbus pour définir l'adresse du registre ou écrire la valeur passée dans le paramètre cmd. L'avantage d'utiliser ce mot clé est que ce paramètre devient optionnel et la méthode ne générera pas d'erreurs si les paramètres sont inexistants.

Couches instruments

Les couches instruments, identifiées en vert, concernent la couche « Abstraction instruments » qui contient les classes génériques des types d'instruments identifiés dans le laboratoire, et sont :

- **DeviceWithSlots** : Spécifie un instrument qui est composé de slots. Ces slot sont des emplacements où l'on peut installer des cartes permettant de réaliser des mesures ou des actions spécifiques. Par exemple, il existe des cartes de mesure de sonde de température, de génération de courant ou de commande de relais. Ce type d'instrument concerne uniquement les modèles Hp/Agilent du laboratoire.
- **HumidityDevice** : Spécifie un instrument de mesure d'humidité.

- **HeatingDevice** : Spécifie un réchauffeur externe.
- **PressureDevice** : Spécifie un instrument de mesure de pression.
- **RegulationDevice** : Spécifie un instrument de régulation thermique.

Ces classes spécifient les types d'instruments, notamment en fonction des types de voies de mesure utilisés. Cette couche contient également la classe « AbstractDevice » qui définit le comportement générique de tous les instruments. Le code de la classe AbstractDevice est fourni à l'annexe 5.

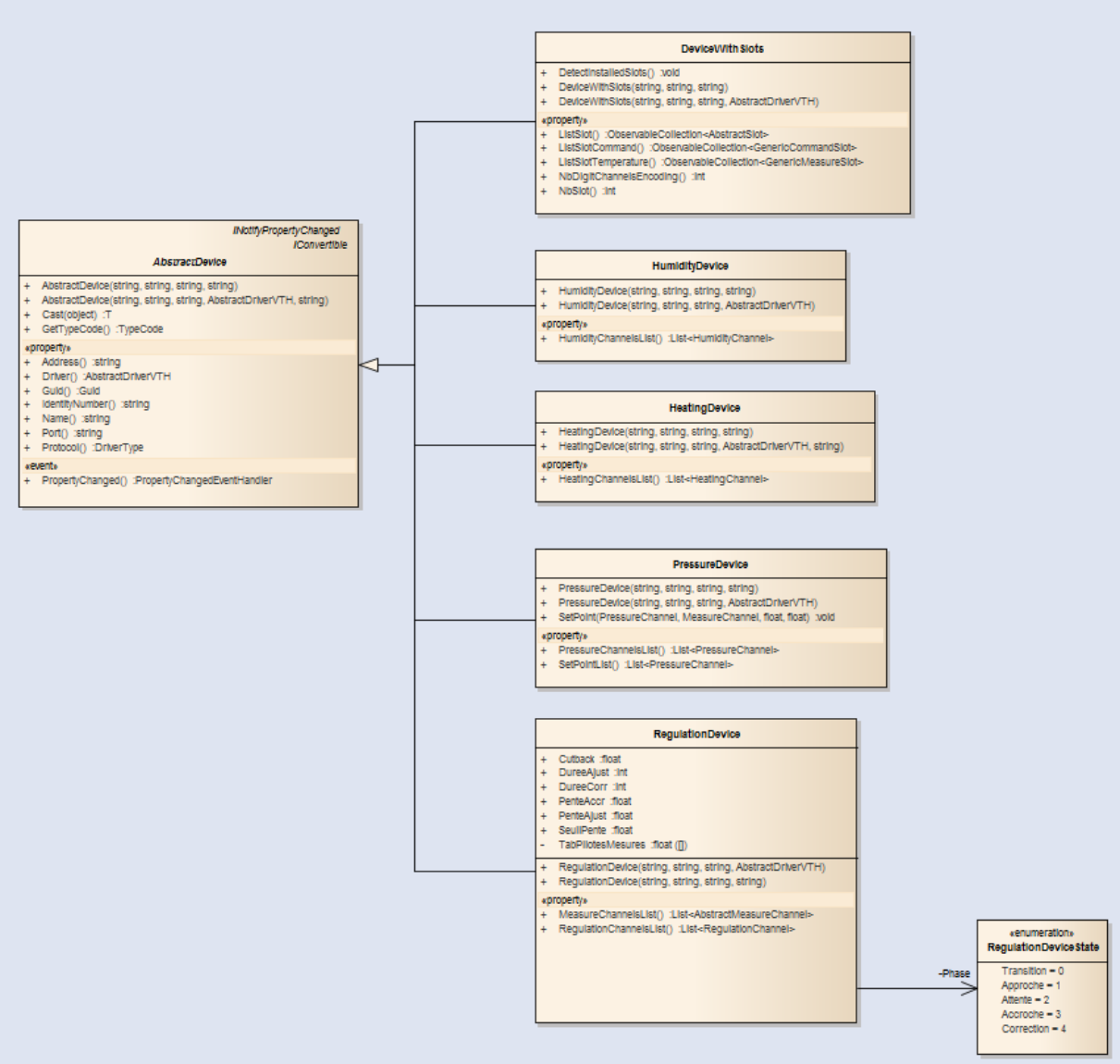


Figure 29 Couches abstraction instruments

La Figure 30, ci-dessus, montre la relation d'héritage entre les classes de la couche d'abstraction. On remarque que les classes d'implémentations sont différenciées en fonction du type de voies de mesure ou de commande qu'elles possèdent.

Concernant la définition des types de voies qui composent un instrument, un effort a également été fait pour rendre ses types facilement maintenable et évolutif par l'utilisation de classe abstraite. Le diagramme de classe ci-dessous présente les types définis et leurs relations.

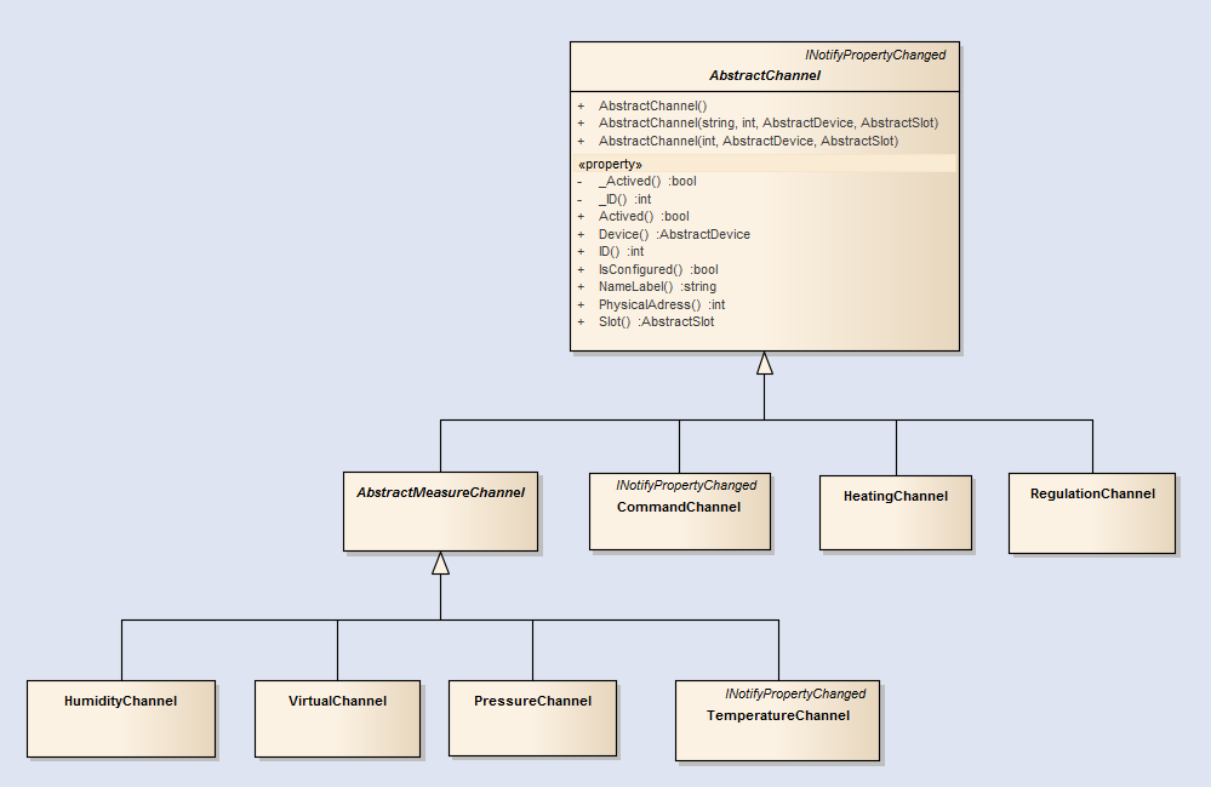


Figure 30 Diagramme de classes type de voie

Chaque instrument du laboratoire est défini par une classe qui réalise le type d'instrument correspondant et qui utilise un driver de communication. Afin de s'interfacer avec le reste de l'application nous avons définis trois interfaces qui sont :

- **ICommandDevice** : Interface avec les instruments de type commande.
- **IMeasureDevice** : Interface avec les instruments de type mesure.
- **IThermalRegulationDevice** : Interface avec les instruments de régulation thermique.

Ces interfaces sont manipulées pour réaliser des opérations de haut niveau. Par exemple, l'interface de régulation thermique définie une méthode qui est WriteSetPoint() et qui correspond à une écriture de consigne, dont la fonction principale attendue par un régulateur thermique est : envoyer des consignes de températures au groupe thermique.

Design patterns mise en œuvre

Pour faciliter la gestion des drivers de communication, des voies d'instruments ainsi que la configuration des instruments, nous avons mis en œuvre des composants de création et de centralisation. Pour répondre à ces problématiques, qui sont courantes dans l'ingénierie logicielle, nous avons mise en œuvre les design patterns. Ceux-ci proposent des solutions génériques d'implémentation à un problème spécifique et récurrent en ingénierie logicielle. Dans notre cas nous avons utilisé les avantages des design patterns Singleton, pour la centralisation, et Factory pour la création d'objet dont le type n'est pas prédéfini.

Les classes ainsi créées sont les suivantes :

- **DriverVTHFactory** : Centralise la création des drivers de communication (Modbus, VISA, Serial).
- **ChannelFactory** : Centralise la création des voies utilisées par les instruments.
- **DeviceFactory** : Centralise la création des instruments utilisés sur le caisson et met en œuvre la détection automatique de ceux-ci sur les ports.

Les choix de conceptions faits pour la réalisation des composants voies, drivers et instruments permettent de fournir un code facilement maintenable et évolutif, ainsi l'ajout d'un nouvel instrument se fait simplement. Ci-dessous, en exemple, les étapes minimales qui permettent de créer la classe définissant un instrument, le modèle Eurotherm3504.

- Création de la classe Eurotherm3504, qui hérite de la classe RegulationDevice. On déclare la classe dans le namespace VTH_Lib.Models.Devices.DeviceDriver afin que celle-ci soit enregistrée au chargement du composant VTH_Lib.

```
public class Eurotherm3504 : RegulationDevice
{
}
```

- Création d'un constructeur à deux paramètres qui prend le numéro d'identification de l'appareil et le nom du port de communication (exemple : COM2) afin d'initialiser le driver de communication grâce à la fabrique DriverVTHFactory. Le constructeur appelle celui de la classe mère en l'initialisant avec le nom de l'instrument.

```
public class Eurotherm3504 : RegulationDevice

{
```

```
        public Eurotherm3504(string idNum, string port)

       : base(idNum, "Eurotherm3504", port)
       {
             //initialisation du driver
            this.Driver=DriverVTHFactory.Instance.CreateDriver(port,
      DriverType.Modbus);
       }
}
```

- Création des voies associés à l'instrument dans le constructeur ; ici le régulateur thermique permet d'utiliser deux voies de régulation et les deux voies de mesure correspondantes. Ainsi, on appelle la fabrique de voies, ChannelFactory. Pour plus de clarté nous avons choisi d'initialiser les voies dans une méthode qui sera appelée depuis le constructeur.

```
private void CreateChannels()

{

            ChannelFactory.Instance.CreateRegulationChannel(2, this);

            ChannelFactory.Instance.CreateRegulationChannel(1026, this);



            ChannelFactory.Instance.CreateMeasureChannel(1, this);

            ChannelFactory.Instance.CreateMeasureChannel(1025, this);
}
```

Pour créer les voies on passe l'id associé, dans note cas l'id correspond à l'adresse du registre Modbus à écrire pour les voies de régulation et à lire pour les voies de mesures, et l'instance de la classe Eurotherm3504, cela permet d'avoir une relation bidirectionnel entre les voies et l'instrument.

- Implémentation des interfaces nécessaire : l'instrument contient des voies de régulation et des voies de mesure, par conséquent on implémente les interfaces IMeasureDevice et IThermalRegulationDevice qui définissent, respectivement, les méthodes permettant d'effectuer une mesure de température et l'écriture la consigne au régulateur.
- Enfin, s'il existe un moyen de détecter la présence de l'instrument de manière automatique, il est nécessaire de créer une classe de détection. Elle contient la logique permettant de déterminer si un instrument donné est présent ou non sur un

port de communication. Cette classe implémente l'interface IDeviceDetection et doit être dans le namespace VTH_Lib.Models.Devices.DeviceDetection afin d'être prise en compte. Enfin, pour que la classe de l'instrument soit instanciée de manière automatique il est nécessaire de surcharger le constructeur avec un troisième paramètre de type AbstractDriverVTH. Les classes Eurotherm3504 et Eurotherm3504Detection sont fournies à l'annexe 6.

5. Réalisation du composant de pilotage

Cette partie présente la réalisation du composant de pilotage. Dans un premier temps nous verrons la manière dont nous avons décomposé les différentes activités réalisées pour la régulation thermique. Dans une seconde partie nous verrons quels ont été les choix de conception qui garantissent la maintenabilité et l'évolutivité du module ainsi que les détails d'implémentation.

Modélisation du pilotage thermique

Pour la réalisation du module du pilotage nous nous sommes appuyés sur la documentation existante afin de comprendre le fonctionnement de l'existant. Celle-ci concerne essentiellement les sécurités, les règles métiers de régulation et le calcul des consignes de température à envoyer aux régulateurs thermique.

Diagramme d'activités

Pour synthétiser les informations des différentes activités du pilotage, nous avons utilisé le diagramme d'activité UML pour modéliser les différents processus de pilotage thermique. Ainsi, nous avons identifiés 6 activités réalisées séquentiellement :

- **Mesure** : Réalise l'acquisition des voies de mesure actives (température, pression, humidité).
- **Sécurité** : Vérifie les différentes sécurités sur les voies de référence à partir des mesures réalisées. En fonction de l'état des sécurités cette activité effectue les actions nécessaires telles que : le déclenchement des alarmes, l'arrêt du groupe thermique, le retour à température ambiante, etc.
- **Initialisation du pilotage :** Contrôle les voies de mesure pilote et met à jour les valeurs des différentes variables utilisés pour le pilotage.
- **Pilotage :** Calcul les consignes à envoyer aux régulateurs et contrôle l'état du palier.

- **Envoi des consignes :** Envoi des consignes calculées aux régulateurs.
- **Enregistrement des données :** Effectue l'enregistrement de l'ensemble des variables de pilotage ainsi que la dernière mesure vers la base de données configurée.

La logique d'enchainement des activités est modélisée dans le diagramme d'activité de la Figure 31 ci-dessous.

