Quiz - Layer Diagrams

Reminder: if you have questions about anything on the quiz, try not to spoil any answers for others who haven't yet taken the quiz.

Email *

bgitego@gmail.com

✗ You are starting at a new company with an established product. You'll be fixing bugs and adding features. Which of these would you want to try to put together the first week?

- [✓] Given the code repository, the hierarchy of control would help me understand what happens as the system runs. ✓
- [✓] Given the schematics, a hardware block diagram so I can understand what hardware I will be working with. ✓
- [✓] Given the higher-level software API, the software layered diagram would help me find and fix bugs in the HAL. ✗
- [ ] Given the company organization chart, a hierarchy of control would show who has the true political power.
- [✓] Given the user manual, a software block diagram discussed with a colleague would help me understand the system. ✓

Correct answer

- [✓] Given the code repository, the hierarchy of control would help me understand what happens as the system runs.
- [✓] Given the schematics, a hardware block diagram so I can understand what hardware I will be working with.
- [✓] Given the user manual, a software block diagram discussed with a colleague would help me understand the system.

Feedback

The hardware and software block diagrams are incredibly useful for understanding the system. The hierarchy of control will help you figure out where the bugs and features are in the code.

✓ What does the layered view add that the software block and hierarchy of control don't have?

- [x] An obvious hardware abstraction layer ✓
- [ ] Path showing flow of control through the code
- [ ] A broad overview of the system
- [ ] Detailed state machine descriptions
- [x] A view of the groups of modules that work together ✓

Feedback

The layered software diagram identifies modules that can be grouped together as one subsystem if they don't interact with other areas. It can also provide horizontal layers to the code (like a HAL) that might divide a project up by low-level software and higher-level software.

✗ When we talk about modular software, which of the following points are important:

- ☑ Information hiding ✓
- ☑ Dependency hiding ✗
- ☑ Few dependencies between modules ✓
- ☑ Modules acting independently of each other ✓
- ☑ Encapsulating modules with a minimum API ✓

Correct answer

- ☑ Information hiding
- ☑ Few dependencies between modules
- ☑ Modules acting independently of each other
- ☑ Encapsulating modules with a minimum API

Feedback

Hiding dependencies makes it harder to understand, maintain, and debug a system.

✗ Global variables are often very useful. What are the downsides?

- [ ] It creates dependencies between modules that are hard to remember and predict.
- [x] They may be modified in places you don’t expect (interrupts, other threads) and that may not be safe. ✓
- [ ] Sometimes there is only one type of a particular processor interface so it makes sense to have a global point to it.
- [x] Variables that are global can be modified from anywhere. ✗
- [ ] The static and volatile keywords make globals more useful.

Correct answer

- [x] It creates dependencies between modules that are hard to remember and predict.
- [x] They may be modified in places you don’t expect (interrupts, other threads) and that may not be safe.

Feedback

A global variable may be modified in places you don’t expect including interrupts and other modules. This causes a dependency that can be difficult to remember (especially if you have a lot of global variables).