Ch18 Documenting Software Architectures 软件架构编档

内容很多主要看红字部分

【理解】架构编档的重要性。

文档是交流的基础。文档就像数据，生命周期远长于代码。更多时候你的上级只看文档，不看代码。

【理解】构文档的用途、读者 Uses and Audience for Architecture Documentation

* 观众（如利益相关者）

新员工

开发商

分析师

* 架构文档既是规范性的（约束），也是描述性的
* 了解利益相关者对架构文档的使用至关重要
* 这些用途决定了要获取的信息

架构文档的三个用途

* 教育

向人们介绍系统

* 利益相关者之间沟通的主要工具

特别是开发中的架构师和未来的架构师

* 系统分析和构建的基础

表示 Notations

* 非正式的表示法

使用通用图表和编辑工具

描述以自然语言为特征

无法正式分析

* 半正规表示法（UML）

标准化符号

缺乏元素的完整语义

可以应用基本分析

* 正式的符号

有一个精确的语义。

语法和语义的形式分析是可能的。

架构描述语言（ADL）

通过相关工具支持自动化

选择表示法Notation

* 权衡Tradeoffs

正式表示法：更多时间和精力，减少模糊性和更多分析机会。

非正式符号：更容易创建，但更少保障。

* 对于表达不同种类的信息，不同的表示法更好（或更糟）。

UML

ADL

视图 Views

* 视图让我们将软件架构划分为多个可管理的系统表示。
* 架构文档原理：

记录相关意见

添加超出视图的文档。

模块视图概述

* 元素

模型

* 关系

Is part of.

Depends on.

Is a.

* 约束

拓扑约束。

行为约束。

* 用法

构建代码的蓝图

变化影响分析

规划增量开发

需求可追溯性分析

工作分配，时间表和预算信息

如果没有至少一个模块视图，任何软件架构的文档都不可能完整。

C&C 视图

* 元素

组件 Components

连接器 Connectors

* 关系

附件 Attachments

接口委托 Interface delegation

* 约束

组件只能连接到连接器。

连接器只能连接到组件。

附件只能在兼容的端口和角色之间进行。

接口委派只能在两个兼容端口（或两个兼容角色）之间定义。

连接器不能单独出现; 必须将连接器连接到组件。

* 用法

显示系统如何交互。

帮助推理运行时系统质量

Notations for C&C Views

* UML组件非常适合C＆C组件。

uml.tiff

* UML连接器不够丰富，无法代表许多C＆C连接器。

没有子结构，属性或行为描述。

* 使用UML连接器 - 线代表“简单”的C＆C连接器。

没有语义，使用构造型来表示连接器的类型。

* 参与者无法用UML连接器表示。

标记连接器末端。

分配视图Allocation Views

* 元素

软件元素

环境因素

* 关系

分配给 Allocated to

* 约束

因视图而异

* 用法

关于性能，可用性，安全性和安全性的推理。

关于分布式开发和团队工作分配的推理。

关于并发访问软件版本的推理。

关于系统安装形式和机制的推理。

质量视图

* 质量视图可以为利益相关者定制，或解决具体问题。
* 通过提取相关的结构视图并将它们包装在一起来形成质量视图。

Examples：

* 安全性视图

具有安全角色或责任的组件

安全视图的行为部分

显示安全协议的操作以及人与安全元素交互的位置和方式。

捕获系统如何响应特定威胁和漏洞。

* 交流视图

显示所有组件到组件通道，各种网络通道，服务质量参数值和并发区域。

此视图的行为部分可以显示如何动态分配网络带宽。

* 异常或错误处理视图

显示组件如何检测，报告和解决故障或错误。

它有助于确定每个错误的来源和适当的纠正措施。

* 可靠性视图

模型机制，如复制和切换。

描述时间问题和交易完整性。

* 性能视图

显示架构中用于推断系统性能的那些方面。

显示网络流量模型，操作的最大延迟等。

【了解】如何选择视图进行编档

不同的视图支持不同的目标和用途。你应记录的视图取决于你希望对文档的用途。每种视图都有成本和收益; 你应该确保维护视图的收益超过其成本。

如果你知道以下内容，你可以确定需要哪些视图，何时创建它们，以及包括多少细节。

* + 什么人，有什么技能可用
  + 您必须遵守哪些标准
  + 手头有什么预算
  + 时间表是什么
  + 重要利益相关者的信息需求是什么
  + 驱动质量属性要求是什么
  + 系统的大致尺寸是多少