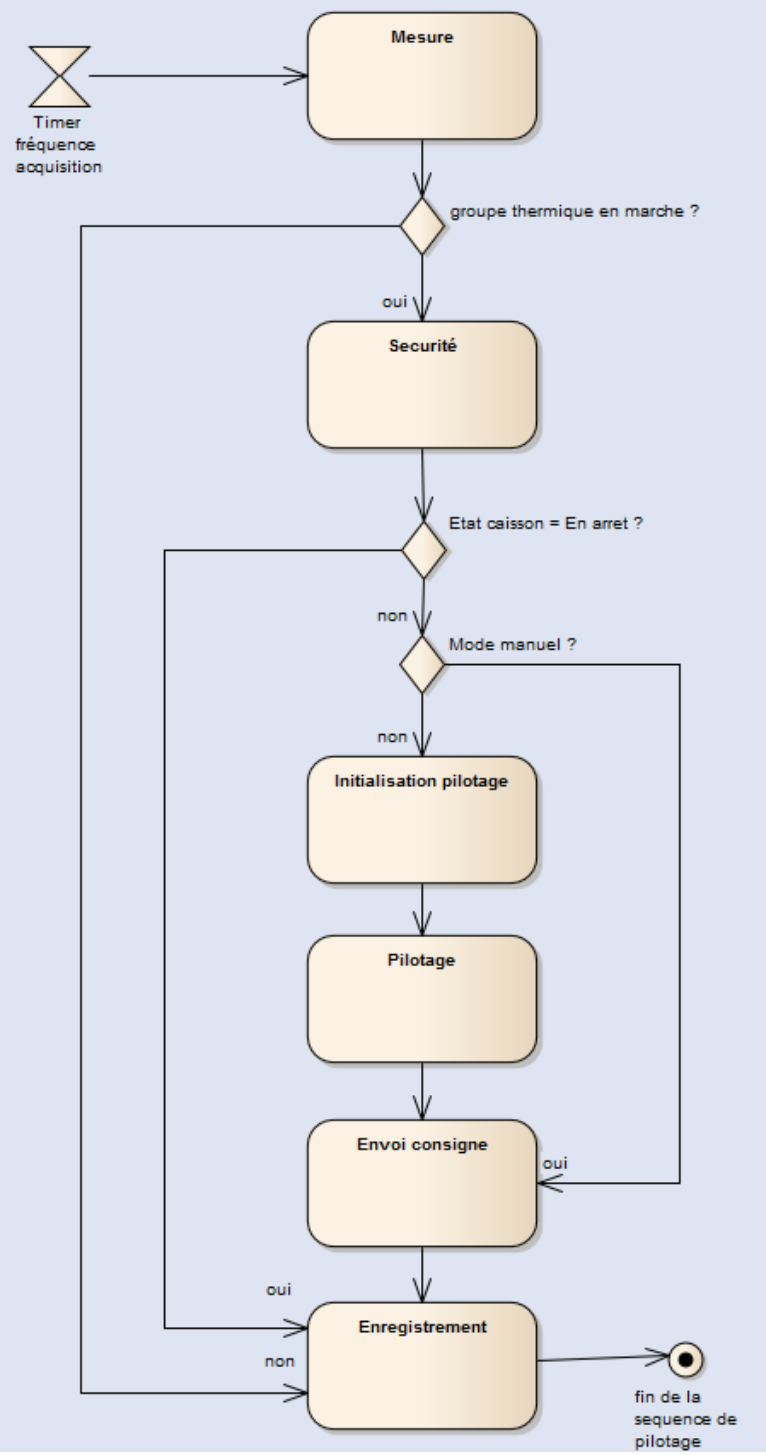


Figure 31 Diagramme d'activité pilotage

On remarque que dans le diagramme d'activité de la séquence de pilotage, l'évènement déclencheur est un timer correspondant à la fréquence d'acquisition. Dans le cas des essais dans le laboratoire VTH la fréquence configurée est d'une minute, en effet dans le domaine de la thermique l'inertie est telle qu'il n'est pas nécessaire d'avoir une fréquence plus importante.

L'étape suivante consiste à détailler l'ensemble des activités ainsi identifiées. Afin de pouvoir valider le comportement de chacune des activités avec le personnel spécialisé dans le domaine de la thermique, nous avons décrit chacune des activités dans un langage naturel. L'utilisation d'un langage naturel permet de valider l'implémentation future en faisant abstraction de la syntaxe d'un langage de programmation. Ces algorithmes facilitent également le travail du développeur.
Les algorithmes des activités qui ont été réalisés sont fournis à l'annexe 7.

Choix de conceptions

Une exigence forte sur la réalisation du composant de pilotage est sa maintenabilité et son évolutivité afin de garantir son amélioration future. Ce composant doit également pouvoir être maintenu en interne par du personnel ayant des connaissances très basiques sur le développement logiciel et l'architecture complète du système. Par conséquent, il est nécessaire que le composant de pilotage ait un très faible couplage avec le reste du système.
Pour résoudre ces problématiques nous avons implémenté chacune des activités dans une classe qui contient l'implémentation de l'algorithme correspondant. Ces classes spécialisent une classe abstraite, AbstractModule, qui définit la méthode execute() qui est appelée pour réaliser l'activité.
Enfin, nous avons choisi de mettre en place un composant unique dont la responsabilité est d'exécuter, selon la séquence définie dans le diagramme d'activités, chacun des modules et de fournir à ces modules l'ensemble des variables et fonctions à manipuler pour les réaliser. Par conséquent, cela impose que ces fonctions soient accessibles de manière simple. Nous avons donc réalisé une classe qui met en œuvre le pattern singleton et qui fournit toutes les fonctions de haut-niveau nécessaire.

Ces choix de conceptions permettent de faciliter la modification de la séquence d'exécution des activités du pilotage et l'ajout d'une activité relativement facilement, cela sans affecter le reste du système.

6. Tests réalisés / Validations

Cette partie présente les tests qui ont été réalisés pour valider les composants de communication et de pilotage présentés précédemment. Chacun des composants a, dans un premier temps, été validé dans l'environnement de développement. Une fois ces tests réalisés nous avons testé les fonctionnalités dans l'environnement de production.

Nous présenterons ici les composants qui étaient à ma charge en illustrant les différents tests réalisés par des captures d'écran du logiciel.

La validation du composant de communication instrument a nécessité de valider les fonctions suivantes :

1. Détection automatique des instruments ;
2. Acquisition de mesures ;
3. Envoie de commande automate ;
4. Envoie de consigne de température.

La validation du pilotage a nécessité de valider les fonctions suivantes :

5. Exécution périodique de la séquence de pilotage ;
6. Fonctionnement des sécurités ;
7. Validation des algorithmes de pilotage.

L'exécution du cas d'utilisation configuration caisson nous a permis de valider les fonctions 1 et 2. Les captures de la Figure 32 montrent les interfaces du processus de détection automatique des équipements. La capture de la Figure 33 montre l'ensemble des équipements de mesure de température disponible (encadré vert) et les voies associés à ceux-ci (encadré orange).

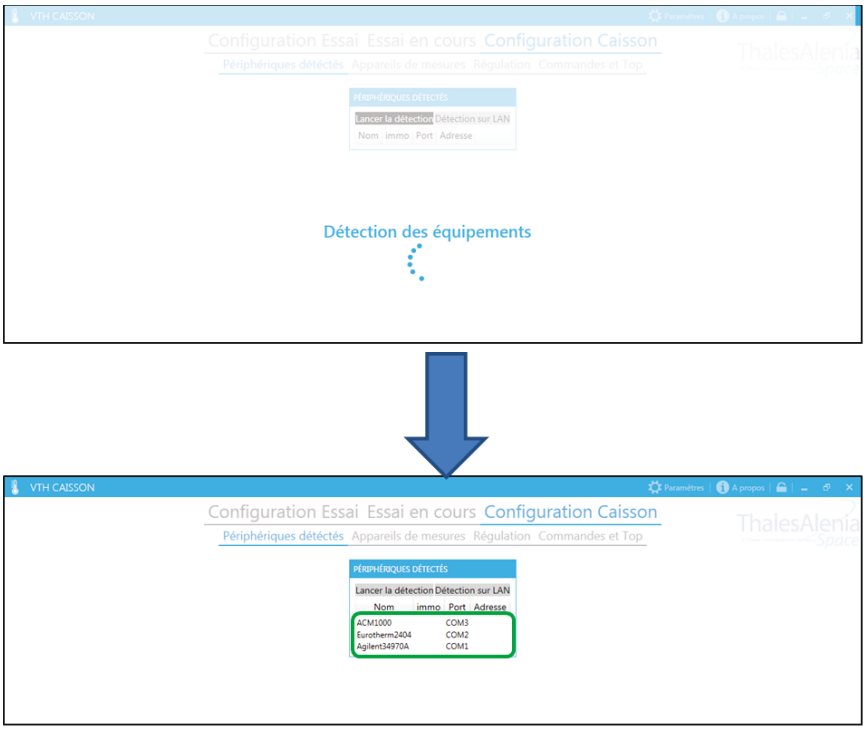


Figure 32 Captures détection auto. des instruments

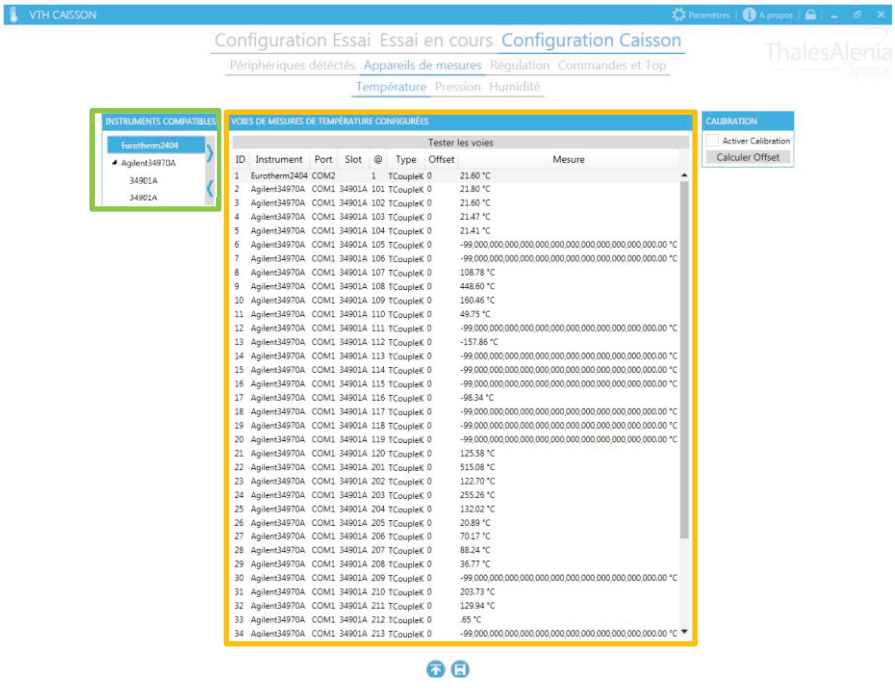


Figure 33 Capture configuration des voies

Enfin pour valider les aspects de pilotage thermique qui concernent les fonctions 3, 4, 5, 6 et 7 nous avons configuré, exécuté et surveillé plusieurs essais, dont certains ont duré

plusieurs jours. La capture de la Figure 34 montre l'écran de pilotage, on remarque que la variation des températures mesurées à l'aide de la courbe et qui correspond au cycle de température configuré.

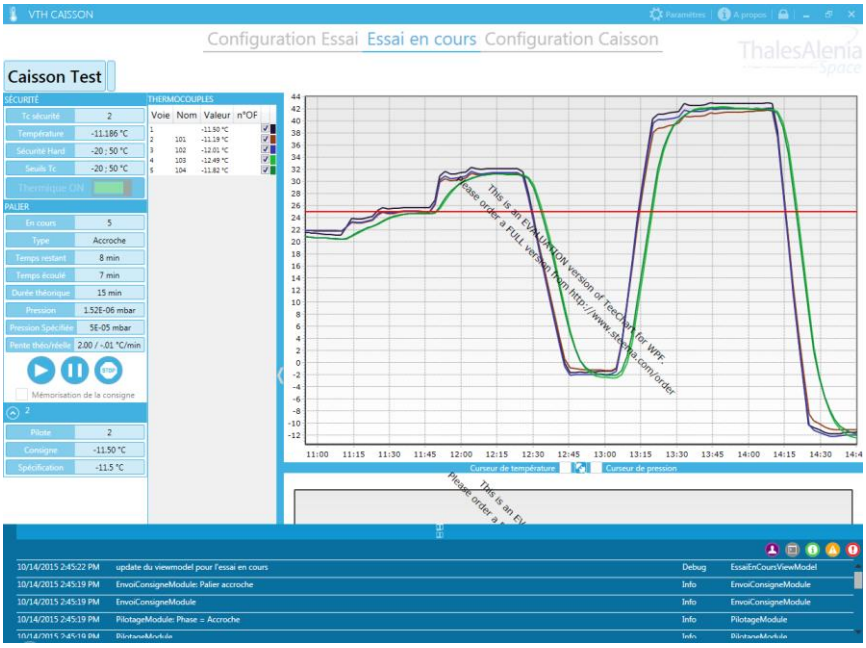


Figure 34 Capture essais en cours

Le lancement d'essais en condition réelle a permis de valider les composants du modèle qui étaient à ma charge et leur intégration avec les composants réalisé par la sous-traitance.

Conclusion

Afin de conclure le mémoire, je présenterai le bilan du projet dans lequel s'est inscrit mon sujet, nous verrons ensuite mes contributions au sein du projet, concernant notamment les méthodes qui ont été proposées afin de répondre aux problématiques identifiées. Par la suite je présenterai les limitations de ces méthodes, et pour finir, les perspectives et opportunités liés à celles-ci.

1. Bilan du projet

Le projet d'amélioration du système des essais réalisés au sein du laboratoire VTH, est actuellement en phase d'essais en conditions réelles (sur caisson thermique). Les premiers tests qui ont été réalisés ont permis de corriger des problèmes mineurs. Ces tests ont montrés que l'architecture mise en œuvre correspond aux attentes. En termes de planning,

d'autres sprints de développement ont été ajoutés car plusieurs composants ont nécessité davantage d'effort que ce qui était prévu. Par conséquent la mise en production est retardée d'environ 1 mois.

En ce qui concerne les problématiques abordées, qui, pour rappel, sont :

1. **Comment réaliser l'étude d'un système logiciel existant dépourvu de documentation détaillée ?**
2. **Comment garantir l'interchangeabilité d'un ensemble hétérogène d'instruments par la mise en place d'une architecture adaptée ?**

Les méthodes appliquées se sont montré efficaces dans le contexte du projet, cependant leurs applications dans un contexte diffèrent peut montrer certaines limitations que nous verrons par la suite.

Enfin concernant l'organisation du développement de la solution; l'utilisation de diagrammes et de design patterns a permis d'avoir la maitrise sur la qualité interne du logiciel pendant le développement des composants et a permis de communiquer efficacement avec les équipes de développement.

2. Contributions

Mes contributions et activités pendant ce projet sont comparables aux fonctions de l'Assistance à Maitrise d'Ouvrage (AMOA) dont le rôle, comme son nom l'indique, est d'assister la Maitrise d'Ouvrage (MOA). La MOA est le client et est à l'origine du projet, elle a la responsabilité du résultat du projet et finance sa réalisation.

