CPP-Summit 2019

全球C++软件技术大会

C++ Development Technology Summit

Boolan

高端IT互联网教育平台

关注“博览Boolan”服务号

发现更多 会议·课程·活动

CPP-Summit 2019

面向产品系列的嵌软架构设计 辅导心得

温　昱
代码重构|架构优化|首席顾问

议程

1

改方法 接地气

分享点：上游接功能、中游抱接口、下游衔技术

2

嵌软架构 产出标准

分享点：文档等问题

3

需求变更/型号变化 应对

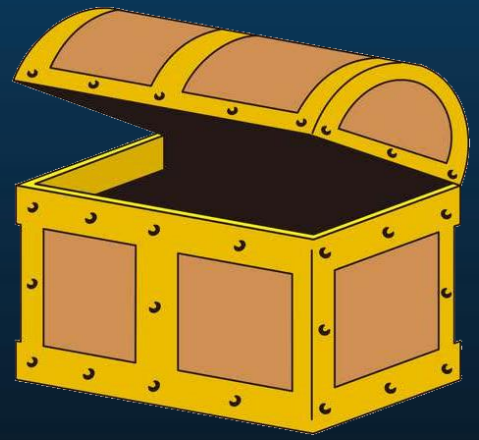
分享点：变更分析技巧、接口设计技巧

4

嵌软平台 部分思考

分享点：平台需求规范 与 平台架构设计框架

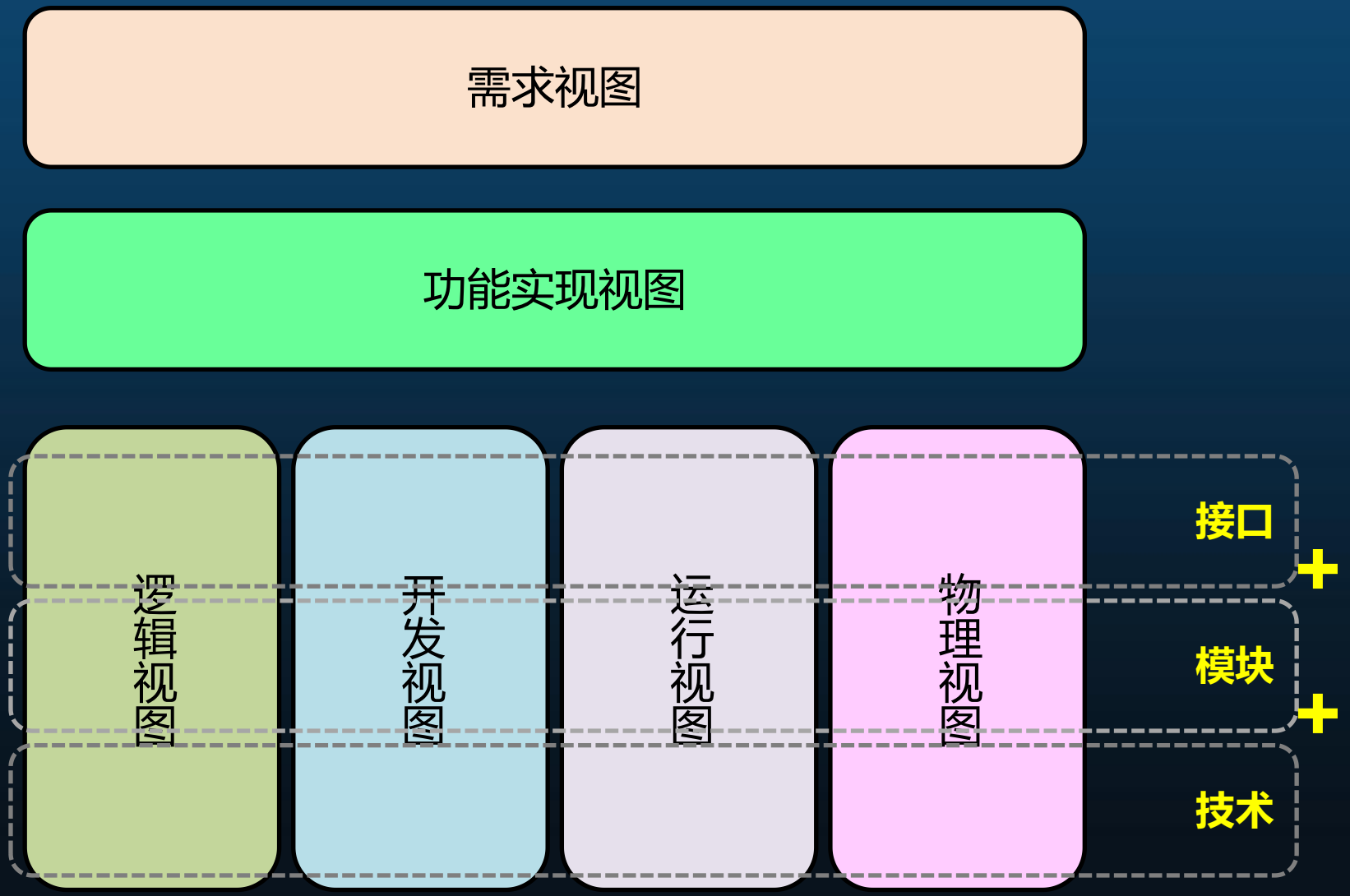
企业问：
很多理论脱离实际，
嵌软可参考的成套实践更少



对，是的！

上游接功能、中游抱接口、下游衔技术

做不到的都是假方法


需求视图
功能实现视图
逻辑视图
开发视图
运行视图
物理视图
接口
模块
技术


需求视图
上下文图
功能范围
操作与处理流程
UI 需求
质量需求
识别来自上游方案的约束
识别来自企业技术规范的约束
关键功能：跨模块/跨任务/跨控制器的功能实现
模块接口
任务接口、线程接口
集成接口
层划分
子系统划分
代码模块划分
工程划分
开发语言选择
库/框架选择
进程/线程划分
任务/中断划分
硬件网络结构
软件部署结构
基础设施选择
群集/主从/互备选择
逻辑视图
开发视图
运行视图
物理视图
接口
+
模块
+
技术

