# 1. Information systems and complexity

**1.1 What is the difference between complicated and complex systems?**
Complicated systems contain parts where the sum of the parts equals the system. This could be a bike or a computer. The system might be complicated but you can understand what the system is going to do based on the parts.

Complex systems contain parts where the behavior of the parts is hard to predict, making the system seemingly impossible to understand fully. The complexity tends to be introduced by the number, the types and the speed of change of integrations, components, and systems.

**1.2 What do we mean by socio-technical systems? Does this focus in any way help us understand how information systems function, fail, succeed and evolve?**
Socio-technical systems include the technology as well as the users, organizations, institutions, laws, and regulations that are related to the system. By focusing on the social-technical aspect, we might find observations that we otherwise might have missed if we only looked at the technical aspect of the system.

As users aren't as deterministic and rational as many software developers would have liked we do experience in the real world that the systems fail and are hard to utilize by its intended users. The socio-technical focus recognizes these challenges and gives us a way to identify, describe and handle these issues.

**1.3 Do you have any examples of information systems?**
Information systems have existed for a long time. Some of the earliest examples of information systems could be cave paintings, where someone would paint information on the wall and show it to others. As long as it is a system of information, it could be an information system. More modern examples include newspapers, books, and computers.

**1.4 What is the role of standards in information systems?**
Be quantifiable metrics to which parties in the information system adhere for purposes of allowing some common ground for interchange. This makes it possible for the information system to communicate its information in a systematic manner.

**1.5 If we define information systems as socio-technical: can you give an example of an organization that does not currently have any information systems?**
If we coin IS to be socio-technical we have expanded our view beyond what is traditional digital information technology. With the socio-technical worldview we would incorporate all work practices and consolations of cooperating people in the organization. Thus it is not possible to have a organization without any information system as the organization in itself is would fit the description of a Information System.

**1.6 If you were to introduce a new software system into a large organization with**

**already existing information systems and practices: are there any possible challenges?**
The main challenge people face is cultivating the installed base. The transitions have a large impact on the installed base in what we call a installed base hostile way. Thus you would need to gain acceptance by the IB and take one step at the time.

**1.7 What do we mean by governance and architecture?**
Architecture is the conceptual blueprint we use to describe a technical solution. In the blueprints, we describe the components of the complex systems, what they do and how they interact with the rest of the system. The architecture is thus a high-level description of the systems building blocks.  When it comes to different architectural configurations it varies between two extremes. Perfectly modular and perfectly monolithic, but most instances fall somewhere along this continuum.

Governance is a central aspect to any organization and it broadly refers to who decides what. In a platform oriented world, governance is manifested in mainly three ways. How decision rights are divided between the owner and the app developers. What formal and informal control mechanisms are used by the platform owner. This could be exemplified thy gatekeeping, performance metrics, defined processes, and informal clannish pressure. Lastly; the pricing structure of the platform and what side gets subsidized.

The governance of the platform impacts the evolutionary dynamics in the ecosystem, and the competitive advantages generated by governance choices can strengthen or diminish with the choice of the platform's architecture.

**1.8 What is modularization?**
Software modularity:
Modularity of a platform ecosystem refers to the degree to which the platform and apps can be designed, implemented, operated, and altered independent of each other (de Weck et al., 2011, p. 188). Modularity in software systems is a property that can reside anywhere along a continuum ranging from perfectly monolithic to perfectly modular. A perfectly monolithic system is one where every app is highly interdependent with the platform. An example of a monolithic architecture is one where a platform and an app are integrated into a single system (e.g., the Mail app in iOS). (Tiwana)

Modularization
Modularization entails minimizing dependence between the platform and apps but maximizing it within each of them (Ethiraj and Levinthal, 2004a). Modularity, however, is not an absolute property of a platform ecosystem. A platform can intentionally be designed to be more or less modular. Platform modularization is a necessary but insufficient precondition for app modularization.

