IT 运维中配置管理的意义和常见问题解答

配置管理的核心工作主要包含两部分：建立配置管理数据库（Configuration Management Database，简称 CMDB）和对配置管理数据库准确性的维护。配置管理数据库是真实环境的逻辑映射，是 IT 工程师和管理人员的技术地图。所以这个地图的准确性和详细度直接影响着使用者后续的判断和决策。通常以面向服务的方式将相关的任何服务资产定义成配置项（Configuration Item，简称 CI），各个 CI 之间确立逻辑或物理关系，典型的 CI 总结有 6 大类：服务、软件、硬件、位置、人员、文档。

CMDB 可用于故障定位、问题分析、变更影响度分析、容量决策、组件失败影响度分析（CFIA）、故障树分析（FTA）等等。与真实环境的匹配度和详细度非常重要，否则辛苦建立起来的 CMDB 就会被放弃使用或是让使用者做出错误的决策。

企业在做配置管理时，通常会面临以下这些问题：

- 配置项的定义太细或是太粗，无法找到合适级别，粒度太细导致维护成本太高，粒度太粗导致提供的信息没有意义。
- 没有定期做好 CMDB 的备份，从而导致 CMDB 在回滚时没有参考的基线。
- CI 项随意地被修改，没有严格的控制机制，导致 CMDB 与真实环境相差甚远，对 IT 人员也就失去了参考意义

场景一： CI 项识别的粒度粗细

场景描述：

某证券公司随着业务的发展，基础架构规模越来越大。面对几百台主机，错综复杂的网络，已经没有人能够清楚总共有多少应用系统、软硬件之间的关系是什么，以及它们的位置信息，给 IT 管理层决策和设备升级更新维护带来了很大的困难。为了提升 IT 管理的效率公司决定实施配置管理。

基于全面完整性的原则，管理层决定 CI 应该被设计得非常详细，覆盖所有的资产，不能有任何遗漏。甚至连主机的一些常规外设也作为范围之内，如：显示器、鼠标、键盘等设备也作为 CI 项来识别定义。CMDB 建成后经历了 3 个月的试运营阶段，配置管理员们发现每天都有非常大量的配置信

息更新需要完成，CMDB 的维护工作变得非常繁重，而且许多 CMDB 的使用者也经常抱怨查询很慢，在查询结果中也包含了很多无用的信息。

解决办法：

在 CMDB 建设的初期很多企业都会面临场景中描述的问题，如何确定 CI 的颗粒度。在识别 CI 时最重要的原则有：

- 三个面向原则（面向服务、面向客户、面向业务）
- 够用原则
- 持续改进原则

在实施配置管理的初期阶段，需要把握好这些原则。场景中我们发现由于 CI 项的粒度太细，广度太广，建设者有一步到位，涵盖全局的想法。从而导致 CMDB 过于庞大，而且并没有从业务的角度来识别 CI，造成了系统中有了很多无用信息。场景中所提到的外设，如显示器、鼠标等设备的识别并不符合三个面向的原则，没有这些设备，同样能提供主机上面应用所支持的服务。

在配置管理流程的运行期，只要真实环境中的组件发生变更，与他对应的在 CMDB 中的 CI 就必须及时更新，任何配置项的更新都需要受到变更管理的控制，这些工作都增加了维护人员的工作量。所以可以考虑先把 CI 的识别集中在所有的硬件和服务方面，如：网络和主机等。这部分比较好识别可以优先去做。等有了初步的成效再扩充到位置和软件信息上面，最后再覆盖所有的，如文档和人员。在识别属性和关系时，考虑在初期先识别物理关系。然后识别逻辑关系，可以有效降低配置管理流程实施的风险。

总的来说采取先易后难，分阶段实施来进行符合实施所有流程的一般规律。

场景二： CMDB 更新后回滚的依据

场景描述：

某 IT 服务企业使用的一套流程管理软件做了一个重大的功能升级补丁，由于此应用软件已经在现实环境中运营一段时间，并且和其他的第三方系统也有一系列接口和交互，导致测试环境很难构

建，所以在完成了单元测试后就上线运营了，由于还引入了一些新的基础架构设备，在上线完成后同时更新了 CMDB。但试运行 3 个月后发现效果不佳，决定暂时先执行回滚。

在真实环境执行回滚后却发现当时在更新 CMDB 之前没有做过备份，并且 CMDB 之前也从来没有做过任何的备份工作，导致逻辑环境无法回滚。

解决办法：

CMDB 可以称作 IT 人员的技术地图，CMDB 与现实环境的不一致，会导致 CMDB 无效，长此以往，就不会有 IT 人员再去使用和信任 CMDB，很多基于 CMDB 的流程也就无法实现。而配置管理的一个重要目标就是保证 CMDB 与现实环境相一致。如何来避免这样的事情再次发生呢？我们要保证 CMDB 在每次变更之前需要做好 CMDB 的快照（snapshot），以此作为配置的基线（baseline），在回滚时 CMDB 也能恢复到变更执行前的状态。除了变更前，定期也需要做快照。每年执行定期检查，消除 CMDB 与现实环境中的差异，避免所有未授权的更改。

案例三： 用 Excel 表作为 CMDB 的存储介质

场景描述：

某企业 CMDB 存储介质使用了 Excel 表格，将此 Excel 表存放在一个共享目录中供 IT 人员使用。但运营一段时间后发现，出现了多个 Excel 表格，给使用者造成了很大的疑惑，不知道那个版本才是最新的。很多 CI 项的信息与实际环境出入很大，信息无法准确反映出基础架构的真实属性和情况。问题出在哪里？

解决办法：

由于采用了 Excel 表作为 CMDB，不像数据库或是 CMDB 工具，可以通过权限管理来控制更改。但这种方式实施的成本很低，在 CI 更新变化不多的情况下是一种非常有效的方法。用这种方式通常会把 CMDB 的 Excel 表文件放在一个共享目录中供全体使用者访问。所以必须严格控制好此配置表的管理。同时设定好配置管理的角色与职责，由谁来负责更新和管理这个 Excel 表。

根据实践经验，除了利用操作系统的用户管理权限以外，使用软件开发中的版本管理软件也是非常有效的，如 VSS、CVS。设定只有管理员可以对配置表其进行修改，而其他使用者只有访问的权限。能够有效避免各个使用者获得的配置表不一致的情况。

明确配置管理人员的角色和职责，设定定期巡检的机制，发现与真实环境不匹配就及时通知管理员确认。CMDB 的准确性直接影响着其他服务管理流程的执行。