Mes compétences d'écoute et de formalisation pour rendre la solution adaptée à l'utilisateur ont permis de réaliser les activités suivantes [15]:

- Décrire les besoins utilisateurs : A partir de l'existant j'ai pu formaliser les besoins en terme de métier et en terme de service que le système doit rendre. Cette activité a été réalisée par l'étude de la solution existante et la capture des besoins par les interviews et réunions que j'ai menées.
- Décrire la solution fonctionnelle : Afin de répondre aux besoins identifiés j'ai réalisé la description de la solution à réaliser. Cette activité correspond à l'analyse fonctionnelle et à la description de l'architecture globale de la solution.

- Contrôler la qualité : Pour vérifier que le système développé corresponde bien aux exigences j'ai participé à la validation des livrables tout au long de la phase de réalisation avec l'aide de la MOA et des utilisateurs.
- Pilotage du projet : Pour respecter les engagements en termes de contenu, de coûts et de délais, j'ai rendu compte de l'avancement et des difficultés rencontrées à la MOA.

J'ai également eu la responsabilité de plusieurs activités de la Maitrise d'Œuvre (MOE) qui est responsable du développement de la solution. Par l'étude des technologies envisageables fait en amont de la phase de réalisation, j'ai orienté le choix des technologies pour concevoir la solution technique. J'ai également participé à la définition de l'architecture technique et ai été responsable du développement de plusieurs composants du système.

3. Limitations

Concernant les limitations des méthodes proposées permettant de répondre aux problématiques identifiées **:** La méthode d'analyse du système existant est adaptée dans le cas où la majeure partie des composants développés ne permettent pas d'être réutilisés ou évolués et, par conséquent, qu'il est nécessaire de refondre entièrement le système. Ainsi pour adapter cette méthode à un plus large panel de systèmes à étudier il est nécessaire d'étudier les composants éventuellement réutilisables afin de les adapter ou d'adapter la solution future à ceux-ci.

Pour la seconde problématique, l'approche de décomposition du problème en couche est adaptée dans le cas où l'on identifie des composants d'un haut niveau de fonctionnalités qui s'appuient sur d'autres composants de bas niveau. Ce problème est souvent identifié lorsque l'on souhaite avoir un haut niveau d'abstraction des fonctionnalités fournies par des équipements matériels de même type. Cette méthode montre ses limitations dans plusieurs cas :

- Si l'on souhaite changer le comportement des couches de haut niveau il sera nécessaire d'effectuer des changements sur les couches inférieures ainsi cela peut impliquer un effort qui peut être conséquent.
- Si les besoins en termes de performance évoluent, étant donné que les informations et les requêtes sont traitées successivement dans chacune des couches ainsi cela a une incidence sur le temps de traitement de la requête. Par conséquent, Il sera

nécessaire de revoir le nombre de couches et donc de trouver un compromis entre l'évolutivité du composant concerné et ses performances.

4. Perspectives

Dans le contexte du laboratoire et dans un premier temps il va être nécessaire d'intégrer l'outil de maintenance au système puis dans un second temps il est prévu de permettre la supervision à distance et hors site cela implique un travail en collaboration avec DSI qui est garant de la sécurité du système d'information.

Les problèmes de compatibilité des outils existants avec le nouveau parc informatique et les besoins d'évolutivité de ceux-ci ne concernent pas uniquement le laboratoire VTH.

Il est donc envisageable de s'appuyer sur la démarche qui a été réalisée et de l'appliquer dans un premier temps sur l'ensemble des moyens d'essais, d'autant plus que les instruments utilisés au service des moyens d'essais ont les mêmes interfaces (Série, GPIB, LAN etc..).

Enfin, les essais thermiques ne sont pas propres au site de Toulouse et il existe d'autres centres d'essais en France (Cannes) et ailleurs en Europe. Dans le cadre d'une homogénéisation des outils de tests sur l'ensemble de ces sites nous avons organisé une présentation de notre projet au centre de L'Aquila (Italie) et à Cannes. L'objectif de cette présentation a été de comparer le workflow de TAS Toulouse avec les centres présents pour voir si le système développé répond aux autres besoins et pour préparer les axes d'améliorations afin de fournir une solution commune à tous les centres. Ceci fera l'objet d'un projet ultérieur.

Bibliographie

[1] « La filière européenne des satellites en crise - Spatial ». [En ligne]. Disponible sur: http://www.usinenouvelle.com/article/la-filiere-europeenne-des-satellites-en-crise.N236759. [Consulté le: 07-oct-2015].

[2] « IEEE-488 — Wikipédia ». [En ligne]. Disponible sur: https://fr.wikipedia.org/wiki/IEEE-488. [Consulté le: 28-sept-2015].

[3] « Architecture logicielle — Wikipédia ». [En ligne]. Disponible sur: http://fr.wikipedia.org/wiki/Architecture_logicielle. [Consulté le: 09-avr-2015].

[4] P. Roques, *SysML par l'exemple*. Paris: Eyrolles, 2009.

[5] « Home - Pencil Project ». [En ligne]. Disponible sur: http://pencil.evolus.vn/. [Consulté le: 29-sept-2015].

[6] « FAQ - Rxtx ». [En ligne]. Disponible sur: http://rxtx.qbang.org/wiki/index.php/FAQ. [Consulté le: 01-oct-2015].

[7] « java-simple-serial-connector - jSSC - java serial port communication library - Google Project Hosting ». [En ligne]. Disponible sur: https://code.google.com/p/java-simple-serial-connector/. [Consulté le: 01-oct-2015].

[8] « JPIB: a Java API for GPIB Devices ». [En ligne]. Disponible sur: http://jpib.sourceforge.net/. [Consulté le: 01-oct-2015].

[9] « JFreeChart ». [En ligne]. Disponible sur: http://www.jfree.org/jfreechart/. [Consulté le: 01-oct-2015].

[10] « Charting Components by Steema for Data Visualization. Java Charts, Maps and Gauges for all major Java IDEs - product page ». [En ligne]. Disponible sur: http://www.steema.com/teechart/java. [Consulté le: 01-oct-2015].

[11] « Overview (Instrument Control (iC)) ». [En ligne]. Disponible sur: https://icontrol.java.net/. [Consulté le: 01-oct-2015].

[12] « National Instruments VISA - National Instruments ». [En ligne]. Disponible sur: https://www.ni.com/visa/. [Consulté le: 03-oct-2015].

[13] « Manifeste pour le développement Agile de logiciels ». [En ligne]. Disponible sur: http://agilemanifesto.org/iso/fr/. [Consulté le: 04-oct-2015].

[14] « Méthodologie Model-View-ViewModel avec WPF ». [En ligne]. Disponible sur: http://japf.developpez.com/tutoriels/dotnet/mvvm-pour-des-applications-wpf-bien-architecturees-et-testables/#LIII. [Consulté le: 02-oct-2015].

[15] « Le métier d'AMOA ». [En ligne]. Disponible sur: http://www.eqlhenix.fr/le-metier-damoa. [Consulté le: 04-oct-2015].

Table des annexes

Annexes

Annexe 1 : Description textuelle des cas d'utilisation

Editer procédure

ID	1
Description	Les opérateurs simples et avancés pourront manipuler les fichiers de procédure. Le cas d'utilisation comprend la création, la recherche en local et sur Alice des procédures, l'édition et l'export des fichiers de procédure. Elles sont exportées en format PDF sur le disque local ou HDD pour validation et l'adresse des procédures validées sont disponibles dans la base de données Alice. L'opérateur peut également la sauvegarder en local sur sa machine pour l'éditer ultérieurement au format XML.

Acteurs	Les opérateurs simples et avancés pourront invoquer ce cas d'utilisation.
Préconditions	• L'opérateur est identifié et dispose des informations de l'équipement à tester (OS, affaire, n° article, …) • Le poste a accès au réseau de l'entreprise • L'IHM « Configuration procédure » est ouverte

Etapes basiques	• L'opérateur indique les informations d'identification de la procédure.
	• Le système recherche les références de la procédure sur Alice.
	  ○ La procédure est présente sur Alice : le logiciel charge le fichier XML de la procédure en lecture seule pour lancer un essai.
	  ○ La procédure n'existe pas : le système propose à l'utilisateur de charger un Template du document pour créer le nouveau fichier de procédure.
	• L'opérateur rempli les différents champs de spécification de la procédure.
	• L'opérateur charge des images et des schémas à inclure dans le fichier de procédure.
	• L'opérateur sauvegarde la procédure en local ou sur le disque réseau HDD au format XML.
	• L'opérateur exporte une version PDF de la procédure sur le disque HDD.

Etapes alternatives	• L'opérateur sauvegarde localement la procédure en cours au format XML. • L'opérateur charge la procédure XML depuis son disque et continu l'édition. • Lors de la recherche, la procédure est présente sur Alice : l'utilisateur avancé peut charger la procédure XML en lecture et écriture pour être modifié. Une

	nouvelle demande d'approbation devra être envoyée après édition.
Exceptions	En cas de perte de connexion réseau, les procédures sont sauvegardées localement.
Règles métier	Les documents de procédure validés (Alice) ne peuvent être modifiés que par les utilisateurs avancés. Les procédures validées (Alice) ne sont disponibles qu'en lecture pour les utilisateurs standards. Ils peuvent les charger afin de lancer les essais.
Post-condition	A la fin de la manipulation d'un fichier de procédure, celui-ci doit être sauvegardé en local ou sur le disque réseau HDD.

UC Configuration Essai

ID	3
Description	L'opérateur configure les voies de mesures à utiliser pour l'essai à lancer (physiques et virtuelles), la méthode de pilotage de l'essai, les alarmes et les objectifs. L'opérateur exécute l'essai et visualise les résultats de mesure.
Acteurs	Les opérateurs simples et avancés pourront invoquer ce cas d'utilisation.
Préconditions	• L'opérateur est identifié • L'opérateur dispose d'une procédure d'essai (ou une procédure de maintenance) • Le système est configuré avec les instruments à utiliser • L'IHM configuration de l'essai est ouverte • Le système est connecté au réseau ou non • L'opérateur est présent sur le poste caisson

Etapes basiques	**Configuration de l'essai avec procédure numérique** • L'opérateur saisie les informations de l'essai : ○ n° Affaire et/ou n° article et l'équipement concerné • Le système recherche les procédures correspondantes sur Alice et en local ainsi que leur état (validée, en attente d'approbation) • L'opérateur sélectionne la procédure qu'il souhaite charger • Le système affiche la liste des essais contenus dans la procédure avec leur état (réalisé, à réaliser, en cours) • L'opérateur sélectionne l'essai qu'il souhaite réaliser. • Le système charge le(s) cycle(s) présent(s) dans la procédure sélectionnée **Configuration des voies de mesures (physique)** • L'opérateur sélectionne les instruments à utiliser • Le système met à jour la liste des voies utilisables • L'opérateur sélectionne les voies à utiliser • L'opérateur choisi de tester les voies à utiliser • Le système interroge les instruments de mesure concernés et affiche les valeurs mesurées • Le système enregistre la configuration dans le fichier essai **Configuration des voies de mesures virtuelles** • L'opérateur choisi d'ajouter une voie virtuelle • L'opérateur configure les voies virtuelles en entrant leurs caractéristiques (nom, formule, commentaire) • L'opérateur choisi de tester les voies configurées • Le système interroge les instruments de mesures et affiche les valeurs calculées • L'opérateur sélectionne les voies calculées à activer • Le système enregistre la configuration dans le fichier essai sur le disque HDD (format XML). **Configuration du pilotage** • Le système affiche le profil du (des) cycle(s) précédemment chargé(s) (sous forme de courbe) ainsi que le calcul des

	sécurités hard à saisir • L'opérateur édite éventuellement le cycle chargé et valide les informations saisies • Le système enregistre le cycle dans le fichier essai **Exécution de l'essai** • L'opérateur choisi d'exécuter l'essai configuré • Le système affiche un message rappelant la configuration de l'essai et propose à l'opérateur de confirmé son choix **Visualisation de l'essai** • Le système affiche une vue qui contient les mesures effectuées sous forme de courbe (pression et température), le palier en cours, la dernière consigne envoyée au(x) régulateur(s) (évolution possible suivant retour utilisateurs).
	Configuration du pilotage avec réchauffeurs • L'opérateur créer un nouveau cycle et l'associe à un régulateur configuré sur le poste • L'opérateur sélectionne la voie de mesure (physique ou virtuelle) qui servira de pilote pour le régulateur **Configuration de l'essai avec procédure papier** • L'opérateur sélectionne le cycle principal pour édition • Il saisit le cycle à respecter en entrant la température à atteindre pour la plaque et l'écran, la durée à respecter, l'attente ou l'envoie de top, la pente pour chaque palier et si nécessaire la voie de mesure (physique ou virtuelle) qui servira de pilote • Le système affiche le profil du cycle saisi (sous forme de courbe) ainsi que le calcul des sécurités hard à saisir • L'opérateur valide les informations saisies **Modification de la configuration de l'essai en cours** • L'opérateur sélectionne la vue de configuration du pilotage • Le système se met en pause : il n'envoie plus de nouvelles

	consignes aux régulateurs mais continue d'acquérir des mesures périodiquement (pression et température) • L'opérateur modifie la configuration de l'essai et valide sa saisie • Le système reprend le palier en cours en tenant compte des modifications effectuées
Exceptions	**Configuration des voies de mesures (physique)** • La communication avec les instruments est défectueuse • Le système envoie un message avec le type d'erreur rencontrée **Configuration des voies de mesures virtuelles** • La formule contient des voies avec des types de mesure différente (exemple : °C, Voltage) • Le système affiche un warning en précisant les unités utilisées • L'utilisateur choisi de modifier sa formule ou non
Règles métier	Dans le cas d'une régulation thermique mono-groupe, la température plaque est égale à la température écran.
Post-condition	Sauvegarde de la configuration OK sur le disque (HDD) Configuration conforme aux exigences du caisson

Editer PV

ID	4
Description	L'essai est terminé et l'opérateur génère le PV à partir des informations de configuration de l'essai et des données de mesure.
Acteurs	Les opérateurs simples et avancés pourront invoquer ce cas d'utilisation.
Préconditions	• L'essai concerné est terminé. • L'opérateur est identifié. • L'IHM génération de PV est sélectionnée sur le desk.

