Software Engineering Chap2

What is System?

* Purposeful collection set of inter-related components working together to achieve common objective ( common goal )
* 어떤 공통된 목표를 달성하기위해 함께 동작하는 상호 작용하는 컴포넌트들의 목표 지향적인 집합이다.

Sociotechnical system stack ( 7layer )

Society

Organization ( 조직의 policy )

Business process ( system의 business goal 을 맞추기 위한 Business process, 이 layer에 따라 다양한 application system이 개발된다 )

Application system

Communications and data management ( network )

Operating system

Equipment ( Hardware )

Complex system

* Software , hardware, people ( user )

Organization affects of a new system

* Process changes
* Job changes
* Organizational changes

Socio-technical system characteristics

* Emergent properties
  + 시스템 전체의 특성 ( 여러 컴포넌트가 통합되었을 때 나타나는 특징 )
  + Component 들의 relationship으로 나타나는 특성
* Non-deterministic
* Complex relationships with organizational objectives

Emergent properties

* Functional properties
* Non- functional properties

Reliability and Failure

* Hardware failure



* Software failure
  + Not a bathtub curve
* Operator failure

각각의 failure는 다른 failure들에도 영향을 미칠 수 있다.

Non-determinism ( 비 결정적 특성 )

* 같은 input은 같은 output을 도출한다 ( deterministic )
* 같은 input도 다른 output을 도출할 수 있다 ( Non – deterministic )

Success criteria

* Stakeholders ( 이해의 당사자 )
* Stakeholders 들의 스펙트럼이 넓을수록 목표 설정이 어려울 수 있다.

<System Engineering>

* Procuring, specifying, designing, implementing, validating
* Services, constraints, ways ( method )

System Engineering stages

* Procurement ( acquisition ) => 조달, 획득 ( 외주, 등 )
  + 목적, high-level system requirement, purchased, 등
* Development
* Operation

System의 사용목적이 다할 때 까지 위의 stages들을 넘나든다.

Security and dependability consideration

* Procurement decisions에 의해 Design option 이 제한될 수 있다.
* Human error들이 system에 의도하지 않은 오류를 만들 수 있다.
* 배포 (deployment) 이전에 testing이 충분하게 이루어지지 않아 오류를 발견하지 못할 수 있다.
* Configuration error는 배포 과정에서 발견될 수, 일어날 수 있다.
* Assumption 자체가 procurement 과정에서 무시되어질 수 있다.

Inter-disciplinary working

* Communication difficulties
* Differing assumption
* Professional boundaries

System procurement

* 특정 조직의 필요성에 맞추어 시스템, 시스템셋을 획득하는 것을 말한다.
* Scope of the system, System budget and timescales, High-level system requirement 에 대한 정보가 필요하다
* 위 정보들을 기반으로 조달, 개발, COTS ( 상용화되어 시장에서 구매 가능한 제품 ) 을 결정한다

Decision drivers

* Organization systems
* External regulations
* External competition
* Business re-organization
* Available budget

개발하는 것 보다 사는 것이 좋다.

Client의 요구사항이 adaptable 한가의 유무에 따라 구매 / 개발의 방향이 정해지게 된다.

System development

* Plan-driven approach
* Parallel development ( need of plan )
* Software가 Hardware보다 비용이 적게 들기 때문에 Software가 Hardware problem까지 커버할 수 있기를 기대한다.
* 다양성을 존중하며 일을 해야한다.

System requirement definition

* Abstract functional requirement
* System properties, Non – functional requirement ( Architecture 과정에서 반영되어야 하므로 반드시 미리 정립되어야 한다. )
* Undesirable characteristics

System design process

* Partitioning requirements => sub-system을 결정하는데 필요함
* Identify sub-systems
* Assign requirements to sub-systems
* Specify sub-system functionality
* Define sub-system interfaces ( Interaction에 대한 약속 ( data-flow ) 등에 대한 약속 )
* 설계와 Requirement 단계의 feedback은 초기관계에 매우 중요하다

Sub-system development

* Parallel project
* COTS or Development
* Lack of communication 이 system의 완성도에 큰 영향을 미칠 수 있다

System Integration

* Hardware, software, people
* 여러 subsystem을 통합하는 과정이 필요함
* 빅뱅 방식 ( 한날 한번에 전체를 통합 )
* Incrementally ( 점진적 통합 )
  + Incrementally integration은 Error, Problem 발견 시 어느 Subsystem의 문제인지 판별하기가 편리하다.
* Interface Problem detection

System delivery and deployment

* Install in the customer’s environment
* Development and dependability

System Operation

* Using the system for defined purpose
* Operational process 가 flexible 해야 한다.

Human Error

* Person approach Error
* Systems approach ( 조기에 막아야한다 )
  + 잘 만들어진 system은 system이 보안할 수 있는 방법을 제공하는 것
* 한번의 실수가 고장으로 연결되지 않는, 몇 개의 layer을 통해 차단할 수 있는 system이 좋은 system이다
* Multiple barriers ( Swiss cheese model for system failure )

System evolution

* 고객의 요구사항, 기술환경, 새로운 정책, 등의 여러 외부요인들에 맞춰 system을 evolution 해야 한다
* 일반적으로 highly costly
* Technical, business에 대한 검토가 필요하다.
* Sub-system interact을 모두 파악, 통제는 쉽지가 않다. 끊임없는 유지 보수는 interface를 망가뜨려 예측하지 못한 문제들을 야기할 수 있다
* 새로운 유지보수를 할 때 이유에 대한 근거와 자료가 필요하다 => 많은 시간이 필요
* 기존의 System structure가 유지보수를 위해 망가져 갈 수 있다.
* Legacy system ( Existing system중 새로 만들어진 system들과 계속하여 같이 동작해야 하는 system을 뜻함 )
* Evolution and dependability