Loi de Conway : lorsque les bonnes pratiques ne suffisent plus

Date : 13 avril 2023
Speakers : Julien Topçu
Format : Conférence (45mn)

IT Zone

Julien nous embarque dans une dystopie gallo-romaine sans ressemblance avec des faits existants. Entrons dans l'IT Zone. A cette époque, la Gaulerie souhaite « réfléchir aux grandes tendances de l'évolutions du secteur de l'enseignement ».
Mercure fait tout naturellement appel au cabinet de conseil McKinsey.

En live, Julien parcourt le **site web de McKinsey** : présentation des personnalités phares de l'entreprise avec un bouton de type « call to action ». Julien recherche une personne dédiée à l'enseignement. Il tombe enfin sur une section Education : clients, personnes du pôles, publications … mais aucun Business Analyst. Julien ne sait donc pas comment ce cabinet pourrait nous aider. Et il se demande pourquoi McKinsey a-t-il conçu son site de cette manière ?

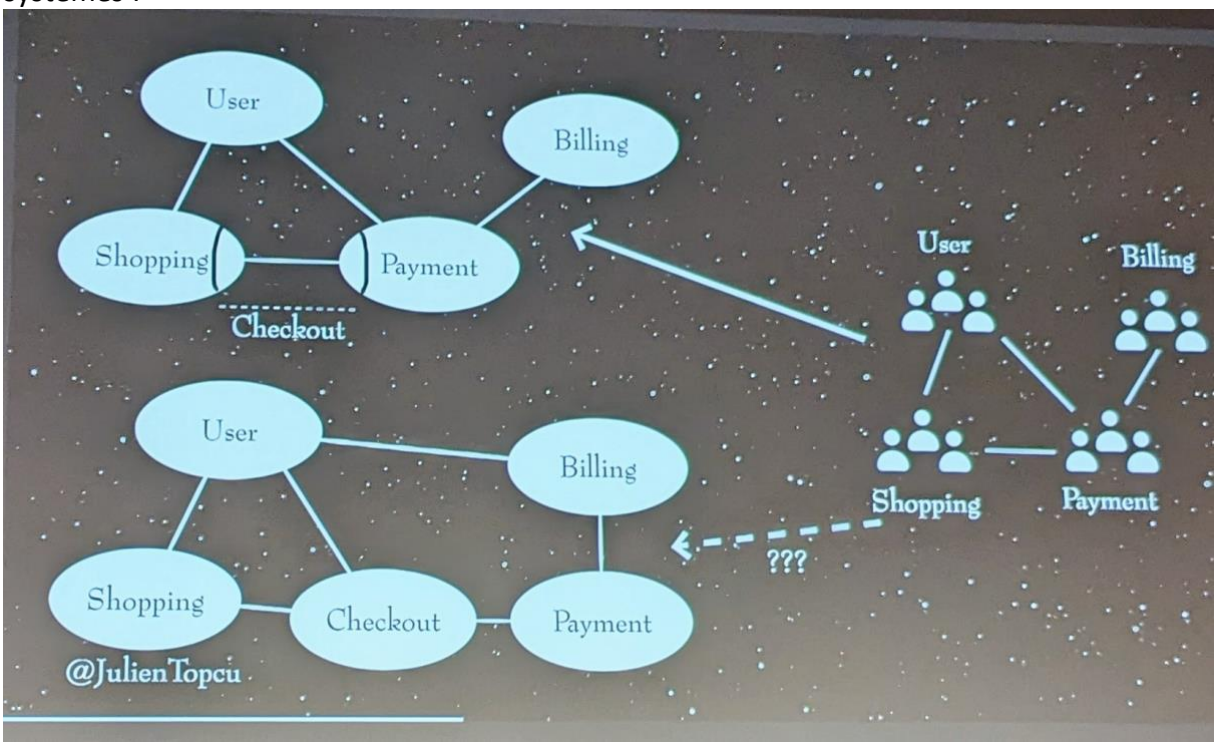
Pour répondre à cette question, remontons en 1998. Jakob Nielsen y est alors l'un des pionniers de l'UX Design. Il affirme que la structure du site permet de déduire la structure de l'organisation. Nigel Bevan ajoute qu'un site reflète plutôt des besoins internes que ceux des utilisateurs (d'où la recherche de nouveaux collaborateurs sur le site de McKinsey).

