Complexity, Emergence, & Cunefin Framework

- SW 아키텍처 요구사항 (ASRs)

2017. 7. 20

김 영온(Young On, Kim)

한국소프트웨어기술진흥협회

SW 아키텍처 요구사항

What are software requirements?

Needs vs requirements

Systems requirements vs software requirements

Software architecture requirements

- ASR(Architecturally Significant Requirements)

SW 아키텍처 요구사항

Why are ASRs concerned?

✓ Difficult to reflect if changed

✓ Big impacts on SW quality and productivity

Software architect must make up-front decisions before detailed design and coding to satisfy ASRs.

Why is software architecture difficult?

✓ Complicated vs. Complexity(Wicked Problems)

✓ Emergence

✓ Cunefin Framework

Complexity 복잡성

- **Disorganized Complexity – "Complicated"**
  - System with **many loosely coupled, disorganized and equal elements**
    - 온도와 압력처럼 어떤 평균적인 속성**(properties)**을 가지고 있음
  - **Statistical analysis techniques – Newton**

- **Organized Complexity – "Complexity"**
  - System with **many strongly coupled, organized and different elements**
    - **경제, 정치 또는 사회적 시스템**에 나타나는 것처럼 어떤 **파생 속성과 현상**을 가지고 있음
  - **Not described well** by traditional analysis techniques **– Darwin**

Guide of the Systems Engineering Body of Knowledge (SEBoK) v1.8 – Full version BKCASE, March 31, 2017

Complexity 복잡성

- **Complicated - Ordered systems**
  - **fixed relationships** between elements and are **not adaptable**.
    - **Watch - complicated,** with many elements **working together.**

- **Complexity - Chaos**
  - **State of disorder or unpredictability - not interconnected and behave randomly** with no adaptation or control.
    - **Simulations of chaotic systems** can be created and, with **increases in computing power, reasonable predictions of behavior** are possible at least some of the time.
    - **Complex systems** may **exhibit counter intuitive behavior** compared to that of more ordered systems.

Guide of the Systems Engineering Body of Knowledge (SEBoK) v1.8 – Full version BKCASE, March 31, 2017

Complexity 복잡성

- **사악한 문제 wicked problems**
  - 종종 인식하기 어려워서 **불완전하고, 모순적이고, 변하는 요구사항** 때문에 **해결하기 어렵거나 불가능한 문제**
  - "사악한 wicked" 용어 사용은 악 evil 이라기 보다는 **해결책에 대한 저항을 표현**하기 위하여 사용한다.
  - **복잡한 상호 의존성** 때문에 **사악한 문제의 한 측면을 해결하는 노력은 다른 문제를 찾거나 만들 수 있다**.

wikipedia 2016

Complexity 복잡성

- **Wicked problems - complexity**

Conklin이 Rittel & Webber의 사악한 문제(1973)를 계획과 정책 이외 영역으로 일반화

1. **문제는 해결책이 공식화된 후까지 이해할 수 없다.**
2. **사악한 문제는 답이 없다**(no stopping rule)
3. 사악한 문제 **해결책은 옳거나 틀리지도(right or wrong) 않다**

4. 모든 사악한 문제는 **본질적으로 새롭고 고유하다**
5. 사악한 문제에 대한 **모든 해결책은 'one shot operation'** 이다
6. 사악한 문제는 **주어진 대안 해결책이 없다**

wikipedia 2016

Complexity 복잡성

- **설계 문제(SW 요구사항?)이 사악한 이유**
  - 종종 잘못 정의되어 있거나, 선행적으로 기술되어 있지 않다.
  - 다양한 관점의 이해당사자가 참여한다.
  - 옳거나 최적의 해결책이 없다.

