Modular Structure of Complex Systems

David Lorge Parnas
Communications Research Laboratory
Department of Electrical and Computer Engineering
McMaster University, Hamilton, Ontario Canada L8S 4K1

Abstract

We describe a systematic procedure for decomposing complex systems into modules. The result is a hierarchical structure that is described in a “module guide”, an informal document that guides a maintainer or designer to the modules affected by a change. The “guide” is largely derived from a requirements document with the result that there it is easy to trace requirements to modules, or to identify the requirements that led to a particular design decision.

Dividing Systems Into Modules

(1) What do we mean by “module”?
- •A work assignment for a programmer or programmer team.
- •No other meaning!
- •No criteria in this definition.

(2) What would make a module structure good?
- •Parts can be designed independently.
- •Parts can be tested independently.
- •Parts can be changed independently.
- •Integration goes smoothly.

(3) What’s the principle?
- •Each module’s implementation is a “secret”.
- •Each module’s interface abstractly and precisely described.
- •We abstract from implementation details likely to change.
- •We parameterise changeable aspects that cannot be hidden.

(4) What are the buzzwords?
- •Information Hiding
- •Abstract Data Types
- •Separation of Concerns
- •Object Orientation

These are all the same principle.

What is different about complex systems?

There are <u>many</u> implementation decisions.

There are <u>many</u> details.

<u>This leads to new issues:</u>

- How can we keep the project under intellectual control?
- How can we maintain conceptual integrity?
- How can we keep the maintenance cost down?
- How do we deal with unstructured lists of modules?
- How can we tell when we have them all?
- How does everyone remember the names?
- How do we avoid duplication?

Why should we group modules into classes?

Put some structure in the list of modules.

Help to check for completeness.

Leads to more helpful naming conventions.

Makes duplications less likely.

Make a specific module easier to find.

What are some possible classification criteria for modules?

A. By similarity of interface

•too vague

B. By ultimate purpose

•can be ambiguous

C. By type of "function" or service provided

•too vague

D. Similar programming problems

•implementation dependent

E. By nature of the secret

•limited to information hiding designs

•Must be unambiguous

•Requirements will be used to disambiguate when needed.

What are the classes of modules in the SCR A-7E Module Structure

Top-level decomposition

1. Hardware-hiding module
2. Behaviour-hiding module
3. Software decision module

If the secret is in the software requirements document, it must be (1) or (2).

If it is not a requirement, it must be (3).

What are the classes of modules in the SCR A-7E Module Structure

Second-level decomposition

1. Hardware-hiding module decomposition

    1.1 Extended computer module

    1.2 Device interface module

If it affects more than one device, consider it part of the computer.

What are the classes of modules in the SCR A-7E Module Structure

Second-level decomposition

2. Behaviour-hiding module decomposition

    2.1 Function driver module

    2.2 Shared services module

There will be one function driver for each controlled variable.

Some judgement is required in identifying variables.

If some behaviour should be consistent in two modules, move it to a shared module?

Watch out for coincidences.

What are the classes of modules in the SCR A-7E Module Structure

Second-level decomposition

3. Software decision module decomposition
   - 3.1 Application data type module (objects)
   - 3.2 Physical model module (active objects)
   - 3.3 Data banker module (self-updating objects)
   - 3.4 System generation module
   - 3.5 Software utility module

What are the classes of modules in the SCR A-7E Module Structure

Third-level Decomposition

1. Extended computer module decomposition
   - 1.1 Data type module
   - 1.2 Data structure module
   - 1.3 Input/output module
   - 1.4 Computer state module
   - 1.5 Parallelism control module
   - 1.6 Sequence control module
   - 1.7 Diagnostics module (restricted)
   - 1.8 Virtual memory module (hidden)
   - 1.9 Interrupt handler module (hidden)

Note: independent change is unlikely but possible simplification results from separation.

Third-level Decomposition

2. Device interface module decomposition
    - 2.1 Air data computer
    - 2.2 Angle of attack sensor
    - 2.3 Audible signal device
    - 2.4 Computer fail device
    - 2.5 Doppler radar set
    - 2.6 Flight information displays
    - 2.7 Forward looking radar
    - 2.8 Head-up display (HUD)
    - 2.9 Inertial measurement set (IMS/IMU)
    - 2.10 Panel

Note: continues on next slide

Third-level Decomposition

2. Device interface module decomposition (cont.)
    - 2.11 Projected map display set (PMDS)
    - 2.12 Radar altimeter
    - 2.13 Shipboard inertial navigation system (SINS)
    - 2.14 Slew control
    - 2.15 Switch bank
    - 2.16 TACAN
    - 2.17 Visual indicators
    - 2.18 Waypoint information system
    - 2.19 Weapon characteristics
    - 2.20 Weapon release system
    - 2.21 Weight on gear

Note: Almost corresponds to hardware structure, but not quite.

Exceptions are closely linked devices.

Third-level Decomposition

3. Function driver module decomposition
   - 3.1 Air Data Computer functions
   - 3.2 Audible Signal functions
   - 3.3 Computer Fail Signal functions
   - 3.4 Doppler Radar functions
   - 3.5 Flight Information Display functions
   - 3.6 Forward Looking Radar functions
   - 3.7 Head-up Display (HUD)functions
   - 3.8 Inertial Measurement Set (IMS/IMU) functions
   - 3.9 Panel functions
   - 3.10 Projected Map Display Set (PMDS) functions
   - 3.11 Ships Inertial Navigation System (SINS) functions
   - 3.12 Visual Indicator functions
   - 3.13 Weapon release functions
   - 3.14 Ground test functions

NOTES: input-only devices are missing!
each module can be further divided.

Third-level Decomposition

4. Shared services module decomposition
   - 4.1 Mode determination module
   - 4.2 Stage director module
   - 4.3 Shared subroutine module
   - 4.4 System value module
   - 4.5 Panel I/O support module
   - 4.6 Diagnostic I/O support module
   - 4.7 Event tailoring module

Third-level Decomposition

5. Application data type module decomposition

   <u>Examples:</u>

   •Angles (several versions)

   •Distances

   •Temperatures

   •Local data types for device modules

   •STE (state transition event) variables

All of the above are objects

Third-level Decomposition

6. Physical model module decomposition

   6.1 Earth model module

   6.2 Aircraft motion module

   6.3 Spatial relations module

   6.4 Human factors module

   6.5 Weapon behaviour module

   6.6 Target behaviour module

   6.7 Filter behaviour module

Third-level Decomposition

7. Data banker module
   - One for each real-time data item
   - Value always up-to-date
   - Secret: When to compute up-to-date value

Third-level Decomposition

8. Software utility module
   - Resource monitor module
   - Other shared resources

Documentation

- The module structure must be documented in a module guide.
- Each module must have a precise and complete interface description.
- Each implementation of a module requires a module internal design document.
- Documents must be kept “alive”.
- Engineers will take advantage of mathematics in the documentation.

VI. Conclusions

In complex systems, Information Hiding can be carried out consistently.

Classify into small, obviously complete, non-overlapping lists.

Show secrets not interfaces or roles.

Requirements Document essential for disambiguation.

Devices Interface modules hide IN and OUT.

Function Driver modules hide NAT and REQ.

Each of the resulting modules creates one or more objects (variables of a newly defined, abstractly specified, data type).

Documentation essential!

- A module guide
- Module Interface Documents
- Module Design Documents (per design)