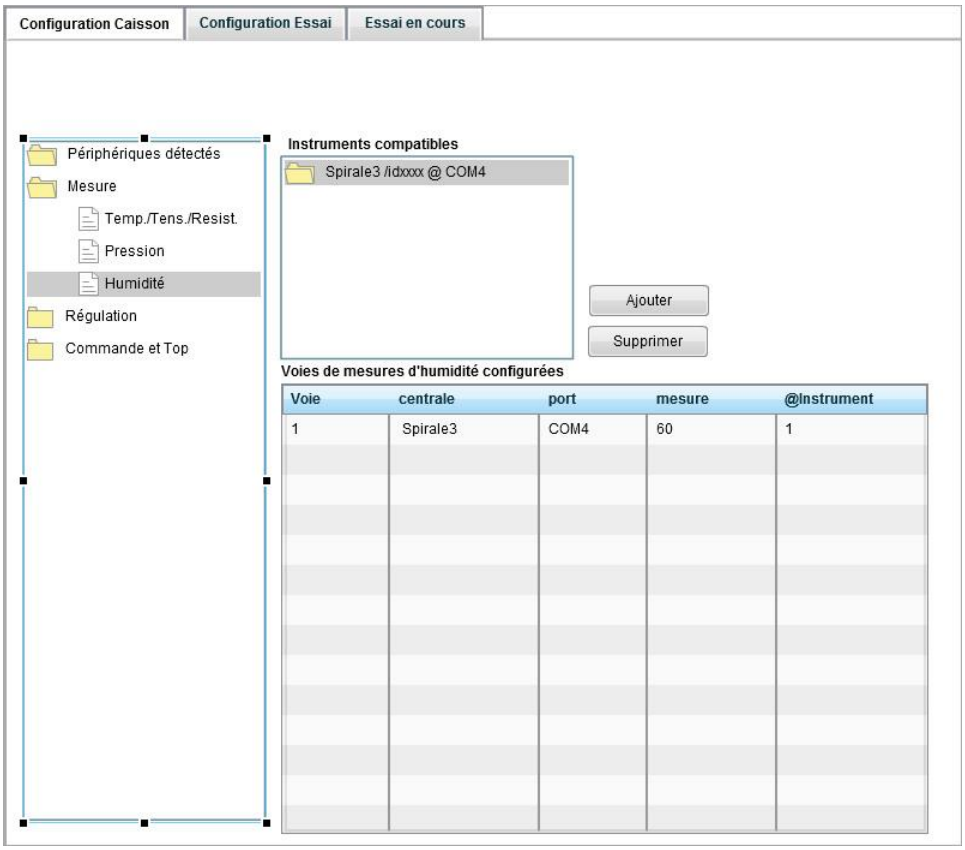
Etapes basiques	• l'opérateur sélectionne le ou les essais pour lesquels il souhaite générer un PV. • L'application affiche une vue présentant toutes les données essais et propose de générer le PV. • L'utilisateur met en forme les données qu'il souhaite faire apparaitre dans le PV, en sélectionnant les voies de mesures qui apparaitront et ajoute éventuellement des commentaires. • L'utilisateur exporte le PV au format PDF afin de l'envoyer au client de l'essai.
Etapes alternatives	• L'utilisateur sauvegarde en local le PV en cours. • L'utilisateur charge un PV en local afin de le compléter.
Exceptions	Exceptions technique (fichiers corrompus)
Règles métier	Un fichier PV est généré par procédure et comprend les résultats de tous les tests définis dans la procédure.
Post-condition	Sauvegarde de la configuration OK sur le disque (HDD).

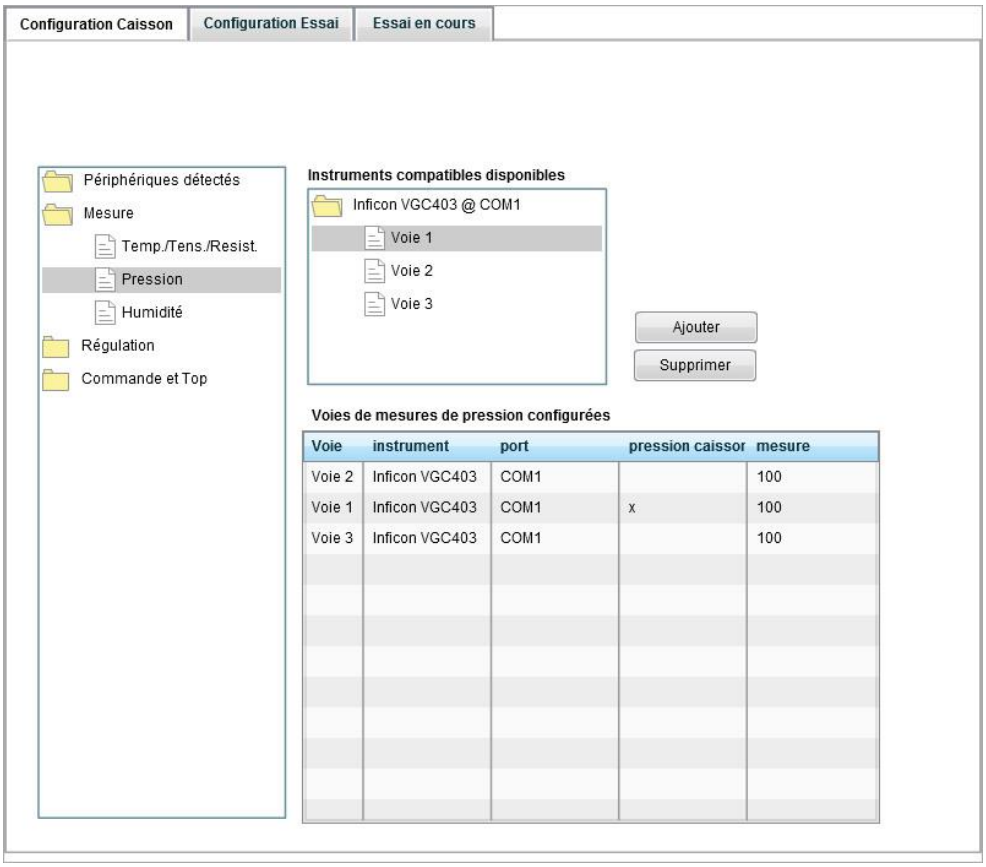
Superviser essai à distance

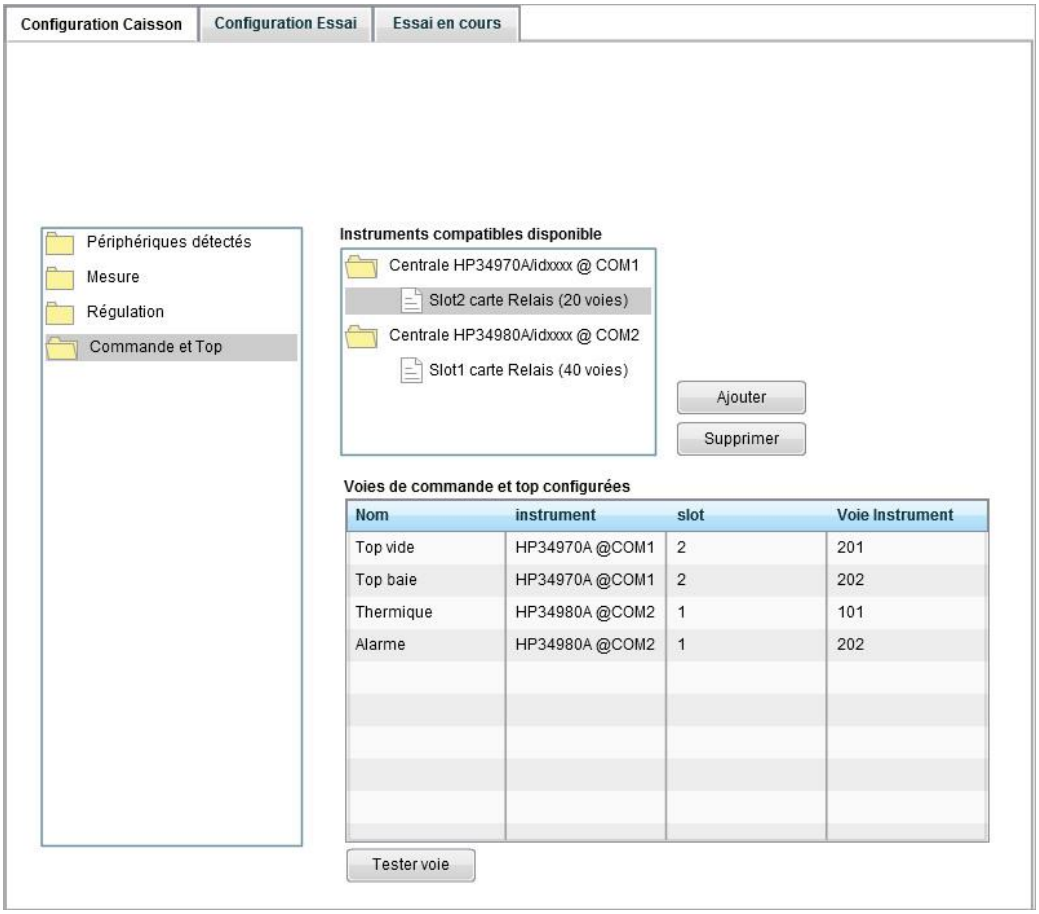
ID	5
Description	L'opérateur visualise un ou plusieurs essais en cours ou terminés
Acteurs	Les clients et les opérateurs simples et avancés pourront invoquer ce cas d'utilisation.
Préconditions	• Un ou plusieurs essais sont en cours ou terminés • Le poste (desk, supervision ou caisson) est connecté à l'intranet (pour supervision à distance)
Etapes basiques	• L'application propose de sélectionner le(s) essai(s) qu'il souhaite visualiser • L'opérateur configure l'affichage pour la supervision • L'application affiche la vue de supervision
Etapes alternatives	**Déclenchement d'une alarme sur un caisson** • Un poste caisson se met en alarme • L'application signale visuellement l'alarme sur la vue de supervision en affichant le caisson concerné
Exceptions	**La connexion réseau est perdue sur un poste :**

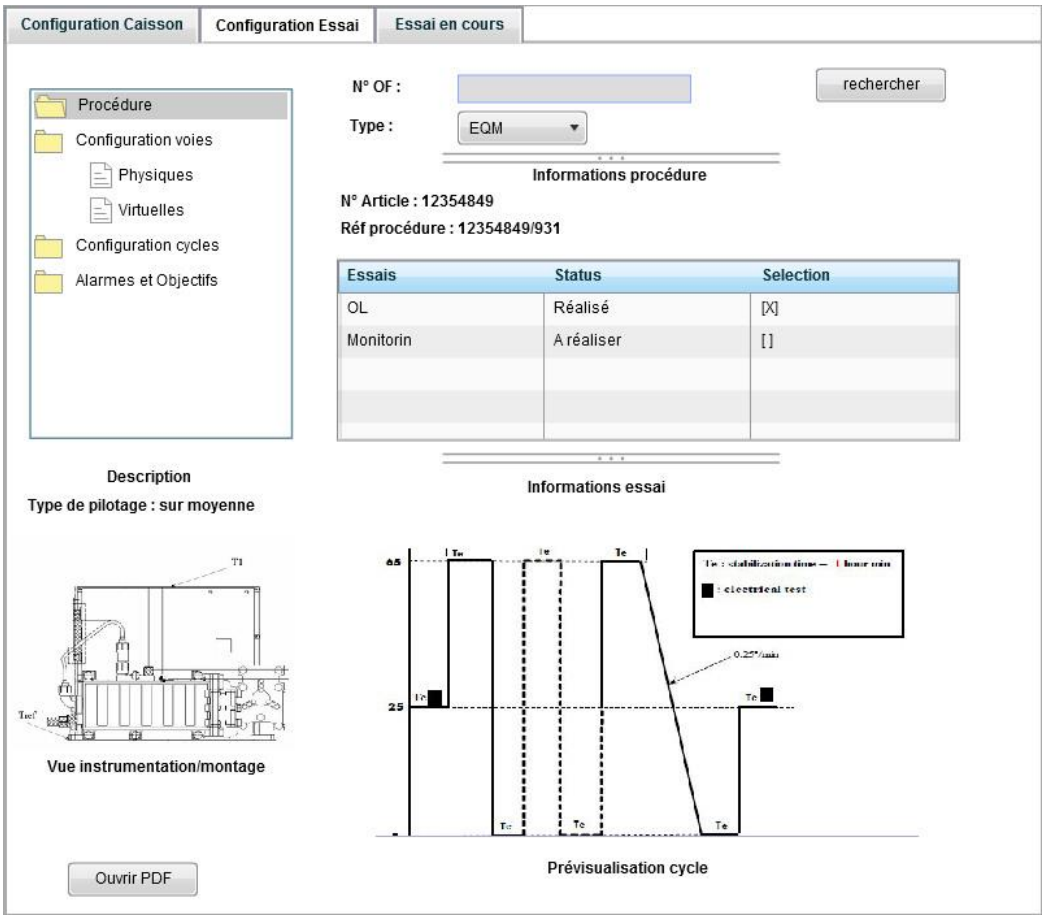
	• L'application signale visuellement le problème tant que le problème n'est pas résolu.
Règles métier	Savoir comment repérer les alarmes
Post-condition	N/A