## 2. Platform Ecosystems – fundamental concepts

**2.1 Tiwana (2013) and Baldwin & Woodard (2008) both define platforms. How do they define them? And are there any differences in their definitions?**

Baldwin & Woodard (2008 (i teksten står det 2009 :o)):
In Essence, a "platform architecture" is a modularization that partitions the system into (1) a set of components whose design is stable and (2) a complementary set of components which are allowed - indeed encouraged - to vary. The combination of stability and variety is accomplished via "stable, yet versatile' interfaces, which govern the interaction of components. The interface specifications are part of the platform; indeed they may be the only components that remain truly stable over a long time. The combination of stability and variety in the architecture makes it possible to create novelty without developing a whole new system from scratch. Thus platform systems are evolvable.

Tiwana (2013):
The extensible codebase of a software-based system that provides core functionality shared by apps that interoperate with it, and the interfaces through which they interoperate.
[...]
The platform's value to a user depends on the number of adopters on the other side. Our focus here is on multisided platforms rather than one-sided platforms, which we do not consider true platforms at all. Instead, they are products or services often confused or mislabeled as platforms

- Their differences is that Tiwana (2013) describes a platform as multi-sided, whereas Baldwin & Woodard (2008) does not explicitly describe it as such.

**2.2 Mention some examples of software platform ecosystems. Select one to describe the characteristics that make it a platform.**

A platform ecosystem refer to the platform core and the apps that interoperate with it.(magl 2018)
Entur er en plattform fordi den har en kjerne med data, informasjon og apper men har også boundary resources i dette tilfellet et api som lar en tredjepart koble seg til og utvikle videre på det som allerede er lagt til grunn.

**2.3 What is a boundary resource? What role do these play in platform ecosystems?**

Boundary resources beskrives i Ghazawneh og Henfridsson (2013) som de verktøy og reguleringer som fungerer som grensesntittet som kobler app utviklere og platformeieren sammen. I praksis blir dette både en generativ mekanisme, men også en regulerende faktor. En finner gode eksempler på BRs i API, SDK, lissensiering osv…

*"boundary resources, i.e. the software tools and regulations that serve as the interface for the arm's-length relationship between the platform owner and the application developer."*

**2.4 What do we mean by multi-sidedness?**
We usually mean how multi-sided a platform is. A multi-sided platform is an intermediary platform having two distinct user groups that provide each other with network benefits. The organization that creates value primarily by enabling direct interactions between two (or more) distinct types of affiliated users is called a multi-sided platform. Multi-sided platforms represent a refinement of the concept of network effects.

**2.5 Are there any advantages of platform architectures?**

- Architectures can reduce complexity by partitioning sub-systems and make the different components less dependent on each other.
- A powerful way to reduce this risk is by lowering the dependencies between the contributions of the three parties in this example. Reducing dependencies reduces interactions among them, hence reducing the structural complexity of the system in Figure 5.5. Architecture is a way to reduce such dependencies among subsystems that constitute a complex system. Thoughtful architectures however are not a silver bullet: They cannot eliminate complexity but can make it more manageable (Tiwana, 2013).
- The more complex a man-made object gets, the harder it becomes to comprehend for any one individual. This incomprehensibility can become the showstopper in a platform's evolvability (Tiwana, 2013).
- A good architecture must exhibit four simple properties that it shares with the architecture of modern cities: simplicity, resilience, maintainability, and evolvability (Tiwana, 2013).

**2.6 What is the role of interfaces in a platform ecosystem? Why do we say that they need the be relatively stable?**

- Interfaces are used for cultivating platform ecosystems through third-party development (Ghazawneh & Henfridsson, 2013).
- They need to be relatively stable since they can be used by many different third-party developers. If the core changes, all the applications connected to the system have to be updated or would fail.

## 3. Design

**3.1 Name some activities that typically are included when designing a software application.**
Mapping, problem definition, solution ideation, prototyping, implementation, testing, maintenance.
prototyping, data collection and more??
**3.2 What benefits are associated with user engagement in software development?**

Helping the designers getting a more complete view of the users real life thus their needs and requirements. This will help develop a more relevant system/application for the end-users.

