Domain-Specific Software Engineering

IF3250 Software Project

Reference

- Richard N. Taylor & Nenad Medvidovic & Eric Dashofy, Software Architecture: Foundations, Theory, and Practice, John Wiley & Sons (2019) – Chapter 15
- Dines Bjorner, from Domains to Requirements

Credits

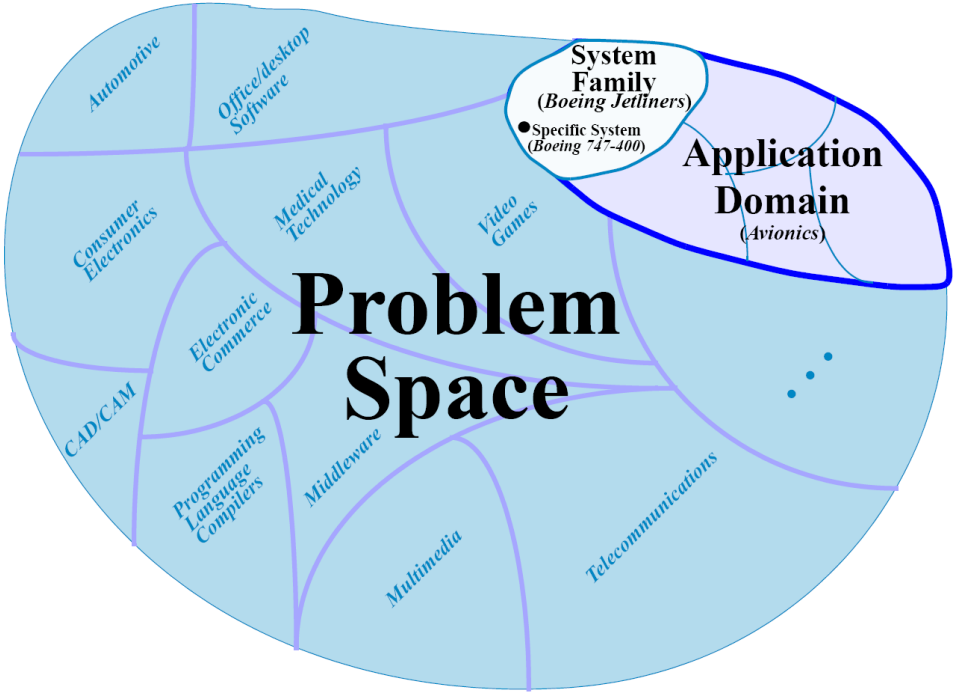
- The slides are adapted from
  - Slide Deck from Richard N. Taylor & Nenad Medvidovic & Eric Dashofy, Software Architecture: Foundations, Theory, and Practice, John Wiley & Sons (2019)

Domain-Specific Software Engineering

- The traditional view of software engineering shows us how to come up with solutions for problems *de novo*
- But starting from scratch every time is infeasible
  - This will involve re-inventing many wheels
- Similar problem → Similar solution → similar software
- Once we have built a number of systems that do similar things, we gain critical knowledge that lets us exploit common solutions to common problems
  - In theory, we can simply build “the difference” between our new target system and systems that have come before

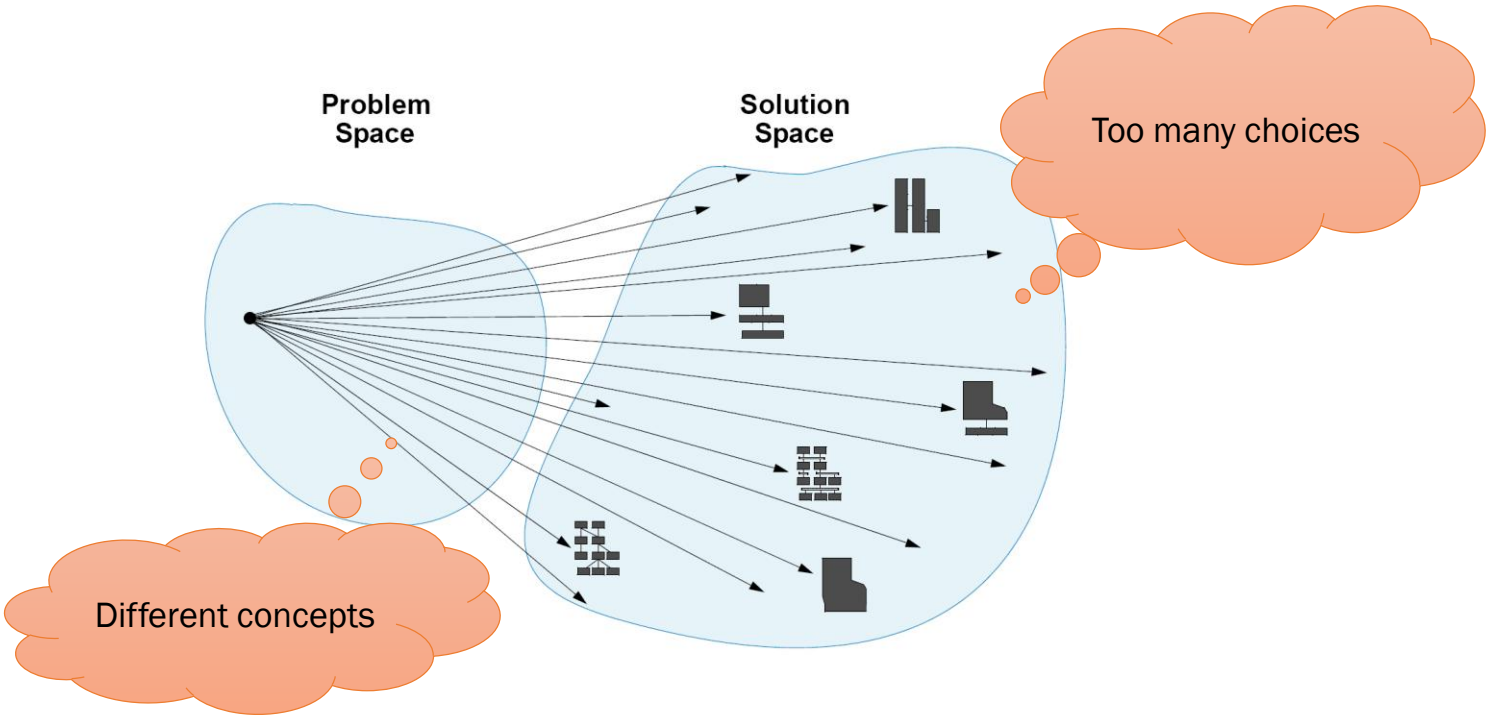
Examples of Domains

- Compilers for programming languages
- Consumer electronics
- Electronic commerce system/Web stores
- Video game
- Business applications
  - Basic/Standard/“Pro”



