Équipe 5

Analyse de traces du démarrage de Chrome
Document d'architecture logicielle

Version 2.0

Historique des révisions

Date	Version	Description	Auteur
2016-02-09	1.0	Configuration du fichier et début de l'écriture du fichier.	Olivier Marchand Lemire
2016-02-10	1.1	Ajout des diagrammes.	Olivier Marchand Lemire
2016-02-11	1.2	Finalisation du document.	Olivier Marchand Lemire
2016-04-10	2.0	Mise-à-jour du document pour qu'il corresponde à l'état final du projet.	Olivier Marchand Lemire

Table des matières

Document d'architecture logicielle

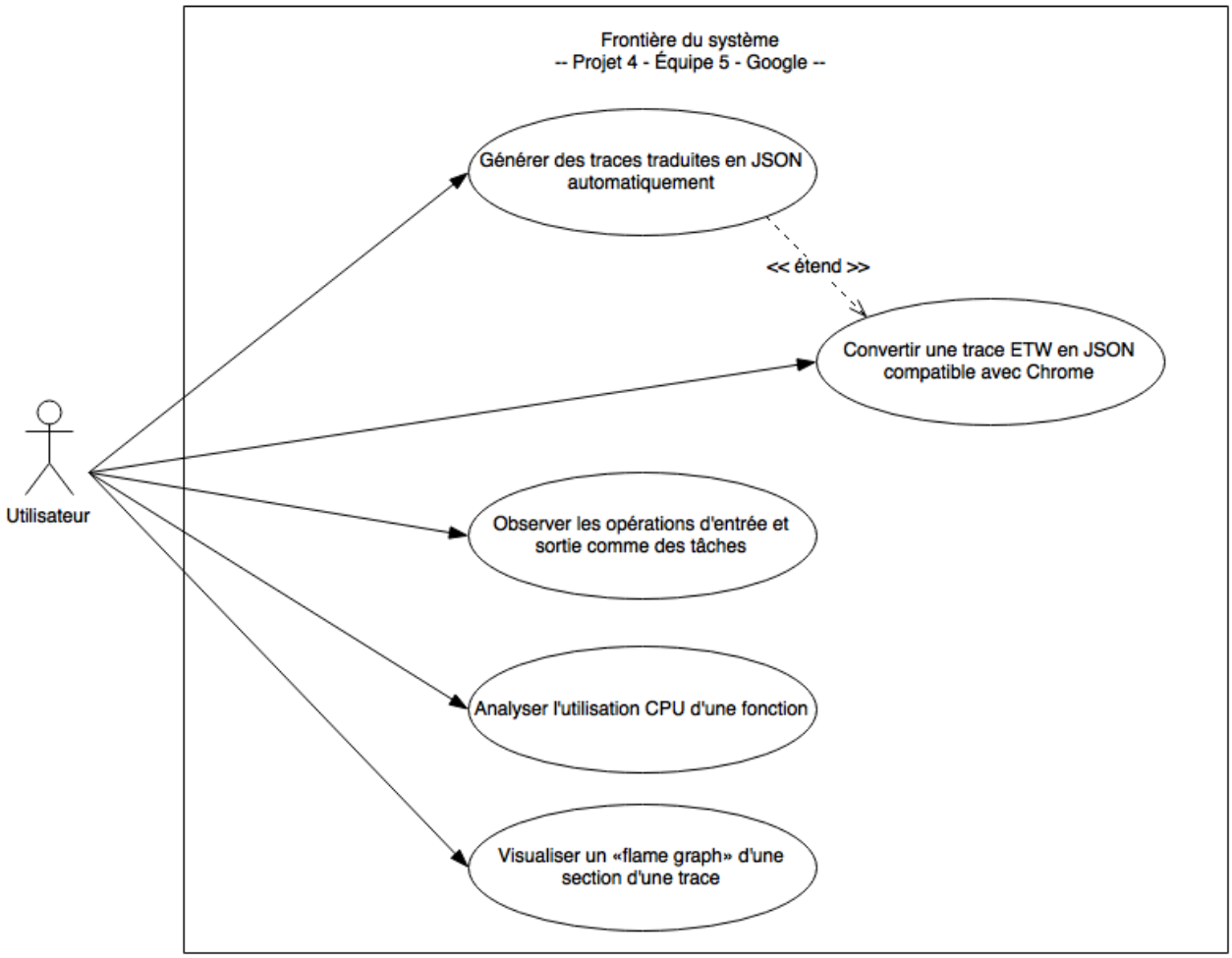
1. Introduction

Ce document décrit l'architecture qui sera utilisée pour le développement au cours de notre projet de 4e année. Pour ce faire, nous décrirons d'abord nos objectifs ainsi que les contraintes architecturales. Dans les sections qui suivront par la suite nous présenterons la vue des cas d'utilisation, la vue logique, la vue des processus, et la vue de déploiement. Nous terminerons ensuite avec des détails sur les caractéristiques de taille et performance pouvant avoir un impact sur notre architecture.

