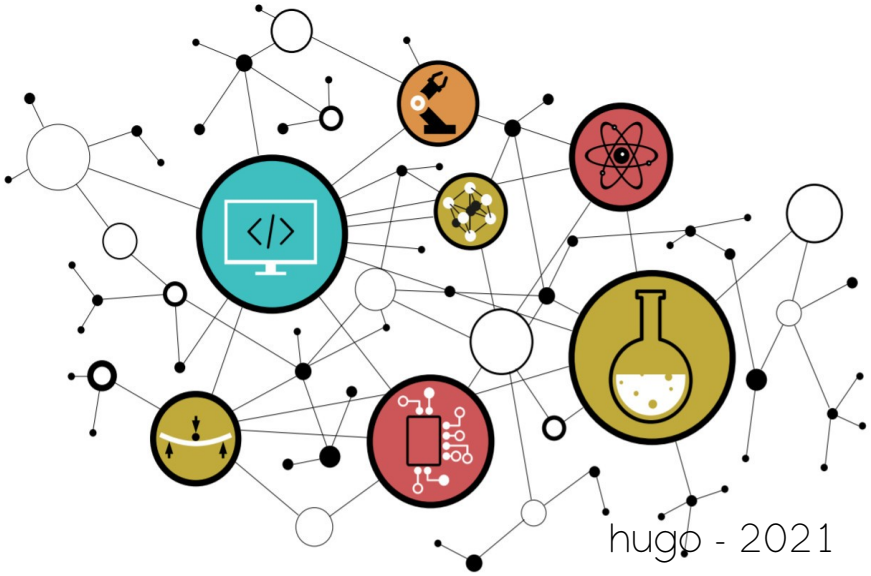
TRAVAUX PRATIQUES

DOCUMENTATION TECHNIQUE ET LIVRABLES



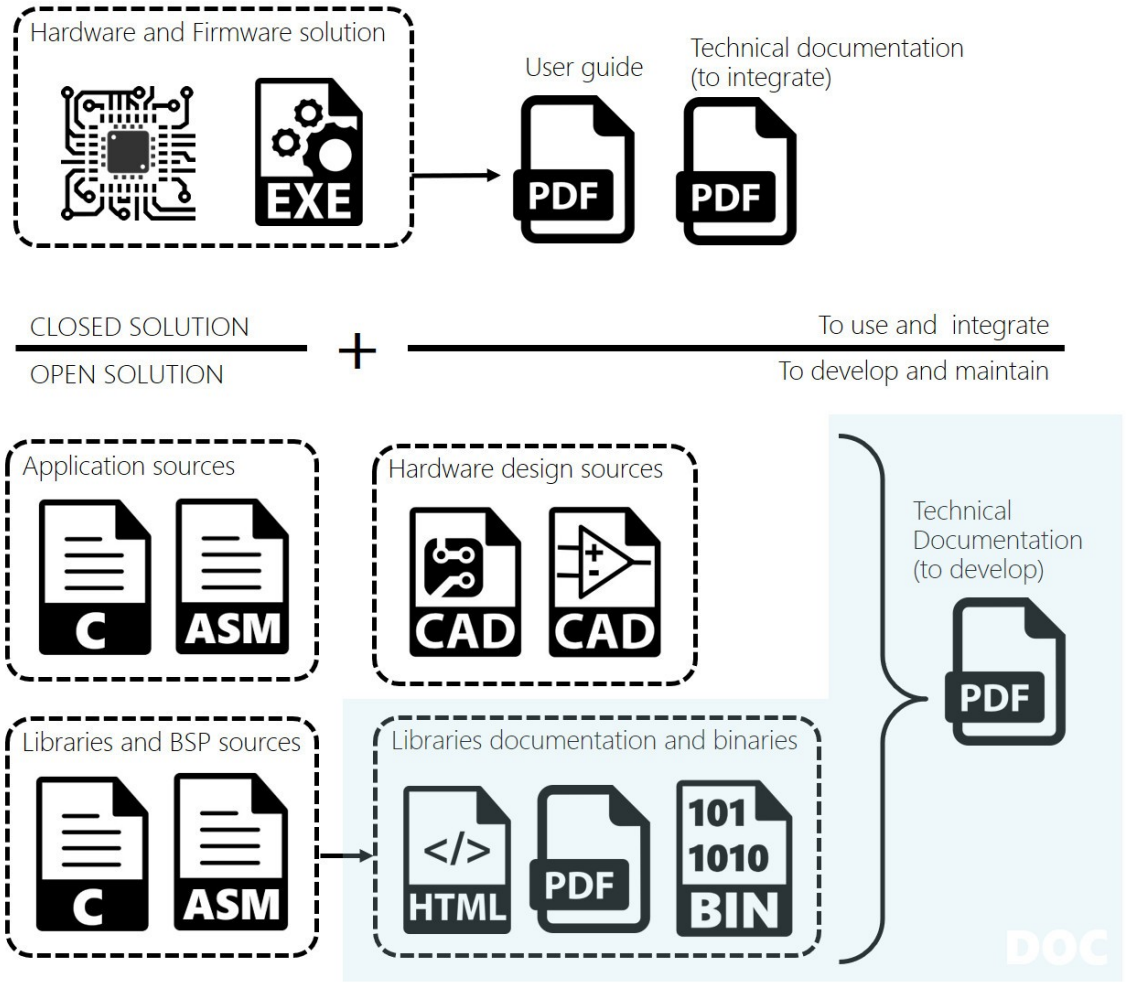
hugo - 2021

SOMMAIRE

DOCUMENTATION TECHNIQUE ET LIVRABLES

9. DOCUMENTATION TECHNIQUE ET LIVRABLES

Le schéma ci-dessous présente l'ensemble des livrables constituant une solution (système embarqué complet). En fonction du contrat signé entre client (MOA ou Maîtrise d'ouvrage) et prestataire (MOE ou Maîtrise d'œuvre), certaines parties de la solution resteront fermées.



La documentation peut être un point essentiel dans le processus de développement, d'intégration, de validation et de maintenance d'une solution. Parlons d'histoires réelles et vécues par bien des ingénieurs. Un ingénieur développe durant des semaines voire des mois une solution pour un client (sous-traitance, équipe interne, société de services, etc). Celui-ci change de mission voire d'entreprise, ou le contrat n'inclue pas forcément la maintenance. Sa solution est fonctionnelle et a été intégrée à une version donnée du produit. De même, elle était maladroitement architecturée et développée (ingénieur junior, stagiaire, incompétences techniques, etc) et maladroitement documentée. Un jour, un autre ingénieur doit se plonger dans sa solution (déverminage, correctif, évolution, intégration vers une nouvelle version du produit, portage, etc). La perte de temps peut être considérable, des jours, des semaines voire bien plus, sachant que le problème peut se reproduire. Cette histoire est très courante en industrie, de nombreux retours de ce type nous sont faits chaque année.

Cependant, pour information, dans le domaine du logiciel, le concept de ''code propre'' émerge. Il met la solution logicielle en avant et au centre de tout, et part du principe qu'une solution bien architecturée, conçue et bien codée (modularité, clarté, simplicité, séparation, etc) se suffit à elle-même. Elle ne nécessitera alors que très peu de documentation. Selon votre majeure et expertise, une formation dédiée en génie logiciel, développement agile et en code propre sera réalisée.

