★Artifact: ARCH.Architecture Notebook, Checklist, and Template

---

- ✓**#artifact# Architecture Notebook**
- ✓**#checklist# Architecture Notebook**
- ✓**#template# Architecture Notebook**

.

.

#artifact# Architecture Notebook

Summary
This artifact describes the rationale, assumptions, explanation, and implications of the decisions that were made in forming the architecture.

Purpose
To capture and make architectural decisions and to explain those decisions to developers.

Relationship
.

Fulfilled Slots	Roles	Tasks
[Technical Architecture]	Responsible: #role# Architect	Output From: • #task# Envision the Architecture • #task# Refine the Architecture
	Modified By: #role# Architect	Input To: • #task# Refine the Architecture • #task# Envision the Architecture • #task# Design the Solution

		• #task# Plan Iteration

.
Main Description
This artifact describes the Software Architecture.

It provides a place for maintaining the list of architectural issues, along with the associated architectural **decisions, designs, patterns, code documented** (or pointed to), and so forth -- all at appropriate levels to make it easy to understand what architectural decisions have been made and remain to be made.

It is helpful for architects to use this artifact to collaborate with other team members in developing the architecture and to help team members understand the motivation behind architectural decisions so that those decisions can be robustly implemented. For example, the architect may put constraints on how data is packaged and communicated between different parts of the system. This may appear to be a burden, but the justification in the Architecture Notebook can explain that there is a significant performance bottleneck when communicating with a legacy system. The rest of the system must adapt to this bottleneck by following a specific data packaging scheme.

This artifact should also inform the team members how the system is partitioned or organized so that the team can adapt to the needs of the system. It also gives a first glimpse of the system and its technical motivations to whoever must maintain and change the architecture later.

Brief Outline
At a minimum, this artifact should do these three things:

- List **guidelines, decisions,** and **constraints** to be followed
- Justify those guidelines, decisions, and constraints
- Describe the Architectural Mechanisms and where they should be applied

Team members who were not involved in those architectural decisions need to understand the reasoning behind the context of the architecture so that they can address the needs of the system. Consider adding more information when appropriate. A small project shouldn't spend a lot of time documenting the architecture, but all critical elements of the system must be communicated to current and future team members. This is all useful content:

- **Goals and philosophy** of the architecture
- Architectural **assumptions and dependencies**
- References to architecturally **significant requirements**
- References to architecturally significant **design elements**
- **Critical system interfaces**
- **Packaging instructions** for subsystems and components
- **Layers and critical subsystems**
- **Key abstractions**
- **Key scenarios** that describe critical behavior of the system

Tailoring

- **Impact of not having**

  Without this artifact, there will be no coordination of the individual design efforts, and it will be difficult to establish and communicate an overall architectural vision of the project.

- **Reasons for not needing**

  This artifact is not required if an existing reference architecture is being used or another approach or set of artifacts are being used to document the architecture. This artifact may not be needed if the design is relatively straight-forward and does not have any architecturally significant risks.

- **Representation Options**

  A set of architectural models can be developed as part of the architectural documentation. For more information, see

#guideline# Modeling the Architecture.

More Information

- **#checklist# Architecture Notebook**
- **#concept# Software Architecture**
- **#guideline# Modeling the Architecture**

.

.

(Work Product Slot) [Technical Architecture]
This slot serves as an abstraction of high level artifacts that represent the documentation of the architecture.

.

.
#checklist# Architecture Notebook

Check Items

- **Is the architecture understandable?**
  - Is the description of the architecture complete, meaningful, and clear?
  - Is the architecture at an appropriate level of detail, given the objectives?
  - Are concepts handled in the simplest way possible?
  - Does the architecture clearly convey not only the solution but also the motivation and objectives related to the decisions that have been made in shaping the architecture?
  - Are the key assumptions and decisions that the architecture is based on documented and visible to reviewers and those who will use the architecture?
  - Is the architecture description current?
  - Have the design guidelines been followed?
- **Have the architectural goals, constraints and requirements been adequately described and handled?**
  - Have the Architectural Goals been clearly described?

