Design Decision Log

Not all requirements can be captured in the beginning of an agile project during one or more design sessions. The initial architecture design can evolve or change during the project, especially if there are multiple possible technology choices that can be made. Tracking these changes within a large document is in most cases not ideal, as one can lose oversight over the design changes made at which point in time. Having to scan through a large document to find a specific content takes time, and in many cases the consequences of a decision is not documented.

Why is it Important to Track Design Decisions

Tracking an architecture design decision can have many advantages:

- Developers and project stakeholders can see the decision log and track the changes, even as the team composition changes over time.
- The log is kept up-to-date.
- The context of a decision including the consequences for the team are documented with the decision.
- It is easier to find the design decision in a log than having to read a large document.

What is a Recommended Format for Tracking Decisions

In addition to incorporating a design decision as an update of the overall design documentation of the project, the decisions could be tracked as Architecture Decision Records as Michael Nygard proposed in his blog.

The effort invested in design reviews and discussions can be different throughout the course of a project. Sometimes decisions are made quickly without having to go into a detailed comparison of competing technologies. In some cases, it is necessary to have a more elaborate study of advantages and disadvantages, as is described in the documentation of Trade Studies. In other cases, it can be helpful to conduct Engineering Feasibility Spikes. An ADR can incorporate each of these different approaches.

Architecture Decision Record (ADR)

An architecture decision record has the structure

[Ascending number]. [Title of decision]

The title should give the reader the information on what was decided upon.

Example:

> *001. App level logging with Serilog and Application Insights*

Hint:

When several developers regularly start ADRs in parallel, it becomes difficult to deal with conflicting ascending numbers. An easy way to overcome this is to give ADRs the ID of the work item they relate to.

Date:

The date the decision was made.

Status:

[Proposed/Accepted/Deprecated/Superseded]

A proposed design can be reviewed by the development team prior to accepting it. A previous decision can be superseded by a new one, or the ADR record marked as deprecated in case it is not valid anymore.

Context:

The text should provide the reader an understanding of the problem, or as Michael Nygard puts it, a value-neutral [an objective] description of the forces at play.

Example:

> Due to the microservices design of the platform, we need to ensure consistency of logging throughout each service so tracking of usage, performance, errors etc. can be performed end-to-end. A single logging/monitoring framework should be used where possible to achieve this, whilst allowing the flexibility for integration/export into other tools at a later stage. The developers should be equipped with a simple interface to log messages and metrics.
>
> If the development team had a data-driven approach to back the decision, i.e., a study that evaluates the potential choices against a set of objective criteria by following the guidance in Trade Studies, the study should be referred to in this section.

Decision:

The decision made, it should begin with 'We will...' or 'We have agreed to ...

Example:

> We have agreed to utilize Serilog as the Dotnet Logging framework of choice at the application level, with integration into Log Analytics and

> Application Insights for analysis.

Consequences:

The resulting context, after having applied the decision.

Example:

> Sampling will need to be configured in Application Insights so that it does not become overly-expensive when ingesting millions of messages, but also does not prevent capture of essential information. The team will need to only log what is agreed to be essential for monitoring as part of design reviews, to reduce noise and unnecessary levels of sampling.

Where to Store ADRs

ADRs can be stored and tracked in any version control system such as git. As a recommended practice, ADRs can be added as pull request in the *proposed* status to be discussed by the team until it is updated to *accepted* to be merged with the main branch. They are usually stored in a folder structure *doc/adr* or *doc/arch*. Additionally, it can be useful to track ADRs in a `decision-log.md` to provide useful metadata in an obvious format.

Decision Logs

A decision log is a Markdown file containing a table which provides executive summaries of the decisions contained in ADRs, as well as some other metadata. You can see a template table at `doc/decision-log.md`.

When to Track ADRs

Architecture design decisions are usually tracked whenever significant decisions are made that affect the structure and characteristics of the

solution or framework we are building. ADRs can also be used to document results of spikes when evaluating different technology choices.

Examples of ADRs

The first ADR could be the decision to use ADRs to track design decisions,

- 0001-record-architecture-decisions.md,

followed by actual decisions in the engagement as in the example used above,

- 0002-app-level-logging.md.

Last update: August 26, 2024