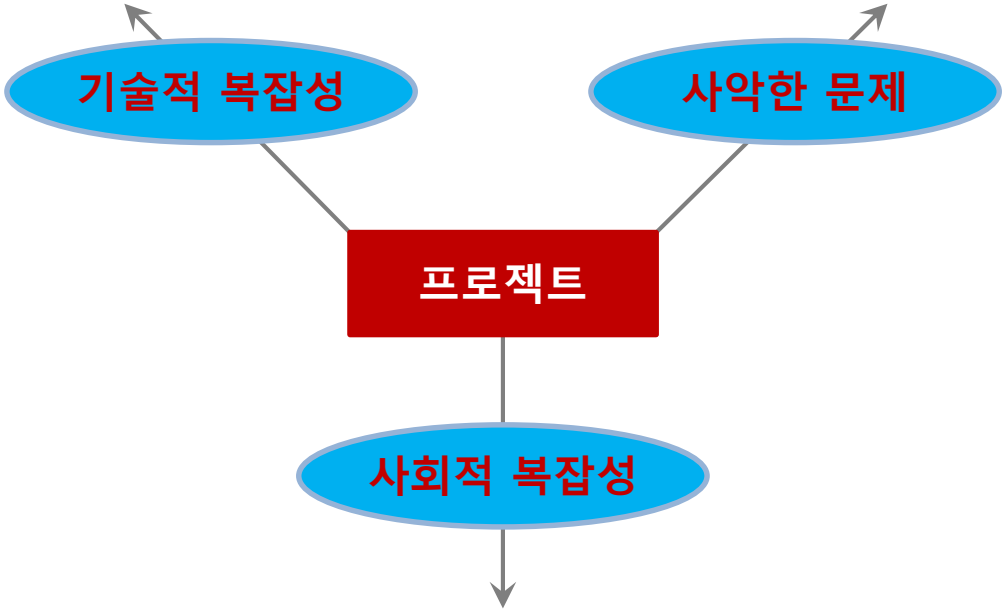
  따라서, 알려진 방법으로 해결할 수 없고, 독창적인 해결책이 필요하다.

- **사악한 문제 해결 전략**
  - 권한에 따른 Authoritative
  - 경쟁적인 Competitive
  - 협동적인 Collaborative

wikipedia 2016

Complexity 복잡성

- **프로젝트에 영향을 미치는 분열 요인**
  - **집단 지성**은 **자연스러운 협업 매개체**이지만,
  - **프로젝트**에는 집단 지성을 방해하여 협업이 어렵거나 불가능하게 하는 힘 **(분열의 요인 - 기술적/사회적 복잡성 & 사악한 문제)**이 있다



Conklin, Jeff; Wicked Problems & Social Complexity, Chapter 1 of Dialogue Mapping: Building Shared Understanding of Wicked Problems, Wiley, November 2005.

Emergence

- **Emergence**
  - Principle that entities exhibit properties which are **meaningful only when attributed to the whole, not to its parts**."
    - **Birds –** 개별 요소인 **깃털, 부리, 날개**는 **중력**을 이겨 낼 수 없으나, **제대로 연결**되면 **나는(flight) "emergent behavior"**를 만든다.

- **Types of Emergence**
  - **Simple emergence** is generated by the **combination of element properties and relationships – 상식적으로 누구나 알 수 있는**
  - **Weak emergence** as **expected emergence** which is **desired(or at least allowed for)** in the system – 전문가가 주의하면 알 수 있는
  - **Strong emergence** is used to describe **unexpected emergence** – simulation, test 또는 실제 운영에서만 발생

Guide of the Systems Engineering Body of Knowledge (SEBoK) v1.8 – Full version BKCASE, March 31, 2017

Emergence

- Appearance of **emergent properties** is the **single most distinguishing feature of complex systems.**
  - The **more complex** a system is, the **more difficult predicting its emergent properties** becomes.

- **Managing emergent properties** is **through iteration.**
  - **True hindsight and understanding** comes from **building multiple systems** of the same type and **deploying** them, then **observing** their **emergent behavior**.
    - **What works** (that is, what emergent behavior and side effects are desirable); and
    - **What does not work** (that is, what emergent behavior and side effects are undesirable).

Guide of the Systems Engineering Body of Knowledge (SEBoK) v1.8 – Full version BKCASE, March 31, 2017

Systems Problems

- Questions about the **nature of systems, organization, and complexity** are **not specific to the modern age.**

- Virtually **every important concept** that backs up the **key ideas emergent in systems** is **found in ancient literature** and in the centuries that follow.
  - 그러나 20세기 중반 부터 – **growing sense of a need for, and possibility of a scientific approach to problems of organization and complexity** in a “science of systems” per se.
  - These limitations, often expressed as **emergent evolution** or "**the whole is more than a sum of its parts**”.

