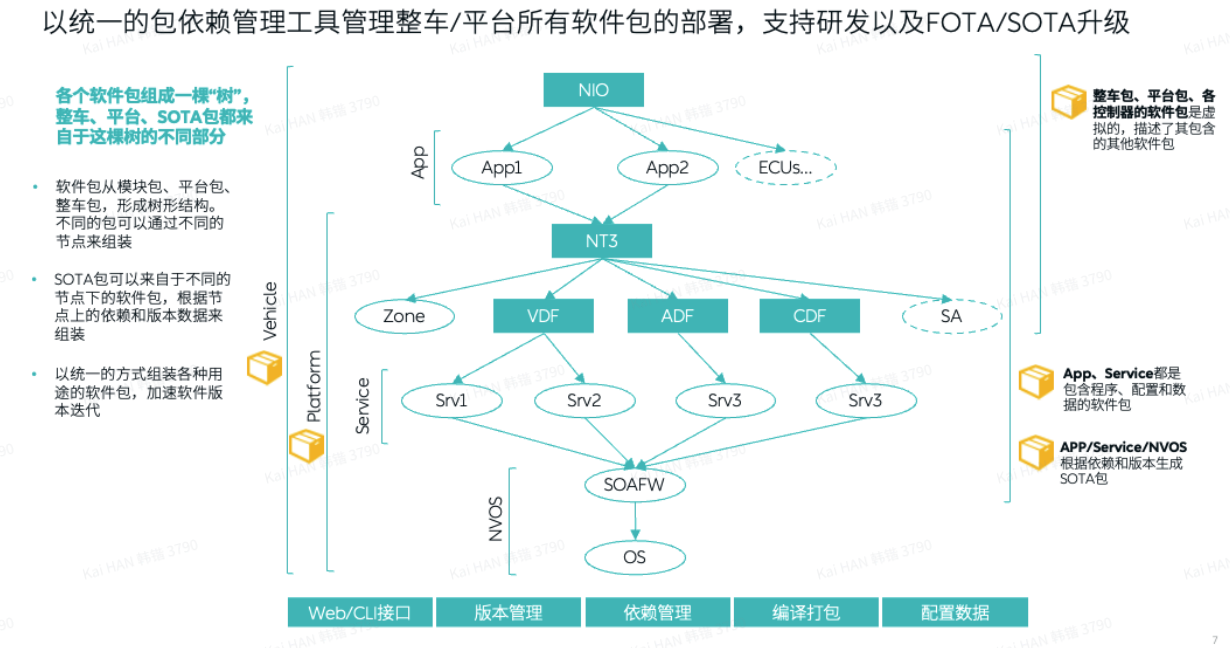
Peanut - A unified NT3 package solution

Peanut Showcase Playbook

POC回放视频：https://nio.feishu.cn/minutes/obcn44uwmc6kg8bg446n475s



NT3软件包管理的挑战

NT3提出的平台化与服务化战略，可以提升软件的复用度以及加快软件的演进；另一方面，NT3也对软件包管理带来新的挑战。

第一，相对于NT1与NT2，NT3的软件包从“集中”走向了“分散”，不同于以前整车软件包只有少数几个子软件包，NT3由平台和大量app组成，而平台又包含大量的service的不同daomain与zone的软件包。

第二，对齐软件包的难度增加了。由于大量的app和service来自于不同的团队，如果存在多种app、service软件包格式、查询机制、版本和依赖管理，将会严重影响集成的效率。

第三，难以支撑更灵活的SOTA。

Peanut Package Solution

基于上述挑战，我们设计了Peanut打包方案，希望可以提供以下的便利性：

- 在不同的团队间使用统一的包格式，可以让跨团队的打包、部署、测试更方便，缩短集成验证的周期。
- 方便不同的系统上面安装细粒度的软件包，如service、app、middleware。
- 使用统一依赖管理和版本信息，方便回溯各个软件包的版本。
- 可以使用统一的命令接口实现不同领域的包的操作，如打包、分发、安装。
- 管理各个软件包的依赖树