**3.3 Are there any challenges related to creating locally relevant applications in large information systems that are used by a heterogeneous group of users?**
Yes. Assumptions made by the developers might not be true for users that the developers have no information on. A design choice that is valid locally, may not be valid for non-locally. These differences can be hard to spot without involving representatives from the whole user-base. Also making something as generic as serving a large and heterogeneous user group would entail making it based on best case practices. These are divided out of the entirety of the vastly diverse group and can result in a state that one size fits none. Rolland and Monteiro have an article (2002) on balancing the local and the global.

**3.4 What advantages might platform architectures have in relation to the design of locally relevant applications and user interfaces?**
Platform architectures might modularize the design of the locally relevant application, thereby reducing complexity, and opening up for evolvability if the app is ever to be used globally.

**3.5 What do we mean by scaffolding and boundary spanning?**
Scaffolding is a loose term used to describe a structure around the artifact that provides support. In a platform context, this could be the API documentation, boundary resources, customization tools, and other abstractions. In DHIS there are also academies and coursing that can fit under the term of scaffolds.

A boundary spanner is something that traverses the regular boundaries between a developer and a platform owner. Yet again, in DHIS2 terms a master student or any other from UiO that takes a field trip, doing action research in an implementing context can work as this spanner bridging two otherwise separate entities.

## 4. Programming

**4.1 Provide a definition of: front-end development, client-side development, back-end development, and server-side development.**

- Front-end development contains everything that the user can see and interact with.
- Back-end development contains everything the user cannot directly see, such as API calls, database queries, and computations.
- Client-side and server-side development describe where the application is being processed.

**4.2 What are the role of HTML, CSS, and JavaScript in front-end web development?**
HyperTextMarkupLanguage provides the structural components for websites. Cascading Style Sheets is WWW makeup, and thus used for styling. JavaScript is considered behavioral.

**is a JavaScript framework, and why do we use them? Name three examples and explain the differences between these.**

1. React is a JS framework made by facebook
2. vue.js
3. angular by google

they are all pretty similar.

JavaScript frameworks are used to make web development faster and easier. They provide built-in functionality for common aspects of development so that we do not have to built everything from scratch.

**4.4 What is an Application programming interface (API)? Why is it relevant to software platforms?**
An Application Programming Interface is an abstract grensesnitt that affords the opportunity to communicate with a module or a system. The API regulates what and how you can interact with the underlying software. GETPUTPOST-MANNEN

**4.5 What is REST?**
It is the architectural style that explains the quality attributes of the World Wide Web, seen as an open, distributed and decentralized hypermedia application, which has scaled from a few Web pages in 1990 up to billions of addressable Web resources today (putasso, 2013).

## 5. Innovation

**5.1 What is digital innovation?**
Innovasjon med 1 og 0
Guess: Vi lever i en tid hvor teknologi er over alt og lett tilgjengelig, digital innovasjon omhandler det å kombinere eksisterende teknologier og pushe ut nye innovative produkter på bakgrunn av den teknologiske kombinasjonen.
The creative combination of social and technical elements in order to create new services.

**5.2 Are there any characteristics of software platforms that might promote innovation?**
Boundary resources
Løse koblinger skaper generativitet i Light og Heavy IT.
Plattformer tillater og encourages en løs kobling mellom Light apper og Heavygreier. Både teknologisk og organisatorisk, slik Bygstad (2017) vil. Vet ikke med standarder…
"The fundamental properties of digital technology are reprogrammability and data homogenization. Together, they provide an environment of open and flexible affordances that are used in creating innovations characterized by *convergence* and *generativity*."
New innovations not intended or anticipated by the initial creators.

# Konsepter

**Multi-sidedness**
The need to attract at least two different and mutual beneficial groups to communicate and interact thru a platform.

**Envelopment**
Describes when a platform begins to offer the functionality of another platform in an adjacent market in addition to its existing bundle of functionality. It sorts of swallows the latter. iOS for instance.