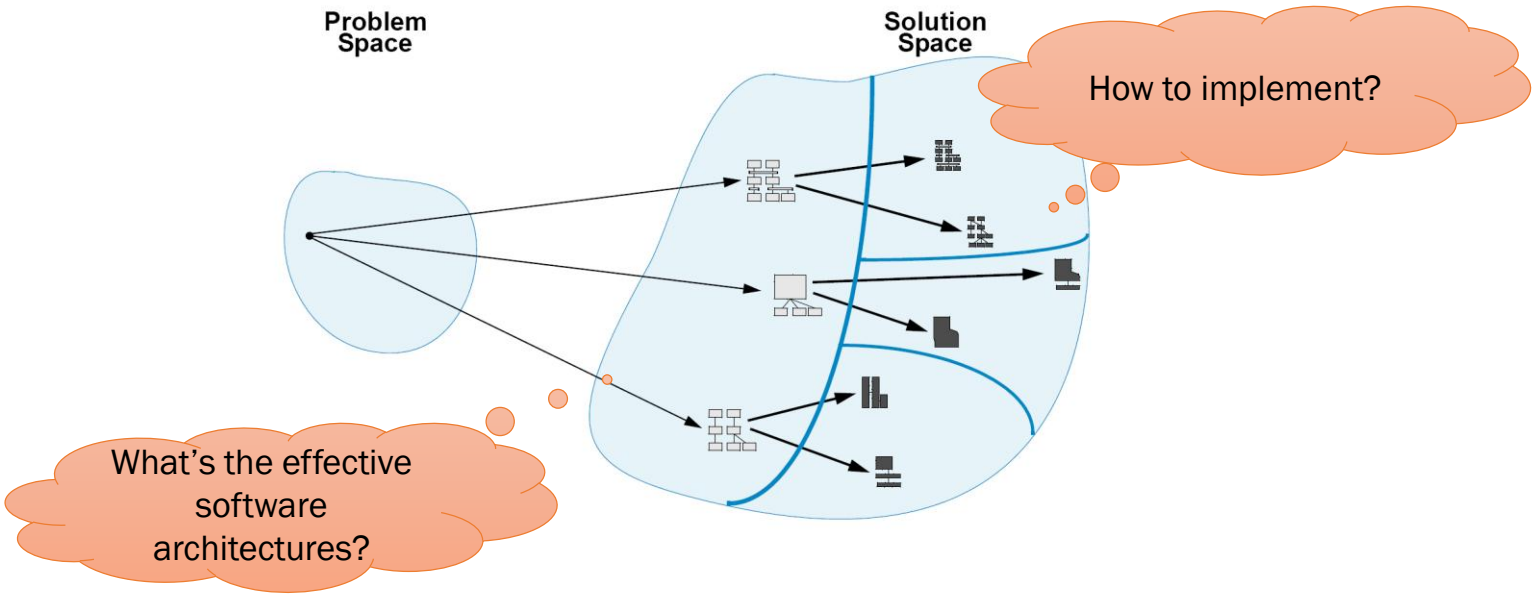
- We can subdivide, too:
  - Avionics systems
    - Boeing Jets
      - Boeing 747-400

Traditional Software Engineering



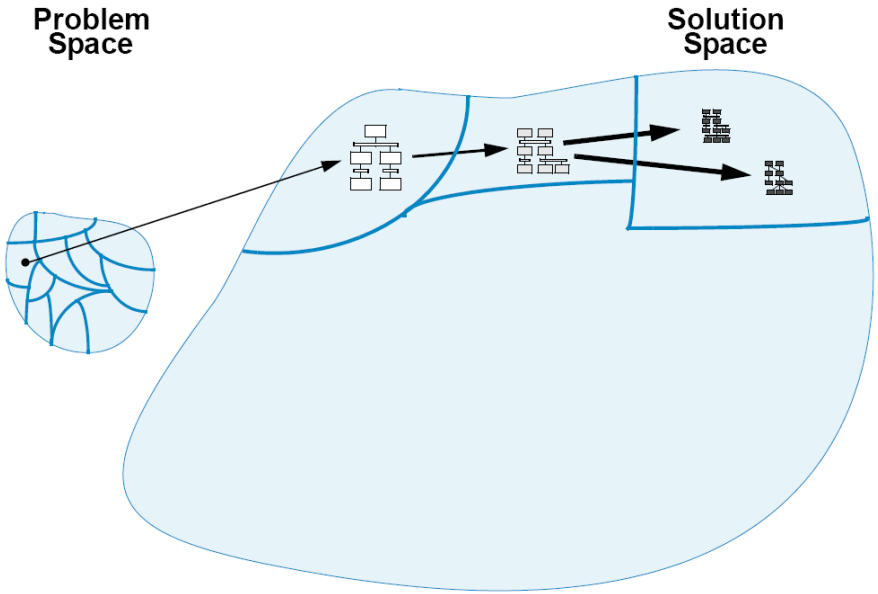
One particular problem can be solved in innumerable ways

Architecture-Based Software Engineering



Given a single problem, we select from a handful of potential architectural styles or architectures, and go from these into specific implementations

Domain-Specific Software Engineering



We map regions of the problem space (domains) into domain-specific software architectures (DSSAs)

These are specialized into application-specific architectures

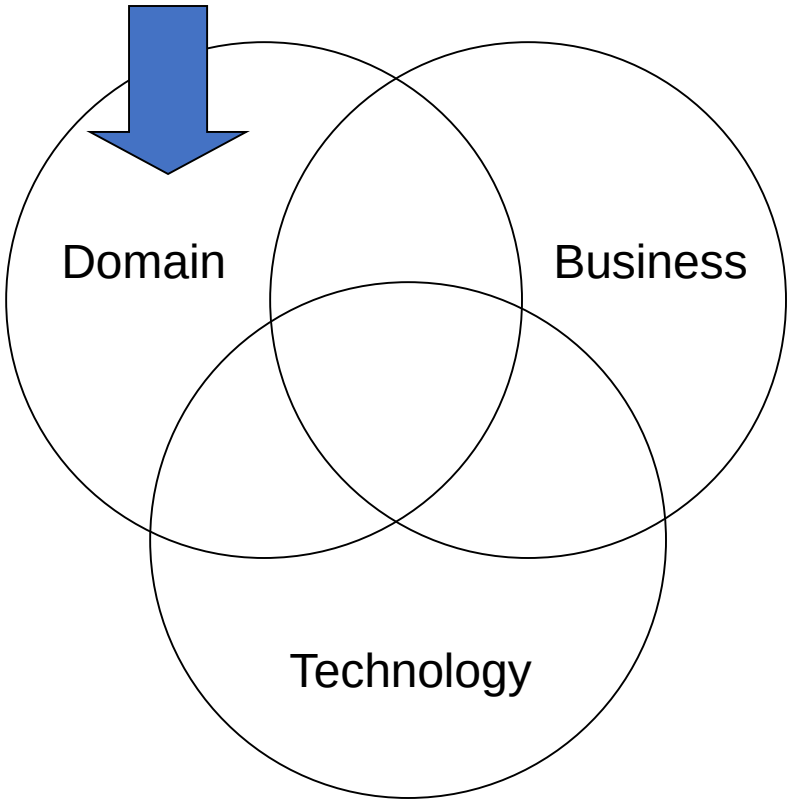
These are implemented

Three Key Factors of DSSE

- Domain
  - Must have a domain to constrain the problem space and focus development
- Technology
  - Must have a variety of technological solutions—tools, patterns, architectures & styles, legacy systems—to bring to bear on a domain
- Business
  - Business goals motivate the use of DSSE
    - Minimizing costs: reuse assets when possible
    - Maximize market: develop many related applications for different kinds of end users

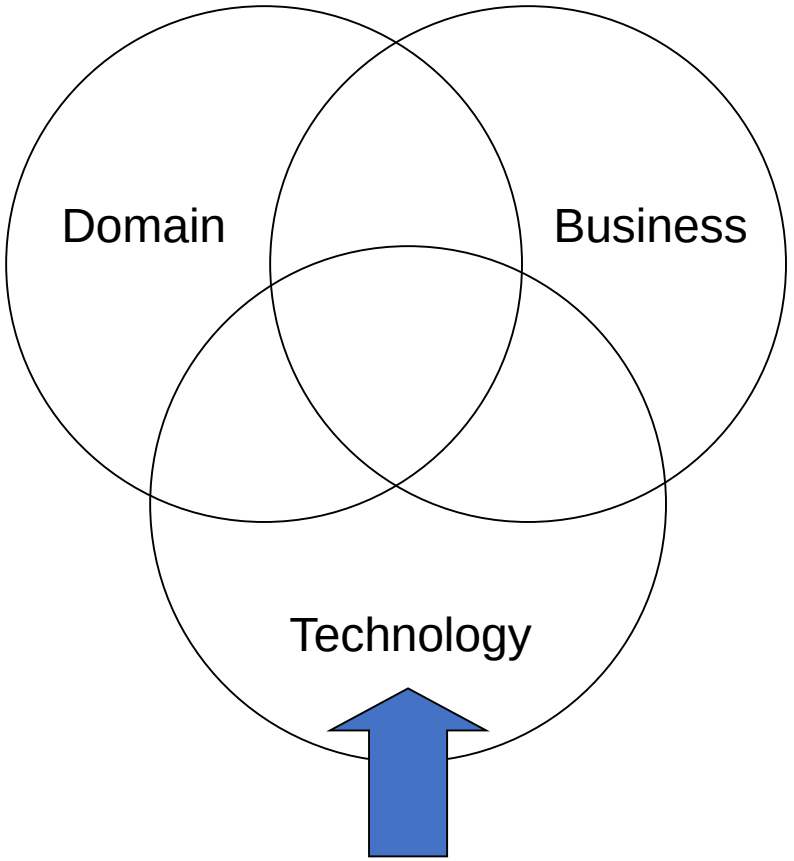
Three Key Factors

- Domain
  - Must have a domain to constrain the problem space and focus development