2. Objectifs et contraintes architecturaux

Comme il s’agit d’un projet qui sera utilisé réellement par notre client nous avons pour objectif de produire un logiciel dont la maintenance est facile à faire et qui soit aussi réutilisable que possible. Par contre, considérant les contraintes sur notre échéancier de 4 mois nous devrons nous restreindre sur la complexité de notre solution.

3. Vue des cas d'utilisation

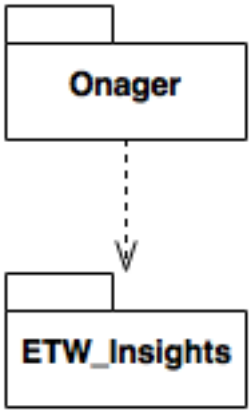


4. Vue logique

4.1 Conversion des traces

Nos classes du paquetage Onager utilisent des définitions, fonctions et classes présentes dans ETW Insights de Catapult (Google). ETW_Insights étant vaste et ayant peu d'influence sur notre architecture nous le conserverons sous sa forme fermée de paquetage.

4.1.1 Diagramme de paquetage

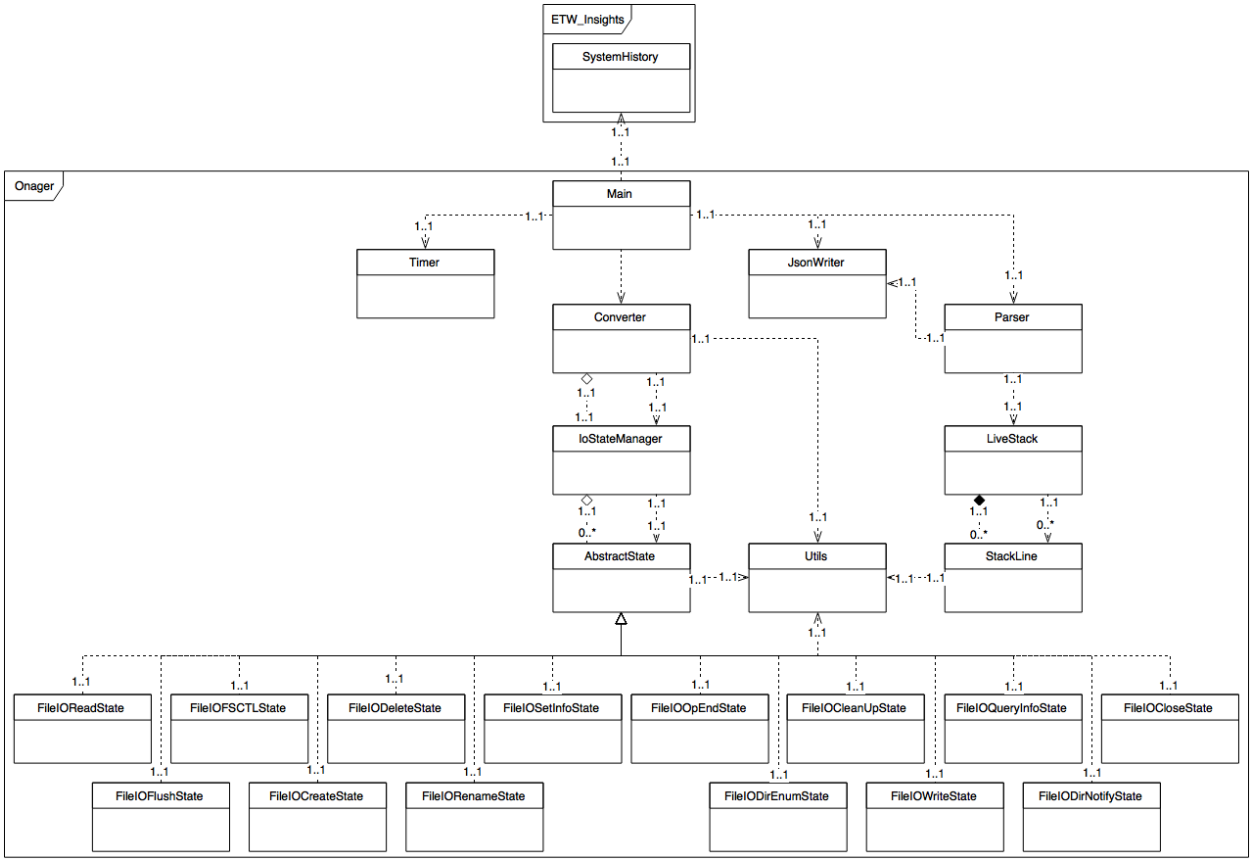


4.1.2 Description des paquetages

Onager	
Description:	Contient les éléments permettant de faire la conversion des traces ETW vers le format JSON approprié pour être compatible avec « Chrome tracing ».
Classes incluses:	