Guide of the Systems Engineering Body of Knowledge (SEBoK) v1.8 – Full version BKCASE, March 31, 2017

Cynefin Framework

- **Cunefin Framework** - **Leader's Framework for Decision Making**

  - "Cynefin"은 웨일즈어, "**쿠-넵-인**"으로 발음
    - "우리에게 **영향을 미치는 환경과 경험**의 여러 가지 요소 들"을 의미함

  - **세계는 질서가 있다**고 가정하는 **뉴튼적 사고**(Newton approach)는 "**단순화"를 조장**하여, **상황 인식과 문제 해결을 어렵게** 할 수 있음

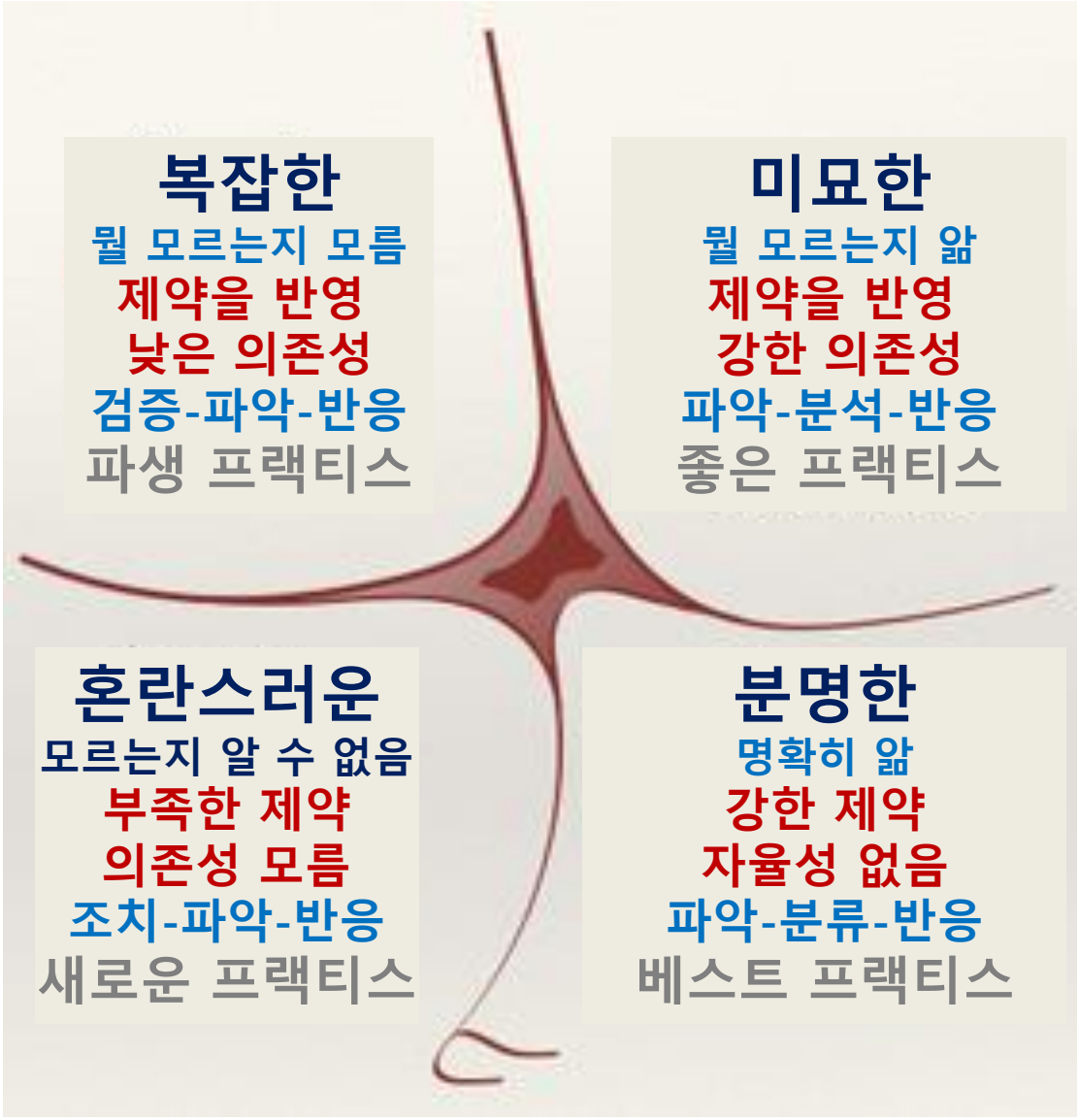
  - Good leadership is not a one-size-fits-all proposition'