Après l'exemple du site de McKinsey, Julien nous parle d'une pub d'Apple de 2008 se moquant des déboires de **Windows Vista**. Cet OS comporte 50 millions de LoC et 3404 binaires. Commanditée par Microsoft, l'étude « The influence of organizational structure on Software Quality: an empirical case study» démontre que l'organisation de Microsoft a joué un rôle clé dans cette débacle.
Julien clame haut et fort : « Ce n'est pas Sonar qui va vous sauver si vous faites du SAFE ».

Pour 3ième et dernier exemple, Julien prend le cas d'une multinationale leader en réservation de billets de train. Ses architectes souhaitent mettre en place une architecture agnostique aux différentes compagnies ferroviaires opérant dans chaque pays. Ceci avec autant d'équipe que de pays. Ce choix amène un effet homomorphique.
Dans le code, on retrouve autant de if que de pays.
Pattern **Feature Team** : permet d'implémenter une feature dans tout le SI. Les 3 features teams des 3 pays différents travaillent sur le même code, ne se parlent pas et mettent des if pour ne pas casse le code du pays voisin.

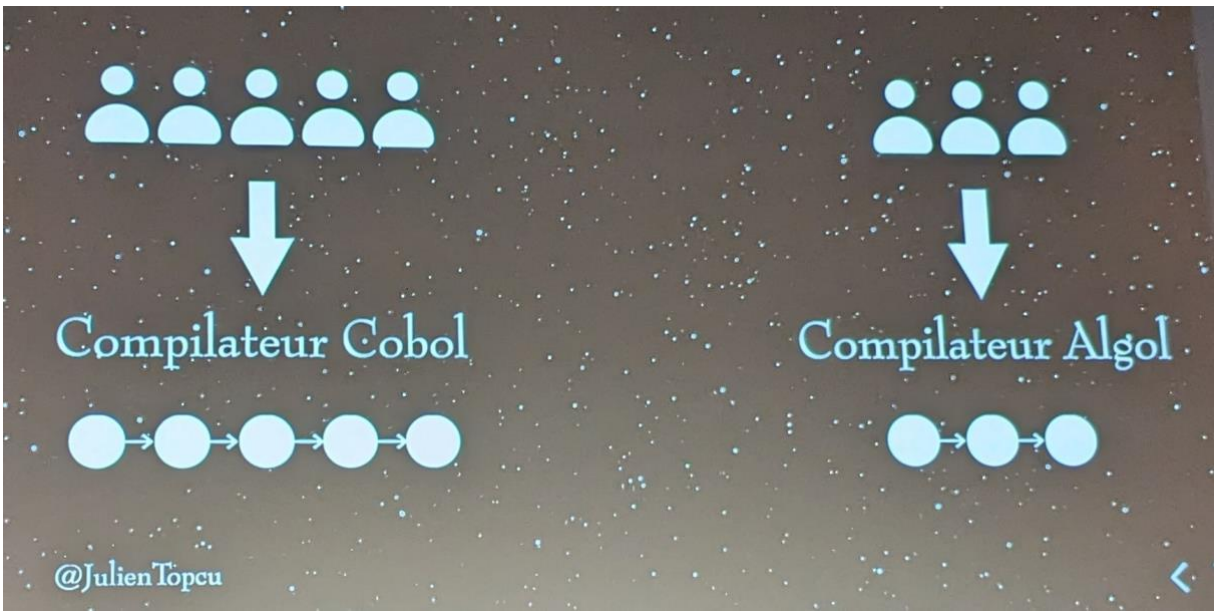
A travers ces 3 exemples, l'IT Zone semble nous envoyer un message : **la qualité du code, l'architecture et la conception de produit semble être lié à l'organisation de la société**.
Ils ont tous les 3 été victims de la loi de Conway : "organizations that design systems are constrained to produce systems which are copies of the communication structures of these organizations."