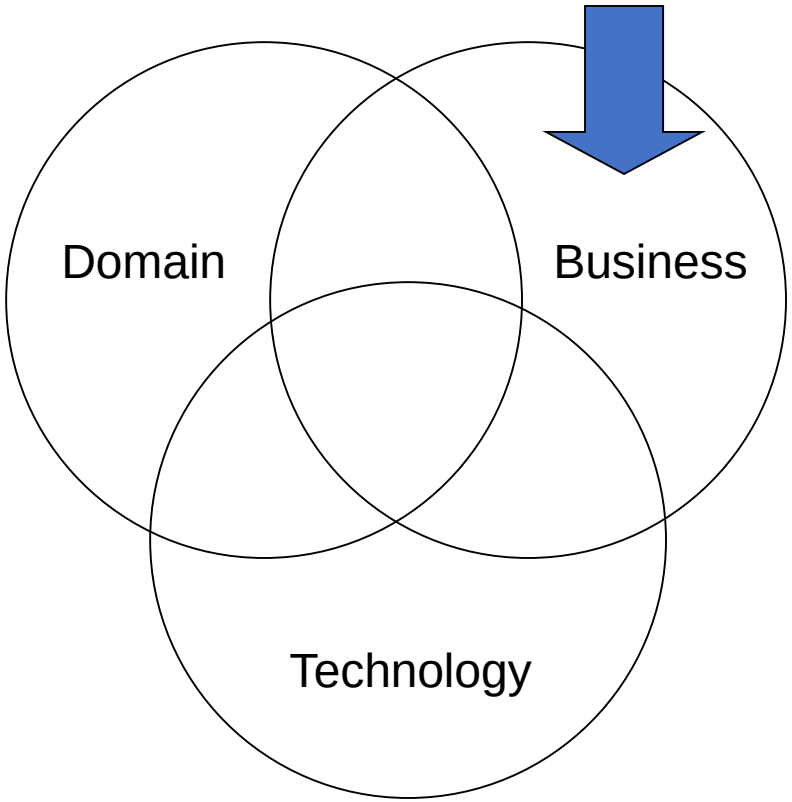
Three Key Factors

- Technology
  - Must have a variety of technological solutions—tools, patterns, architectures & styles, legacy systems—to bring to bear on a domain



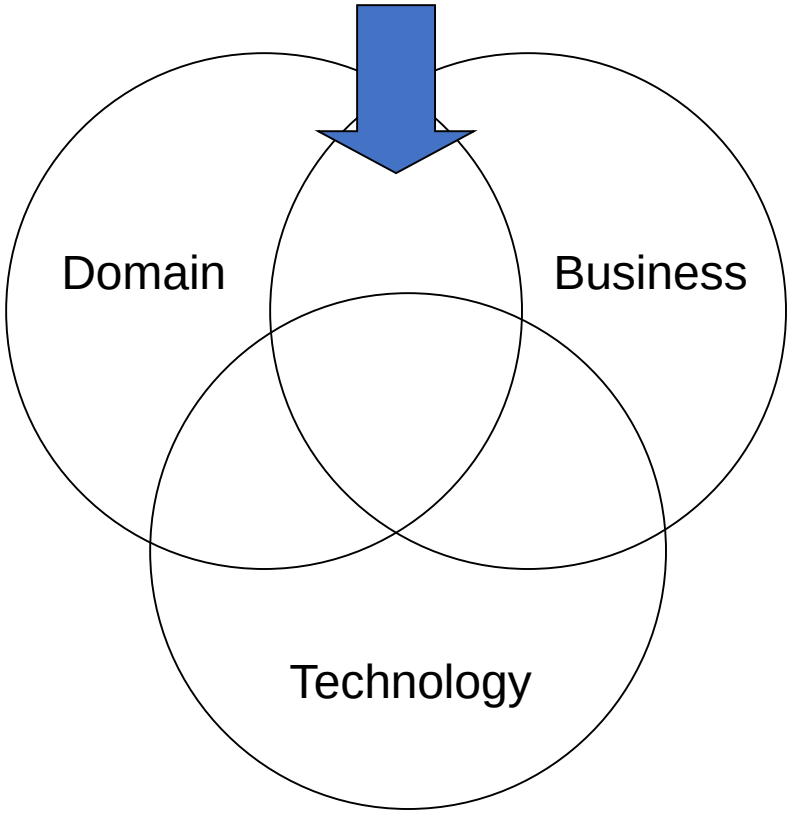
Three Key Factors

- Business
  - Business goals motivate the use of DSSE
    - Minimizing costs: reuse assets when possible
    - Maximize market: develop many related applications for different kinds of end users



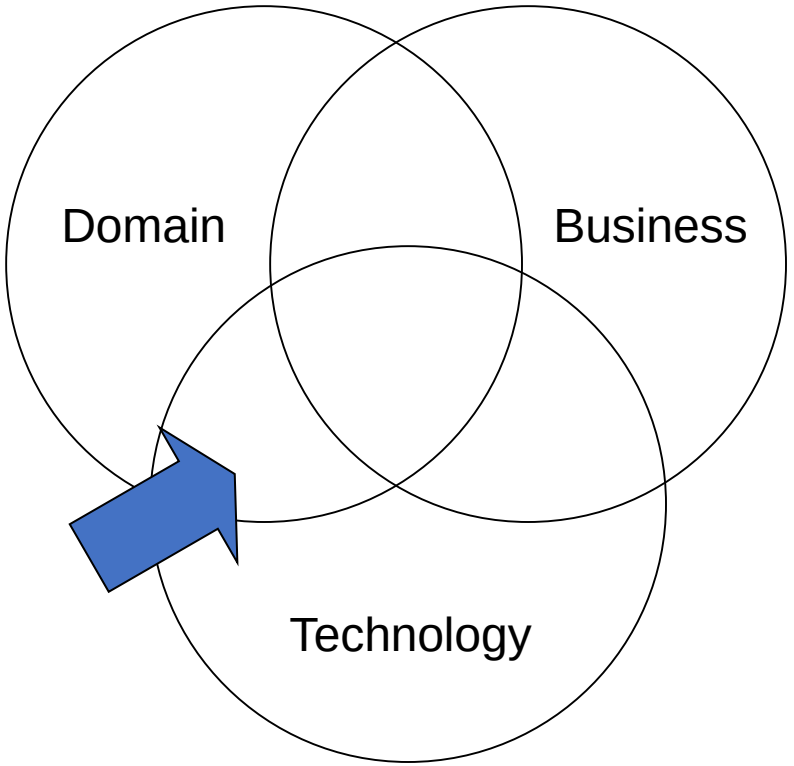
Three Key Factors

- Domain + Business
- “Corporate Core Competencies”
  - Domain expertise augmented by business acumen and knowledge of the market



