Conway’s Law:
The Structure of Products, Processes, and Organizations
Related reading: Herbsleb & Grinter, “Conway’s Law Revisited”

What do we mean by “law”?
An observed regularity, with explainable causes and consequences
Like Moore’s law, not like the laws of physics
A summary of complex phenomena that helps us think about practical problems
Like: How should I divide this project between the team in Ontario and the team in Milano?
Like: Which interfaces should I worry most about?

Organization Structure
Organizations are hierarchically structured
There are interfaces (interactions) between their parts
This includes organizations that design and build software systems

(Software) System Structure
Systems are hierarchically structured
There are interfaces (interactions) between their parts
Architectural design (“software architecture”) comprises several relations; “part of” and “uses” are two of them
“Uses” relation

Module A “uses” module B:
Functioning of A depends on functioning of B
• Actually a little more subtle: Feature a of A may depend on feature b of B. This is important in project planning (what must be in version v?) and in constructing families of related software systems
It is the “uses” relation that must reflect communication channels in an organization

The “responsible for” relation

Each module or subsystem is assigned to some part of the organization
One part of the organization may be responsible for zero or more modules

Conway’s observation: System structure reflects organization

Suppose we violated the law ...

Little or no interaction between us ...
... but our modules share an interface. So who do I call when it doesn’t work?
Information Hiding and Conway’s Law

Information hiding says:
- each module holds a design secret
  “Secret” may change without affecting other modules
- Abstract interface does not expose the secret
Translation: We have to talk about the interface, but not module internals

Abstract interface examples

TCP/IP protocol suite
  Abstracts over possible transport layers: wi-fi, ethernet, others
DOM interface to XML structure
  Abstracts over different implementation packages from Apache, Sun, others
java.sql
  Abstracts over several choices of relational database

What do we mean by “abstract”?

Abstract does not mean “vague” or “formal”
Abstract does mean: can be represented (or implemented) by different concrete examples
  - An “abstraction” reveals what is common among its possible instances, and hides their differences
  - And we ought to be able to describe those alternatives

Process and Conway’s Law

The build order of a project should allow everyone to make and measure progress
  Avoid making some wait for others
Key milestones are in terms of available functionality at interfaces
Distributed Development and Conway’s Law

Distributed teams magnify all the issues
- Even more important for an interface to be stable, abstract, well-documented
- But there will be questions, bugs, maybe even changes; module interfaces are still human interfaces

Brazilian team part

Distance and differences in language, culture, time zone

Irish team part

Summary

Conway’s law: The structure of a system reflects the structure of the organization that designed it
- Because: Module interfaces are human interfaces. They are what we have to talk about.

Consequences
- The key interfaces for architectural design, project planning and monitoring, and change control are those between different parts of an organization. They should be few and simple, the most stable and the most carefully designed and documented.

Read more at ...