Melvin Conway démontre qu'il existe une relation étroite entre les sous-systèmes d'une organisation et l'organisation qui les a produits. En général, il existe une équipe pour chaque sous-sytème. Les sous-systèmes sont interfacés. Les équipes doivent être coordonnées (par l'équivalent d'un Scrum Master).
L'homomorphisme peut être expliqué par le schéma suivant : lorsqu'une organisation doit mettre en place une architecture, elle va retenir la solution qui se calque sur son organisation. L'organisation a polarisé la manière de réfléchir et va donc produire le 1er des 2 systèmes :

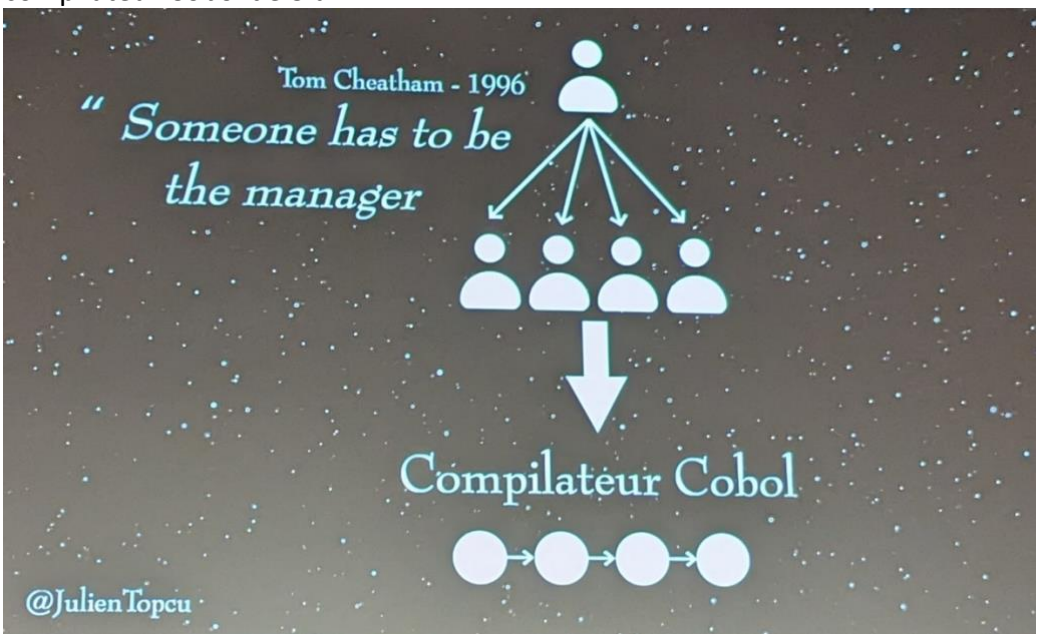


Choisir une organisation, c'est déjà contraindre la structure des produits qu'elle produira.
Pour illustrer, Conway prend l'exemple de 8 personnes devant implémenter un compilateur Cobol et Algol. On les répartit de la manière suivante : 5 développeurs Cobol et 3 développeurs Algol. Devinez quoi ? Le compilateur Cobol est construit autour de 5 étapes de

compilation et celui d'Algol autour de 3 étapes seulement. Les développeurs se sont répartis les tâches.



Tom Cheatham répond à Conway en 1996 que pour coordonner une équipe de 5 personnes, on nommera très souvent un manager, ce qui ferait passer le nombre d'étapes du compilateur Cobol de 5 à 4.



