CST –Analyse multi-facettes et opérationnelle pour la transformation des systemes d'information

HOUEKPETODJI Mahugnon Honoré

1 Description du sujet de la thèse

CIM est une SAS au capital social de 200k détenu a 100 par DL Software. CIM est éditeur, intégrateur, hébergeur et infogéreur de solutions pour l'assurance de personnes en santé, prévoyance. Elle offre une expertise Santé et Prévoyance acquise après plus de 30 ans auprès de ses clients. CIM est hébergeur de ses solutions pour 90 de ses clients et plus de 1000 utilisateurs. Toutes les thématiques d'infrastructure et de surveillance des flux sont intégrées à cette offre. CIM est propriétaire de ses infrastructures serveurs, tous les éléments actifs des systèmes et tous les éléments de stockage sont achetés par CIM, gérés et supervisés par les équipes de CIM. Aucun sous-traitant n'intervient dans les opérations quotidiennes d'hébergement, d'exploitation des solutions et des données hébergées.

CIM est certifiée Microsoft GOLD Partner. Elle est l'éditeur des progiciels de la gamme Izy Links et assure l'intégration de l'ensemble des briques de cette gamme ainsi que des briques partenaires nécessaires à la bonne réussite du projet. Cette solution est développée en PowerBuilder sur base de données DB2. L'équipe de développement vient d'upgrader en PowerBuilder 2017 (été 2018) et est en cours de passage sur DB2 v11 (avec l'aide d'un DBA IBM – prestation 2018).

Le système de gestion est centré sur le back office Izy Protect, autour duquel gravite l'ensemble des briques complémentaires répondant à l'assemble des besoins, et pouvant être activées ou non. La société CIM a effectué une analyse de risque pour son évolution et croissance en 2017 d'où il ressort que Izy Protect souffre des problèmes (1) vieux langage, (2) logiciel vieillissant, (3) perte savoir, (4) changements à haut risque. Ces problèmes sont récurrents chez les organises gérant des systèmes d'information [**?**].

Ce travail de doctorat consiste de proposer des modèles et des mécanismes permettant d'assurer une ré-ingénierie des systèmes d'information. Les expériences et validation des prototypes se feront dans le contexte de l'application du système d'information écrit en PowerBuilder de la société CIM

2 Etat de l'art

Cette section présent Izy Protect, ainsi que les mécanismes de ré-ingénierie des systèmes d'information patrimoniaux.

2.1 Présentation de Izy Protect

Izy Protect est un système de plus de 3 MLOC écris en Powerbuilder et maintenu depuis plus de 20 ans par les développeurs la CIM. Le code source est organisé par bibliothèques Powerbuilder. Vu la complexité actuelle du système, les développeurs ont de plus en plus du mal a le maintenir. Les ancienne versions du système sont stocker sur un disque dure. Pour des raison interne à la CIM, les versions de Izy protect ont été perdu jusqu'en 2010. De plus les développeurs risquent a tout moment d'écraser leurs travaille. Les développeurs modifient Izy protect en réponse a qui décrit le travaille a faire.

Dans l'entreprise, les tickets sont stockés dans la base de données des tickets depuis 2000. Un ticket représente un travail unitaire. La base de données des tickets pilote l'ensemble du processus d'évolution du logiciel : attribution du travail aux développeurs, gestion du flux de travail pour répondre à une demande du client, informations de facturation sur chaque tâche. Il existe des tickets pour la correction de défauts, la rédaction de documentation, l'ajout de nouvelles fonctionnalités, etc.

Un ticket comporte entre autres les caractéristiques suivantes :

— la date de création
— la date de clôture
— l'estimation du temps nécessaire au développeur pour travailler sur le ticket
— temps passé par un développeur

    - le temps d'analyser
    - le temps de mettre en œuvre une solution
    - le temps de test
- le(s) bibliothèque(s) impactées

2.2 Analyse de l'évolution de l'état d'un logiciel patrimonial

Les systèmes patrimoniaux sont des systèmes en constante changements : production de nouvelles fonctionnalités. La deuxième loi de Lehman [**?**] stipule qu'à mesure que les logiciels évoluent, la complexité croissante et l'augmentation des défauts entraîneront une baisse de la satisfaction des parties prenantes, à moins que les équipes de projet n'entreprennent le travail nécessaire pour maintenir la qualité. Dans ce sens nombreuses travaux de la littérature propose des techniques, pour suivre l'évolution de l'état de ces systèmes.

[**?**] utilise les données des occurrences bugs et le temps pour modéliser l'évolution d'un système logiciel avec c-charts.

[**?**] propose un système de recommandation d'action au développeur pour un nouveau bug. Le system se base sur l'historique des bugs, le code source ainsi que qu'un algorithme de prédiction. Pour que ça marche, le code source doit être gérer dans un système de contrôle de version. Ce qui n'est pas le cas avec Izy Protect.

[**?**] utilise les modèles et l'analyse de l'historique des défauts pour évaluer la qualité d'un système et prédire l'effort nécessaire pour améliorer le système. Les métriques mesurés sont : le taux de bugs sur une période de temps, le ratio entre le taux de d'augmentation de la taille du système et le taux de bugs, le temps moyen entre la découverte des bugs, l'effort pour résoudre les bugs, estimation des risque de future bugs, etc. Certain de ces métriques comme : le taux de bugs par période de temps, l'effort pour résoudre les bugs sont intéressantes dans le cadre de Izy Protect. Les autres ne se conforme pas au contexte car les tickets de Izy protect ne sont directement liés au code de façon standard. Chaque développeur note le numéro de tickets a sa façon dans le code.

[**? ?**] propose un modèle de prédiction des défauts avec des algorithmes d'apprentissage en utilisant l'historique des bugs de système. Les algorithmes de prédiction de bugs ne sont pas toujours consistant [**?**]. De plus, ils ne tiennent pas compte des changements qui peuvent être imprévisible dans le code. De plus Izy Protect est système commercial multi-utilisateurs, et donc chaque utilisateur a des fonctionnalités ou des changements qui lui est spécifique.

[**?**] Utilise l'historique le nombre de lignes de code changer pour un bug fix ou une nouvelle fonctionnalité pour prédire la densité de bug le code. Pour que ceci soit réalisable, il faut que le code soit préalablement versionné dans un system de contrôle de version.

[**?**] à étudier différents algorithme de modélisation des défauts d'un système a partir des donnée des défauts relevé sur huit projet open source. Il en ressort que la moyenne glissée modélise mieux les défauts des systèmes.

2.3 Ré-ingénierie

3 Avancées actuelles

3.1 Analyse des fiches navettes

3.2 Route vers le DevOps

3.3 Outil d'aide a la ré-ingénierie logiciel

3.3.1 Outil de visualisation des appelle entre les objets

3.3.2 Outil de détection de code mort

3.3.3 Outil de visualisation de code dupliqué

3.3.4 Outil de présentation de code source

4 Roadmap

5 Publications

Cette section présente la liste des soumissions liée à cette thèse.

5.1 Papiers soumis

1.
2.