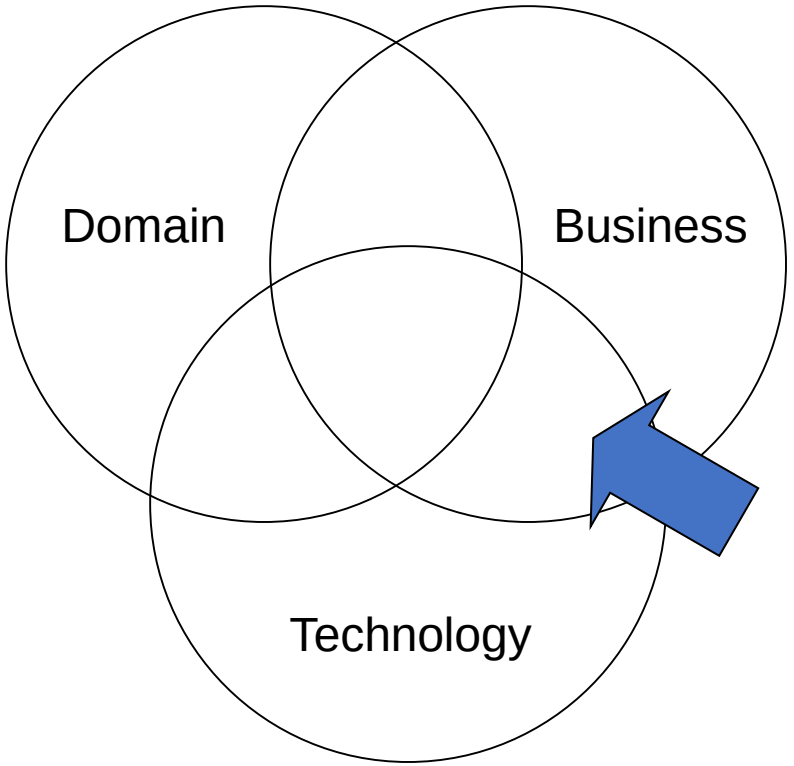
Three Key Factors

- Domain + Technology
- "Application Family Architectures"
  - All possible technological solutions to problems in a domain
  - Uninformed and unconstrained by business goals and knowledge



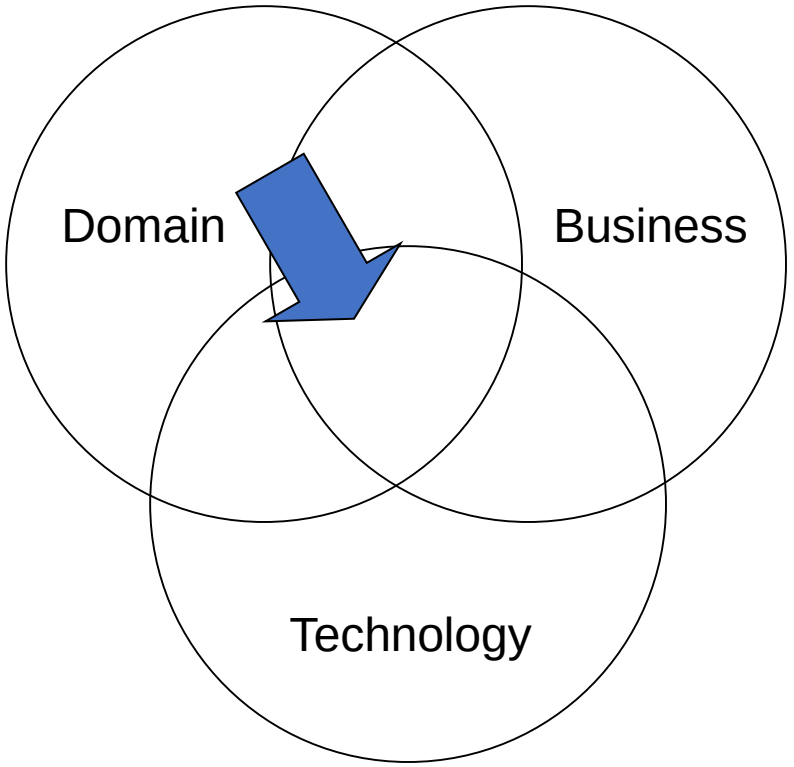
Three Key Factors

- Business + Technology
- "Domain independent infrastructure"
  - Tools and techniques for constructing systems independent of any particular domain
  - E.g., most generic ADLs, UML, compilers, word processors, general-purpose PCs



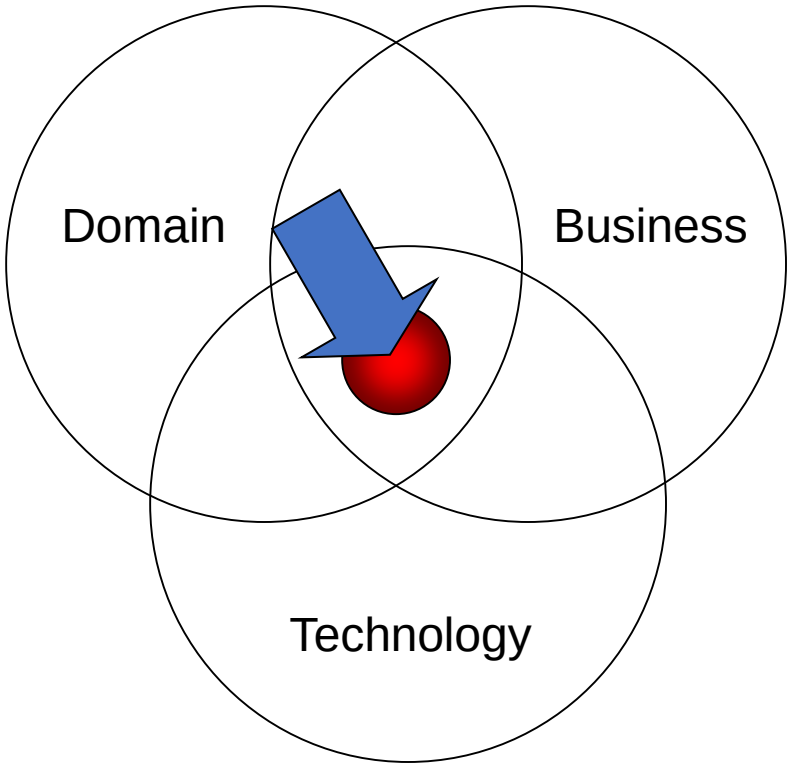
Three Key Factors

- Domain + Business + Technology
- “Domain-specific software engineering”
- Applies technology to domain-specific goals, tempered by business and market knowledge



Three Key Factors

- Product-Line Architectures
- A specific, related set of solutions within a broader DSSE
- More focus on commonalities and variability between individual solutions

