adr.github.io

Homepage of the ADR GitHub organization

Architectural Decision Records

An Architectural Decision (AD) is a software design choice that addresses a functional or non-functional requirement that is architecturally significant. An Architecturally Significant Requirement (ASR) is a requirement that has a measurable effect on a software system's architecture and quality. An *Architectural Decision Record (ADR)* captures a single AD, such as often done when writing personal notes or meeting minutes; the collection of ADRs created and maintained in a project constitute its *decision log*. All these are within the topic of Architectural Knowledge Management (AKM).

The aim of the GitHub adr organization is to:

1. Motivate the need for and benefits of AD capturing and establish a common vocabulary
2. Strengthen the tooling around ADRs, in support of agile practices as well as iterative and incremental software engineering processes
3. Provide pointers to public knowledge in the context of AKM and ADRs (for instance, this website)

   Note: The term "architecture decision record" can be used interchangeably.

ADRs in the media

- (German) Gut dokumentiert: Architecture Decision Records by @obfischer published at heise online.

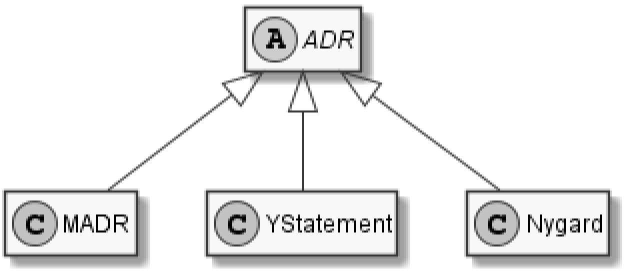
Lightweight Architectural Decision Records Should be Adopted

ThoughtWorks lists architectural decision records as "adopt" at their technology radar vol. 18. It is still listed as "adopt".

A "lightweight" ADR consists of title, status, context, decision, and consequences (according to @mtnygard).

We think that the considered options are crucial to understand the reason of a chosen option. Thus, we propose MADR — The Markdown Architecture Decision Records (MADR: [ˈmæɾɚ]) as alternative in this ADR organization.

Relation of ADRs, MADR, and Others



Sustainable Architectural Decisions

We base our work on the guidelines and principles in Sustainable Architectural Decisions by Zdun et al., for instance the Y-statement format suggested in that article.

However, we are open to other formats of ADRs as shown at @joelparkerhenderson's repository.

In short, the Y-statement is as follows:

> In the context of `<use case/user story u>`, facing `<concern c>` we decided for `<option o>` to achieve `<quality q>`, accepting `<downside d>`.

The long form of it is as follows (extra section "because"):

> In the context of `<use case/user story u>`, facing `<concern c>` we decided for `<option o>` and neglected `<other options>`, to achieve `<system qualities/desired consequences>`, accepting `<downside d/undesired consequences>`, because `<additional rationale>`.

You can find more explanations and examples on Medium Y-Statements — A Light Template for Architectural Decision Capturing.

A Definition of Done for Architectural Decision Making proposes five criteria and a checklist to decide when it is time to set the status of a single decision to "done": evidence, criteria and alternatives, agreement, documentation, and realization/review plan. Here, we focus on the 'D' in *ecADR*.

Existing ADR Templates

- Overview: Architectural Decision Records: collection of markdown templates converted to Markdown
- MADR: The Markdown Architecture Decision Records (MADR: `[ˈmæɾɚ]`). Lean ADRs to quickly document architectural decisions in code. - Slogan: architectural decisions that matter `[ˈmæɾɚ]`.
- Comparison of seven templates: Zimmermann et al.: Architectural Decision Guidance Across Projects - Problem Space Modeling, Decision Backlog Management and Cloud Computing Knowledge. WICSA 2015: 85-94.
- Context, background and examples of good and bad justifications can be found in this blog post.
- DecisionCapture: Templates for agile projects and explanation of the ADR universe. example.

Tooling

- ADMentor Architectural Decision Modeling Add-In for Sparx Enterprise Architect
- adr-tools - bash scripts to manage ADRs in the Nygard format. example.
    - Ansible script to install adr-tools: ansible-adr-tools
    - C# rewrite: adr-cli
    - Go rewrite: adr
    - Java rewrite: adr-j
    - Node.js rewrite: adr
    - PHP version: phpadr
    - Powershell module: adr-ps
    - Python rewrite: adr-tools-python
    - Another Powershell module: ArchitectureDecisionRecords
- adr-log: Generates a architectural decision log out of MADRs.
- adr-manager: Craft MADR templates directly in the Web Browser.
- adr-viewer - python application to generate a website from a set of ADRs.
- Embedded Architectural Decision Records, which shows how a distributed AD log can be embedded in Java Code via ADR annotations.
- Log4brains: CLI and web UI to log and publish your ADRs as a static website

Tooling related to architecture management

- ArchUnit; unit tests for architecture
- docToolchain: docToolchain is an implementation of the docs-as-code approach for software architecture plus some additional automation.
- Structurizr: Structurizr is a collection of tooling to help you visualise, document and explore your software architecture using the C4 model.

Interesting, but unmaintained tooling

- eadlsync: Synchronizes embedded architectural decision records with a repository of architectural decisions.
- SE Repo: Software Engineering Repository. A repository for versioning software engineering artifacts, which can be architectural decisions, patterns, and others.

More Related Work

- Documenting Architecture Decisions by Michael Nygard
- Architectural Knowledge Management (AKM) overview by HSR FHO
- Sustainable Architectural Decisions
- Overview on all variants of ADRs
- Method Selection and Tailoring
- Work by Daniel Popescu
- Architecture Decision Records in Action by Michael Keeling (IBM Watson Group) and Joe Runde (IBM) - presentation including empirical numbers.
- Documenting Architecture Decisions - Blog entry by Fabian Keller
- From Architectural Decisions to Design Decisions and ADR = Any Decision Record? - two blog posts by Olaf Zimmermann proposing to exted the scope of ADRs.

Published with GitHub Pages