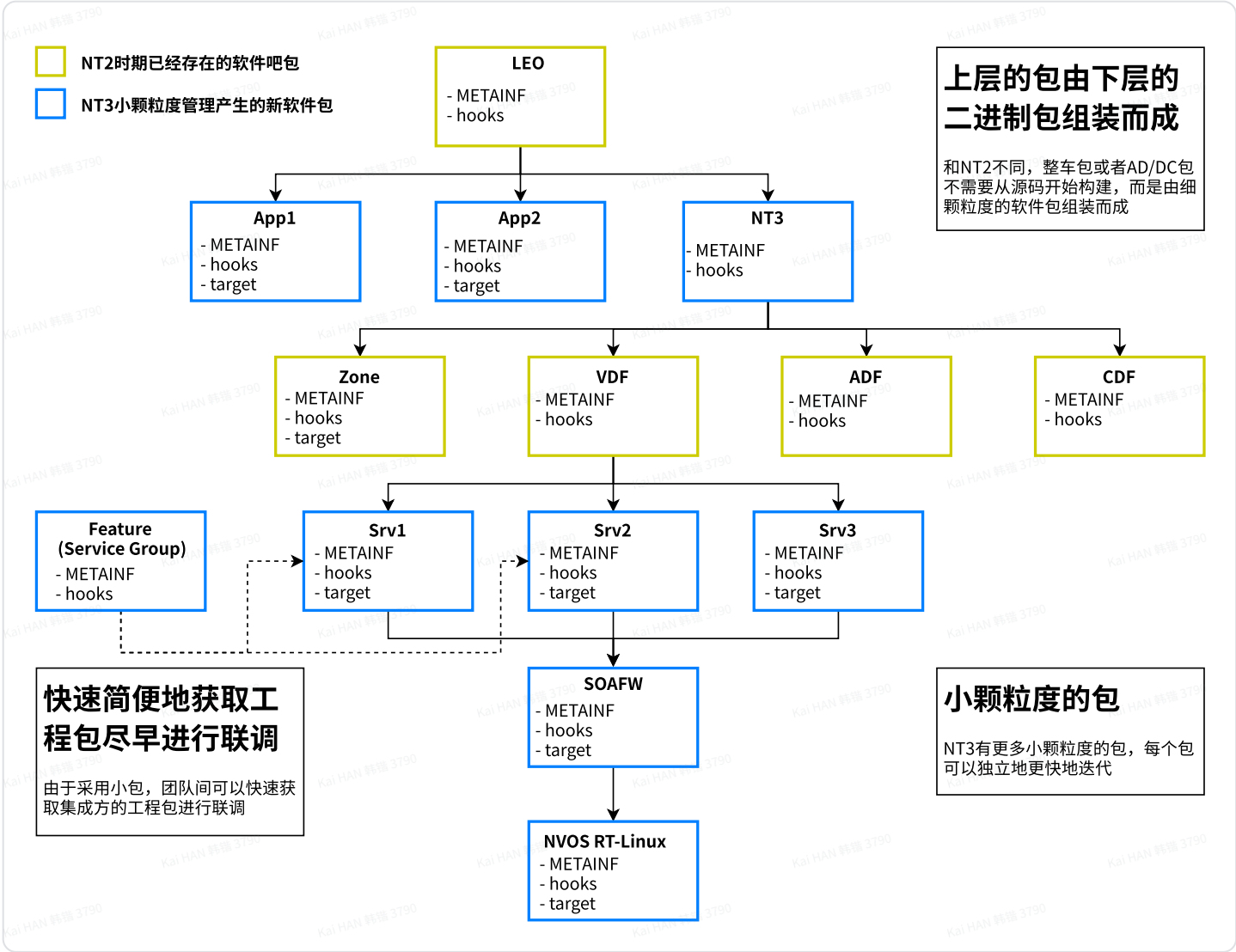
> 💡 **为什么需要一个新的包格式，而不是用rpm、deb、conan、npk、apk ...?**
>
> 1. 首先，现有的“软件包”格式分为两种，一种是用于打包、分发、传输的格式，比如rpm、deb、conan、apk；一种是可执行的包格式，比如bin、exe、jar。当然有些包同时具有两种功能。这里，我们需要解决打包、分发、传输中的统一格式、版本对齐的问题，只是NTxPKG格式的定位。
> 2. 现有NIO整车软件涉及多种架构、平台，但是没有统一的包格式可以用于多个平台上的包依赖管理和版本对齐。体现在现实情况就是，经常出现zip格式以及里面异构的包原数据描述信息，这阻碍了包的集成、测试和发布效率。

