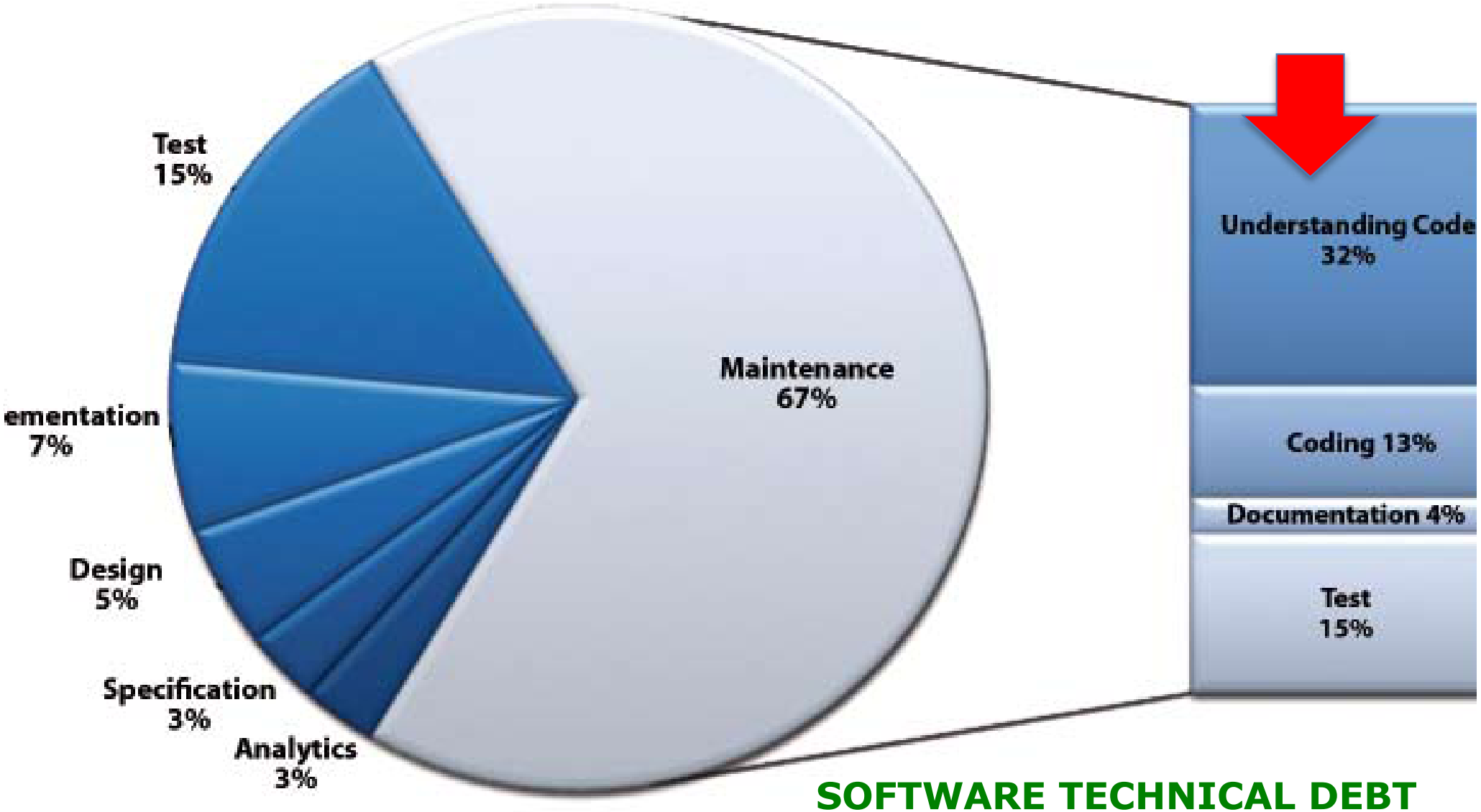
Modular Design

Simanta Mitra

% Time spent during life of software



SOFTWARE TECHNICAL DEBT

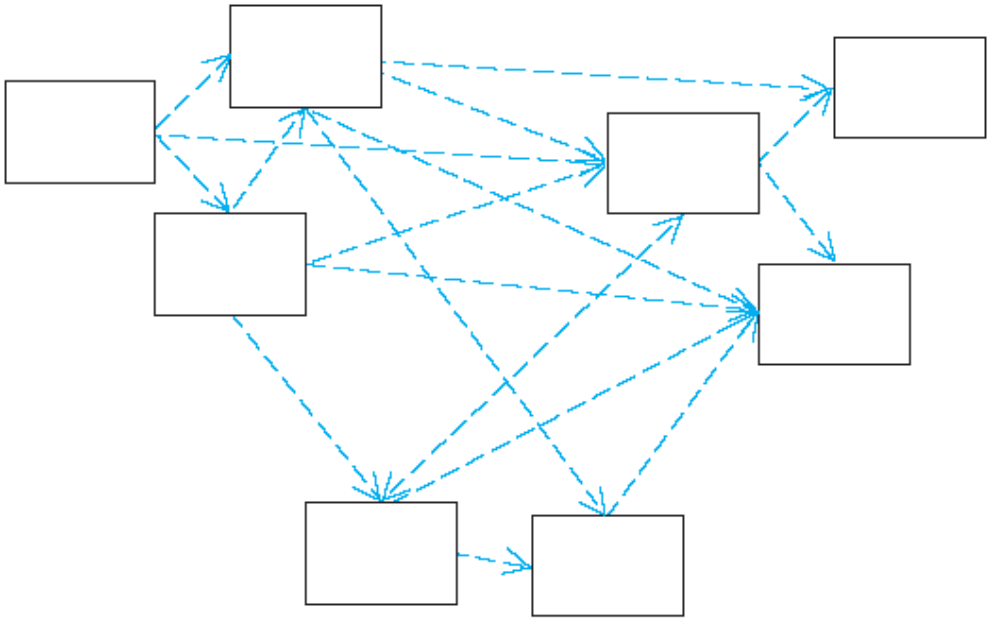
Modules

A module is an implementation unit that provides a coherent set of responsibilities

- typically package (or folder or namespace)

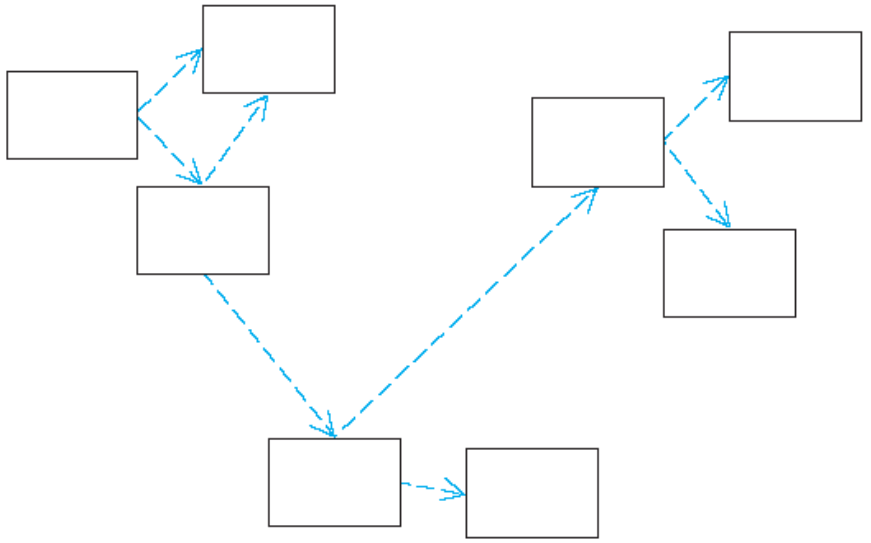
A static (not run-time) construct. Here we are not talking about organization of parts INSIDE a component

Example



High coupling

Lots of interconnections



Low coupling

Few interconnections

COUPLING

Coupling is a measure of how modules are interconnected.

High coupling means lots of interconnections (BAD)

Low coupling means not many interconnections (GOOD)

Why high coupling is bad!

1. If you change one module, that usually forces **a ripple effect of changes** in other modules.
2. Assembly of modules might require more effort and/or time due to the increased inter-module dependency.
3. A particular module might be harder to **reuse** and/or **test** because dependent modules must be included. A bug in one module affects working of other modules.

COHESION

Cohesion is a measure of how strongly-related or focused the responsibilities of a single module are.