    - Have any Architectural Constraints been identified and documented?
    - Have the Architecturally Significant Requirements been identified and are they clearly described.
    - Is the architecture is consistent with the architectural goals, constraints and requirements?
- **Have necessary architectural mechanisms been identified and described?**
    - Is it clear when each Architectural Mechanism should be applied?
    - Is there a clearly defined design pattern in place to support each mechanism?
    - Does each mechanism adequately address the requirements it is intended to meet?
- **Have the system partitions been adequately defined?**
    - Is partitioning approach clearly described and applied consistently?
    - Does the partitioning approach reduce complexity and improve understanding?
    - Have the partitions been defined to be highly cohesive within the partition, while the partitions themselves are loosely coupled?
- **Have the key elements been adequately defined?**
    - Have the Key Abstractions adequately defined?
    - Have the the key design elements (i.e., Components) adequately defined?
        - Do the components have well-defined interfaces?
        - Have the system's responsibilities been allocated to the components?
        - Are the number and types of components reasonable?
- **Have interfaces to external systems been adequately represented?**
    - See **#guideline# Representing Interfaces to External Systems**,
- **Has all reuse been identified?**
    - Have all reusable assets been identified -- either those reused by the system, or those elements within the system that have been built to be reused. For more information, see **#guideline# Software Reuse.**
- **Has the architecture been built to evolve?**
    - Can the architecture easily evolve, so that expected changes can be easily accommodated?
    - Are all technical risks either mitigated or addressed in a contingency plan?

    - Has the architecture been overly structured to handle unlikely change at the expense of simplicity and comprehensibility? (Hint: "Yes" to this question is not good.)
- **Can the architecture be delivered by the team?**
    - Does the architecture provide a suitable basis for organizing the development teams?
    - Does each team have the skills required to implement their allocated components?
    - Are responsibilities divided well between teams?
    - Do all team members share the same understanding of the architecture as the one presented by the architect?
    - Can team members understand enough from the architecture to successfully design and code their allocated components?
- **Has the software been adequately mapped to the hardware?**
    - Have the deployable software components been mapped to physical nodes?

More Information

- **#concept# Architectural Constraints**
- **#concept# Architectural Goals**
- **#concept# Architecturally Significant Requirements**
- **#concept# Architectural Mechanism**
- **#concept# Component**
- **#concept# Key Abstractions**
- **#guideline# Representing Interfaces to External Systems**
- **#guideline# Software Reuse**

.
.

.

#template# Architecture Notebook

1. Purpose

2. Architectural goals and philosophy

3. Assumptions and dependencies

4. Architecturally significant requirements

5. Decisions, constraints, and justifications

6. Architectural Mechanisms

- Architectural Mechanism 1

7. Key abstractions

8. Layers or architectural framework

9. Architectural views

- Logical:
- Operational:
- Use case:

architecture_notebook_tpl.dot
2020/3/2 上午 08:20, 51.5 KB

<Project Name>
Architecture Notebook

1. Purpose

This document describes the philosophy, decisions, constraints, justifications, significant elements, and any other overarching aspects of the system that shape the design and implementation.
[Always address Sections 2 through 6 of this template. Other sections are recommended, depending on the amount of novel architecture, the amount of expected maintenance, the skills of the development team, and the importance of other architectural concerns.]

2. Architectural goals and philosophy

[Describe the philosophy of the architecture. Identify issues that will drive the philosophy, such as: Will the system be driven by complex deployment concerns, adapting to legacy systems, or performance issues? Does it need to be robust for long-term maintenance?

Formulate a set of goals that the architecture needs to meet in its structure and behavior. Identify critical issues that must be addressed by the architecture, such as: Are there hardware dependencies that should be isolated from the rest of the system? Does the system need to function efficiently under unusual conditions?]

3. Assumptions and dependencies

[List the assumptions and dependencies that drive architectural decisions. This could include sensitive or critical areas, dependencies on legacy interfaces, the skill and experience of the team, the availability of important resources, and so forth]

4. Architecturally significant requirements

[Insert a reference or link to the requirements that must be implemented to realize the architecture.]

5. Decisions, constraints, and justifications

[List the decisions that have been made regarding architectural approaches and the constraints being placed on the way that the developers build the system. These will serve as guidelines for defining architecturally significant parts of the system. Justify each decision or constraint so that developers understand the importance of building the system according to the context created by those decisions and constraints. This may include a list of DOs and DON'Ts to guide the developers in building the system.]

- Decision or constraint and justification
- Decision or constraint and justification

6. Architectural Mechanisms

[List the architectural mechanisms and describe the current state of each one. Initially, each mechanism may be only name and a brief description. They will evolve until the mechanism is a collaboration or pattern that can be directly applied to some aspect of the design.]

Architectural Mechanism 1

[Describe the purpose, attributes, and function of the architectural mechanism.]

Architectural Mechanism 2

[Describe the purpose, attributes, and function of the architectural mechanism.]