A Leader's Framework for Decision Making, Dave Snowden, HBR 2007

https://en.wikipedia.org/wiki/Cynefin_Framework

Cynefin Framework

- Agile/Lean
- DevOps
- System thinking
- Design Thinking

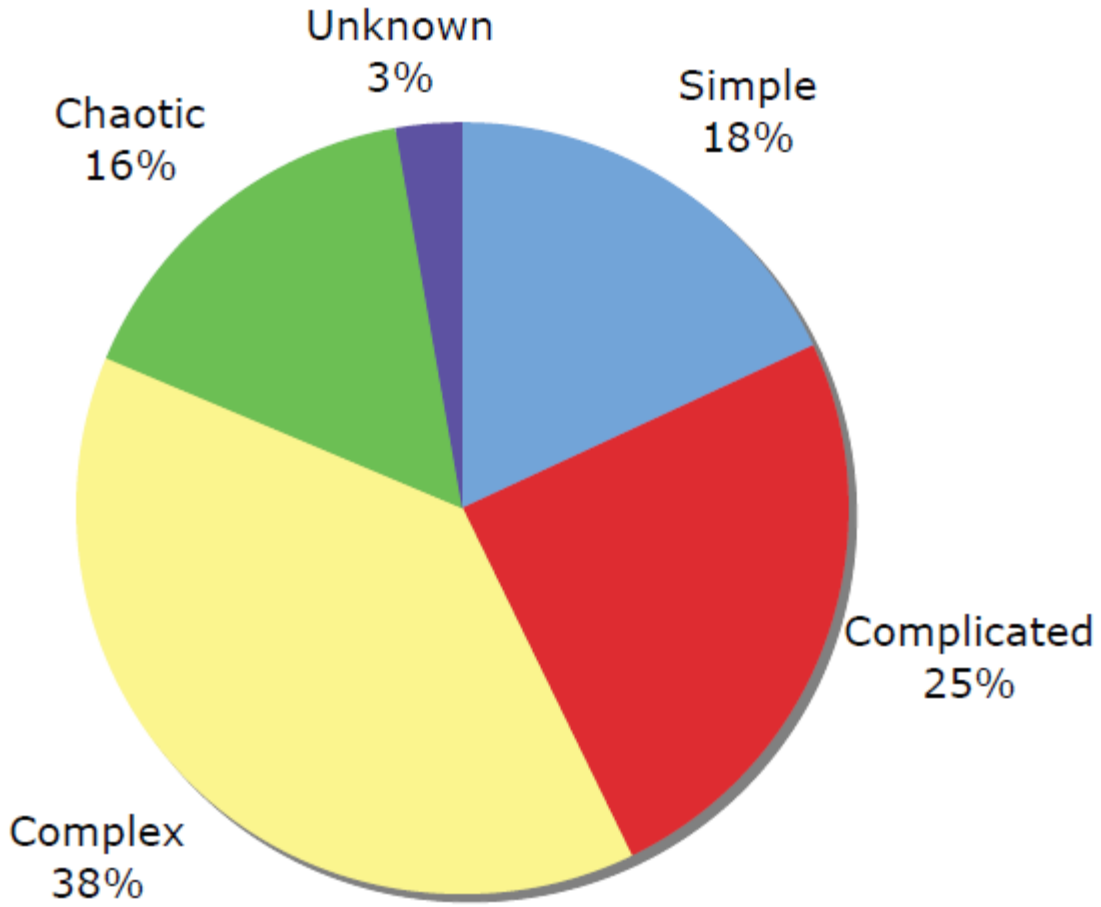


- Process Improvement
- TQM, CMMI
- Quality Assurance
- Iterative & Incremental
- Lean 6 Sigma

- Quality Control
- Waterfall
- Lean 6 Sigma

A Leader's Framework for Decision Making, Dave Snowden, HBR 2007

Cynefin Framework



On Understanding Software Agility—A Social Complexity Point Of View, Joseph Pelrine

Cynefin Framework - 인과관계

Complex - 인과 관계 파악이 어렵고 사후 관계 파악 가능

complex

C E

패턴
완성은 없음
이해당사자가 좋은 것을 결정

complicated

C----------->E

Complicated - 원인과 결과가 시공간 상 분리 되어 난해

분석
전문가 mindset
가능한 정답 군(range)