9.2. Doxygen et documentation d'une bibliothèque

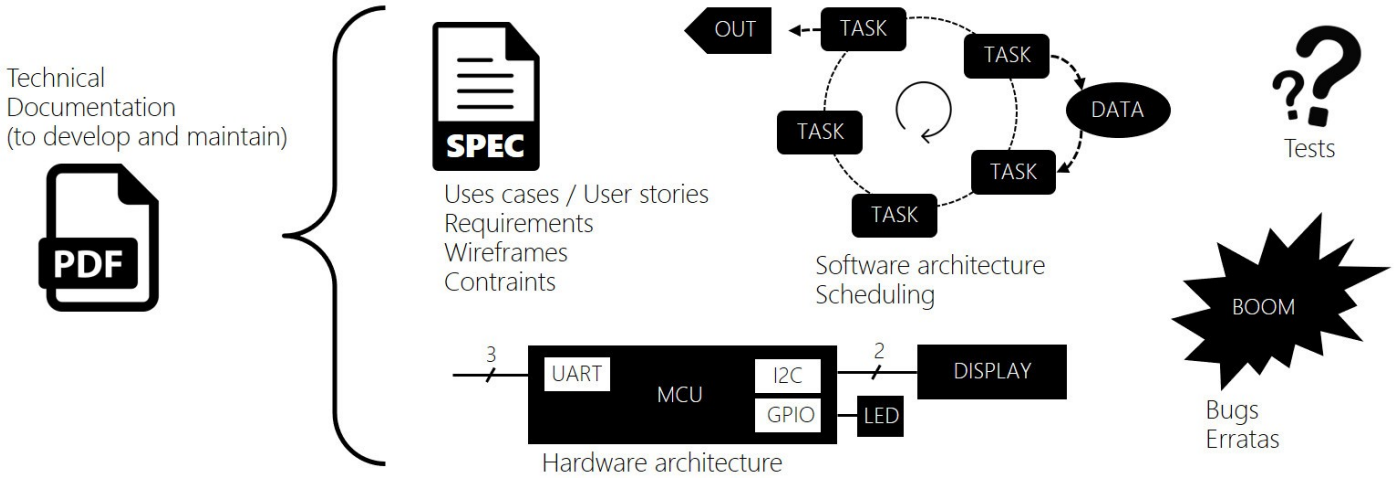
http://www.doxygen.nl

Doxygen est un générateur de documentation sous licence libre dédié à la production de documentation logicielle à partir du code source d'un programme. Il tient compte de la syntaxe du langage ainsi que des commentaires devant respecter un formalisme spécifique. Nous parlons de balises *Doxygen* (balises préfixées par @ présentes dans les fichiers d'en-têtes). Vous avez déjà rencontré ces balises dans l'ensemble des fichiers d'en-tête du BSP. Sur ce point, la majorité du travail a déjà été réalisée de notre côté. L'architecture modulaire de la documentation *Doxygen* a été conçue et développée. Le fichier racine de la documentation est le suivant *disco/bsp/doc/doxygen/doxygen.h* . Vous allez devoir maintenant générer une documentation HTML compilé (.chm) explicitement estampillée à votre nom. Ce travail passe d'abord par une phase d'installation d'outils, de création d'un projet *Doxygen* (pour génération d'un fichier *Doxyfile*) puis de génération de la documentation au format demandé.

- Installer Doxygen/Doxywizard
- Installer Dot : https://graphviz.gitlab.io/download/
  - Sous Doxywizard > Expert > Dot > DOT_PATH > C:\<*dot_path*>*bin*
- Installer HCC : https://www.microsoft.com/en-us/download/details.aspx?id=21138
  - Sous Doxywizard > Expert > HTML > HHC_LOCATION > C:/<*hhc_path*>*/hhc.exe*
  - Sous Doxywizard > Expert > HTML > CHM_FILE > *bsplib_tiny.chm*
  - Sous Doxywizard > Expert > HTML > *disable SEARCHENGINE*

- Générer une documentation *Doxygen* du BSP au format HTML compilé (.chm) nommée *bsplib_<student_name>.chm*. Le nom du projet *Doxygen* devra être ''*bsplib - <student_name> - v1.0*'' et sera sauvé, tout comme le *Doxyfile*, dans le répertoire *disco/bsp/doc/doxygen/*. Une fois générée, copier la documentation présente dans le dossier *disco/bsp/doc/doxygen/html/* dans le répertoire *disco/bsp/doc/. S'aider d'internet pour ce travail et notamment du site officiel de Doxygen*

9.3. Documentation technique



La documentation technique est directement destinée aux équipes techniques de développement et de maintenance du produit. L'objectif premier étant de permettre à un ingénieur développeur, intégrateur ou architecte d'appréhender le plus rapidement possible les architectures et solutions logicielles comme matérielles anciennement développées, ainsi que le positionnement du sous système dans un projet global de plus grande envergure.

Éditer une documentation technique présentant l'ensemble de vos développements (BSP et application Bluewave). Pour ce travail, la construction du plan et donc la définition de l'architecture du document est la première chose à réaliser. Ce travail demande un esprit de synthèse, une vision d'ensemble du projet et une approche progressive. Toujours se mettre à la place du lecteur (un ingénieur) qui découvre votre projet et qui doit comprendre rapidement où travailler et opérer. S'aider de représentations graphiques et de schémas commentés afin de présenter vos travaux de développement. Être concis et précis, aller à l'essentiel en s'efforçant d'être le plus rigoureux possible quant au vocabulaire et descriptions techniques utilisés. Voici les points à présenter :

- Présenter le *projet dans son ensemble* : Expression du besoin, périmètre d'action du produit, guide d'utilisation et cas d'usages (use cases and user stories), cahier des charges, spécifications techniques et contraintes (requirements), etc
- Présenter les *outils de développement et technologies* déployées sans oublier les *versions* de chaque outil (IDE, toolchains, sonde de programmation, processeurs, cartes, etc)
- Présenter à l'aide d'un schéma fonctionnel commenté l'*architecture matérielle du produit* (interfaces externes utilisateur, bus et signaux d'interconnexion avec le processeur, périphériques internes au processeur, brochages et nommage des signaux, etc). Mettre en documents annexes les schémas électriques et layout de routage (version PDF et sources CAO).
- Présenter à l'aide d'un schéma fonctionnel commenté l'*architecture logicielle de l'application* (tâches applicatives, ressources partagées, description du travail de chaque tâche, stratégie d'ordonnancement, etc)
- Présenter les *procédures de test* de la solution (boîte blanche et/ou boîte noire) ainsi que les *résultats des mesures physiques* liés au comportement de l'application (temps d'exécution des tâches applicatives, charge CPU, réactivités des interfaces, consommation énergétique en veille et en fonctionnement, etc)
- Présenter *explicitement et clairement les bugs et erratas connus* voire des méthodes de résolution ou de contournement
- Proposer des *améliorations et évolutions techniques voire architecturales* du produit (sans pour autant avoir essayé de les développer)


notes