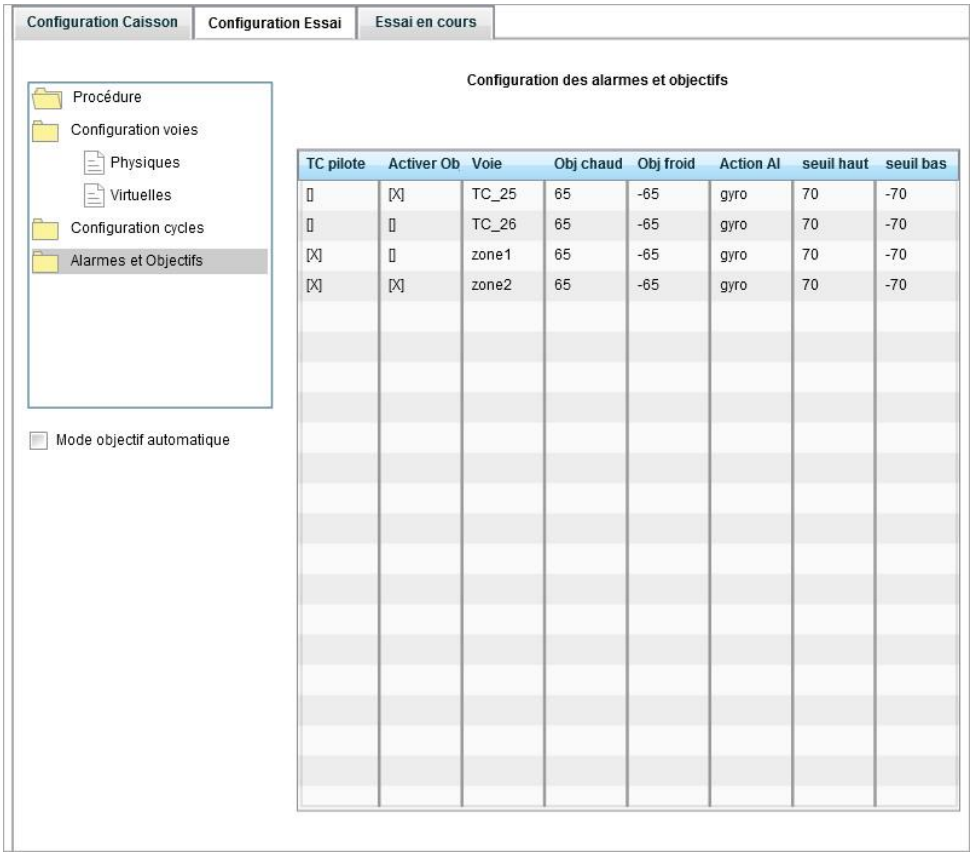
Annexe 2 : Prototypes IHM

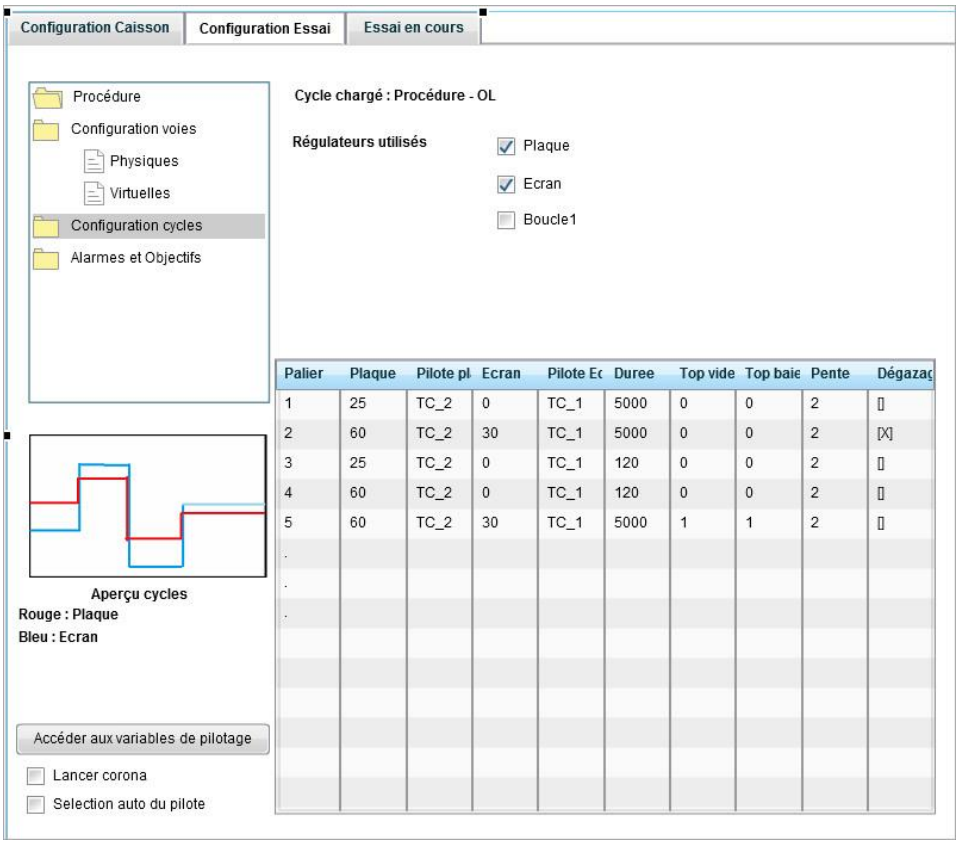


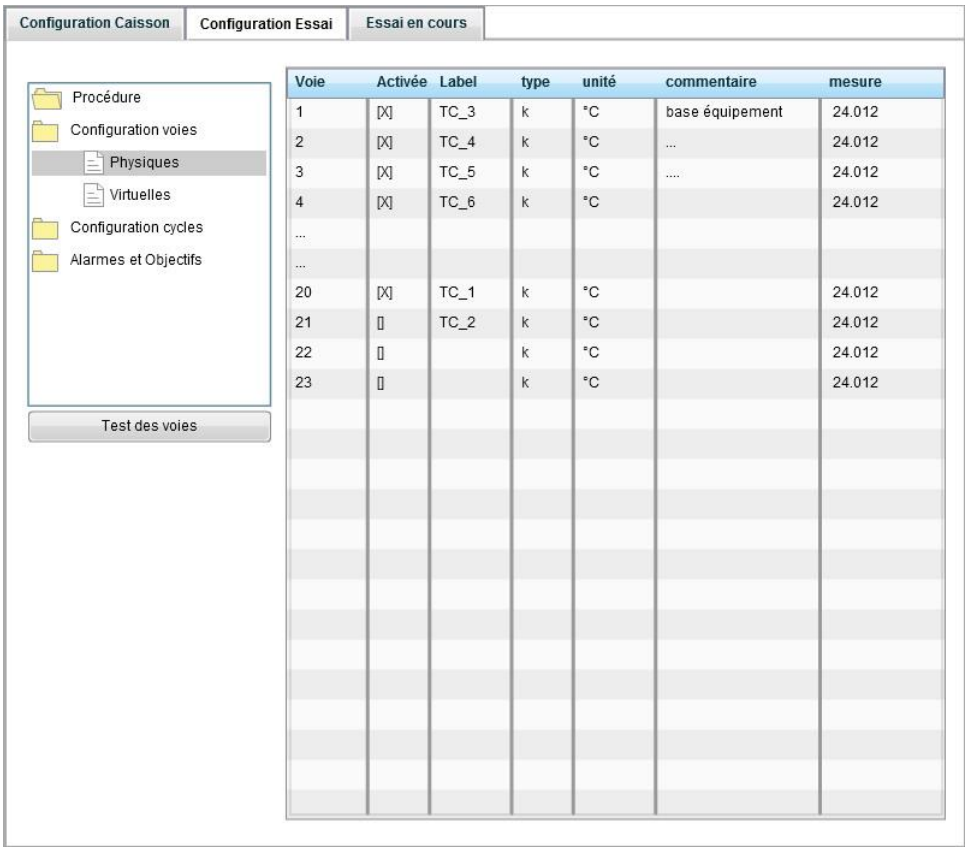

Configuration Caisson
Configuration Essai
Essai en cours
Périphériques détectés
Mesure
Temp./Tens./Resist.
Pression
Humidité
Régulation
Commande et Top
Instruments compatibles disponibles
Inficon VGC403 @ COM1
Voie 1
Voie 2
Voie 3
Ajouter
Supprimer
Voies de mesures de pression configurées
Voie
instrument
port
pression caissor
mesure
Voie 2
Inficon VGC403
COM1
100
Voie 1
Inficon VGC403
COM1
x
100
Voie 3
Inficon VGC403
COM1
100

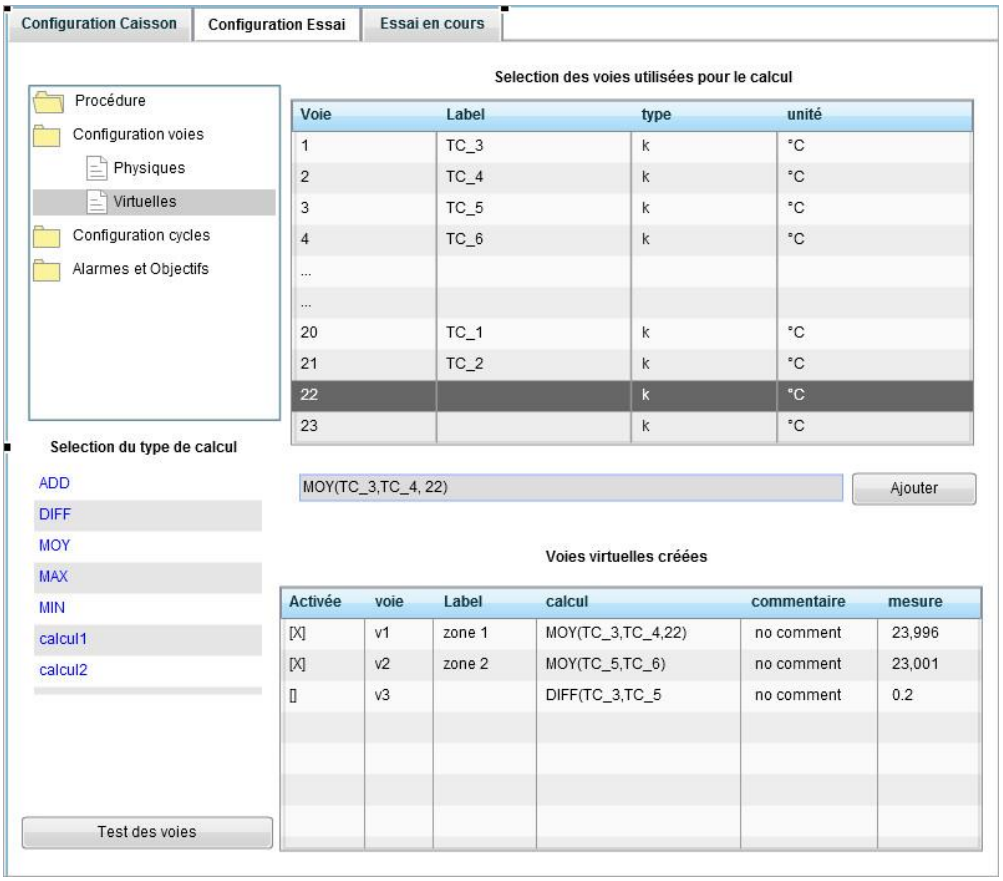

Configuration Caisson
Configuration Essai
Essai en cours
Périphériques détectés
Mesure
Régulation
Commande et Top
Instruments compatibles disponible
Centrale HP34970A/idxxxx @ COM1
Slot2 carte Relais (20 voies)
Centrale HP34980A/idxxxx @ COM2
Slot1 carte Relais (40 voies)
Ajouter
Supprimer
Voies de commande et top configurées
Nom
instrument
slot
Voie Instrument
Top vide
HP34970A @COM1
2
201
Top baie
HP34970A @COM1
2
202
Thermique
HP34980A @COM2
1
101
Alarme
HP34980A @COM2
1
202
Tester voie

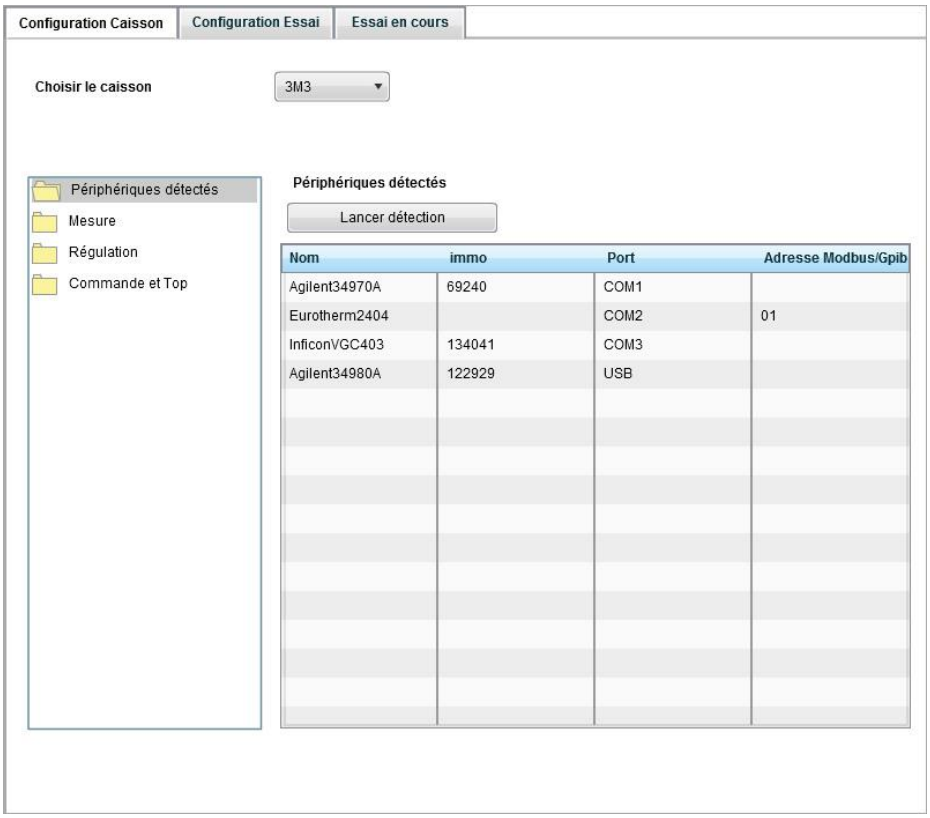

Configuration Caisson
Configuration Essai
Essai en cours
Procédure
Configuration voies
Physiques
Virtuelles
Configuration cycles
Alarmes et Objectifs
N° OF :
rechercher
Type :
EQM
Informations procédure
N° Article : 12354849
Réf procédure : 12354849/931
Essais
Status
Selection
OL
Réalisé
[X]
Monitorin
A réaliser
[ ]
Informations essai
Description
Type de pilotage : sur moyenne
Vue instrumentation/montage
Prévisualisation cycle
Ouvrir PDF

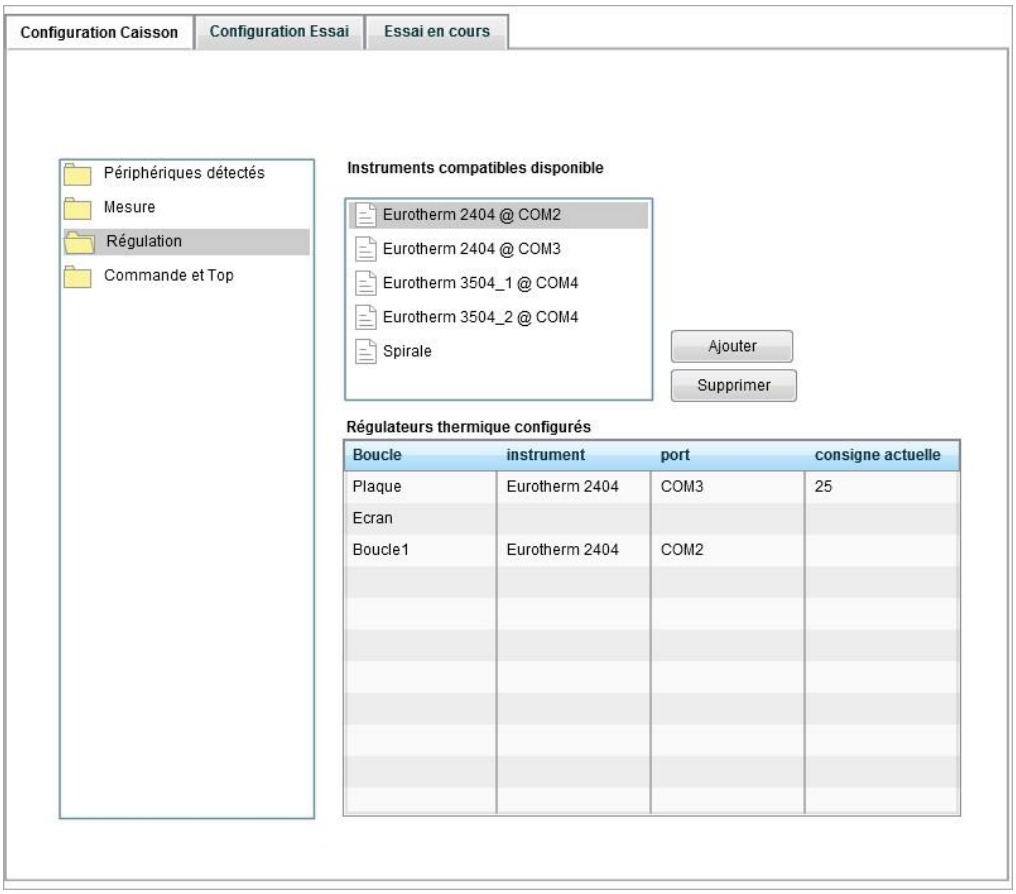

Configuration Caisson
Configuration Essai
Essai en cours
Procédure
Configuration voies
Physiques
Virtuelles
Configuration cycles
Alarmes et Objectifs
Mode objectif automatique
Configuration des alarmes et objectifs
TC pilote
Activer Ob
Voie
Obj chaud
Obj froid
Action AI
seuil haut
seuil bas
[]
[X]
TC_25
65
-65
gyro
70
-70
[]
[]
TC_26
65
-65
gyro
70
-70
[X]
[]
zone1
65
-65
gyro
70
-70
[X]
[X]
zone2
65
-65
gyro
70
-70


Configuration Caisson
Configuration Essai
Essai en cours
Procédure
Configuration voies
Physiques
Virtuelles
Configuration cycles
Alarmes et Objectifs
Cycle chargé : Procédure - OL
Régulateurs utilisés
Plaque
Ecran
Boucle1
Palier Plaque Pilote pl Ecran Pilote Ec Duree Top vide Top baie Pente Dégazag
1 25 TC_2 0 TC_1 5000 0 0 2 []
2 60 TC_2 30 TC_1 5000 0 0 2 [X]
3 25 TC_2 0 TC_1 120 0 0 2 []
4 60 TC_2 0 TC_1 120 0 0 2 []
5 60 TC_2 30 TC_1 5000 1 1 2 []
Aperçu cycles
Rouge : Plaque
Bleu : Ecran
Accéder aux variables de pilotage
Lancer corona
Selection auto du pilote


