# Systems Design

Alex Sverdlov
alex@theparticle.com

## 1 Introduction

It is often easier to solve big problems by decomposing them into smaller problems. We will refer to such aggregations of components to solve big business problems as *systems*.

Decomposing can be accomplished in different ways. The most intuitive one is hierarchical—the system is the overall generalization of the solution—while individual components are specialized components to address specific needs.

While modeling things at this level is itself complex, a lot more challenges arise when modeling interactions between components. There are only so many ways that things may be related to each other—but the interactions between things can be a whole order of magnitude more complex.

There are several ways of approaching this system building process. Automating or enhancing existing business or IT functions, or starting from scratch—by concentrating on business processes, and automating those using technology.

In either case, the process often starts by defining the *scope*. This drives the size of the overall effort, as well as what modules/components get built or automated.

Scope lays out the landscape of things to be defined, constructed, and deployed. These include:

1. System boundary: Things are either part of the system or not. Often this indicates what is under our control, or what should be assumed as part of the environment.
2. Environment: Assumptions about the environment our system will operate in.
3. Inputs: The inputs the system consumes.
4. Outputs: The outputs the system produces.
5. Components: Activities or processes within the system. These components could themselves be systems (sub-systems), etc.
6. Interfaces: How components interact, and how the system interacts with the environment.
7. Storage: Where and how the system stores temporary and permanent data.

# 2 The Writeup

When approaching a complex project or a business system, it is often helpful to write down major components. A short essay/narrative that outlines what the business does, who the customers are, how customers interact with the business, how purchasing works, how delivery works, how products get built, etc.: an executive summary of what the overall picture is.

If the description itself is hard to put down on paper, then write down a scenario—a customer walks in and wants to purchase product $X$, what happens? A few of such scenarios will serve as a guide on what is being built.

In this writeup, we can underline nouns and verbs. Nouns often become components, or entities in the database (or properties of those entities), and verbs become interactions between those entities. The goal is to come up with a functional view of what is happening within an organization.

## 2.1 Existing Processes

It is important in this writeup process to distinguish between legacy processes, and core business processes. For example, the creation of an invoice: businesses use invoices to bill their customers, who often respond with a payment. If modeling a legacy process, then there is an obvious need for an invoice (perhaps an electronic one)—but if we are starting from scratch, perhaps the invoicing capability should not be the primary way of documenting a financial transaction (perhaps payments can be automatic upon shipment, or receipt of product?).

By modeling things using just the core processes we can avoid re-creating unnecessary components, and perhaps streamlining operations.

# 3 Decomposing Perspectives

Decomposing problems into smaller ones has already been mentioned. Something that may not be obvious is the need to view problems from different perspectives. View the situation from the auditor perspective, or the customer's perspective, or the supplier perspective. Perhaps the original perspective is distorting some facts that need to be considered.

For example, when organizing data, what use case gets priority on access: many systems consume data from multiple sources—often the timestamp of when the record is processed is not the same as when the record was generated. For some use cases, it would make sense to organize data by the processed timestamp, in other scenarios, re-sorting the data on the time when the record was generated makes more sense. Both scenarios cannot be easily accommodated with a single dataset.

The other decomposition needs to be temporal. We need to subdivide the system not just by components, but also in time when things are built. What gets built first, and what gets the benefit of revisions is critical—things that are built are often much harder to alter: and every system ever built discovers new understandings of the problem domain.

Another large part of this decomposition idea is to decouple the components as much as possible—to the extent that a component may be replaced or carved out of the system without impacting other pieces.

# 4 Businesses as systems

A business is a system—not a technological system, but still a system. Businesses have components, etc. So designing a system is often just as complicated as building an efficient business.

Just like with a computer system, there are ways of restructuring the business: organize things around outcomes, not tasks (tasks are often legacy things that get done for no reason), assign responsibility to business units that use those outputs of processes—eliminating middle-man groups, ensure data gets recorded at the time of generation, etc.

# 5 Controls

Systems needs controls. Both access/security controls, as well as auditing oversight.