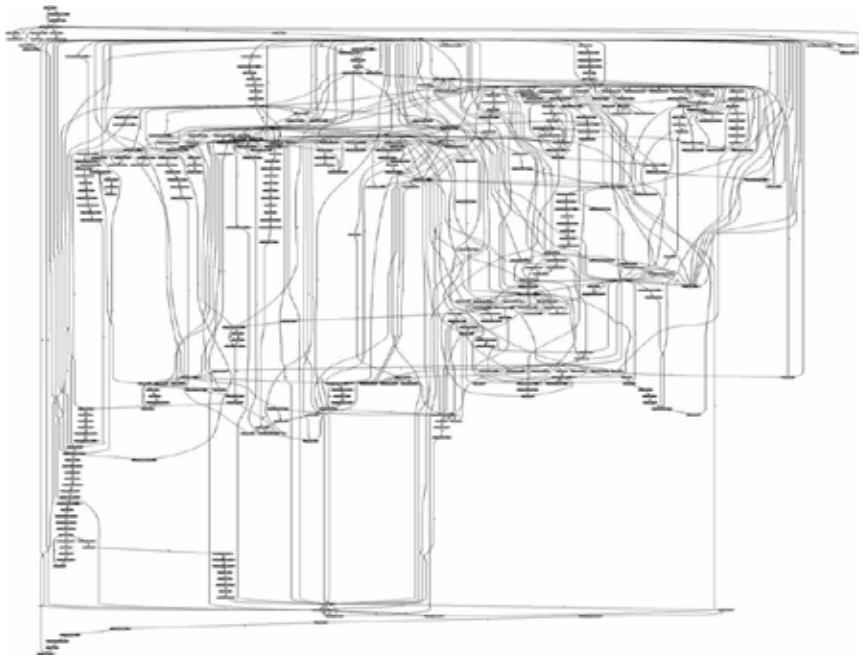
An element with low cohesion does many unrelated things

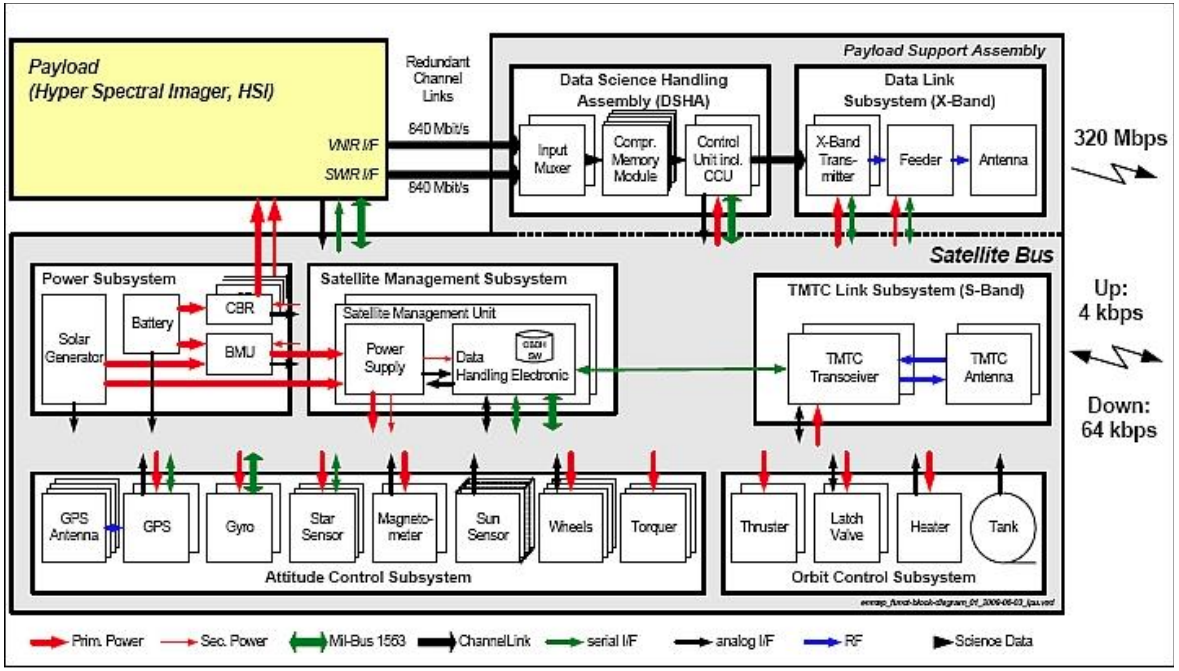
Why low cohesion is bad!

- Increased difficulty in understanding modules.
- Increased difficulty in maintaining a system, because logical changes in the domain affect multiple modules, and because changes in one module require changes in related modules.
- Increased difficulty in reusing a module because most applications won't need the random set of operations provided by a module

Modular Design

- Modular design means High Cohesion
  - Each module has clear and related responsibilities
- Modular design means Low coupling
  - Small number of interconnections between subsystems (i.e. Low coupling)
  - Few or no crisscrossing connections





Modular design = Clearly separated subsystems/modules (BOXES) so that modules have boundaries.