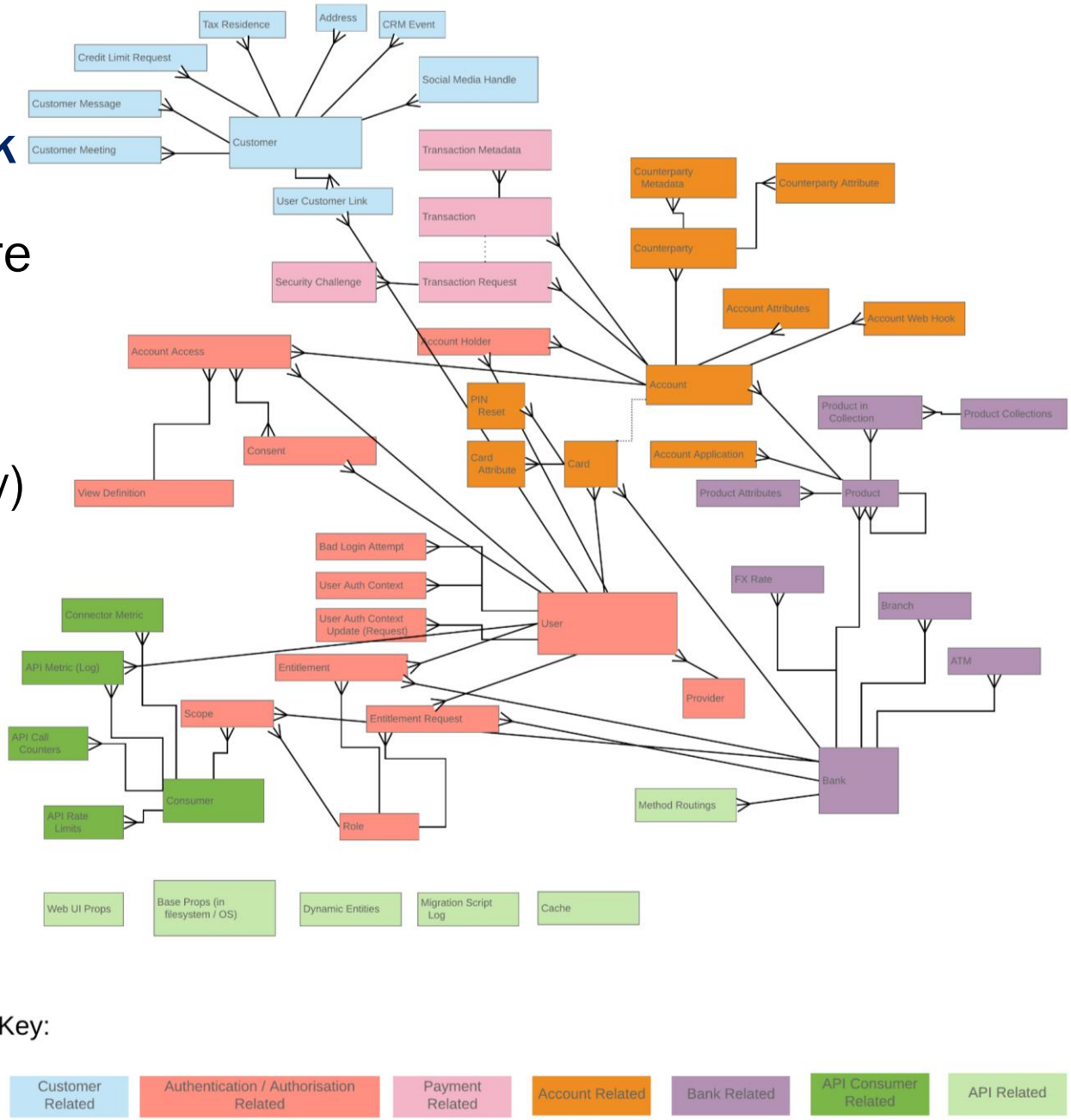


Becoming More Concrete

- Applying DSSE means developing a set of artifacts more specific than an ordinary software architecture
  - Focus on aspects of the domain
  - Focus on domain-specific solutions, techniques, and patterns
- These are
  - A domain model
  - Architecture & Styles (domain-specific software architecture DSSA)
  - Tools
  - Software product line

Domain Model –
Banking by OpenBank

- Develop the software based on existing domain model
- Speed up the (entity) analysis
- Piggyback the knowledge in the domain problem



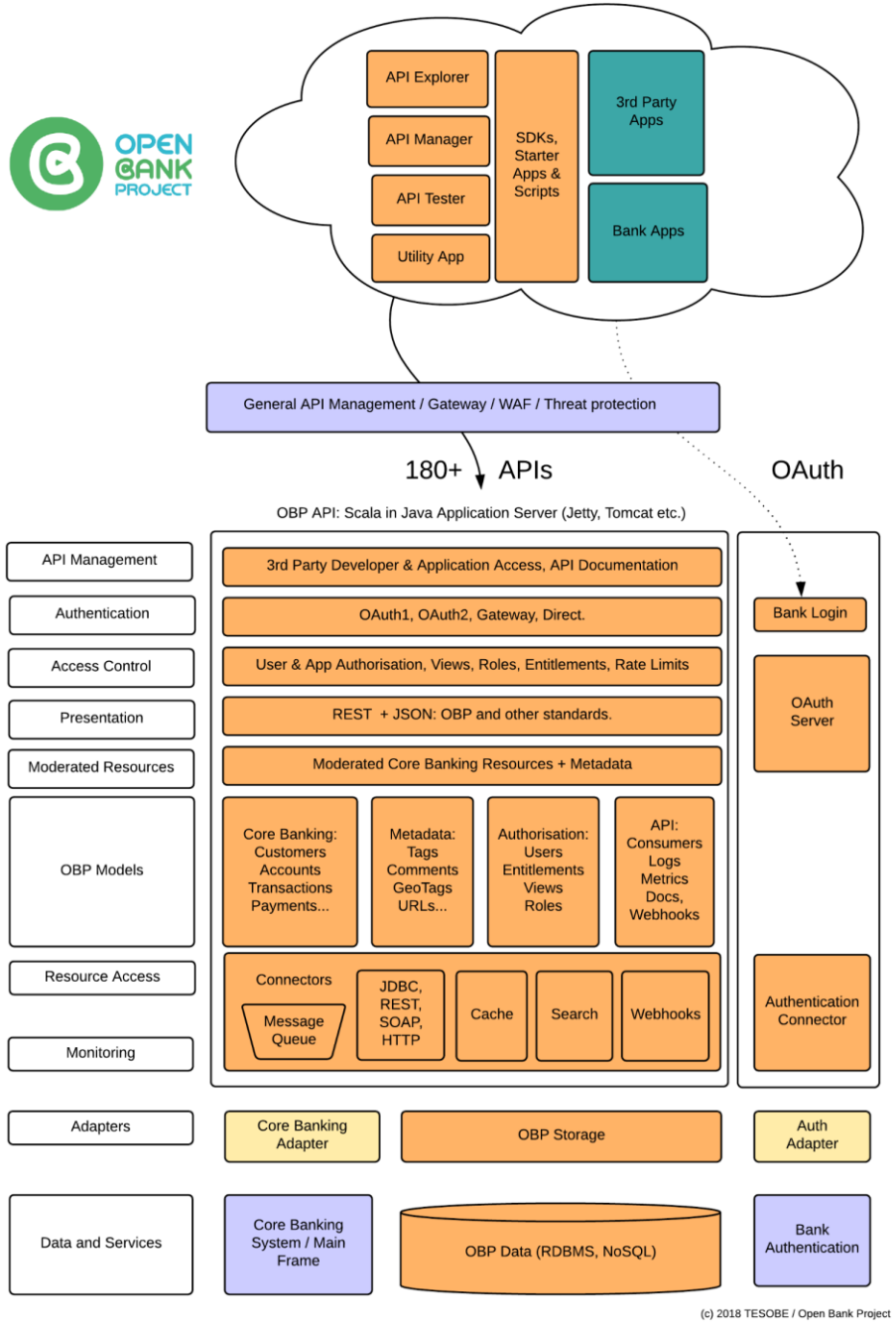
https://apiexplorersandbox.openbankproject.com/glossary

Contoh Domain

- Domain Vertikal (Industry-specific)
  - Domain Perbankan
  - Domain Pertambangan
  - Domain Pendidikan
  - Domain Airline
  - ...
- Domain Horizontal
  - Domain Office Automation
  - Domain ERP
  - Domain CRM
  - Domain CASE Tools
  - Domain Compiler pada Bahasa Pemrograman
  - ...

Reference Architecture by Open Bank Project

- Reference Architecture proposed by Open Bank
- A tested solution to a problem domain; and reduce complexities in designing a software architecture
- Enable delivery of a solution - quicker and fewer errors
- Source https://github.com/OpenBankProject/OBP-API/wiki/Open-Bank-Project-Architecture



Domain-Specific Tools

- CASE Tool or Language
  - Unity3D
  - Actulus Modeling Language
- Faster development
- Native problem concep and programming concepts

```
1.  riskmodel RiskLifeDeath(p : Person) : LifeDeath(p) where
2.  intensities =
3.  alive -> dead by gompertzMakehamDeath(p)
```