Configuration Caisson
Configuration Essai
Essai en cours
Procédure
Configuration voies
Physiques
Virtuelles
Configuration cycles
Alarmes et Objectifs
Test des voies
Voie Activée Label type unité commentaire mesure
1 [X] TC_3 k °C base équipement 24.012
2 [X] TC_4 k °C ... 24.012
3 [X] TC_5 k °C 24.012
4 [X] TC_6 k °C 24.012
20 [X] TC_1 k °C 24.012
21 [] TC_2 k °C 24.012
22 [] k °C 24.012
23 [] k °C 24.012


Configuration Caisson
Configuration Essai
Essai en cours
Procédure
Configuration voies
Physiques
Virtuelles
Configuration cycles
Alarmes et Objectifs
Selection des voies utilisées pour le calcul
Voie Label type unité
1 TC_3 k °C
2 TC_4 k °C
3 TC_5 k °C
4 TC_6 k °C
20 TC_1 k °C
21 TC_2 k °C
22 k °C
23 k °C
Selection du type de calcul
ADD
DIFF
MOY
MAX
MIN
calcul1
calcul2
MOY(TC_3,TC_4, 22)
Ajouter
Voies virtuelles créées
Activée voie Label calcul commentaire mesure
[X] v1 zone 1 MOY(TC_3,TC_4,22) no comment 23,996
[X] v2 zone 2 MOY(TC_5,TC_6) no comment 23,001
[] v3 DIFF(TC_3,TC_5 no comment 0.2
Test des voies


Configuration Caisson
Configuration Essai
Essai en cours
Choisir le caisson
3M3
Périphériques détectés
Mesure
Régulation
Commande et Top
Périphériques détectés
Lancer détection
Nom immo Port Adresse Modbus/Gpib
Agilent34970A 69240 COM1
Eurotherm2404 COM2 01
InficonVGC403 134041 COM3
Agilent34980A 122929 USB



Annexe 3 : Exigences capturées

ID	Nom	Obligatoire	Définition	Cas d'utilisation	Tests	Remarque
1	SPEC-INT-001	Obligatoire	Le produit doit pouvoir écrire dans le fichier de gestion des temps	Configurer essai		Fichier Excel ou txt. attention aux accès concurrents. Voir si intérêt de remplacer fichier par BDD
2	SPEC-FONC-002	Obligatoire	Le produit doit gérer la création et la modification des fichiers de configuration des caissons au format XML.	Configurer caisson		Le format du fichier sera fourni par Thalès.
3	SPEC-FONC-003	Obligatoire	Le produit doit gérer la création et la modification des fichiers de calibration des appareils de mesure au format XML.	Configurer essai		Le format du fichier sera fourni par Thalès. Le fichier actuellement utilisé est un fichier texte.

4	SPEC-INT-004	Obligatoire	Le produit doit gérer la création des fichiers de maintenance au format txt.	Piloter essai		Le format du fichier sera fourni par Thalès.
5	SPEC-INT-005	Obligatoire	Le produit doit gérer la création des fichiers de SuperCalcul au format .calcul.	Piloter essai		Le format du fichier sera fourni par Thalès.
6	SPEC-INT-006	Obligatoire	Le produit doit gérer la communication avec les serveurs 3it et ECM pour le stockage des fichiers pdf.	Gérer procédure Configurer essai Piloter essai		Les interfaces de connexion seront fournies par Thalès. Une BDD Thalès (Alice) permet de récupérer les refs des documents applicable à une affaire donnée.
7	SPEC-FONC-007	Optionnel	Le produit doit pouvoir interagir avec un caisson virtuel (émulation des instruments) afin de pouvoir valider les fonctions logiciel.	Configurer caisson		Voir les possibilités des drivers simulés IVI.
8	SPEC-FONC-008	Obligatoire	L'opérateur doit pouvoir créer une nouvelle procédure.	Gérer procédure		La procédure est créée si les différents champs d'identifiant sont renseignés et si elle n'existe pas déjà. Il peut s'agir d'une procédure de maintenance
9	SPEC-FONC-009	Obligatoire	L'opérateur doit pouvoir charger une procédure existante à partir : du nom de l'équipement du nom de l'affaire du n° article	Gérer procédure Configurer essai		Requête à faire sur BDD TAS (Alice)
10	SPEC-FONC-010	Obligatoire	L'opérateur doit pouvoir éditer une procédure en y ajoutant : images	Gérer procédure		Se référer à une procédure générée.

			commentaires cycles			
11	SPEC-FONC-011	Obligatoire	L'utilisateur doit pouvoir exporter la procédure au format pdf sur le serveur ECM.	Gérer procédure		Les procédures doivent être envoyées sur le réseau pour être validée. Elle est stockée également en local afin de pouvoir la modifier.
12	SPEC-FONC-012	Obligatoire	L'utilisateur doit pouvoir importer et exporter les données de la procédure dans un format XML.	Gérer procédure Configurer essai		La version numérique servira à paramétrer les essais. Le format du fichier XML sera fourni par Thalès.
13	SPEC-ERG-013	Obligatoire	L'utilisateur doit pouvoir éditer la liste des types d'équipement via une vue d'option avancée.	Configurer caisson		L'information sur le type d'équipement est présent dans la procédure d'essai.
14	SPEC-ERG-014	Obligatoire	Le système doit permettre la création d'un cycle sous forme de tableau.	Configurer essai		Se référer à l'ancien logiciel
15	SPEC-ERG-015	Obligatoire	Le système doit proposer un aperçu du cycle entré (cet aperçu ne représentera pas la pente)	Configurer essai		Voir TeeChart. Fourniture Thalès
16	SPEC-FONC-016	Obligatoire	L'utilisateur doit pouvoir préciser si le palier du cycle est un palier Normal ou Transitoire	Configurer essai		Définition d'une liste de sélection paramétrable. Ce paramètre influe sur la condition d'accrochage des paliers.
17	SPEC-FONC-017	Obligatoire	Le système doit permettre la création, la sauvegarde et le chargement d'un fichier de configuration de caisson au format XML.	Configurer caisson		
18	SPEC-ERG-018	Obligatoire	Le système doit afficher une vue spécifique en fonction du type d'instruments à configurer	Configurer caisson		Fourniture Thalès

19	SPEC-FONC-019	Obligatoire	Le système doit être capable de tester la communication avec les instruments	Configurer caisson		Fourniture Thalès
20	SPEC-FONC-020	Obligatoire	Le produit doit permettre de configurer la liste des voies de mesures disponibles sur les équipements.	Configurer essai		
21	SPEC-FONC-021	Obligatoire	L'utilisateur doit être capable de tester les valeurs de mesure des instruments et de valider le bon fonctionnement de ceux-ci.	Configurer essai		La validation des données mesurées sera de la responsabilité de Thalès.
22	SPEC-FONC-022	Obligatoire	L'utilisateur doit pouvoir créer des voies de mesures virtuelles.	Configurer essai		Les voies virtuelles sont calculées en fonction des voies de mesures physique (matérielles).
23	SPEC-FONC-023	Optionnel	Le système doit permettre l'ajout de nouveau type de calcul (exemple : appel d'une fonction définie dans un script groovy)	Configurer essai		Fourniture Thalès
24	SPEC-FONC-024	Optionnel	L'utilisateur doit pouvoir appliquer plusieurs calculs pour une voie virtuelle (exemple : MOY(1102,1104) - (1105 + 1101))	Configurer essai		Fourniture Thalès
25	SPEC-FONC-025	Obligatoire	L'utilisateur doit pouvoir configurer des alarmes pour chaque voie de mesures	Configurer essai		
26	SPEC-FONC-026	Obligatoire	L'utilisateur doit pouvoir configurer une action à exécuter en cas d'alarme.	Configurer essai		La liste des actions possibles sera fournie par Thalès.
27	SPEC-FONC-027	Obligatoire	L'utilisateur doit pouvoir configurer des objectifs pour chaque voie de mesures	Configurer essai		
28	SPEC-ERG-028	Obligatoire	Le système doit afficher les cycles présents dans	Configurer		Fourniture Thalès

			la procédure si l'essai est associé à une procédure	essai Piloter essai		
29	SPEC-FONC-029	Obligatoire	Le système doit permettre la modification des paramètres de l'essai en cours.(cycles, voies pilotes, variables de pilotage)	Piloter essai		Ces modifications ne sont disponibles que pour les utilisateurs avancés.
30	SPEC-FONC-030	Obligatoire	Le système doit permettre le choix d'un TC pilote pour chaque palier si la case « autoriser le changement de pilote » est activée	Configurer essai		
31	SPEC-ERG-031	Optionnel	Le système doit pouvoir être traduit d'en d'autres langues aisément par l'utilisation de fichiers langue.			Toutes les traductions devront être fournies par THALES ALENIA SPACE
32	SPEC-ERG-032	Obligatoire	Le système doit être adapté aux résolutions supérieures ou égales à 1024*768 dans les proportions 4/3, 16/9 et 16/10.			Tests dans les résolutions à intégrer
33	SPEC-ERG-033	Optionnel	Le système doit proposer une même interface adapté à la fois à une utilisation tactile et à une utilisation clavier/souris.			Contraintes de conception / design
35	SPEC-ERG-034	Obligatoire	Le système doit proposer une interface permettant de superviser environ 30 essais en temps réel (onglets, fenêtres internes)	Superviser essai		Fourniture Thalès
36	SPEC-FONC-035	Obligatoire	Le système doit permettre de paramétrer l'affichage des courbes (couleurs, axes, données à afficher…) -> voir TeeChart.	Configurer essai Piloter essai Superviser essai Gestion PV essai		Fourniture Thalès

37	SPEC-FONC-036	Obligatoire	Le système doit permettre de suivre l'avancement de l'essai suivant le cycle qui aura été configuré	Piloter essai Superviser essai		Indicateur du palier en cours
38	SPEC-FONC-037	Obligatoire	Le système doit permettre de donner une estimation de la durée de l'essai en temps réel en fonction des données mesurées.	Configurer essai Piloter essai Superviser essai		L'estimation de la durée de l'essai est calculée en fonction des durées de paliers définies dans le cycle, ainsi qu'en fonction des pentes réelles observées
39	SPEC-ERG-038	Obligatoire	Le système doit permettre de protéger toutes les fonctions permettant d'influencer le déroulement de l'essai par un mot de passe.	Configurer essai Piloter essai		Règle à fournir par THALES ALENIA SPACE
40	SPEC-FONC-039	Obligatoire	Le système doit proposer une fenêtre de logs avec la possibilité de paramétrer niveaux de détails des messages afficher (erreurs, warning, etc…)	Piloter essai Superviser essai		Permet de gérer des alarmes et des warnings pour la supervision des essais.
41	SPEC-FONC-040	Obligatoire	L'utilisateur doit pouvoir définir plusieurs cycles de température dans le cas d'un essai avec baie de réchauffage	Configurer essai		
42	SPEC-FONC-041	Obligatoire	Le logiciel doit pouvoir créer une procédure localement si le réseau n'est pas disponible	Gérer procédure		La procédure est sauvegardée localement mais ne peut être envoyé pour validation sur le serveur que lorsque le réseau est disponible.
43	SPEC-FONC-042	Obligatoire	L'utilisateur peut choisir de superviser un essai localement ou à distance	Piloter essai Superviser essai		Les données des essais doivent être mises à disposition sur le réseau pour la supervision à distance.

44	SPEC-FONC-043	Obligatoire	Les mesures devront être enregistrées et datées dans différents format de fichier (.csv, .txt, xml)	Piloter essai		Formats des fichiers à fournir par Thalès.
45	SPEC-ERG-044	Obligatoire	L'affichage de la supervision devra être totalement configurable (nombre de fenêtres, nombre de voies à afficher, visualisation des sécurités configurées) voir TeeChart	Superviser essai		Fourniture Thalès
46	SPEC-ERG-046	Obligatoire	L'utilisateur doit pouvoir zoomer sur une zone de courbe	Configurer essai Piloter essai Superviser essai		Utilisation de TeeChart (licence fournie par THALES ALENIA SPACE).
47	SPEC-FONC-047	Obligatoire	L'utilisateur doit pouvoir rejouer un essai dans son ensemble (évolution de la pression, température, alarmes, message de logs)	Configurer essai Piloter essai		Sauvegarde en local des paramètres de la dernière exécution.
48	SPEC-FONC-048	Obligatoire	Le système doit être capable de piloter les outils de régulation en fonction des données de mesures.	Piloter essai		Fourniture Thalès
50	SPEC-FONC-049	Obligatoire	L'utilisateur doit pouvoir configurer autant de boucles de régulations que souhaité avec paramétrage de cycle indépendant	Configurer essai		Fourniture Thalès
51	SPEC-FONC-050	Obligatoire	L'utilisateur doit pouvoir modifier le/les cycles défini(s) pour un essai en cours	Piloter essai		
52	SPEC-FONC-051	Optionnel	L'utilisateur doit pouvoir éditer les méthodes de pilotage sous forme de script	Configurer essai Piloter essai		Fourniture Thalès
53	SPEC-FONC-052	Obligatoire	L'utilisateur doit pouvoir changer la méthode de	Piloter essai		

			pilotage pour un essai en cours			
54	SPEC-FONC-053	Obligatoire	L'utilisateur doit pouvoir paramétrer l'affichage des courbes à afficher dans le PV.	Gestion PV essai		Utilisation de TeeChart (licence fournie par THALES ALENIA SPACE).
55	SPEC-FONC-054	Obligatoire	L'utilisateur doit pouvoir insérer la totalité des informations relative à l'essai (configuration du poste, nom client, thermicien, durée de l'essai, mesures...)	Gestion PV essai		
56	SPEC-FONC-070	Obligatoire	L'utilisateur doit pouvoir ajouter des commentaires au PV	Gestion PV essai		
57	SPEC-FONC-055	Obligatoire	L'utilisateur doit pouvoir exporter le PV au format pdf .	Gestion PV essai		Fourniture Thalès
58	SPEC-FONC-056	Obligatoire	L'utilisateur doit pouvoir choisir l'emplacement des données sur le disque (fichiers essais, emplacement des scripts de pilotage)	Configurer caisson Configurer essai Gestion PV essai		
60	SPEC-FONC-057	Obligatoire	L'utilisateur doit pouvoir définir le matériel de laboratoire communiquant avec le logiciel (régulateurs, mesure de pression, mesure de température)	Configurer caisson		Sous réserves que tous les appareils soient connus par le logiciel.
62	SPEC-PERF-058	Obligatoire	Le logiciel doit permettre une acquisition de 300 voies de mesures dans un temps inférieur à 5 sec (si les moyens de communication le permettent).	Piloter essai		Fourniture Thalès
63	SPEC-PERF-060	Obligatoire	Le système doit être capable de fonctionner pendant plusieurs mois (environ 3 mois).	Piloter essai Superviser essai		Fourniture Thalès
64	SPEC-CONC-061	Obligatoire	Le logiciel doit pouvoir	Configurer		Fourniture Thalès.

			ajouter de nouveaux instruments facilement (drivers)	caisson		Voir intégration des drivers VISA/IVI fournis par les constructeurs
65	SPEC-FONC-062	Optionnel	Le logiciel doit pouvoir vérifier automatiquement si une mise à jour est disponible			Les ordinateurs n'ont pas accès au réseau externe. Les mises à jour devront être rendu disponibles sur le réseau interne de Thalès.
66	SPEC-CONC-063	Obligatoire	Le système devra utiliser la librairie TeeChart pour la gestion des courbes	Gérer procédure Configurer essai Piloter essai Superviser essai		Fourniture Thalès
67	SPEC-OPE-064	Obligatoire	Le système doit être capable de fonctionner avec une perte de réseau	Gérer procédure Configurer essai Piloter essai Gestion PV essai		Sauvegarde des résultats en local et partage sur le réseau lorsque celui-ci redeviens disponible.
68	SPEC-OPE-065	Optionnel	Le système doit être capable de reprendre un essai après une coupure électrique ou un redémarrage involontaire	Piloter essai Superviser essai		Fourniture Thalès Detection d'un essai interrompu au demarrage de l'application
69	SPEC-OPE-066	Obligatoire	Le système doit être capable de gérer une perte des données d'acquisition et le signaler à l'operateur	Piloter essai Superviser essai		Liste des warnings et des alarmes à définir par Thalès. Possibilité d'utiliser le buffer des instruments de mesures.
70	SPEC-CONC-067	Optionnel	Le produit doit pouvoir être adaptable à différentes utilisations sans impact sur l'architecture générale, essentiellement les			Le logiciel doit proposer des fonctionnalités différentes dépendamment s'il est installé sur un

			interfaces du logiciel.			caisson, sur un écran de supervision ou dans un bureau.
71	SPEC-CONC-068	Optionnel	L'installation du produit doit être configurable -> possibilité de désactiver certaines fonctionnalités.			Le logiciel doit proposer des fonctionnalités différentes dépendamment s'il est installé sur un caisson, sur un écran de supervision ou dans un bureau.
72	SPEC-CONC-069	Optionnel	Le développement du produit doit tenir compte que la supervision des essais pourra être portée sur smartphone/Tablet (Android).			Le chiffrage n'intègre pas le portage vers Android.

