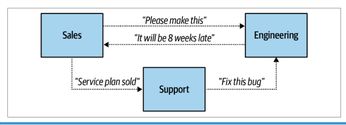
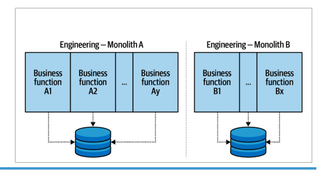
Adam Bellemare - Building Event-Driven Microservices

1. Why Event-Driven Microservices

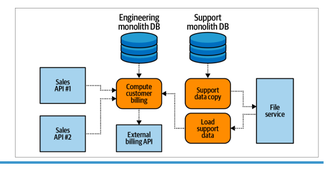
- What Are Event-Driven Microservices?
  - Architectures
    - 1. Traditional monolith-style
    - 2. Service-oriented architectures (SOAs)
      - multiple microservices synchro-nously communicating directly with one another
    - IT IS WHAT THIS BOOK IS ABOUT
    - 3. Event-driven microservice (EDM) architectures
      - Async consuming of events
  - Services/microservices
    - Small - can be implemented in 2 weeks
    - Fit in one head
- Domain-Driven Design and Bounded Contexts
  - Domain
  - Subdomain
  - Domain (and subdomain) model
  - Bounded context
    - ⚠ Should be built around business requirements and not technological requirements
      - Because it's rare for a company to need to change the underlying implementation of any given product without accompanying business requirement changes
      - ⚠ Tradeoffs
        - Code may be replicated a number of times, and many services may use similar data access patterns
- Three levels of Communication Structures
  - Business Communication Structures



  - Implementation Communication Structures



  - Data Communication Structures



  - Conway's Law
    - *Organizations which design systems...are constrained to produce designs which are copies of the communication structures of these organizations. (c) Melvin Conway—How Do Committees Invent? (April 1968)*
- Problems in Traditional Computing (how we use and evolve Communi-cation Structures)
  - When new business requirement arrives
    - Option 1: Make a New Service (move from Monolith to SOA)
      - Hard, slow, requires extra testing and monitoring, requires data sync
      - Loose coupling
      - ⚠ However synced data is still coupled!
    - Option 2: Add It to the Existing Service (stay in Monolith)
      - Easy, fast
      - Tight coupling
      - Most team would select this option
    - ⚠ The problem is due to a weak or nonexistent data communication structure
  - When team grows and need to separate into two teams
    - business communication structure cannot work here - because need to assign requirement to one team
    - implementation communication structure cannot support reassign-ments and needs to be broken down
      - What does it mean?
    - ⚠ The problem is in weak, ill-defined means of communicating data between implementation communi-cation structures
- Event-Driven approach (introduces different approach to using and evolving Communication Structures)
  - Absolutely different approach (not a simple replacement to traditional request/response)
    - It decouples the production (ownership) of data from the access to it
  - Features
    - Events Are the Basis of Communication
    - Event Streams Provide the Single Source of Truth
    - Consumers Perform Their Own Modeling and Querying
    - Data Communication Is Improved Across the Organization
    - Accessible Data Supports Business Communication Changes
- Asynchronous Event-Driven Microservices (EDM)
  - Benefits
    - Granularity
    - Scalability
    - Technological flexibility
    - Business requirement flexibility
    - Loosely coupling
    - Continuous delivery support
    - High testability
  - No problems in Event-Driven Microservices
    - When new business requirement arrives
      - Make new microservice
    - When team grows and need to separate into two teams
      - Easy to reassign the microservice ownership
- Synchronous Microservices
  - ⚠ Neither point-to-point request-response microservices nor asynchronous event-driven microservices are strictly better than the other
  - Drawbacks of Synchronous Microservices
    - Point-to-point couplings
      - makes future changes more difficult
    - Dependent scaling
      - can be a bottleneck on scalability
    - Service failure handling
      - becomes increasingly difficult
    - API versioning and dependency management
      - can add a lot of complexity
    - Data access tied to the implementation
      - puts the onus of data access and scalability back on the implementa-tion communication structure
    - Distributed monoliths
      - with many intertwining calls being made between them
    - Testing
      - can be difficult
  - Benefits of Synchronous Microservices
    - Certain data access patterns are favorable to direct request-response couplings
      - authenticating a user
      - reporting on an AB test
    - Integrations with external third-party solutions almost always use a synchronous mechanism
    - Tracing operations across multiple systems can be easier in a synchronous environment
    - Services hosting web and mobile experiences are by and large powered by request-response designs
    - Many developers in today's market tend to be much more experienced with synchronous, monolithic-style coding