Disorder - 인과관계를 파악하지 못한 상태

Chaotic - 원인과 결과의 연관성이 없는 혼동 상태

chaotic

C≠E

조치
다른 업무로 전환
(Push into the other domains)

simple

C=E

Simple - 인과관계가 단순 반복적, 예측 가능

표준 운영 절차
베스트 프랙티스
하나 또는 몇가지 정답

C : Cause

E : Effect

https://www.youtube.com/watch?v=5mqNcs8mp74

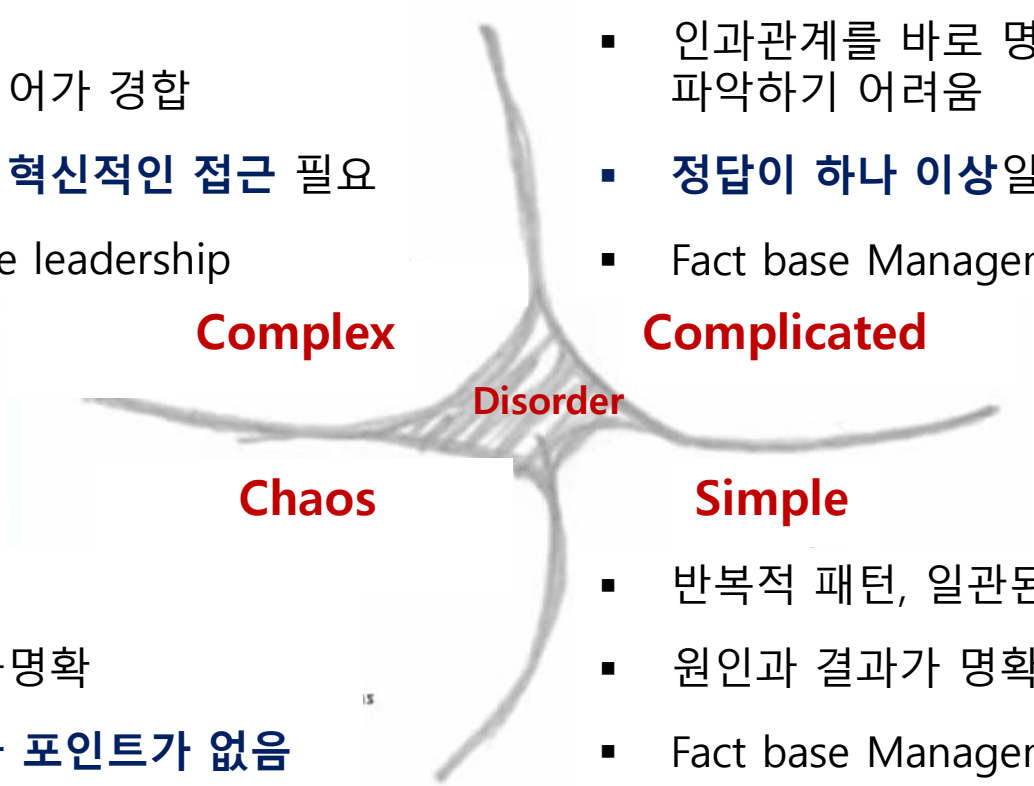
Cynefin Framework - 상황별 특징

- 유동적이고 예측이 어려움
- **정답이 없으나** 돌발적 교훈적 **패턴 있음**
- 많은 아이디어가 경합
- **창의적이고 혁신적인 접근** 필요
- Pattern base leadership