Main, Timer, JsonWriter, Parser, LiveStack, StackLine, Utils, Converter, IoStateManager, AbstractState, FileIOReadState, FileIOFSCTLState, FileIOFlushState, FileIOCreateState, FileIODeleteState, FileIORenameState, FileIOSetInfoState, FileIOOpEndState, FileIODirEnumState, FileIOCleanUpState, FileIOWriteState, FileIOQueryInfoState, FileIODirNotifyState, FileIOCloseState
Relations:	ETW_Insights
Sous-paquetages:	--

ETW_Insights	
Description:	Contient des éléments qui facilitent la lecture des traces ETW qui sont sous le format CSV.
Classes incluses:	--
Relations:	Onager
Sous-paquetages:	--

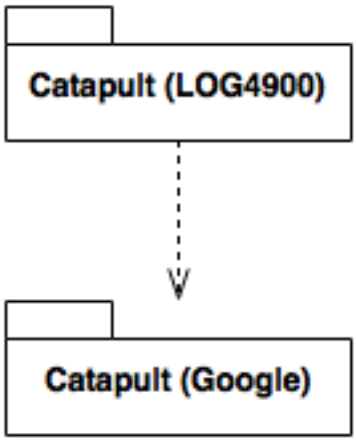
4.1.3 Diagramme de classes



4.2 Amélioration de « Chrome tracing »

Notre paquetage Catapult (LOG4900) ne contient pas de classe, mais nous présenterons le modèle Polymer que nous avons utilisé puisqu'il est conceptuellement proche d'une classe. Évidemment, comme Catapult (Google) est vaste, nous présenterons seulement les quelques modèles Polymer que nous avons modifiés pour nos besoins.

4.2.1 Diagramme de paquetages

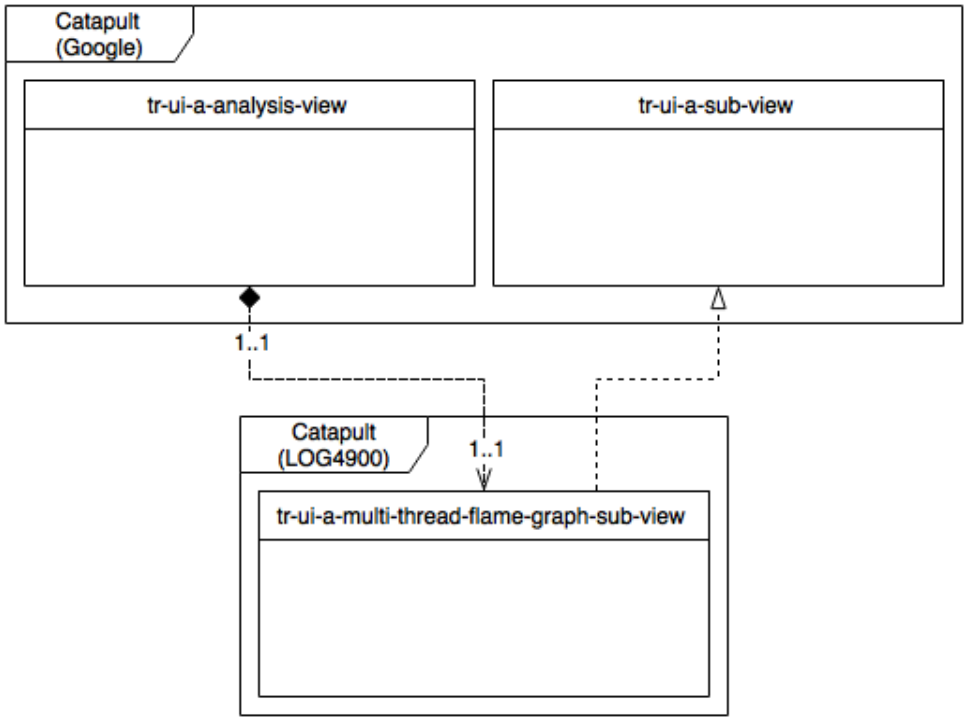


4.2.2 Description des paquetages

Catapult (LOG4900)	
Description:	Contient le code nécessaire à l'affichage des temps CPU, des IOs en tant que tâche et d'un « flame graph » dans « Chrome tracing ».
Classes incluses:	tr-ui-a-multi-thread-flame-graph-sub-view
Relations:	Catapult (Google)
Sous-paquetages:	--