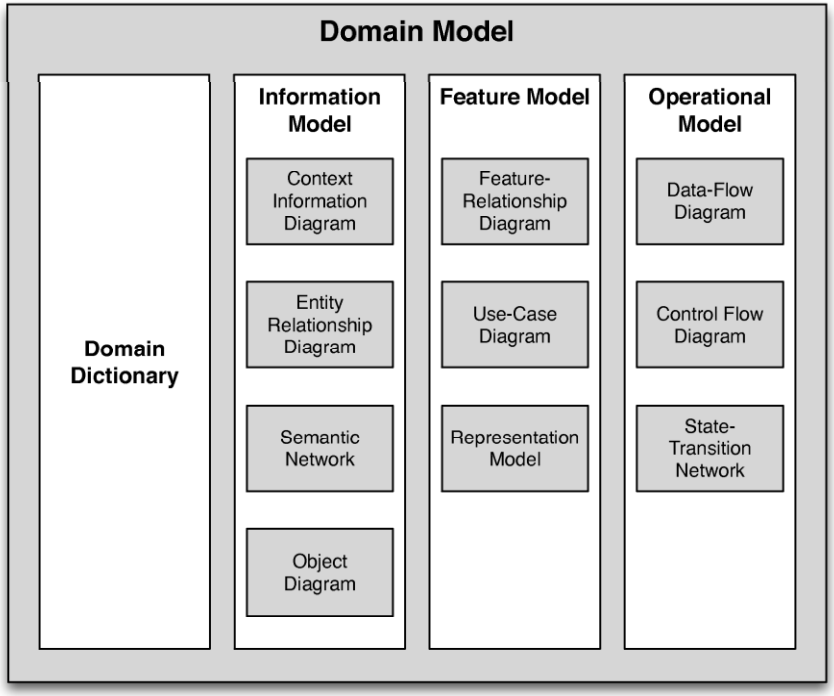
Representing Domain

By Tylor et al

Domain Model

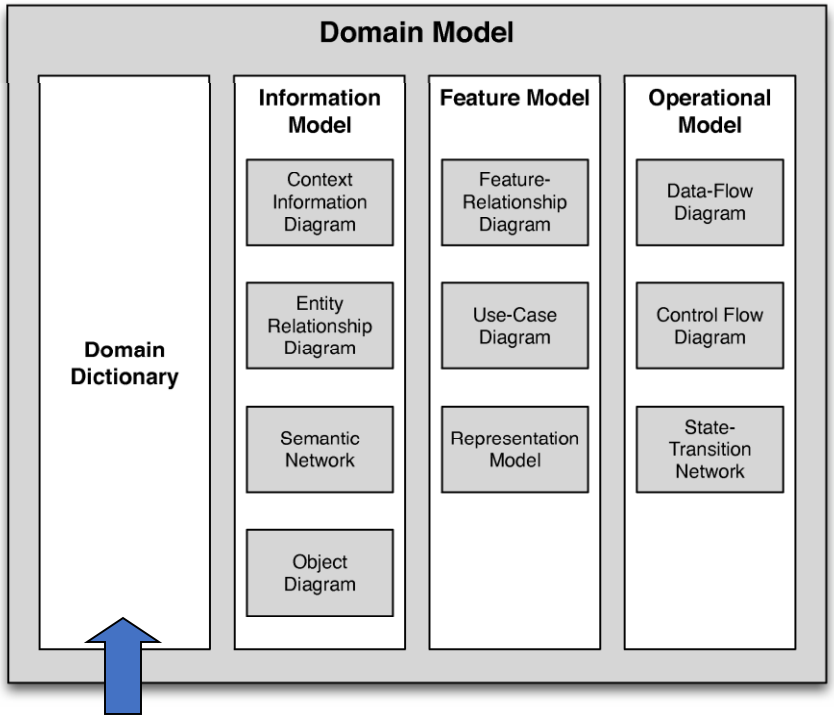
- A domain model is a set of artifacts that capture information about a domain
  - Functions performed
  - Objects (also known as entities) that perform the functions, and on which the functions are performed
  - Data and information that flows through the system
- Standardizes terminology and semantics
- Provides the basis for standardizing (or at least normalizing) descriptions of problems to be solved in the domain

Domain Model



Domain Model

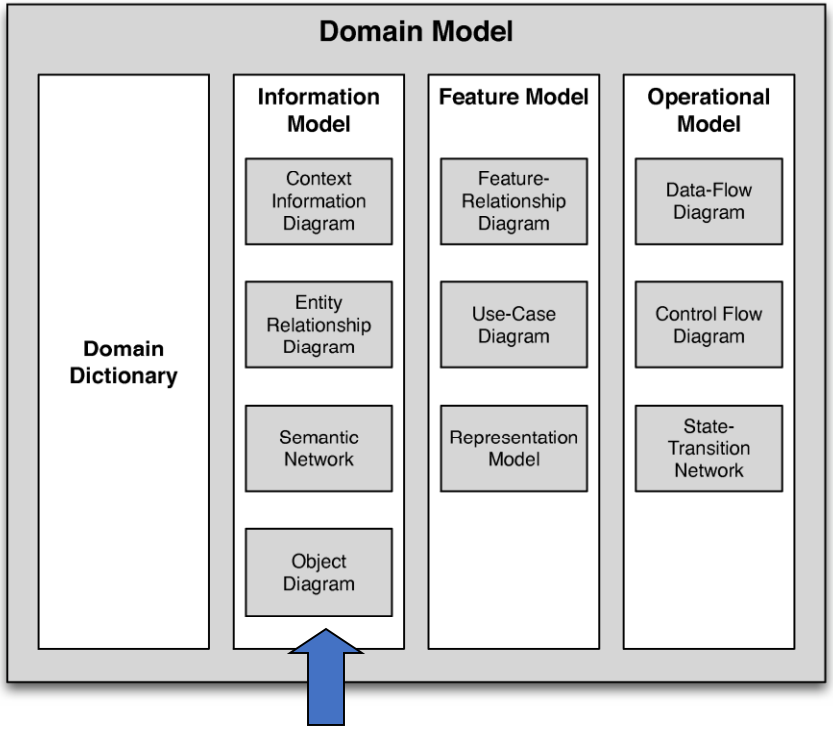
- Defines vocabulary for the domain



Software Architecture: Foundations, Theory, and Practice; Richard N. Taylor, Nenad Medvidovic, and Eric M. Dashofy; (C) 2008 John Wiley & Sons, Inc. Reprinted with permission.