Complex

- 혼란스러움
- 인과관계 불명확
- **정답을 찾을 포인트가 없음**
- Pattern base leadership

Chaos



Complicated

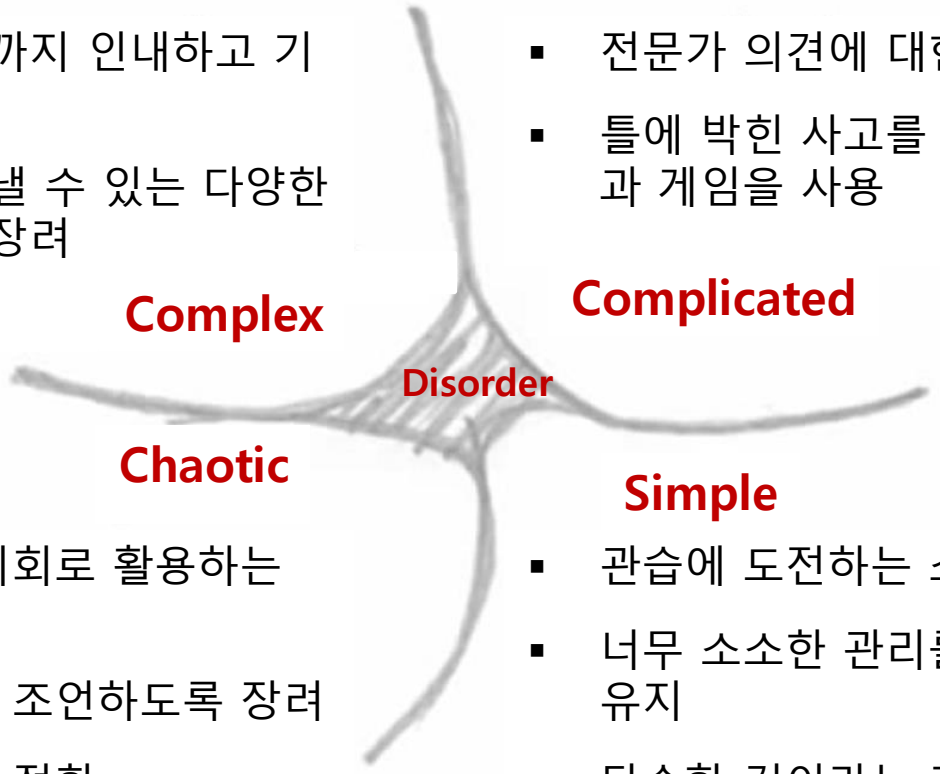
- **전문가 진단** 필요
- 인과관계를 바로 명확하게 파악하기 어려움
- **정답이 하나 이상**일 수 있음
- Fact base Management