Catapult (Google)	
Description:	Contient le code actuel de « Chrome tracing » avec quelques modifications pour intégrer les éléments de Catapult (LOG4900).
Classes incluses:	tr-ui-a-analysis-view, tr-ui-a-sub-view
Relations:	Catapult (LOG4900)
Sous-paquetages:	--

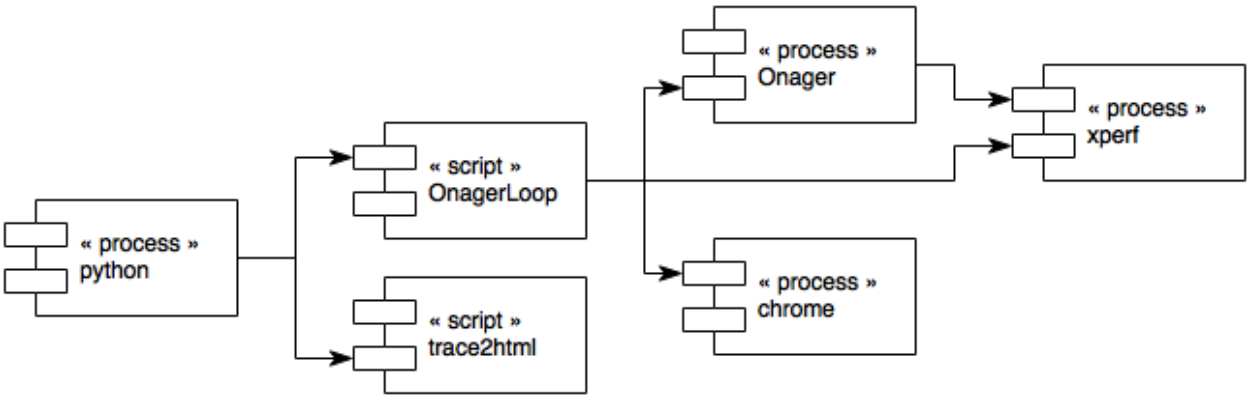
4.2.3 Diagramme de classes



5. Vue des processus

5.1 Conversion des traces

La structure de notre application est plutôt simple et elle n'utilise pas plusieurs fils d'exécutions. Nous avons donc un seul processus et aucun fil d'exécution autre que le principal.

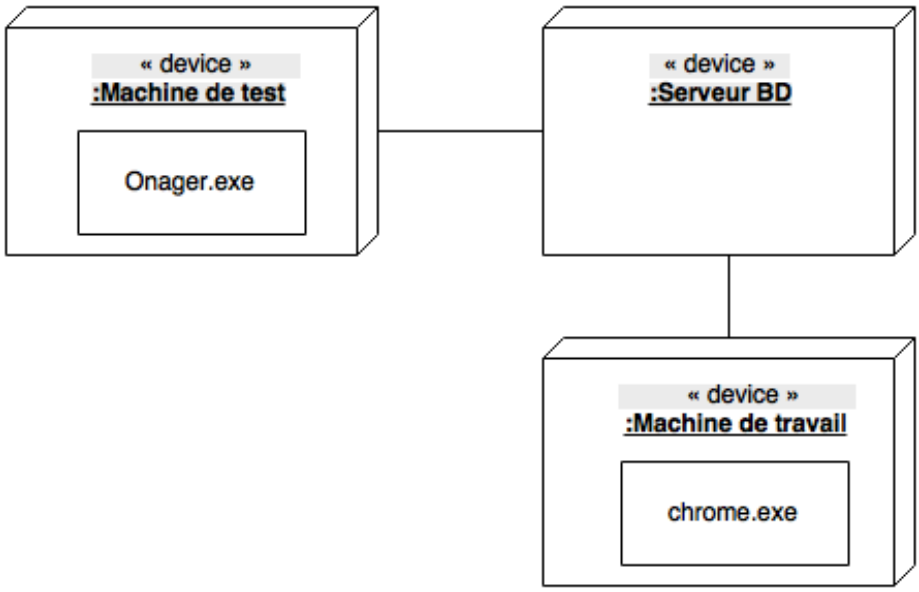


5.2 Amélioration de « Chrome tracing »

Cette partie du projet consiste à modifier du code JavaScript qui est utilisé par une page web, il n'y a donc pas de processus appartenant au projet.

6. Vue de déploiement

Notre convertisseur de trace ETW sera utilisé sur des machines de test qui téléchargeront les fichiers de format JSON sur une base de données afin que des utilisateurs puissent les visualiser sur la vue « Chrome tracing ».



7. Taille et performance

Le client a requis que la partie de conversion des fichiers .CSV à .Json soient traités à une vitesse d'au moins 150 MB/s. Il faudra donc que l'architecture soit suffisamment légère et efficace pour obtenir ce taux.