Peanut (https://git.nevint.com/nt3-pet/poc/peanut)包含三部分，

1. ntxpkg包的规范；
2. peanut命令行工具，提供ntxpkg包的制作、发布、管理依赖等功能；
3. 是软件包存储仓库（Artifactory）。

1. ntxpkg包格式

Peanut将各个层级的软件包抽象为一个树型结构，最顶层是整车软件包，包含了app和平台软件包，平台软件包包含Zone和CCC，等等，如下图所示，（下图重点示意了NT3平台软件包，关于整车软件包的管理，请参考：整车包管理方案Proposal ）



各层的软件包尽管形态各异，不过整体上分为两类，一类是作为其他包的“容器”的composite类型，比如VDF包、NT3平台包、LEO整车包，另一类是包含执行程序、数据的实际软件包component类型（compoenent类型中还包括一个特例：os类型）。这两种类型统一封装为ntxpkg格式，并且在这个统一的格式上进行依赖、版本、分发的等管理。

NTxPKG包的内部结构以及NTxPKG.yml的规范，请参考 NTxPKG Specification (v0.0.3)

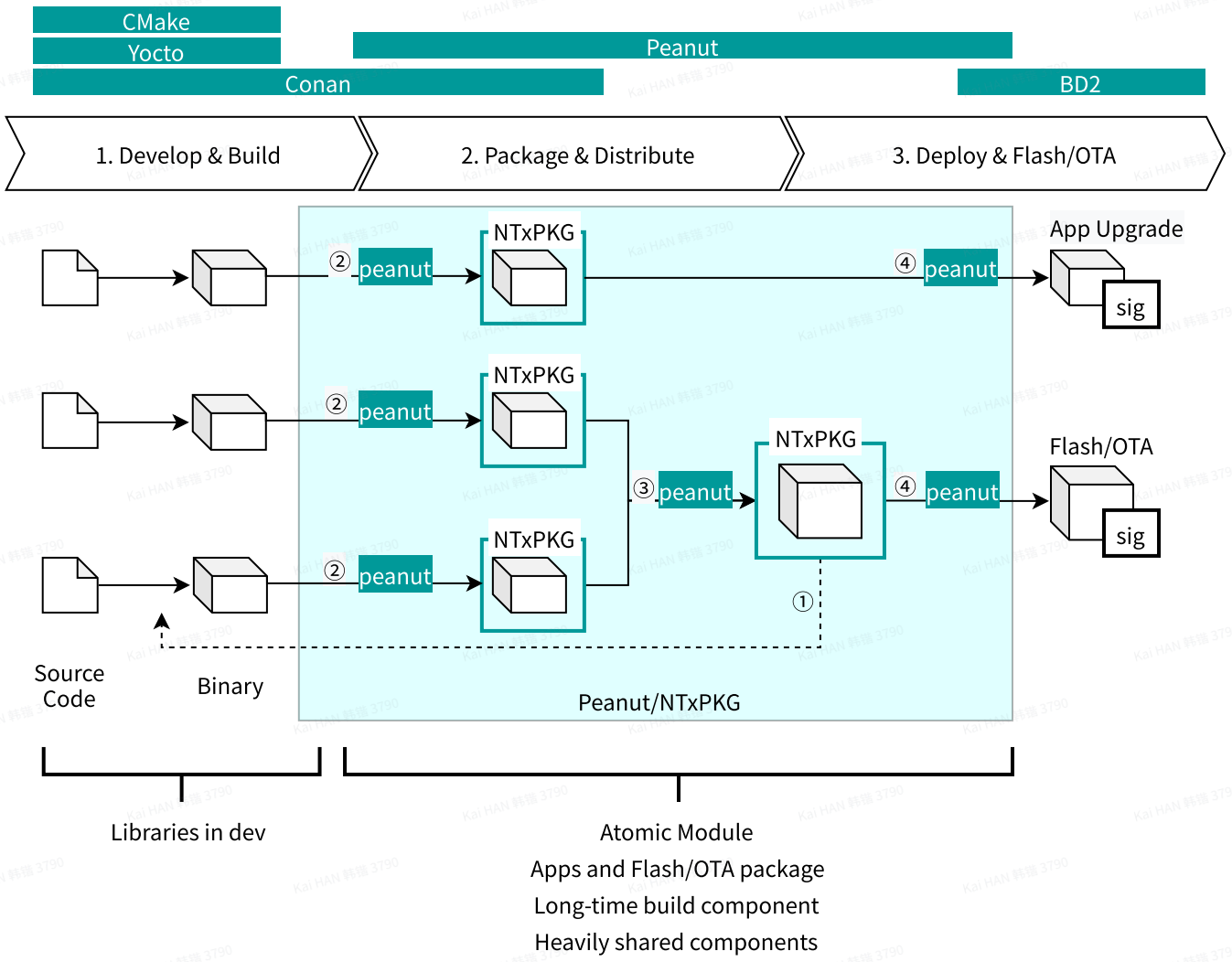
下面示例了一个包含在服务源码目录下的ntxpkg的元数据文件：

```yaml
name: LoBeamMgr
type: component
version: 1.23.456
os:
    - nvos >= 1.3
    - Ubuntu >= 18.04
arch:
    - x86
    - aarch64
devDependencies:
    - soa-framework-dev >= 1.3.1
deployDependencies:
```

```
    - soa-framework >= 1.3.1
runtimeDependencies:
    - LoBeamBehavior >= 1.1.1
hooks:
    start: my-start.sh