Trop souvent l'Organisation est mise en place avant de savoir ce qu'on produit. On découpe l'organisation du travail afin que chacun ait une tâche à faire.

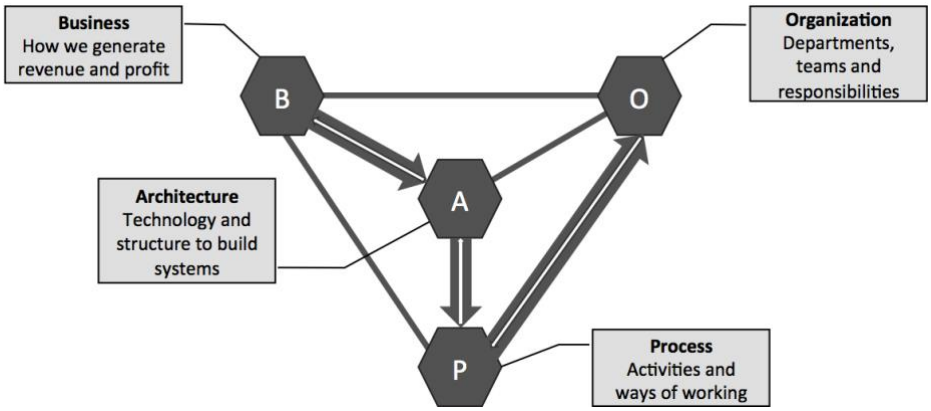
Exemple de l'escalade : lorsque 2 personnes de même niveau n'arrivent pas à se mettre d'accord. Des choix techniques vont incomber à des responsables qui ne sont plus dedans et ceci en une demi-heure.

Dans le livre « Le Mythe du mois-homme » de Frederick Brooks publié en 1975, 3 personnes entrainent 3 canaux de communication alors que 11 personnes induit 55 canaux de communication. L'effort de travail n'est pas linéaire.
Ajouter des gens pour atteindre la deadline ajoute des coûts. Pour les diminuer et laisser les gens travailler, on va devoir retirer des canaux de communication.

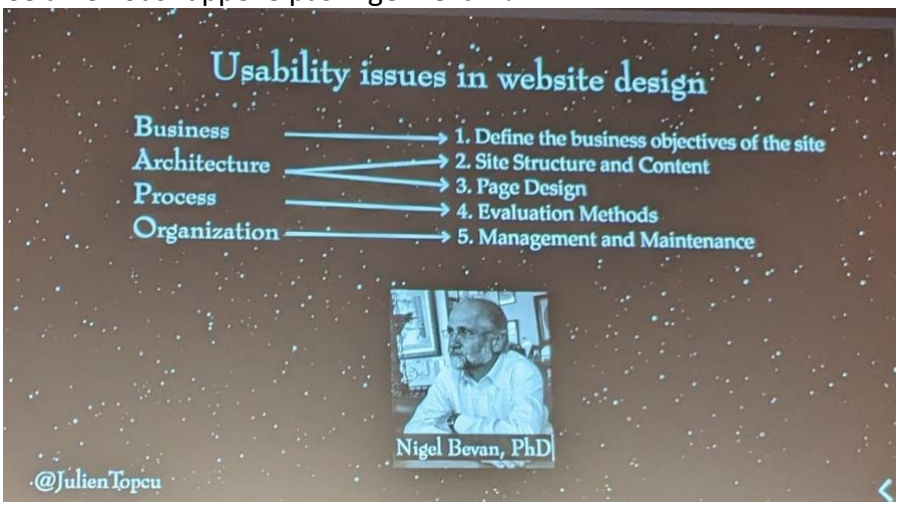
Une nouvelle fois, Julien fait référence à SAFE 6.0 : le **monolith distribué**

Est-on condamné à subir la loi de Conway ? Malheureusement oui. Mais on peut réduire son emprise. Par exemple en mettant en place une organisation qui soit en phase avec le design. Un design pouvant changer, il faut que la structure soit flexible. En 2010 est apparue la **Manœuvre de Conway inversée** ou « Inverse Conway Maneuver ». Les silos empêchent d'être efficaces et doivent être cassés.

