Playbook Methods Repository

# **Embedded Development**

Implement software which runs on a specialized edge device with considerations for integration with sensors and input/output.

### Remote Agility: **•** High

### Linked Tactic(s): Agile Development

## Why we do it:

As technology advances and becomes more complex, manufacturers embed software into automobiles, homes, appliances, and other things that people use on a regular basis. These embedded systems can control anything from a basic digital calculator to a large industrial robot or guided missile.

Embedded systems are primarily designed to perform a specific task by combining computer hardware and software. It helps you to complete things in the simplest and most efficient manner possible. These are designed to operate autonomously or with little or no human involvement inside large applications. However, there are significant distinctions between a general-purpose system and an embedded system dedicated to a specific job. Embedded systems may be either without a graphical user interface (UI) or with a complicated graphical user interface (GUI) through sensors or remote capability. These are often developed in such a manner that they avoid human interaction and effectively control functions.

There are many reasons why embedded development is important. First, it helps to ensure that products are reliable and meet customer expectations. Secondly, embedded development can help to save time and money by reducing the need for external testing and debugging. Finally, embedded development can also improve the security of a product by making it more difficult for malicious code to be injected. Consequently, embedded development is a critical part of the product development cycle for many companies. By understanding the importance of embedded development, companies can ensure that their products are of the highest quality and meet the needs of their customers.

## When to apply it:

* Xx

## Best Practices & Considerations:

* **Design and code for reliability:** Embedded systems typically have to run 24/7 for years on end, often without needing a reset. This requires that code for this type of thing be planned and built with this in mind. Watchdogs should be used as a last option.
* **Be cautious of dynamic memory allocation:** It is not only memory leaks but also memory pool fragmentation that might cause your code to fail. If you can prevent dynamic memory allocation, it is fantastic. If not, there are techniques such as configuring a single dynamic allocation at startup or ensuring that all dynamic allocations are identical in size.
* **Maintain diligence in your coding:** Before relying on any function parameter or stored variable, do a test on it.
* **Prevent the sharing of global variables:** This is an excellent method for obtaining unexpected/undocumented behaviour. Global variables should ideally be accessible using accessor functions/methods.
* **Avoid reinventing the wheel:** If there is already existing code that does the task (or comes close), use it.
* **Avoid copying and pasting code:** If you often find yourself reusing the same code in many locations, consider making it a function (or a macro, or an inline function). Later on, the maintainers will thank you.
* **Reviews:** Submit your designs for peer or senior assessment. If you work in a project management setting, the process will need frequent evaluations. If not, you must make a deliberate effort to do so. It's as if there's a loophole or a weakness, a flaw that you've neglected and it's there in front of you, right under your nose. As a result, it pays to have your design examined by at least one non-affiliated third party.

## Responsible roles:

* xx

## Tools:

* Online tools/platforms/services
  + MATLAB, Arduino, PyCharm, WebStorm, Qt Creator, MPLAB X, Visual Studio, Eclipse, NetBeans, ARM Keil
* Websites
  + xx
* Databases
  + xx
* Other
  + xx

## 

## Thoughtworks Examples - Linked

* Client working docs, airtable, miro/mural boards
  + xx
* Client polished presentations/deliverables
  + xx
* Internal assets - clinic materials / guild docs
  + xx

## 

## Learn more: How we do this?

* Templates (docs, decks, sheets, miro, etc.)
  + xx
* How-To Resources (external or internal)
  + <https://radixweb.com/blog/embedded-systems-concept-worth-understanding-implementing>
  + <https://www.comptia.org/blog/what-is-embedded-development>
  + <https://www.quora.com/What-are-some-embedded-systems-design-best-practices>
  + <https://www.sam-solutions.com/blog/top-ten-embedded-software-development-tools/>
* Outside References (articles, books, etc.)
  + xx
* Sub-set Activities
  + xx