Gestion de la dette d'architecture dans le contexte d'hypercroissance

Date : 13 avril 2023
Speakers : Cyril Beslay (ManoMano)
Format : Université (45mn)

Cyril est Architecte d'Entreprise. Fondé en 2013, ManoMano est le leader européen du marketplace de bricolage et de jardinage. Cyril y est arrivé en 2019. En 4 ans, l'IT est passé de 200 à 500 personnes. Cette hypercroissance a nécessairement créé de la dette technique, dette qui leur aura permis de rester le leader européen.

Dette technique globale

Concept créé en 1992 par **Ward Cunningham** et adopté 10 ans plus tard.
"Shipping first time code is like going into debt. A little debt speeds development so long as it is paid back promptly with a rewrite... The danger occurs when the debt is not repaid. Every minute spent on not-quite-right code counts as interest on that debt. Entire engineering organizations can be brought to a stand-still under the debt load of an unconsolidated implementation, object-oriented or otherwise."

Cyril donne un exemple simple de dette : coder en dur les jours fériés dans l'agenda. Nécessite du travail supplémentaire chaque année + redéploiement.

La **dette financière** est connue de toutes et tous. Pour accélérer un projet maintenant (maison, voiture), contraction d'un crédit et donc d'une dette. Un emprunt pris à la banque doit être remboursé. C'est implicite.

Chez ManoMano, se pose la question : « les développeurs, ingénieurs, products managers sont-ils de bons payeurs de dette ? »

Dette d'Architecture

Il existe de multiples types de dette : technique (code, infra, tests), fonctionnel (bugs), sécurité, légale, architecture …
Pas de définition stricte. Voici celle de ManoMano : « **Dans un composant ou un process fonctionnel, écart entre le système actuel et les principes d'architecture / standards de l'entreprise** ».
Mesurer la différence entre le système actuel et la cible.

Exemple 1 : Pattern Circuit Breaker : tout appel d'une API vers un service externe doit implémenter le pattern Circuit Breaker. Lorsque l'équipe A développe un micro-service faisant appel à l'API de La Poste sans Circuit Breaker, elle créée de la dette car son MS est en écart avec le standard.

Exemple 2 : le POC de Machine Learning arrivé en prod pour vérifier l'attrait du client est resté. Il n'a jamais été industrialisé.

Plusieurs typologies de dette d'architecture :

- Choix facile d'une solution aujourd'hui par rapport à une meilleure approche
- Mauvais choix de framework ou d'outil
- Produit qui stagne dans un contexte évolutif
- Projet de migration non terminé
- Création / changement de standards : lorsqu'on crée un standard ou on le fait évoluer, on crée un écart et on crée donc de la dette

SonarQube est un outil très populaire dans le monde du dév. Il permet de mesurer les écarts du code par rapport au profil qualité fixé. Une correction incrémentale est possible.
A contrario, il est impossible aujourd'hui de lancer une analyse statique pour mesurer la dette d'architecture. Les quelques outils connus ne sont pas matures
Alors par où commencer ?

En premier lieu, déterminer les causes principales externes à l'équipe :

1. Business :
    - Pression et délais raccourcis
    - Non compréhension du concept de dette
    - Domaine métier volatile
    - Peu de vision long terme
2. Standards d'architectures non ou mal définis : souvent dans des petites boites
3. Temps qui passe

En second lieu, trouver les causes internes à l'équipe :

1. Manque de connaissance fonctionnel ou technique
2. Mauvaises décisions d'architecture : c'est souvent l'enchainement de décisions qui en est à l'origine
3. Manque d'anticipation, système non évolutif
4. Facteurs humaines divers

Les équipes ne sont pas toujours à blâmer. La dette peut être involontaire.

<u>Symptômes</u> :

- « Il n'y a que John qui sait gérer çà »
- 2 Microservices sont toujours livrés en même temps
- Temps pour analyser / résoudre un bug
- Complexité excessive (3 systèmes en parallèle)

<u>Conséquences</u> :

- Augmentation du coût des fonctionnalités
- Augmentation du nombre de bugs
- Augmentation des risques : on ne peut plus recruter que des seniors sur ce projet
- Diminution globale de la qualité des produits

Management de la dette d'architecture

Comment peut-on gérer la dette d'architecture ?

Cyril revient sur l'analogie de la dette financière : la banque rappelle qu'il faut payer la dette. En architecture, on le paie sans réellement le voir. On peut arriver à la banqueroute, à savoir tout jeter pour tout réécrire.