**Multihoming**
Multihoming is the practice of connecting a host or a computer network to more than one network. This can be done in order to increase reliability or performance.

**Tipping**
A network effect only kicks in after a certain threshold has been reached. A platforms critical mass or tipping point is referring to when this phenomenon kicks in. After this point, we can see a noticeable and potential self-reinforcing feedback loop. A platform would, therefore, require a vastly different strategy before and after reaching the tipping point.

**Lock-ins**
Technology lock-in is a form of economic path dependence whereby the market selects a technological standard and because of network effects the market gets locked-in or stuck with that standard even though market participants may be better off with an alternative.

**Competitive durability**
(Økonomi) Durability of competitive advantage determines how long the competitive advantage can be sustained and is considered in terms of the ability of competitors to imitate through gaining access to the resources on which the competitive advantage is built.

**Platform envelopment**
Platform envelopment refers to one platform provider moving into another one's market, combining its own functionality with the target's, to form a multi-platform bundle.

**CRUD**
Create, read, update, delete

**Scaffolding**
- You can't find it in nature.
- It is some kind of lens we can use.
- The social setup around the implementation of the platform.
- In HISP they have focused on creating local knowledge for easier maintainability.
- We build some kind of scaffolding initiative around the information system.

- Difference between systems and houses. Information systems are never finished, so the scaffolding is never removed.
- To keep developing the platform in the right way, HISP uses the scaffolding that exists, different nodes around the world are feeding the experiences back to the system. Email lists, workshops.
- Everything that supports the systems, this could be the documentation for the system e.g.

**Drivers**
What drives things towards platforms.

**Multi-sided network**
- A two-sided market, also called a two-sided network, is an intermediary economic platform having two distinct user groups that provide each other with network benefits. The organization that creates value primarily by enabling direct interactions between two (or more) distinct types of affiliated customers is called a multi-sided platform (MSP).
- Two-sided markets represent a refinement of the concept of network effects.

**Network effects**
The positive effect described in economics and business that an additional user of a good or service has on the value of that product to others. When a network effect is present, the value of a product or service increases according to the number of others using it

**Bootstrap problem**
A problem with user adaptation if most of a platform or applications value lies within having many users. Its is hard to gain new users if there is no users already there.

**Boundary resources**
- API
- Laws and regulations

**Generative**
Be able to take components and recombine them to create something new with them.
The ability of technical and social elements to interact and recombine to produce or expand new solutions.

**Heavyweight: Could be the core of the systems.**
Denotes the well-established knowledge regime of large systems, developing ever more sophisticated solutions through advanced integration.

**Lightweight: Innovate on users needs, these could be the apps.**
A term used on new knowledge regimes of mobile apps, sensors, IoT and such. The differentiator is the cost and availability of the tech, and that implementing and deploying

such new arrangements could largely be done without the need of a dedicated IT department.
“a socio-technical knowledge regime, driven by competent users’ need for solutions, enabled by the consumerization of digital technology, and realized through innovation processes.”

Information systems

- Systems that makes it possible to interact with information.

Complexity
Socio-technical
Standards
Architecture
Governance
Modularization
Interfaces
Decoupling

**Ecosystem**
“An ecosystem is a community of living organisms in conjunction with the nonliving components of their environment (things like air, water and mineral soil), interacting as a system.” - Urelatert internett definisjon på biologiske økosystemer.

**Consumerization**
“Bring your own device” mine enheter og programvarer funker og er mer effektive enn de jobben min har/tilbyr.

**RE**presentational
**S**tate
**T**ransfer
Arkitektonisk stil en kan benytte seg av når en designer Web APIer, som beskriver den ‘korrekte’ bruken av HTTP protokollen. Finnes i 4 nivå, 0 gjennom 3. Ikke REST, Flere URI, GET PUT POST DELETE, Hypermedia.

Simple, easy to describe
Resilient
Evolvable
Dependable
Reduces structural complexity
Maintainable, steady interfaces