Annexe 4 : Classe instrument iControl

```
/*
 *[..]
 */
package icontrol.drivers.instruments.agilent;
import icontrol.AutoGUIAnnotation;
import icontrol.Utilities;
import icontrol.drivers.Device;
import icontrol.iC_Annotation;
import icontrol.drivers.Device.CommPorts;
import java.io.IOException;
import java.util.Random;
import java.util.logging.Logger;
import java.util.zip.DataFormatException;
import javax.script.ScriptException;

// promise that this class supports RS232 communication
@iC_Annotation(CommPorts=CommPorts.RS232,
                InstrumentClassName="Agilent34970A")
```

```
public class Agilent34970A extends Device {


    ////////////////////
    // member variables

    /**
     * The Logger for this class. Note that the Logger Level is set in the
     * constructor of the <code>Device</code> class.
     */
    // change the Logger name to the name of your class
    private         static         final         Logger         m_Logger         =
Logger.getLogger("iC.Instruments.Agilent34970A");

    /**
     * get temperature for the given channel
     * @param Channel
     * @return temperature measured
     * @throws java.util.zip.DataFormatException
     * @throws java.io.IOException
     * @throws javax.script.ScriptException
     */
     @AutoGUIAnnotation(
       DescriptionForUser = "Get the temperature.",
       ParameterNames = {"Channel"},
       DefaultValues= {"101"})
    @iC_Annotation(MethodChecksSyntax = true )
    public double getTemp(int Channel)
            throws DataFormatException, IOException, ScriptException{

          ////////////////
        // Syntax-Check

        if (Channel < 100 || Channel > 128) {
            String str = "Channel must be between 100 or 128\n";
            throw new DataFormatException(str);
        }

        // return if in Syntax-check mode
```

```
        if (inSyntaxCheckMode())
            return Double.NaN;

        // exit if in No-Communication-Mode
        if (inNoCommunicationMode())
            return Double.NaN;
        // get a measurement
        String Answer = QueryInstrument("MEAS:TEMP? TC,k,1,0.1,(@" + Channel
+")");
        // strip off the unit: remove everything other than a number or a .
        String NumberOnly = Answer.replaceAll("[^0-9\\.]", "");
         // convert to a Double
        double ret = Utilities.getDouble(NumberOnly);
        return ret;
    }
}
```

Annexe 5 : Classe AbstractDevice

```
using System;
using System.ComponentModel;
using System.Xml.Serialization;
using VTH_Lib.Models.Drivers;
namespace VTH_Lib.Models.Devices
{
    public abstract class AbstractDevice : INotifyPropertyChanged, IConvertible
    {
        #region GUID
        [XmlIgnore]
        public Guid Guid { get; private set; }
        #endregion

        #region Identification
        public string Name { get; set; }
        public string IdentityNumber { get; set; }
        #endregion

        #region Communication
        public string Port { get; set; }
        public string Address { get; set; }
        public DriverType Protocol { get; set; }
        public AbstractDriverVTH Driver { get; set; }
        #endregion
        public AbstractDevice(string identityNb = "", string name = "", string
port = "", string address = "")
```

```
        {
            //GUID
            Guid = Guid.NewGuid();
            //Identification
            Name = name;
            IdentityNumber = identityNb;
            //Communication
            Port = port;
            Address = address;
        }
        public  AbstractDevice(string  identityNb,  string  name,  string  port,
AbstractDriverVTH driver,  string address = "")
         :this(identityNb, name, port, address)
        {
            Driver = driver;
        }

        public T Cast<T>(object obj)
        {
            return (T)obj;
        }
        #region Properties changed
        /// <summary>
        /// Permet de notifier la vue lorsqu'une propriété est mise à jour
        /// </summary>
        public event PropertyChangedEventHandler PropertyChanged;
        private void NotifyPropertyChanged(String propertyName)
        {
            PropertyChangedEventHandler handler = PropertyChanged;
            if (null != handler)
            {
                handler(this, new PropertyChangedEventArgs(propertyName));
            }
        }
        #endregion
    }
}
```

Annexe 6 : Classes Eurotherm3504

```
using System;
using VTH_Lib.Models.Channels;
using VTH_Lib.Models.Devices.Interfaces;
using VTH_Lib.Models.Drivers;

namespace VTH_Lib.Models.Devices.DeviceDriver
{
    public  class  Eurotherm3504  :  RegulationDevice,  IThermalRegulationDevice,
IMeasureDevice
    {
        /// <summary>
        /// Constructeur par défaut, l'adresse esclave utilisé est 1
```

```
        ///
        /// </summary>
        /// <param name="idNum"></param>
        /// <param name="port"></param>
        public Eurotherm3504(string idNum, string port)
            : base(idNum, "Eurotherm3504", port)
        {
            //initialisation du driver
            this.Driver       =       DriverVTHFactory.Instance.CreateDriver(port,
DriverType.Modbus);
            this.Driver.SetTimeOut(2000);
            CreateChannels();

        }

        public Eurotherm3504(string idNum, string port, int adrrSlave)
            : base(idNum, "Eurotherm2504", port)
        {
            //initialisation du driver avec l'adresse esclave passée en paramètre
            this.Driver       =       DriverVTHFactory.Instance.CreateDriver(port,
DriverType.Modbus, new string[] { adrrSlave.ToString() });
            this.Driver.SetTimeOut(2000);
            CreateChannels();

        }

        /// <summary>
        /// Constructeur utilisé pour la detection et création automatique dans
DeviceFactory
        /// </summary>
        /// <param name="idNum"></param>
        /// <param name="port"></param>
        /// <param name="driver"></param>
        public Eurotherm3504(string idNum, string port, AbstractDriverVTH driver)
            : base(idNum, "Eurotherm3504", port, driver)
        {
            CreateChannels();
        }

        private void CreateChannels()
        {
            //initialisation du nombre de voie de regulation correspondant au
modèle (boucle de régulation)
            //l'adresse physique de la voie correspond ici au numero de registre
modbus du SetPoint (2)
            ChannelFactory.Instance.CreateRegulationChannel(2, this);
            ChannelFactory.Instance.CreateRegulationChannel(1026, this);

            //ajout des voies de mesures qui correspondent aux variables de
process disponibles (@ registre 1 et 1025)
            ChannelFactory.Instance.CreateMeasureChannel(1, this);
            ChannelFactory.Instance.CreateMeasureChannel(1025, this);
        }

        public void WriteSetPoint()
        {
            foreach           (RegulationChannel           channel           in
RegulationChannelsList.FindAll(RegulationChannel  =>  RegulationChannel.Actived  ==
true))
```

```
            {
                float setPoint = channel.SetPoint * 10;
                if (setPoint < 0.0f)
                {
                    setPoint += 65536.0f;
                }

                //ecriture de la consigne à l'adresse registre correspondante
                this.Driver.SendCommand(setPoint.ToString(),    new    string[]    {
channel.PhysicalAdress.ToString() });
            }
        }
        /// <summary>
        /// Lecture de la variable de process
        /// </summary>
        public void Measure(bool saveResultAsMeasure)
        {
            DateTime time = DateTime.Now;
            foreach (MeasureChannel channel in MeasureChannelsList)
            {
                float measuredValue;
                //acquisition  et  conversion  de  la  valeur  en  float  à  l'adresse
spécifiée
                if
(float.TryParse(this.Driver.QueryCommand(channel.PhysicalAdress.ToString(),    new
string[] { "1"}), out measuredValue))
                {
                    //la valeur récupérée est un short non signé ushort.
                    //conversion dans une valeur signé
                    if (measuredValue > 32767.0f)
                    {
                        measuredValue -= 65536.0f;
                    }
                    //l'eurotherm utilise des entiers naturels
                    measuredValue = measuredValue / 10.0f;

                    if(saveResultAsMeasure)
                        channel.SetValue(time, measuredValue);
                    else
                        channel.Value = measuredValue;
                }
            }
        }

        public void MeasureAll(bool saveResultAsMeasure)
        {
            Measure(saveResultAsMeasure);
        }

        public bool IsDeviceDetected()
        {
            string  result  =  this.Driver.QueryCommand("122",  new  string[]  {  "1"
});
            if(null != result)
            {
                string hex = int.Parse(result).ToString("X");
                if (hex.Contains("3504"))
                {
                    return true;
```

```
                }
            }
           // string hexValue = hex.ToString("X");
            return false;
        }
    }
}
```

```
using VTH_Lib.Models.Devices.Interfaces;
using VTH_Lib.Models.Drivers;

namespace VTH_Lib.Models.Devices.DeviceDetection
{
    public class Eurotherm3504Detection : IDeviceDetection
    {
        public Eurotherm3504Detection() { }

        public bool IsDeviceDetected(AbstractDriverVTH driver)
        {
            //configuration du port avec les paramètres de l'instrument
(baudrate, parity, data, stop, handshake)
            driver.SetParameters(new string[] { "9600", "none", "8", "one",
"none" });
            //lecture du registre 122 pour identification de l'appareil
            string result = driver.QueryCommand("122", new string[] { "1" });
            if (result == "58432")
            {
                return true;
            }
            // TODO: Voir pourquoi la conversion en HEX ne donne pas le modèle
comme le fait le 2404

            return false;
        }

        public DriverType GetDriverType()
        {
            return DriverType.Modbus;
        }
        public string GetDeviceClassName()
        {
            return "Eurotherm3504";
        }
    }
}
```

Annexe 7 : Description algorithmique des activités de pilotage

Fonctions:

AlarmeSupervision(NiveauAlarme) : Envoi l'alarme à la supervision avec son type en paramètre

AlarmeClient() : Ouvre le contact de l'alarme client

ZoneAccrochage(Tspec) : Permet de calculé la fourchette d'accrochage autour de la

température spécifiée passée en paramètre.(5.10-5*Tspec²+ValAccr)

MesurePente(mesure1, mesure2) : pente entre deux mesures passées en paramètre

Compteurs :

CpterSecuPression : déclenche un arrêt thermique ou un retour à 25° au bout de 2 min si la pression remonte.

CpterSecuMinMax : Compteur utilisé dans le contrôle des sécu min max

CpterStab : compteur utilisé pour la stabilisation du régulateur à la température spécifiée

CpterAjus : Compteur utilisé pour amener le régulateur à la température spécifiée

CpterDuréePalier :Compteur utilisé pour mesurer la durée du palier

Constantes

OffsetSecuMinMax : sécurité sur les températures extrême de l'essai (10° par défaut)

OffsetAlarmOrange :sécurité utilisée en palier (5° par défaut)

OffsetSecuOvershoot :sécurité utilisée en transition (8° par défaut)

OffsetSecuPalier : sécurité utilisée en palier (8° par défaut)

SeuilSecuPression : seuil de sécurité pression en mb

PressionSpecifiée : pression spécifiée dans la procédure par le client

FrequenceAcquisition : fréquence de timer d'acquisition des mesures (1min par défaut)

CoefCor :Coefficient de correction utilisé pour le calcul de la consigne (2 par défaut)

Variables par régulateur

DureeAjust: Durée entre chaque ajustement pour atteindre la température spécifiée

DureeCorr : Durée d'attente entre chaque correction pour revenir à la température spécifiée

Spente: Seuil de la pente (pente max)

ValAccr: Valeur de la différence maxi entre la température spécifiée et la température du pilote pour l'accrochage des paliers

CutBack: Valeur de la différence entre la température spécifiée et la température pilote pour l'ajustement de la consigne

PenteAccr: Pente max pour déclarer le température stabilisée dans la zone d'accrochage palier (0.05°/min)

PenteAjust :Pente qui permet de définir si l'on a besoin d'ajuster la consigne pour atteindre la temperature spécifiée

Phase : défini dans quel phase se situe le régulateur {Transition, Approche,

Attente, Accroché, Correction}

Tspec : Température spécifiée

TPilMes[] : tableau des températures du pilote mesurées

PenteSpec : Pente régulateur spécifiée

PenteMes : Pente réelle mesurée

Cons :Consigne en cours

ConsMem : Consigne mémorisée du dernier palier similaire coché (tableau[nbpalier])

Variable essais :

EtatCaisson : {Arret , Retour25, Nominal} l'etat du caisson à l'instant

DuréePalier :Durée totale du palier

MemCons :Défini si la mémorisation de consigne doit être utilisée pour la palier en cours

PalierEnCours : n° du palier en cours

Sens : 1 si positif -1 sinon

Mode : Défini le mode de pilotage de l'essai {Conductif, Radiatif, Convectif}

OffsetCycle : Offset à appliquer aux températures saisies dans le cycle (IHM conf essai)

Remarque :

= : prend la valeur

== : test si la valeur est égale à

Mesure :

```
POUR CHAQUE VoiesDeMesure FAIRE
        SI Voie == Active
                Acquisition(Voie)
        FIN SI
FIN POUR
```

