介绍UIT的结构层次以及各种概念，描述UIT REPO与UIT Project的关系，以及这两者的迭代、更新方式

UIT 结构

德赛西威

[ 文档适用人群 3](#_Toc4231)

[ 几个概念 3](#_Toc18266)

[【UIT框架】： 3](#_Toc23645)

[【UIT REPO】： 3](#_Toc8780)

[【UIT SDK】： 3](#_Toc27977)

[【UIT Project】： 3](#_Toc29412)

[ UIT REPO 4](#_Toc14225)

[ 架构图 4](#_Toc3455)

[ BaseLine 4](#_Toc29317)

[ UIT Project 5](#_Toc22257)

[ SDK的选择 5](#_Toc22374)

[ 分支管理 5](#_Toc21573)

[ 模块组成 5](#_Toc3229)

[ 输出物 6](#_Toc6570)

[ UIT REPO与UIT Project周期 6](#_Toc18294)

[总结 7](#_Toc6569)

# 文档适用人群

* 框架制定以及维护者
* UIT框架技术模块开发者
* KANZI GUI工程开发者、Qt GUI 工程开发者
* 使用UIT框架的平台
* 使用UIT框架的项目

# 几个概念

对于【UIT框架】这个概念，一般而言，我们说的都是广义的。因为UIT在给外部提供支持时，除了提供必要的SDK部分，往往还包含运行时的整体套件，比如运行时脚本，配置文件，以及运行时加载的各种资源如KZB，font等。总而言之，【UIT框架】提供的是一整套与UI相关的解决方案，它是完整的，独立的，可持续迭代的。

为了后面在叙述UIT相关内容时不产生混淆，我们先明确这几个概念：UIT框架，UIT REPO，UIT SDK，UIT Project。

## 【UIT框架】：

提供与UI相关的一整套独立，完整的解决方案框架，它在概念上是最大的一个集合。

## 【UIT REPO】：

包含了各技术模块的源码和环境整合，编译后，它将输出UIT SDK。

## 【UIT SDK】：

由UIT REPO输出，提供给具体项目进行二次开发的软件开发包。

## 【UIT Project】：

是具体的平台或者项目，它使用UIT SDK进行具体实施，并集成所有UIT套件输出到相关的客户项目。

这样划分，可以保证UIT REPO和UIT Project之间的独立性。做到，即兼顾UIT REPO的规划性、迭代性和可控制性，使它的技术路线图不受当前支持的单个或多个项目的影响；又保证了使用UIT的各Platform或者客户项目，得到可控的长期稳定版本支持（相当于LTS）。

下图为UIT框架与应用项目的之间的示意图。

UIT REPO

SDK

UIT Project

UIT框架

CES

...

Mazda Project

Geely Project

Haval Project

SGMW Project

# UIT REPO

UIT REPO并非是单独的一个模块，它统一管理控制着各种UI技术模块，包括：编译环境、基础日志模块、Runtime环境、数据驱动、kanzi GUI、Qt Gui、图层管理等。也就是说，UIT REPO是一个技术方案组，它的职责是制定和实现与UI相关的技术方案的评估和实施，并有着自身的迭代路线rollmap和版本控制。它对外部的的输出为UIT SDK。

## 架构图

请查看<【uit】【框架】Arch.pptx>

## BaseLine

UIT REPO将采用BaseLine机制，即可控的分支版本管理，如UIT 1.0, UIT 1.1, UIT1.2，UIT2.0等。这样的管理方式，可以有效地对UIT REPO的各模块进行统一的版本管理控制。

一般而言，大版本的迭代更新，意味着较多较大的新特性支持，而小版本的迭代，一般是较小的更新。例如，UIT 1.x支持数据驱动，到UIT 2.x开始支持多GUI方案，UIT3.x开始支持云服务等，这些是大版本的更新迭代。而更诸如更换编译器、追加第三方开源库支持等，这些则属于小更新，迭代我1.1, 1.2这样的小版本。

这样说来，是否启用新的版本，是由新特性的迭代作为依据的。

# UIT Project

UIT Project是为了满足各应用项目或平台的长期稳定支持而产生的。

## SDK的选择

一方面，UIT Project的输入是UIT SDK， UIT Project需要评估选择UIT REPO的BaseLine，也就是说，每个不同的UIT Project对应的UIT REPO BaseLine是可以不一样的。而选择不一样的依据，则需要根据需要的UIT REPO特性来决定。例如，我需要UIT 1.0的特性，那么我就选用UIT 1.0 版本的UIT SDK，反之如果我需要UIT 2.0的特性，那么我就选用2.0版本的SDK。

## 分支管理

与UIT-REPO类似，UIT Project也有它的分支管理。不同的是，UIT Project的分支名字与具体项目名非常密切，因为它已经是跟具体项目紧密联系了。分支名可以叫Mazdaxxx，g6sh等。

简单来说，一个分支对应一个具体项目，这样既可进行长期稳定的支持。

## 模块组成

与UIT REPO一样，UIT Project并非单独的模块，而是包含了多个部分。主要部分是UI工程，如kanzi UI工程、Qt UI工程等；另外，也可能包含app工程（虽然UIT一般不负责app部分，但如果是比较特殊的项目如CES等，则需要包含该部分）。

它的结构大致如下：

UIT Project

UIT SDK

KANZI UI Projects

QML Projects

APPS

## 输出物

在交付给使用UIT框架的第三方时（BU项目、外包等），将基于UIT Project的具体分支进行集成，也就是说，如果UIT Project有两个分支，分别对应两个具体项目，则集成动作是分开进行的，输出物为集成后的整体UIT包。

集成方式请查看<【uit】【框架】：编译与集成.docx>

# UIT REPO与UIT Project周期

它们周期和支持关系如下图所示：

***UIT REPO***

***BL 1.0***

***BL 2.0***

***BL 3.0***

***Project A***

***Project B***

***Project C***

***Project D***

图中我们可以看出，UIT REPO的BaseLine会一直向前推进和迭代，不受当前支持项目的影响。

Project则依据需要的UIT REPO特性，选定一个BaseLine，UIT REPO在此BaseLine上提供长期稳定支持。直到所有基于此BaseLine的Project都顺利release。

# 总结

综上所述，对内，UIT框架将具备完好的迭代能力；UIT框架将提供完整的UI解决方案和相关开发支持，通过BaseLine的选择和分支的建立，达到任意项目的长期稳定维护和支持。

BU在使用UIT框架时，几乎不需要关心与UI相关的方案评估和具体实现，而只需把精力放在底层功能和业务逻辑即可。如此，可大大地提高工作效率和项目交付能力。