```

2. peanut命令行工具 - 创建、发布、安装、依赖关系图



peanut工具可以根据源码目录下的ntxpkg文件创建和发布ntxpkg软件包，并且可以安装component和composite类型的软件包到local、QEMU模拟器以及测试台架上。

创建和发布ntxpkg软件包

```
peanut create .
```

peanut命令会在当前目录(.)下面找到ntxpkg.yaml文件，根据“type”字段进行不同的处理。

- 如果type=composite，create命令会检查其包含的子包的版本是否能够对齐，例如检测所有service依赖的soa-framework版本是否一致，包版本是否在artifactory中存在。不过在create阶段，composite包并不会包含所有子包的二进制内容，只有在install阶段，composite中所有子包的二进制内容才会被下载安装。
- 如果type=component|os，create命令需要被告知如何获取软件的二进制文件（前文包规范所定义的“目标文件”），也就是软件如何被编译和打包。这个过程可以由软件包维护者自行定义，比如使用make、conan、yocto、npk、pkg等。ntxpkg会将二进制内容，以及ntxpkg.yaml和钩子脚本打包到最终的ntxpkg包中。

关于依赖项的版本

在软件目录下的ntxpkg.yaml文件中描述了依赖项的版本，即包格式规范中的packages和dependencies字段。为了支持持续集成，版本信息可以使用>=，ntxpkg打包执行时会找到latest version的依赖包。因此，在创建ntxpkg包的时候，依赖项的版本信息将由“>=”转为“==”，指向一个确定的版本，实现lock机制，这使得软件包的内容是可确定的、可回溯的。

软件包创建成功后，可以通过命令

```
peanut publish .
```

发布到Artifactory中。这个命令需要特定的权限来执行，发布的过程需要安全校验。

安装ntxpkg软件包

安装ntxpkg软件包需要面对不同的场景，比如安装一个单独的component包还是安装带有多个包的composite包，安装的目的地是local、QEMU模拟器或者台架。如下表所示，

	app/service/middleware	VDF(nvos +middleware+ service + app)	Nio(vdf+cdf+adf+zone)
local	direct	direct	N/A
simulator	ssh/direct	QEMU	QEMU
bench	ssh/adb	ssh/flash tools(uds/serial)	ssh/flash tools

尽管安装有很多种情况，peanut提供了一致的命令行接口，示例如下，

```
peanut install --target bench_001  .  # 安装当前工作区目录下创建的包至测试台架;
peanut install --target local  .      # 安装当前工作区目录下创建的包至local环境;
peanut install --target local -pkg soa.uda.1.0.2     # 安装指定Component包到local
peanut install --target simulator_001 -pkg vdf.1.1.2 # 安装指定Composite包到模拟器
```

peanut会定位到应对目标环境的包，执行里面的install脚本，将软件包安装到指定的环境上。进行包安装前，peanut会进行一致性检测，检测不同包之间依赖树冲突等问题，检测失败则无法安装。