Securite :

```
//contrôle de la pression
SI TypeCaisson != Clim
// on ne contrôle pas la pression sur les enceintes climatiques
        SI PressionCaisson == null
// pas de mesures de pression on arrete le termique et on declenche les alarmes
```

```
                AlarmeSupervision(Rouge)
                EtatCaisson = Arret
                AlarmeClient()
        SINON SI PressionCaisson > SeuilSecuPression
                SI PalierEnCours == degazage || PalierEnCours == 1
                        CpterSecuPression++
                        SI CpterSecuPression >= 2 min
                                AlarmeSupervision(Rouge)
                                EtatCaisson = Arret
                                AlarmeClient()
                        FIN SI
                SINON
                        EtatCaisson = Retour25
                        CpterSecuPression++
                        SI CpterSecuPression >= 2 min
                                AlarmeClient()
                FIN SI
                FIN SI
        SINON SI PressionCaisson > PressionSpecifiée * (1+1/5)
        // si la pression est caisson est remontée depuis la dernière mesure
                AlarmeSupervision(Orange)
        FIN SI
FIN SI



//TC secu
//TODO reset compteur
SI TcPilotes[].size == 1 && TcSecu == TcPilotes[0]
// il y a un seul Tc pilote qui est egalement le tc secu
        SI TcSecu >= SecuMax || TcSecu <= SecuMin //secu max
                CpterSecuMinMax++
                SI  5min > CpterSecuMinMax >= 3 min
                        SI TcPilote =! TcRedondant
                                ChangerTCPilote(TcRedondant)
```

```
                        FIN SI
                SINON SI CpterSecuMinMax >= 5 min
                        AlarmeSupervision(Rouge)
                        EtatCaisson = Arret
                        AlarmeClient()
                FIN SI
        SINON
        // configuré avec plusieurs tcs pilote
                SI TcSecu >= SecuMax || TcSecu <= SecuMin //secu min-max
                        SI 5min > CpterSecuMinMax >= 3 min
                                SI TcPilote =! TcRedondant
                                        ChangerTCPilote(TcRedondant)
                                        AlarmeClient()
                                        EtatCaisson = retour25
                        SINON SI CpterSecuMinMax >= 5 min
                                AlarmeSupervision(Rouge)
                                EtatCaisson = Arret
                        FIN SI
                FIN SI
        FIN SI
FIN SI




//contrôle de la temperature
SI Tspec >= 25
//chaud
        //TODO séparé secu tc secu et pilote
        SI Phase == Accroché
                SI TPilMes >= Tspec + OffsetAlarmOrange//alarme orange
                        AlarmeSupervision(Orange)
                SINON SI TPilMes >= Tspec + OffsetSecuPalier
                        CpterSecuPalier++
                        SI CpterSecuMinMax >= 3 min
```

```
                        //sécurité en palier
                        AlarmeSupervision(Rouge)
                        EtatCaisson = Arret
                FIN SI
        FIN SI //TODO reinitialiser les compteurs
    SINON SI TPilMes >= TSpec + OffsetSecuOvershoot //secu overshoot
        CpterSecuOvershoot++
        SI CpterSecuOvershoot >= 3 min
            AlarmeSupervision(Rouge)
            EtatCaisson = Arret
        FIN SI
    FIN SI
FIN SI
SI Tspec <= 25
//froid
    SI Phase == Accroché
        SI TPilMes <= Tspec - OffsetAlarmOrange //alarme orange
            AlarmeSupervision(Orange)
        SINON SI TPilMes <= Tspec - OffsetSecuPalier //securité en palier
            AlarmeSupervision(Rouge)
            EtatCaisson = Arret
        FIN SI
    SINON  TPilMes <= TSpec - OffsetSecuOvershoot //secu overshoot
        AlarmeSupervision(Rouge)
        EtatCaisson = Arret
    FIN SI
FIN SI
```

Initialisation variable

```
SI Tspec  > 25
    //MAJ de la spec en fonction de l'offset
    TCycleVisee = Tspec + OffsetCycle
SINON SI Tspec  < 25
    //MAJ de la spec en fonction de l'offset
```

```
        TCycleVisee = Tspec - OffsetCycle
SINON
        TcycleVisee = Tspec
FIN SI
//Initialisation de la valeur de CoefCor en fonction du mode
SI Mode == Conductif
        CoefCorr = 2
SINON SI Mode == Radiatif
        CoefCorr = 1
SINON SI Mode == Convectif
        CoefCorr = 2
FIN SI
SI PalierEncours.MemCons == VRAI
        TCycleVisee = ConsMem // du dernier palier similaire coché


FIN SI


SI SelectionAutoPilote == true
//si la selection auto du pilote est activée on selectionne le tc le plus chaud ou plus froid
        POUR i=0 ; i<ListeTcPilote[].size ; i++
                SI Tspec >= 25 //chaud
                        SI ListeTcPilote[i] > TcPilote
                                TcPilote = ListeTcPilote[i]
                        FIN SI
                SINON // froid
                        SI ListeTcPilote[i] < TcPilote
                                TcPilote = ListeTcPilote[i]
                        FIN SI
                FIN SI
        FIN POUR
FIN SI
```

Gestion objectifs :

```
//declarer palier en fonction des TC à verifier
```

```
Pilotage :

SI Tspec > 25 //Chaud
        SI TPilMes < Tspec – CutBack
                SI Cons < Tspec // La consigne a-t-elle atteint la spec ?
                        Phase = Transition
                SINON
                        Phase = Attente
                FIN SI
        SINON SI Tspec – CutBack < TPilMes < Tspec – ZoneAccrochage(Tspec)
                SI Cons < Tspec
                        Phase = Transition
                SINON
                        Phase = Approche
                        CpterAjustPlaque++
                FIN SI
        SINON SI Tspec – ZoneAccrochage(Tspec) < TPilMes < Tspec + ZoneAccrochage(Tspec)
                DemarrageCpterAccrochage()
                SI CpterStab == DureeStab ET PenteMes < PenteAccr
                        Phase = Accroché  // utilisé fonction pour accrochage ?
                FIN SI
        SINON SI Tspec +  0.5 < TPilMes // En dépassement
                Phase = Correction
                CpterCorrection++
        FIN SI
SINON //Froid
        SI TPilMes > Tspec + CutBack
                SI Cons > Tspec // La consigne a-t-elle atteint la spec ?
                        Phase = Transition
                SINON
                        Phase = Attente
                FIN SI
        SINON SI Tspec + CutBack > TPilMes > Tspec + ZoneAccrochage(Tspec)
                SI Cons > Tspec
                        Phase = Transition
```

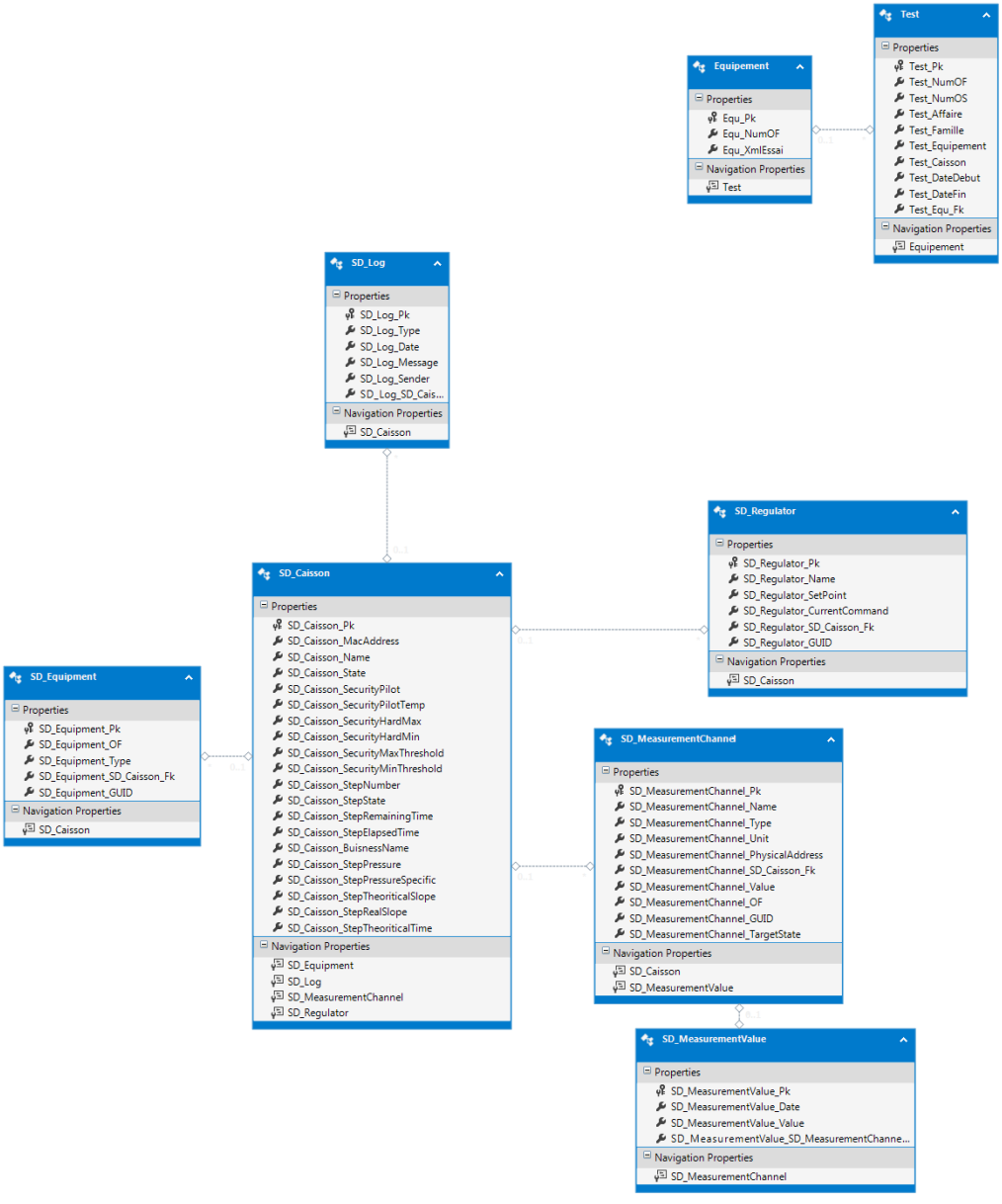
```
                SINON
                        Phase = Approche
                        CpterAjustPlaque++
                FIN SI
        SINON SI Tspec - ZoneAccrochage(Tspec)< TPilMes < Tspec + ZoneAccrochage(Tspec)
                DemarrageCpteAccrPlaque()
                SI CpterStab == DureeStab ET PenteMes < PenteAccr
                        Phase = Accroché
                FIN SI
        SINON SI Tspec – 0.5 > TPilMes // En dépassement
                Phase = Correction
        FIN SI
FIN SI
```

Envoi consigne

```
// calcul de la consigne à envoyer
SI Phase == Transition
        // On fait évoluer la consigne selon la pente spécifiée
        Cons = Cons + ( PenteMes * Sens)
SINON SI Phase == Attente
        // TODO ?
SINON SI Phase == Approche
        SI CpterAjustPlaque == DureeAjust
                SI MesurePente(TPilMes[t-1],TPilMes[t]) <= PenteAjust
                        Cons = Cons + ((TPlS – TPilMes)/ CoefCorr)
                FIN SI
        FIN SI
SINON SI Phase == Correction
        SI CpterCorrection >= DureeCorr
                Cons = Cons + ((TPlS – TPilMes)/ CoefCorr)
        FIN SI
FIN SI
SI  (Regulateur1 == Accroché ET Regulateur2  == Accroché ET) || check objectifs …
        SI CpterDuréePalier == 0
```

```
                DemarrageCpterPalier()
        SINON SI CpterDuréePalier >=  DuréePalier
                //fin de palier -> passage au palier suivant
                PalierEnCours = PalierEnCours+1
                InitialiserVariable(PalierEnCour)
        FIN SI
FIN SI
```

Annexe 8 : Modèle de la base de donnée VTH



Annexe 9 : IHM de pilotage du système existant

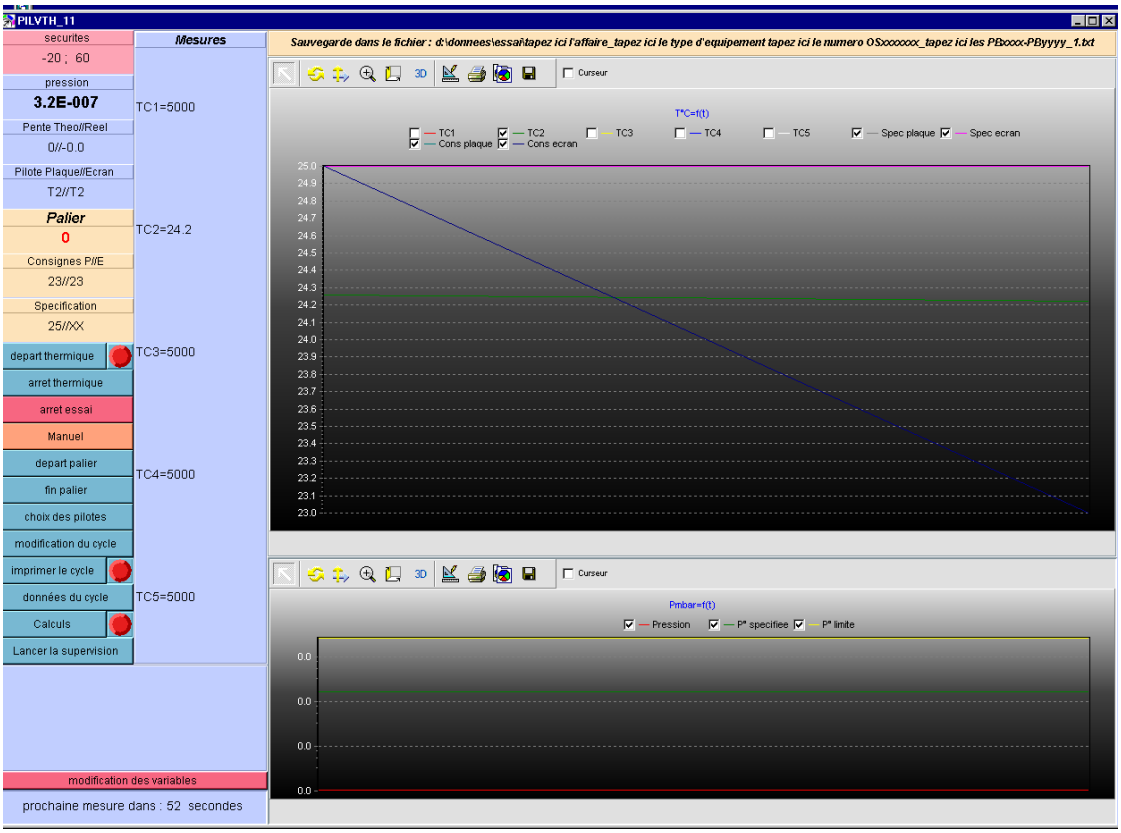


Table des figures

Table des tableaux

Amélioration d'un système d'essais thermiques automatisés

Mémoire d'Ingénieur C.N.A.M, Toulouse 2015

Résumé

Le laboratoire d'essais thermiques sous-vide réalise des essais d'équipements satellitaires de la procédure d'essai jusqu'à la génération de PV. Des outils logiciels, permettant d'automatiser certaines activités, ont été développées et présentent aujourd'hui des problèmes de maintenabilité et d'évolutivité.

Dans le cadre de l'amélioration de ces outils, une étude du système existant a été réalisée afin d'en dégager les fonctions, les exigences et les axes d'amélioration. La mise en place d'une architecture adaptée et basée sur les technologies C# .NET a permis de répondre aux contraintes d'évolutivité et de maintenabilité notamment au niveau de l'interchangeabilité des instruments mis en œuvre. La phase de réalisation du système a été conduite sur les principes méthode agile scrum.

Mots clés : essais thermiques, architecture, existant, évolutivité, interchangeabilité, agile

---

Summary

The laboratory tests of thermal vacuum test satellite equipments from procedure to minutes. Software tools to automate activities have bee, developed and now have maintainability and evolutivity issues.

As part of the tools improvement, a study of the existing system was realized in order to identify functions, requirement and areas for improvement. The use of an appropriate architecture, based on the C# .NET technologies has permitted to answer the demands of evolutivity in terms of instrument interchangeability. Implementation was conducted according to agile scrum methodology.

Key words: thermal tests, architecture, existing, evolutivity, interchangeability, agile