In Software – you need to forcibly create concrete boundaries by creating packages, interfaces etc etc.

IMPORTANCE OF MODULAR DESIGN

1. Build on budget!

- Because you can divide up the development and testing work.

- Because each person need not wait for other parts to be done before starting. Because each person does not have to keep asking others how they would need to interact with their parts (save on lots of communication time) – if interfaces are well defined.

- Because – less time will be spent on fixing bugs.

2. Build Maintainable systems

- Because it is easier to pinpoint the cause of bugs to a module and then focus and fix the module.

- Because it is easier to now isolate and test a module before integrating.

- Because it is easier to understand the system as a whole and to identify how a new feature can be added.

3. Build Reliable systems

- Because you can understand the whole design and identify flaws and fix them

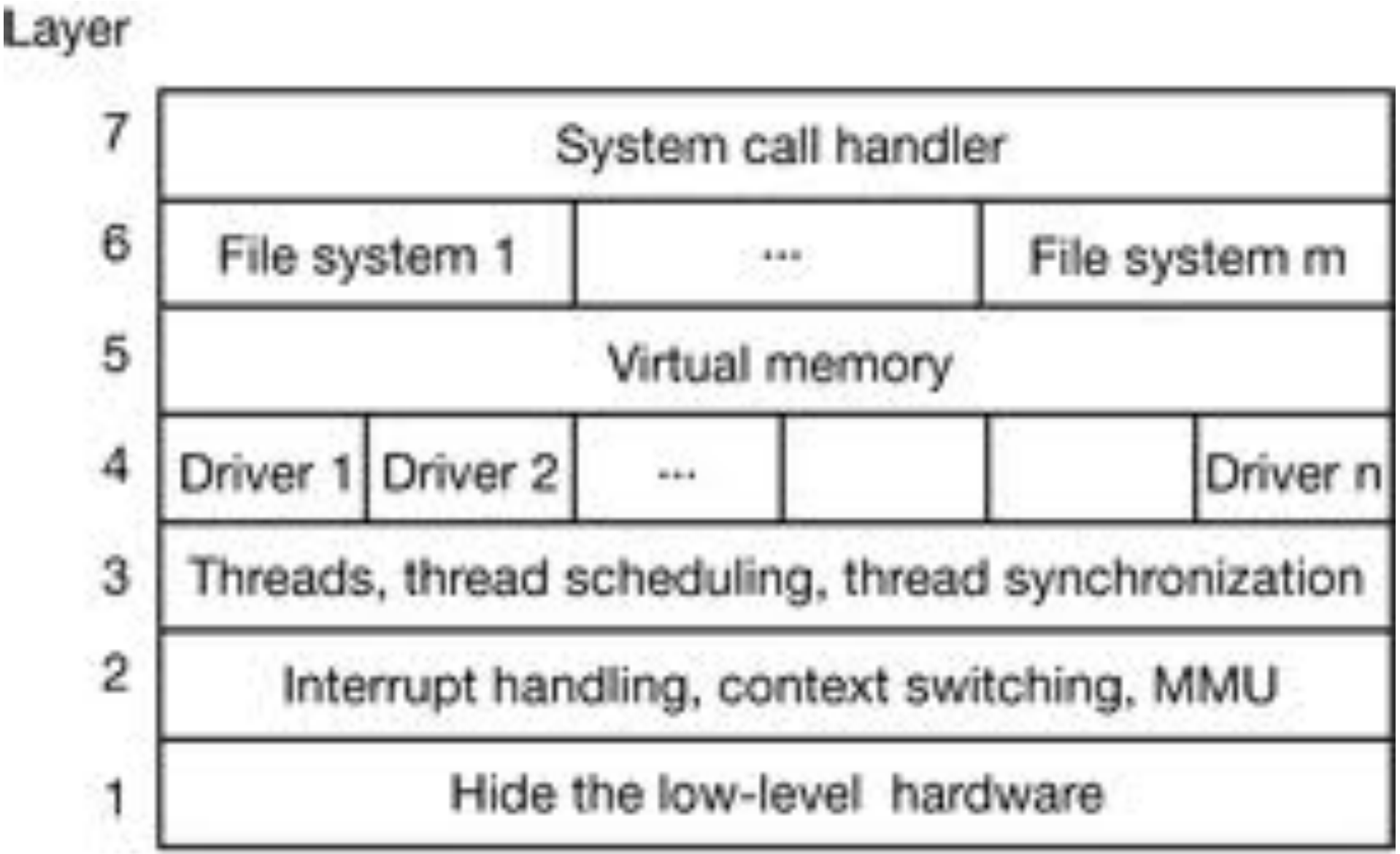
- Because you can TEST each part and make sure they work right.