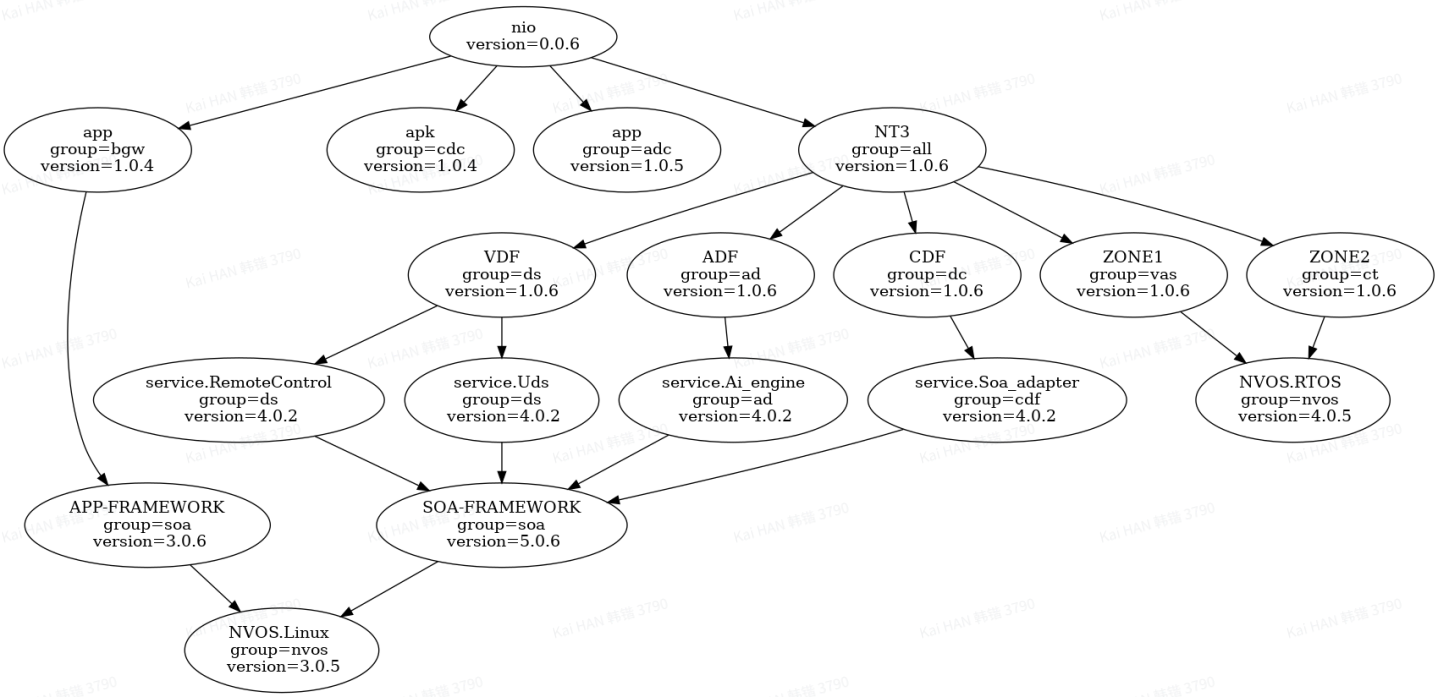
peanut会使用一个轻量的数据库来维护已安装ntxpkg软件包的信息，这个信息将用于软件包的依赖检查与更新。数据库的实现方案目前还在调研中。

依赖关系图

peanut可以生成ntxpkg包的所有子包的版本信息，

```
1 peanut dep-graphic  -pkg nio.1.0.3 # 生成1.0.3版本的nio整车包包含的所有子模块的依赖与
```