En 2015 : l'essor des **Microservices** dans l'IT Zone.
En 2017 : modèle **BAPO** « Structures eats Strategy » de Jan Bosch. Les entreprises font l'inverse : partent de l'organisation et non pas du produit.
BAPO et le **Domain-Driven Design** tendent vers le même principe : remettre au centre le métier. Biais d'un ingénieur : penser aux solutions quand on nous parle des problèmes.

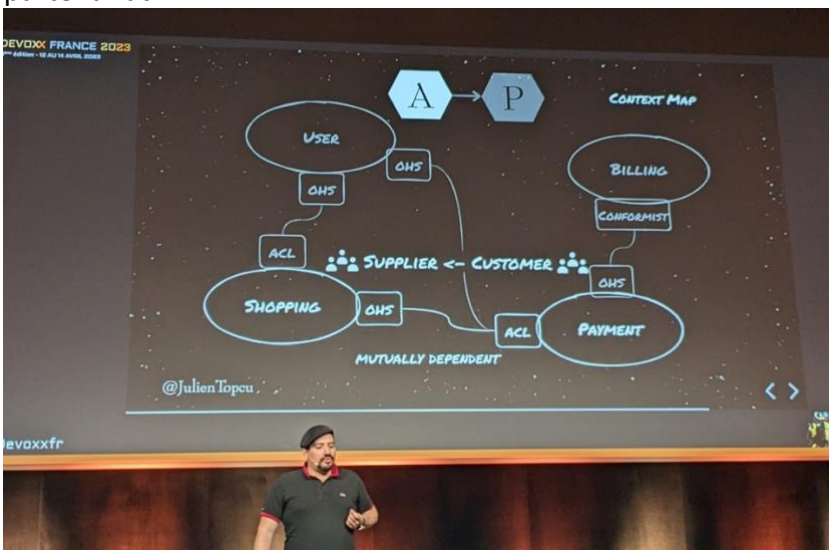


Cela ne vous rappelle pas Nigel Bevan ?



Pattern Stratégique : atelier **event storming** permettant de lister tous les évènements métiers sur une timeline. On arrive aux **Bounded Context** implémentés sous forme de sous-systèmes. On passe ensuite au Context Map.

Mutually Dependent : si l'un des 2 sous-systèmes échoue, l'autre également.
Dans le schéma ci-dessous, le lien customer -> supplier pourrait être remplacé par un lien de partenariat.



Le DDD ne donne pas d'infos sur comment s'organiser une fois les bounded-context identifiés.

Sortie en 2019, le livre "Team Topologies" de Matthew Skelton et Manuel Pais donne ce conseil : "limit the size of software services/products to the cognitive load that the team can handle". La taille du domaine A doit être construit de manière à ne pas surcharger leur charge de travail. Possibilité de monter une nouvelle équipe comme la team Checkout.
Le Everything-as-a-Service (XaaS) : réduire les interactions entre les différentes équipes.
La Manoeuvre de Conway inversée ne résout pas tout.
Ce n'est pas parce qu'on change l'organisation que le système va être aligné avec ce qu'on souhaite. Une réorganisation coute cher, en termes humain et financier.

Julien termine son show par 3 réalités :

1. Une grande perte de membres de l'équipe affecte la rétention des connaissances et donc la qualité.
2. Plus les contributeurs à un composant appartiennent à la même équipe, plus grande est la qualité.
3. Plus la personne qui a le pouvoir de prendre une décision sur le composant est proche hiérarchiquement des ingénieurs qui l'éditent, meilleure est la qualité