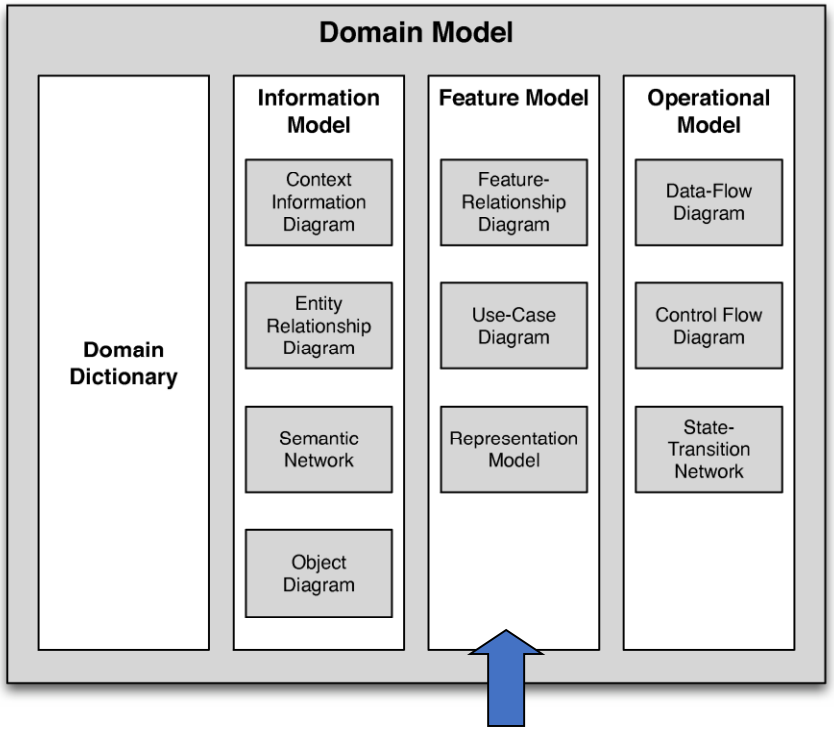
Domain Model

- Describes the entities and data in the domain



Domain Model

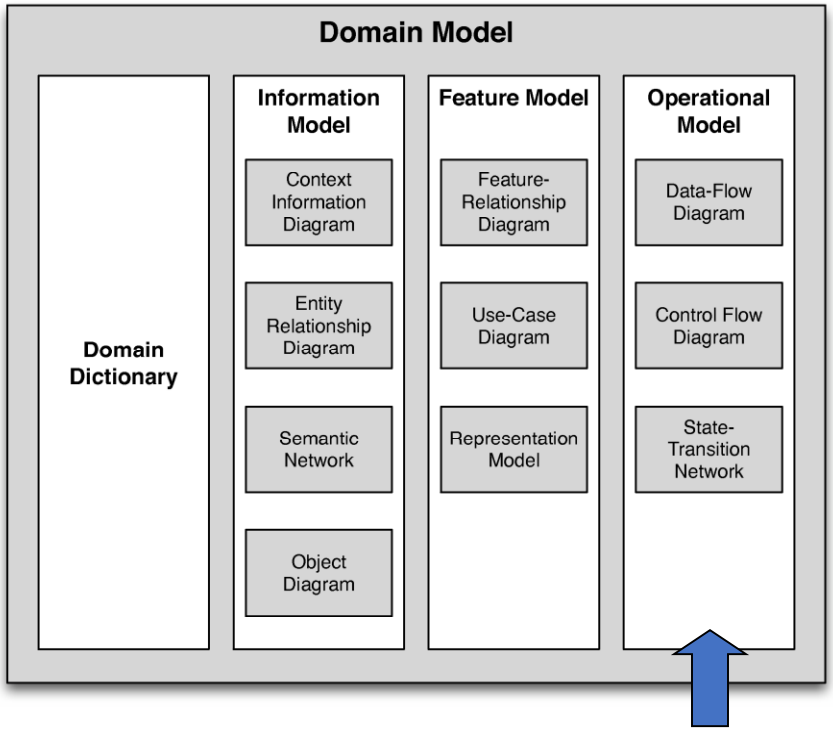
- Defines how entities and data combine to provide features



Software Architecture: Foundations, Theory, and Practice; Richard N. Taylor, Nenad Medvidovic, and Eric M. Dashofy; (C) 2008 John Wiley & Sons, Inc. Reprinted with permission.

Domain Model

- Defines how data and control flow through entities



Software Architecture: Foundations, Theory, and Practice; Richard N. Taylor, Nenad Medvidovic, and Eric M. Dashofy; (C) 2008 John Wiley & Sons, Inc. Reprinted with permission.

Feature Model: Feature Relationship Diagram

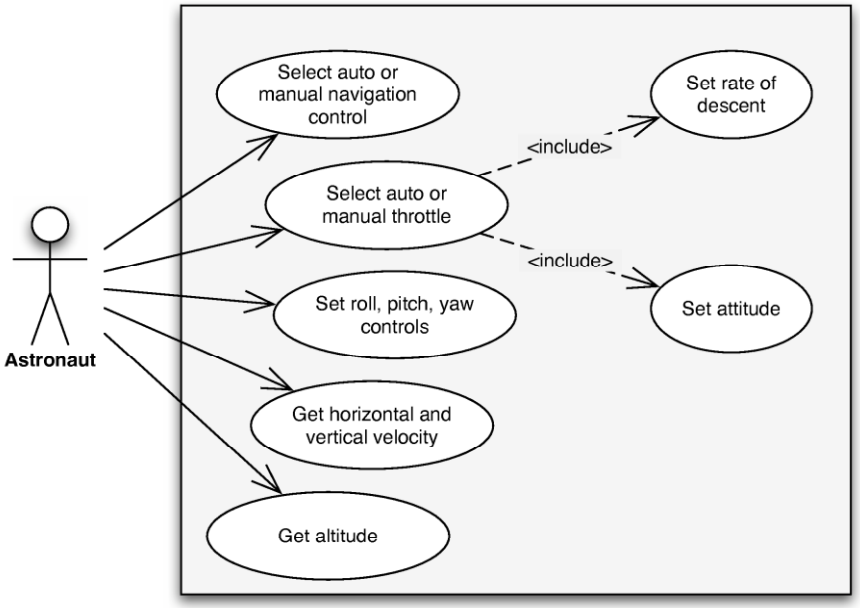
- Describes overall mission operations of a system
- Describes major features and decomposition

Feature Relationship Diagram – Landing Phase

Mandatory: The Lunar Lander must continually read altitude from the Landing Radar and relay that data to Houston with less than 500 msec of latency. Astronauts must be able to control the descent of the Lunar Lander using manual control on the descent engine. The descent engine must respond to control commands in 250msec, with or without a functioning DSKY…
Optional/Variant: Lunar Lander provides the option to land automatically or allow the crew to manually steer the spacecraft.
Quality Requirements:
Real-time requirements: The thrusters and the descent engine must be able to respond to commands from the computer system in real-time.
Fault tolerance: Lunar Lander must be able to continue in its flight-path even when the main computer system (Primary Navigation Guidance & Control) goes down. Lunar Lander must be able to maintain system altitude even when one of the thrusters and propellant supplies goes down in the Reaction Control System.

Feature Model: Use Case Diagram

- Defines use casescwithin the domain
- Similar to use case models in UML



Software Architecture: Foundations, Theory, and Practice; Richard N. Taylor, Nenad Medvidovic, and Eric M. Dashofy; (C) 2008 John Wiley & Sons, Inc. Reprinted with permission.

Feature Model: Representation Diagram

- Defines how information is presented to human users

Representation Diagram

DSKY Unit – Display and Keyboard Unit

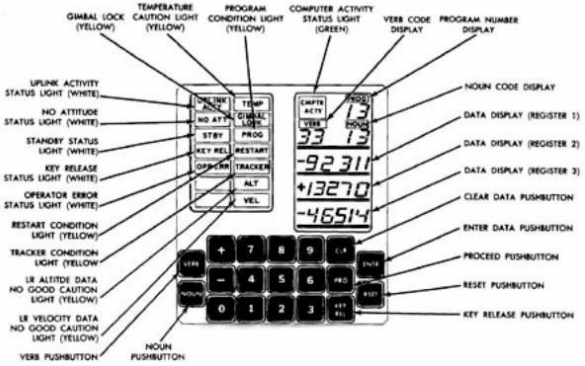


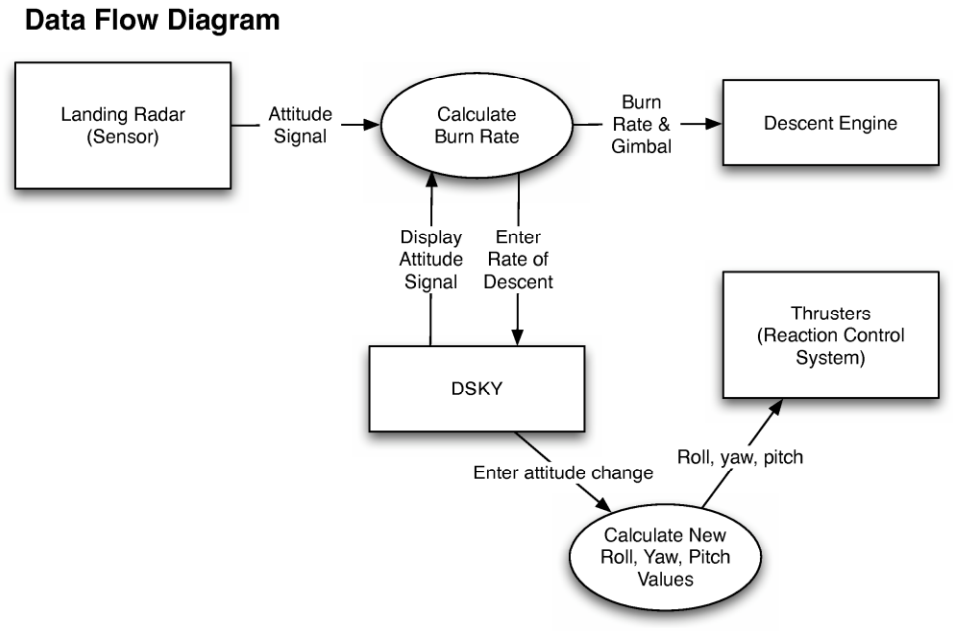
Figure 3: Lunar Module Display and Keyboard Unit (DSKY)