2 approches de managements : contrôler la création de la dette et réduire la dette existante

Contrôle de la qualité de la dette

Parfois on doit prendre des raccourcis et s'endetter pour saisir des opportunités.

Bonnes pratiques :

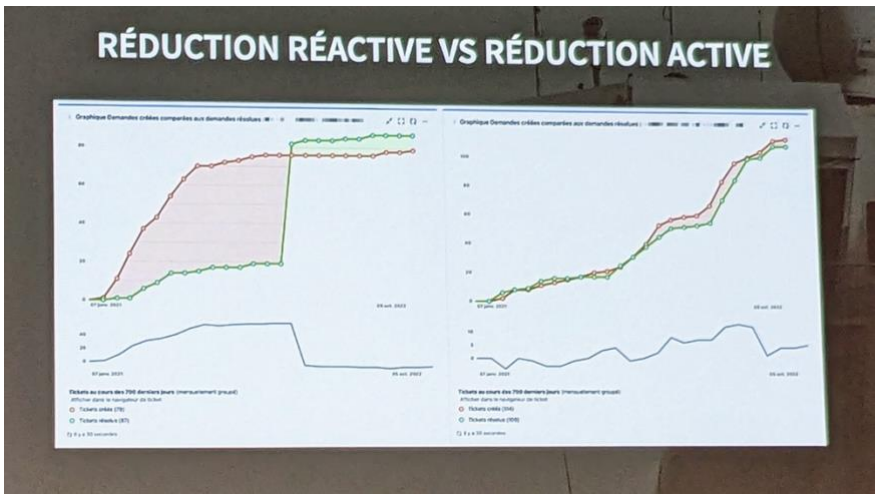
- Penser l'architecture / faire de la **conception**. Nous sommes tous architectes. Avant d'écrire une ligne de code, faire de la conception.
- **Anticiper** les besoins fonctionnels
- Noter les décisions et leurs impacts via des **Architecture Decision Record** (ADR)
- Créer des **tickets de dette dans Jira**

Comment réduire la dette existante ?

Plusieurs stratégies :

1. Stratégie active de management
    - Boy scout rule
    - 20% de temps par Sprint
    - Investir plus de temps dans les plans de refactoring / architecture
2. Stratégie réactive de management
    - Correction opportuniste (nouveau gros projet)
    - Gel des nouvelles fonctionnalités
    - Réécrie en entier un composant / process
3. Stratégie passive de management
    - Délaisser (rapport coût/risque trop élevé)

Il est beaucoup plus simple de tout jeter et tout refaire que d'améliorer continuellement : les gens n'apprennent pas.



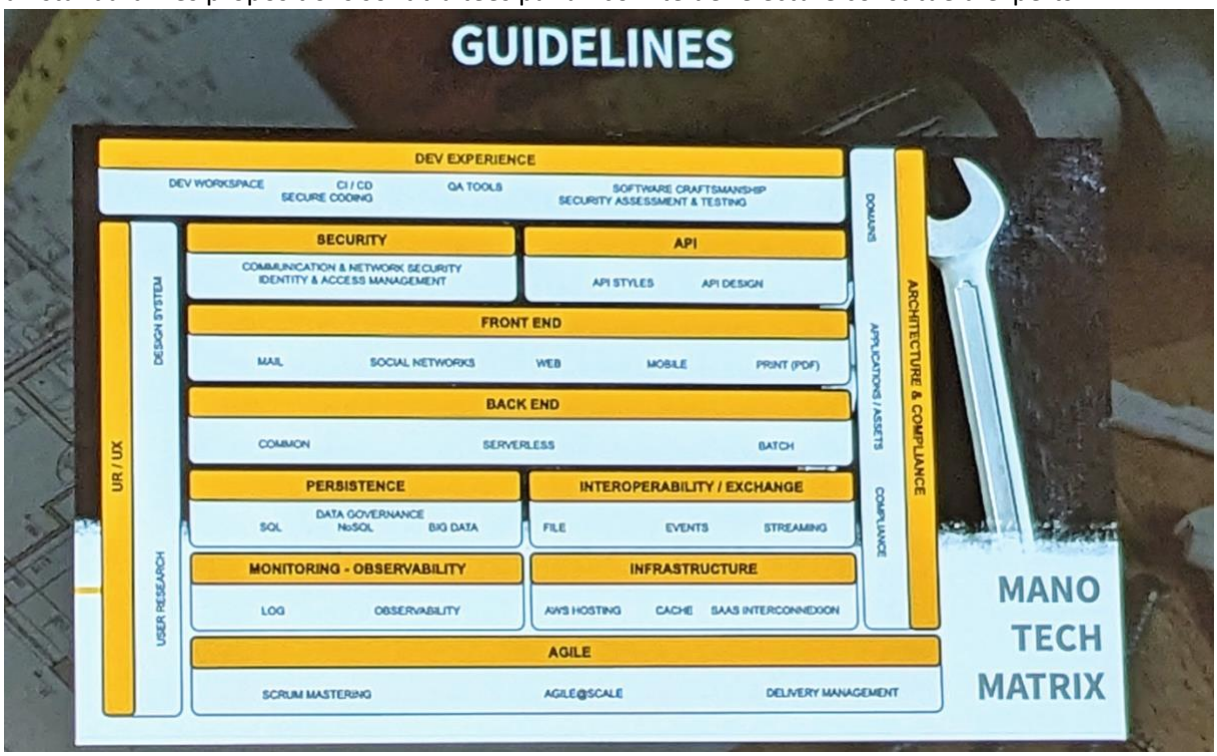
Construction d'un framework pour identifier et mesurer cette dette ?