这个命令会生成类似于下图的依赖版本图，

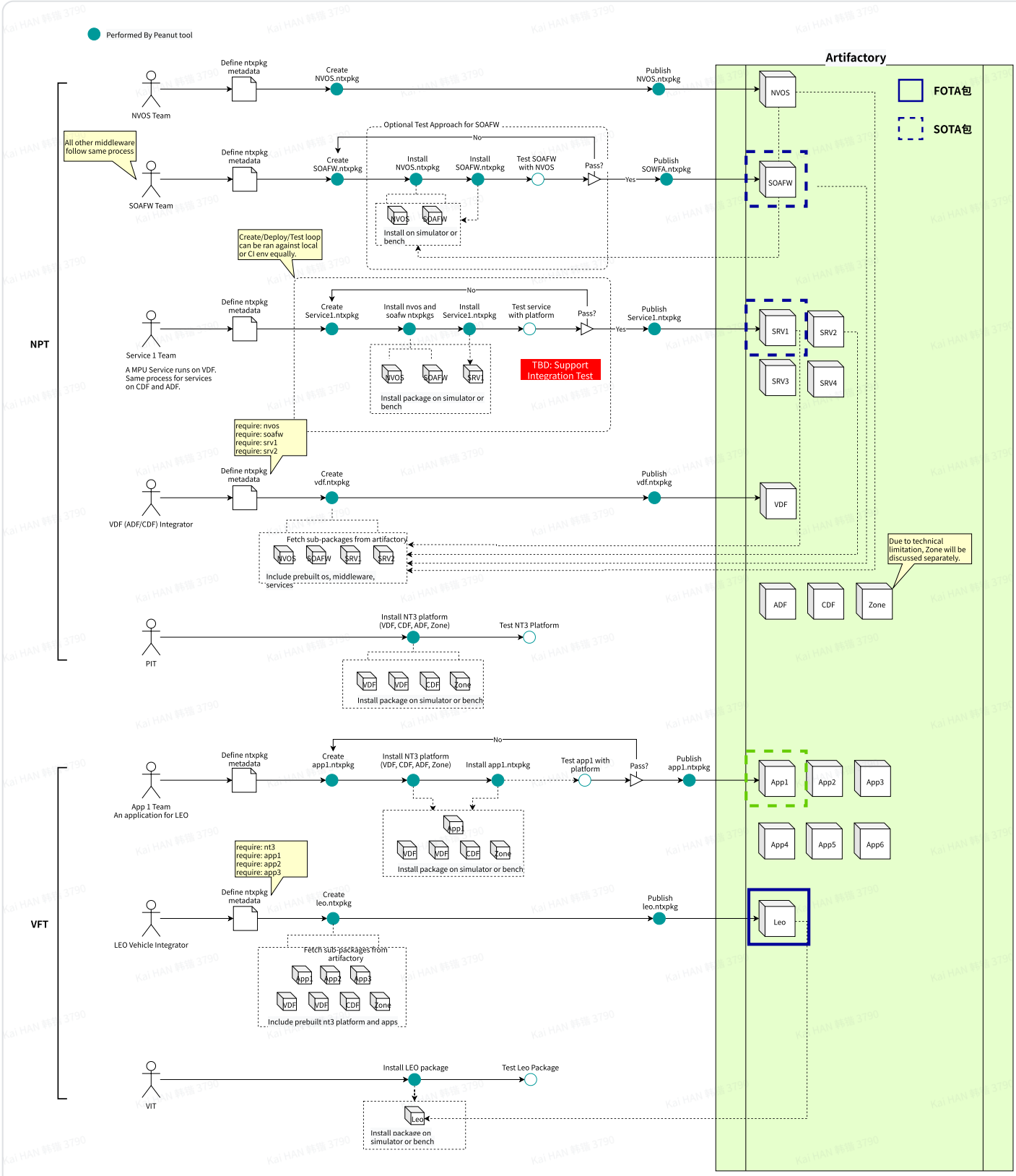


3. 软件包存储仓库

软件包存储仓库并不属于peanut方案，但是peanut需要一个实体存储机制来持久化所有ntxpkg包。业界有很多COTS产品，可以提供易用、稳定、高效、可扩展的方案。目前peanut使用JForg的Artifactory作为ntxpkg的包仓库。

场景

这部分概述了几种不同角色通过peanut进行包创建、发布、安装、测试的完整场景。



效率目标关键举措及Roadmap拆解

- POC & Showcase @ 21/Oct
- Q4在VDF/Zone上落地

目标：	**使用peanut和ntxpkg支持整车各层级软件包的管理**	
举措/ Roadmap	**D2**	
子目标1	完善peanut与ntxpkg功能，支持整车包的构建与发布	
举措1：整合ntxpkg包格式与其他包格式	完善ntxpkg包格式specification	实现与npk、apk包的
举措2：实现完备的peanut工具，支持包的构建、分发、安装、依赖管理等功能	- 建设包仓库基础设施 - 基本支持service、app和RT-Linux的ntxpkg包	- 优化命令行接口 - 加入本地缓存以提高 - 与Seed/NTxTool集成
子目标2	基于ntxpkg小包的发布节奏，改进现有的研发流程	
举措1：规范基于小包的发布、紧急修复流程	制定平台级的常规包、紧急包发布的流程	在平台研发团队实施发
举措2：左移基于小包的功能测试	- 左移服务的功能测试，在开发环境能够安装服务以及依赖的中间件和OS - 左移跨服务的集成测试	- 加快集成测试的准备 - 基于依赖关系分析集