Simple

- 반복적 패턴, 일관된 이벤트
- 원인과 결과가 명확 ; **정답 존재**
- Fact base Management

A Leader's Framework for Decision Making, Dave Snowden, HBR 2007

Cynefin Framework - 상황별 대응



- 반응이 올 때까지 인내하고 기다림
- 패턴을 찾아 낼 수 있는 다양한 접근법 찾기 장려

- 전문가 의견에 대한 도전을 장려
- 틀에 박힌 사고를 벗어나는 실험과 게임을 사용

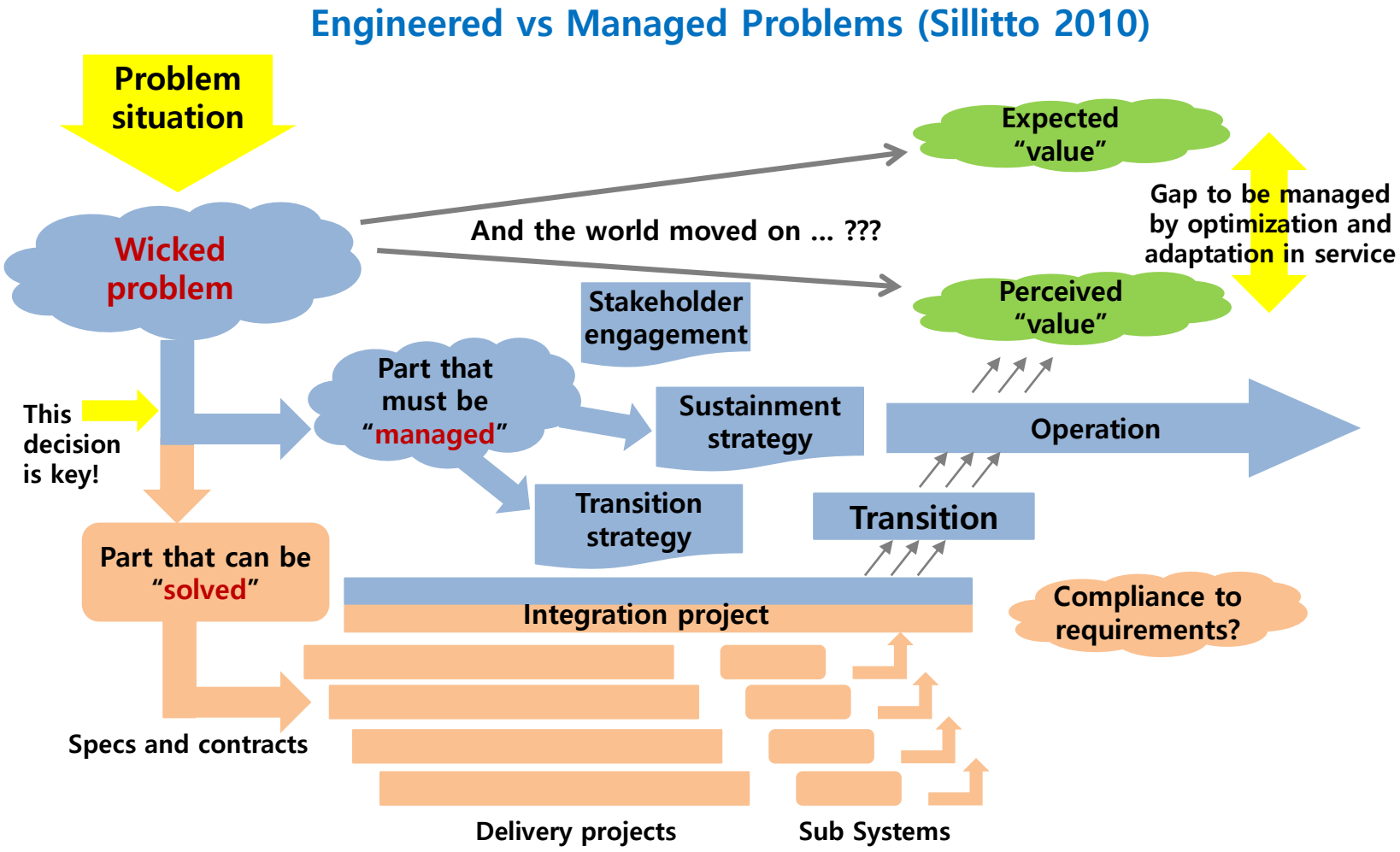
- 혼돈 상황을 기회로 활용하는 매커니즘 정립
- 리더의 견해에 조언하도록 장려
- 다른 상황으로 전환

- 관습에 도전하는 소통 채널 구축
- 너무 소소한 관리를 피하면서 관계 유지
- 단순할 것이라는 가정 피하기
- Best Practices의 가치와 한계를 동시에 인식

A Leader's Framework for Decision Making, Dave Snowden, HBR 2007

How to manage complexity in ASRs?

Balance among best, good, emergent & novel practices



Guide of the Systems Engineering Body of Knowledge (SEBoK) v1.8 – Full version BKCASE, March 31, 2017

How to manage complexity in ASRs?

Continuum of Project Life Cycle

Highly Predictive → **Highly Adaptive**

▪ **Requirements** are **mostly defined up-front** before product development begins	▪ **Requirements** are defined **iteratively and elaborated at periodic intervals** during the requirements process	▪ **Requirements** are **elaborated at frequent, recurring intervals for each iteration**
▪ **Risk and cost** are controlled by **detailed planning based on in-depth analysis** of requirements and constraints prior to development	▪ **Risk and cost** are **controlled by progressively detailed planning** based on timely specification of requirements and constraints during development	▪ **Risk and cost** are **fixed and controlled for each iteration**
▪ **Key stakeholders** are **involved as scheduled** milestone	▪ **Key stakeholders** are **involved at specified intervals**	▪ **Key stakeholders** are **continuously involved and actively engaged**
폭포수 **Obvious**	**반복/점진적** **Complicated**	**애자일/DevOps** **Complex**

Requirements Management: A Practice Guide, Project Management Institute, 2016

How to manage complexity in ASRs?

- **Agile & DevOps (Iterative & Incremental?), not waterfall**
  - Benchmarking, observations, prototyping & perpetual beta
- **Thinking, not thought**
  - **Systems Thinking**
    - 부분이 아닌 전체 관점에서 보다 좋은 답 찾기
  - **Design Thinking**
    - **공학(engineering)**에 **예술/공예(art/craft)** 덧 붙이기
    - **Empathize – Define – Ideate – Prototype – Test** : Stanford d.school
    - Balance between **divergent thinking** and **convergent thinking**

감사합니다.

yokim31@daum.net