Le SI de ManoMano est passé de 2 monoliths à des dizaines de MS.
Pour savoir comment, Cyril nous invite à lire l'article ManoMano IT Odyssey sur Medium.

Exemple de dettes :

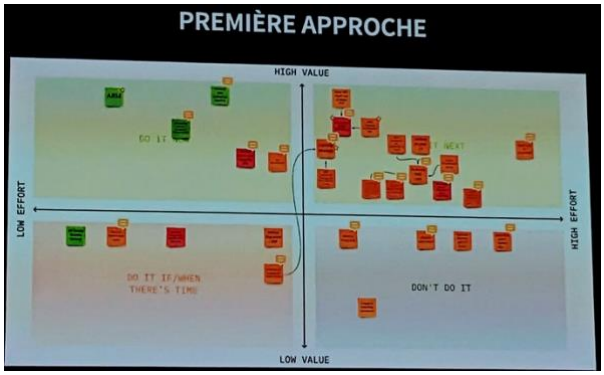
En 2016, Banqueroute de ManoMano. Gel fonctionnel. 10 dévs ont migrés le code de PHP vers Symfony.
Facturation à la main lancée sur Google Sheet avec 300 marchands.

Leur framework a démarré en 2022 et toujours en construction.
La définition des standards chez ManoMano se fait en mode collaboratif : tout le monde peut initier un standard. Les propositions sont traitées par un comité de relecture constitué d'experts.



Création d'un outil nommé Radiologist prenant en compte le code, les logs, l'infra de production …
Autre outil utilisé : Tech Radar, l'ensemble des outils adoptés, documentés …

Réduction de la dette
Première approche par post-it limitée à l'équipe



Seconde approche : récolte de données par méthode qualitative. Entretiens en 2 parties avec les équipes techniques (manager et Tech Lead)

Exemples de questions posées lors de l'entretien :

1. Si vous aviez carte blanche, quelle partie de votre périmètre vous souhaiteriez refaire ?
2. Quel est votre retard quant à la montée de version de vos dépendances OSS ?

Pour chaque architecture, building blocks :

1. Risque : quel est le risque que cela crée des problèmes dans le futur ?
2. Temps : quand cela risque-t-il d'arriver ?
3. Impact : quel impact sur l'utilisateur final ?
4. Efficacité : est-ce que cela nuit à l'efficacité de l'équipe ?
5. Déviation : quel est l'écart avec les standards ?

L'idée est d'avoir une vue d'entreprise.

Challenger
Mettre en corrélation les résultats des enquêtes avec :

1. Fréquence des incidents par domaine
2. Vélocité des équipes depuis Jira
3. …

Prioriser

1. Présentation des cartes et des alertes aux CTO et directeurs techniques
2. Sensibilisation du business sur les notions de dette technique

Concept « La dette est trop importante dans cette partie du système, il faut la nettoyer avant de développer de nouvelles fonctionnalités ».

Quelques retours d'XP : reconnaitre que la dette n'est pas toujours mauvaises.

Takeways

1. Définir ses standards/principes d'architecture
2. Démystifier la dette dans toute l'entreprise
3. Identifier et caractériser régulièrement
4. Adopter différentes stratégies de management
5. Construire ses propres outils