选择视图的方法

1. 构建利益相关者/视图表

行：利益相关者

列：适用于系统的视图。

某些视图（例如分解，使用和工作分配）适用于每个系统，而其他视图（各种C＆C视图，分层视图）仅适用于某些系统。

使用以下内容填写每个单元格：none，仅概述（overview only），中等细节（moderate detail）或高细节（high detail）。

1. 合并视图以减少其数量

这些观点经常自然结合：

各种C＆C观点。

具有SOA或通信进程视图的部署视图。

分解视图和任何工作分配，实现，使用或分层视图。

1. 确定优先级和阶段

分解视图提前发布。

不必满足所有信息需求。

提供80％可能“足够好”，以便利益相关者可以完成他们的工作。

如果信息的子集足够，请与利益相关者核实。

在开始另一个视图之前不必完成一个视图。

可以使用概述级别的信息取得进展

广度优先的方法通常是最好的。

构建文档包

包括：视图、超出视图的文档

【掌握】如何对视图进行编档 Documenting a View

1. 主要演示文稿 The Primary Presentation

它显示了视图的元素和关系。

包含您希望传达的有关系统的信息 - 在该视图的词汇表中。

主要演示文稿通常是图形化的。

非正式，半正式或正式。

包括解释符号的关键点。

有时是文本的，例如表格或列表。

如果有风格规则，则应通过引用说明或纳入这些规则。

1. 元素目录 The Element Catalog

元素目录至少详细说明了主要演示文稿中描述的那些元素。

如果在主要演示文稿中省略了与此视图相关的元素或关系，则应在此处介绍和解释这些元素或关系。

目录的组成：

元素及其属性

关系及其属性

元素接口

元素行为

1. 上下文图 Context Diagram

此视图中描述的系统如何与其环境相关。

环境中的实体可以是人，其他计算机系统或物理对象。

1. 可变性指南 Variability Guide

可变性指南显示变化点。

1. 理由 Rationale

基本原理

所选模式的架构问题，以及选择模式的理由

【掌握】视图模板

viewtemplate.tiff

记录超出视图的信息

文件控制信息。

列出发布组织，当前版本号，发布日期和状态，更改历史记录以及向文档提交更改请求的过程。

通常捕获在前面的事物

1. 文档路线图

什么信息，在哪里以及在哪里找到它。

范围和总结。 解释文档的目的并简要总结所涵盖的内容。

如何组织文档。

查看概述。 描述架构师已包含在包中的视图。

利益相关者如何使用文档。

考虑至少用户，收单机构，开发人员和维护人员的顾虑。

1. 如何记录视图。

解释您用于记录视图的标准构成

1. 系统概述

系统功能，用户以及任何重要背景或约束的简短描述。

为您的读者提供一致的系统心理模型及其目的。

1. 视图之间的映射。

帮助读者深入了解架构如何作为统一的概念整体运作。

通常，架构中的视图之间的元素之间的关联是多对多的。

可以将视图到视图关联捕获为表。 例子

“用于实现”，用于从组件和连接器视图映射到模块视图

“包含在”中，用于从分解视图到分层视图的映射

1. 解释 Rationale

记录适用于多个视图的架构决策。

1. 目录

一套参考资料，可帮助读者快速找到更多信息。

条款索引

词汇表

缩略语列表

【掌握】如何对系统行为进行编档

行为编档

* 行为文档通过描述该视图中的体系结构元素如何相互交互来补充每个视图。
* 行为文档可以进行推理

系统陷入僵局的可能性

系统能够在所需的时间内完成任务

最大内存消耗

和更多

* 行为在视图模板的元素目录中有自己的部分。

行为编档的符号

面向跟踪的语言

* 跟踪
* 跟踪描述了特定的活动序列或结构元素之间的相互作用。
* 例子

用例

序列图

通信图

活动图

消息序列图表

时序图

业务流程执行语言

(一堆uml的东西。。。略过)

【掌握】如何对质量属性进行编档

* 质量属性在文档中显示在哪里？

基本原理。

提供服务的架构元素通常具有分配给它们的质量属性边界。

体系架构文档通常包含对需求的映射，以显示如何满足需求。

路线图告诉利益相关者文档中的哪个位置可以找到问题。

记录变化比编档快的架构

* + 在运行时或由于高频发布和部署周期而发生变化的体系结构的变化比文档周期快得多。
  + 在这种情况下：

记录所有版本系统的真实情况。 记录那些不变量。

记录允许架构改变的方式。 这样做的地方称为变异性指南。

* 采用模板或标准来捕捉你的设计决策。
* 记录一个视图，如果它有一个确定的利益相关者选区。
* 当信息可用时，请填写视图模板的各个部分以及视图之外的信息。
* 生成足够的设计信息以转移到代码。
* 没必要填写模板的所有部分。
* 敏捷团队有时会通过白板进行简短的讨论。 拍照并将其用作主要演示文稿。