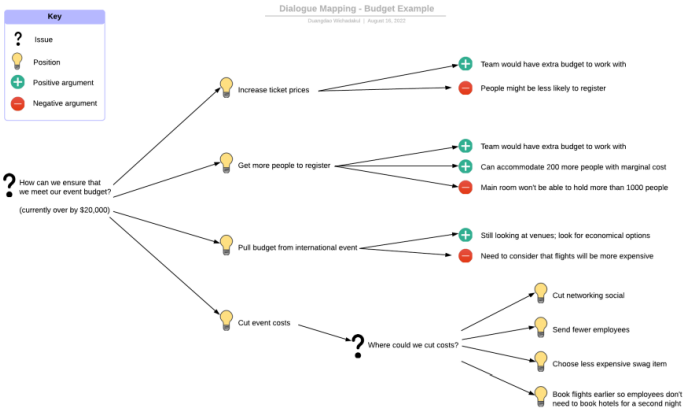
Chapter 2 Architectural Decision Record (ADR)

ADR Related items

- **Architecture Decision (AD)** เป็นการตัดสินใจ (เกี่ยวกับ Architecture) ภายใน context ใด ๆ
- **Architectural Decision Record (ADR)** คือ record (document) ที่เก็บรวบรวม AD ภายใต้ context ต่าง ๆ รวมถึงรวบรวมผลลัพธ์ (consequences) ของการตัดสินใจนั้น ๆ ด้วย
- **Architectural Decision Log (ADL)** เมื่อมี ADR หลาย ๆ ตัวแล้วจะกลายเป็น ADL หรือก็คือ log ของระบบ
- **Architecturally-Significant Requirement (ASR)** คือ requirement ที่มีผลต่อ architecture ของเรา
- **Architecture Knowledge Management (AKM)** ทั้งหมดที่กล่าวมารวมกับเป็น AKM หรือก็คือ knowledge base

How to start using ADRs?

- มีการจัด ranking ว่าเรื่องไหน urgent เรื่องไหน important เรื่องไหนรอได้
- มีทำ **Decision TODO** ซึ่งคือลิสของสิ่งที่ต้องตัดสินใจ เพื่อทำให้ **Product TODO** เดินต่อไปได้
- มี Decision-making technique แนะนำคือ **Dialogue mapping**
  - Issue >> ประเด็นที่กำลังตัดสินใจ
  - Position >> วิธีการแก้ปัญหา
  - Positive/Negative argument >> ข้อโต้แย้ง



Decision Enforcement

- **ADR** เป็นส่วนหนึ่งของ **software design** ดังนั้นต้องได้รับการยอมรับจาก stakeholder ไม่ใช่อยากทำอะไรก็ทำ
- ตามหลักเมื่อมี **new technology** ที่น่าสนใจ ก็ควรจะมีการพิจารณา AD ด้วย
  - แต่ตามความเป็นจริง ถ้า ignore แบบปล่อยทิ้งไว้ 1-2 ปีก็พอรับได้ (ประมาณว่าไม่ได้ปรับ tech stack)

Decision Sharing

- ควรเก็บการตัดสินใจ รวมถึงบทเรียนไม่ว่าจะดีหรือแย่ไว้เป็นบทเรียน โดยเก็บไว้ใน **AKM** (Knowledge base)

> How to start ARDs with tools >> Google Drive, Git, VS Code, Jira หรืออะไรก็ได้

Suitable ADR

Characteristics of a Suitable ADR (PRIS)

- **Point in Time** >> บอกว่า AD นั้น ๆ เกิดขึ้นเมื่อใด

- **Rationality** >> บอกว่าตัดสินใจทำไม
- **Immutable record** >> ไม่แก้ไข ADR ที่ published ไปแล้ว
  - ถ้าการตัดสินใจครั้งใหม่ ๆ นั้น ๆ แย้งกับครั้งก่อน ๆ ก็สร้าง ADR อันใหม่เลย ไม่ต้องแก้ของเก่า
- **Specificity** >> 1 ADR ต่อ 1 AD
  - หมายความว่า 1 record ต่อ 1 การตัดสินใจ (ไม่ใช่เขียนรวม ๆ กว้าง ๆ หลาย ๆ เรื่อง)

Characteristics of a Good Context

- ต้องอธิบาย organization's situation และ business priority และควรอธิบายสภาพแวดล้อมด้วย ยกตัวอย่างเช่น ทีมมีกี่คน แต่ละคนมีทักษะอะไรบ้าง resource ของบริษัทเป็นอย่างไร budget ต่อปีเป็นอย่างไร

Characteristics of a Good Consequence

- บอกไปเลยว่าเราจะเลือกใช้อะไร อย่าบอกแค่ข้อดีข้อเสีย

Twelve factor app

บอกว่า Application ในสมัยใหม่ควรมีลักษณะเป็น as a service on cloud โดยควรมีคุณสมบัติครบทั้ง 12 ข้อนี้

1. **Codebase** >> เช่นเป็น Git deploy ได้หลาย ๆ ที่
2. **Dependencies** >> ประกาศ dependencies ที่จะใช้ให้ชัดเจน
3. **Config** >> มีการ stored config ใน .env
4. **Backing services** >> อะไรก็ตามที่ app เราเชื่อมต่อผ่าน network ควรแยกออกไป เช่น database ก็ควรแยกออกจาก front ไม่ใช่เก็บข้อมูลใน array อย่างเดียว
5. **Build, Release, Run**
6. **Process**
7. **Port Binding** >> port number มีความ dynamic เมื่อ deploy แล้ว port เปลี่ยนก็ยังสามารถเข้ามาถึง app ได้อยู่ (ไม่ควร fixed port)
8. **Concurrency**
9. **Disposability** >> System สามารถ start หรือ shutdown ได้อย่างรวดเร็ว และ state ไม่ผิดพลาด
10. **Developing/Production parity** >> development และ production ควรมีความคล้ายคลึงกัน ไม่ใช่ต่างกันมาก
11. **Logs** >> มีการทำ logging
12. **Admin process** >> admin สามารถ run ครั้งเดียวแล้วที่เหลือ Autonomous ได้ (one-off process)

> 💡 มีนำเสนอ Fifteen-factor App มาอีกในภายหลัง
> 13. **API First**
> 14. **Telemetry** >> คล้าย ๆ การทำ monitoring (ต่างจาก log ตรงที่)
> 15. **Authentication and Authorization**