Layering is a special case of modular design.

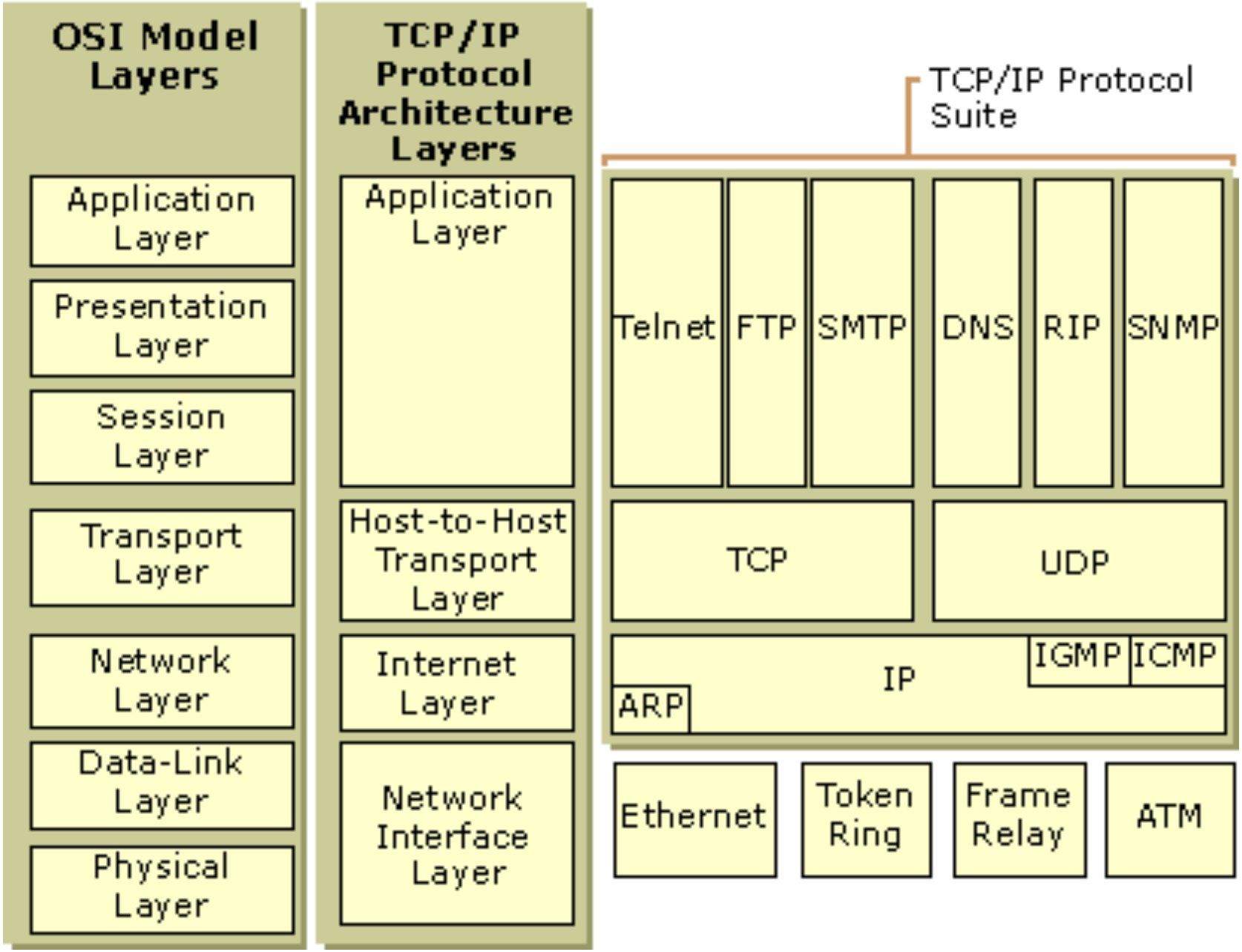
The system modules are organized into layers.

Modules in an upper layer are **allowed to use** the modules in the lower layer below it.

EXAMPLE1: OPERATING SYSTEM



EXAMPLE 2: NETWORKING



ADDITIONAL Benefits of layered

- important tool for managing complexity

- portability? maintainability
  - change in a layer can be hidden behind its interface
  - a layer can be substituted by another implementation of the layer.

- a blueprint for constructing the system
  - too many modules etc -> layering helps focus on a part
  - specialized teams can be assigned to layers (skilled GUi developers to GUI layer)