Source: TALES FROM THE LUNAR MODULE GUIDANCE COMPUTER – figure
Apollo 11 The NASA Mission Reports Vol 2 pp 166

- 3 five-digit registers – general purpose
- 3 two-digit registers – indicate phase for landing
- 19 keys
- Warning lights
- Issue commands via VERB & NOUN
  - VERB is the action
  - NOUN is the object to which the action is applied
  - Ex: VERB 6 NOUN 20
    VERB 6 = Display in decimal
    NOUN 20 = Angles
- 70 predefined PROGRAMS
  - Ex: PROGRAM for each descent phase executes trajectory
    - P63 – Braking Phase
    - P64 – Approach Phase
    - P65 – Landing Phase

Operational Model: Data Flow Diagram

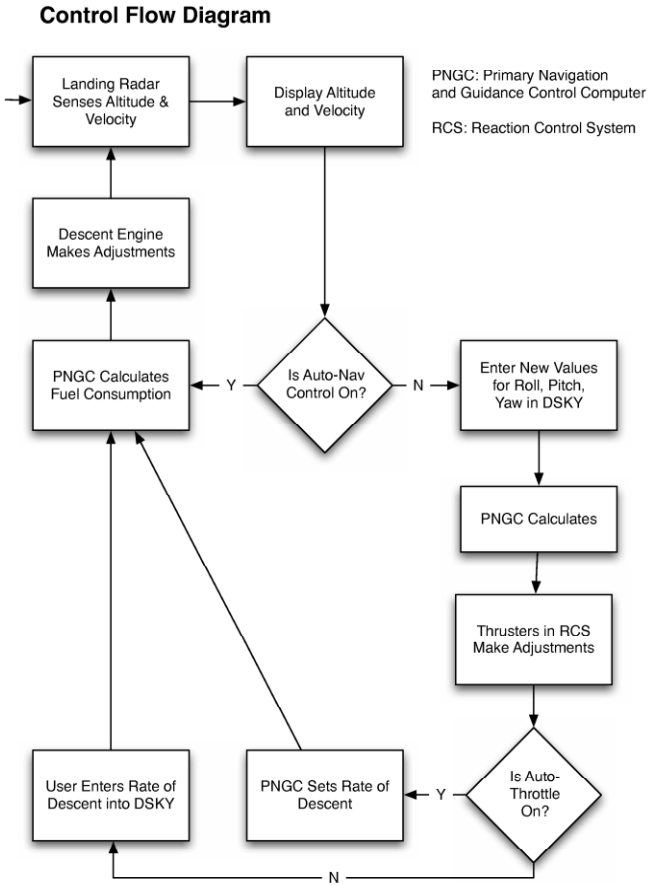
- Focuses on data flow between entities with no notion of control



Software Architecture: Foundations, Theory, and Practice; Richard N. Taylor, Nenad Medvidovic, and Eric M. Dashofy; (C) 2008 John Wiley & Sons, Inc. Reprinted with permission.

Operational Model: Control Flow Diagram

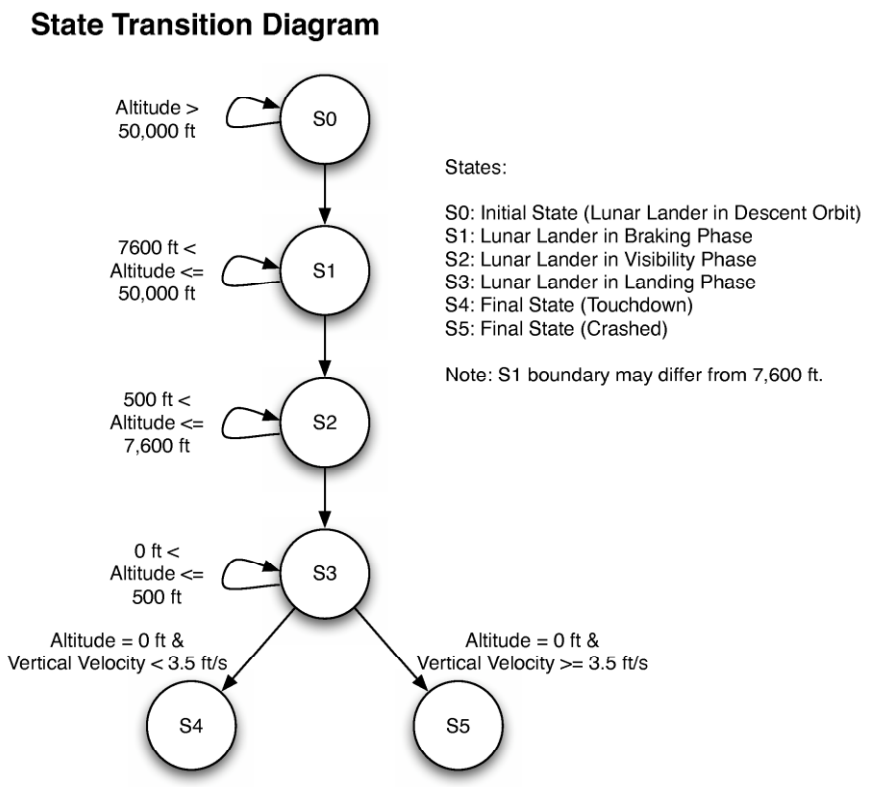
- Focuses on control flow between entities separate from data flow



Software Architecture: Foundations, Theory, and Practice; Richard N. Taylor, Nenad Medvidovic, and Eric M. Dashofy; (C) 2008 John Wiley & Sons, Inc. Reprinted with permission.

Operational Model: State Transition Diagram

- Focuses on states of systems and transitions between them
- Resembles UML state diagrams



Reference Requirements

- Mandatory
  - Must display the current status of the Lunar Lander (horizontal and vertical velocities, altitude, remaining fuel)
  - Must indicate points earned by player based on quality of landing
- Optional
  - May display time elapsed
- Variant
  - May have different levels of difficulty based on pilot experience (novice, expert, etc)
  - May have different types of input depending on whether
    - Auto Navigation is enabled
    - Auto Throttle is enabled
  - May have to land on different celestial bodies
    - Moon
    - Mars
    - Jupiter's moons
    - Asteroid

Representing Domain

By Bjorner

from Domains to Requirements by Dines Bjorner
http://www2.imm.dtu.dk/~dibj/2013book.pdf

Definition 1 Domain: *By a 'domain' we shall understand a set of 'entities': 'endurant's, that is, phenomena that "lasts", and 'perdurant's, that is actions, events and behaviours over endurants, where the focus (of such entities) is on domains created by humans, domains that are components of a 'societal infrastructure', and where, hence, these users interact, more-or-less casually with their domain, and these human users are not expected to excert greater technical skills in their interaction with the domain* ■

Definition 2 Domain Engineering: *By 'domain engineering' we shall understand the process of analysing and describing a domain, that is, of* determining *the range of* domain stake-holders, *of* gathering *and* analysing *information about the domain, of* describing *the domain, of* testing, checking and verifying *the evolving* domain description *and of* validating *that description* ■

Definition 3 Domain [Requirements] Stake-holder: *By 'domain stake-holder' ('requirements stake-holder') we shall understand a person, or a group of persons, "united" somehow in their common interest in, or dependency on the domain (requirements); or an institution, an enterprise, or a group of such, (again) characterised (and, again, loosely) by their common interest in, or dependency on the domain (requirements)* ■

Domain is characterized

- Entity
- Function/Action
- State
- Behavior
- Event
- Axiom
- Stakeholder

Definisi domain?

- Derajat abstraksi
- Himpunan atau anggota
- Transport atau bus ?
- Class: atribut+method;

  feature: characteristik+ability
- Di dunia: noun, verb, adjective/adverb

Latihan

Identifikasi feature dari:

- Domain tiket:

Bioskop	Sepak Bola	Kereta Api	Pesawat	....

- Domain pengelolaan stasiun transportasi umum:

Pelabuhan Kontainer	Bandar Udara	Terminal Bus	Stasiun KA	....

Entity
Function
Behavior
Event
Axiom
Stakeholder