议程

1

改方法 接地气

分享点：上游接功能、中游抱接口、下游衔技术

2

嵌软架构 产出标准

分享点：文档等问题

3

需求变更/型号变化 应对

分享点：变更分析技巧、接口设计技巧

4

嵌软平台 部分思考

分享点：平台需求规范 与 平台架构设计框架

心得分享 10分钟

议程

1

改方法 接地气

分享点：上游接功能、中游抱接口、下游衔技术

2

嵌软架构 产出标准

分享点：文档等问题

3

需求变更/型号变化 应对

分享点：变更分析技巧、接口设计技巧

4

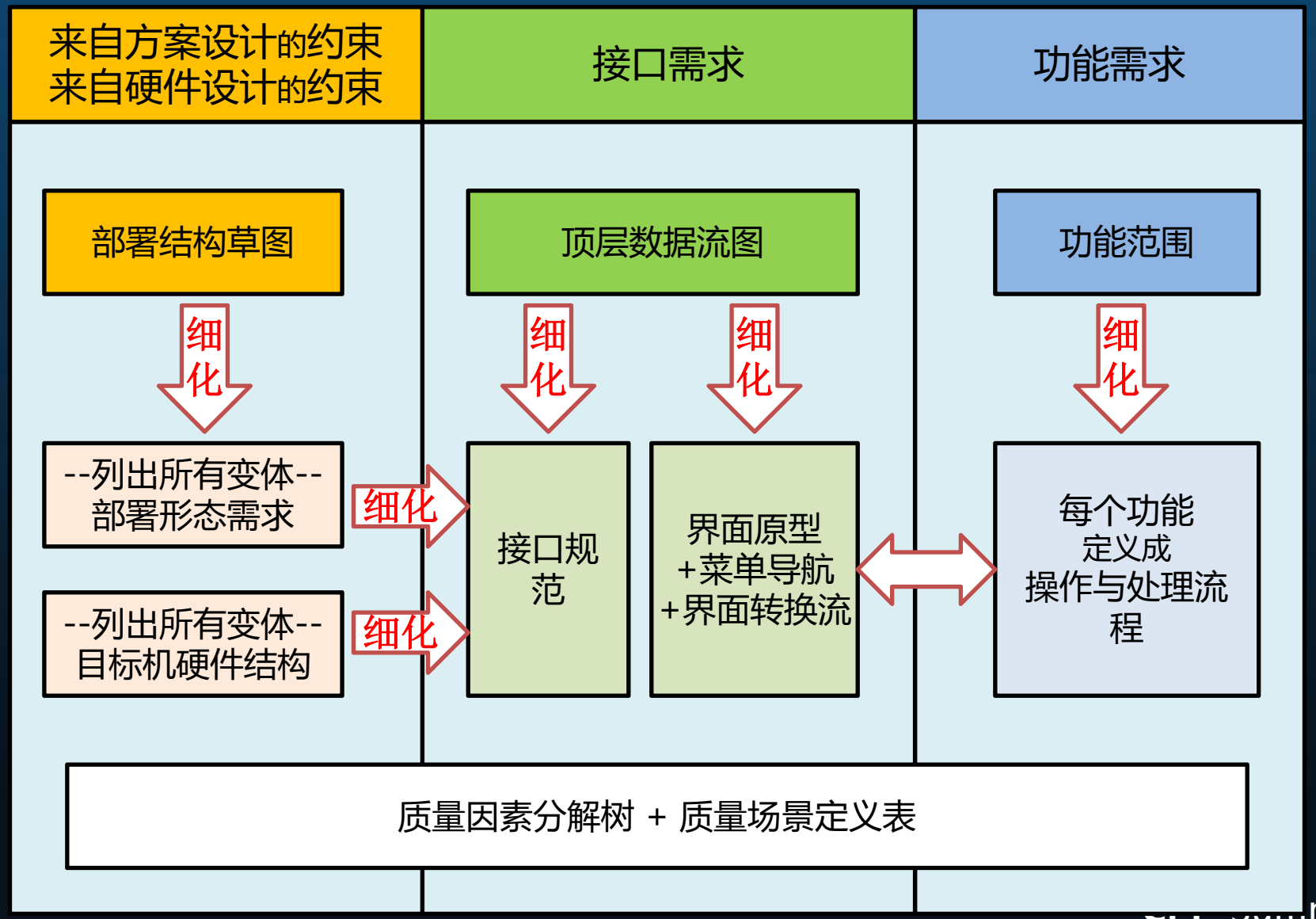
嵌软平台 部分思考

分享点：平台需求规范 与 平台架构设计框架

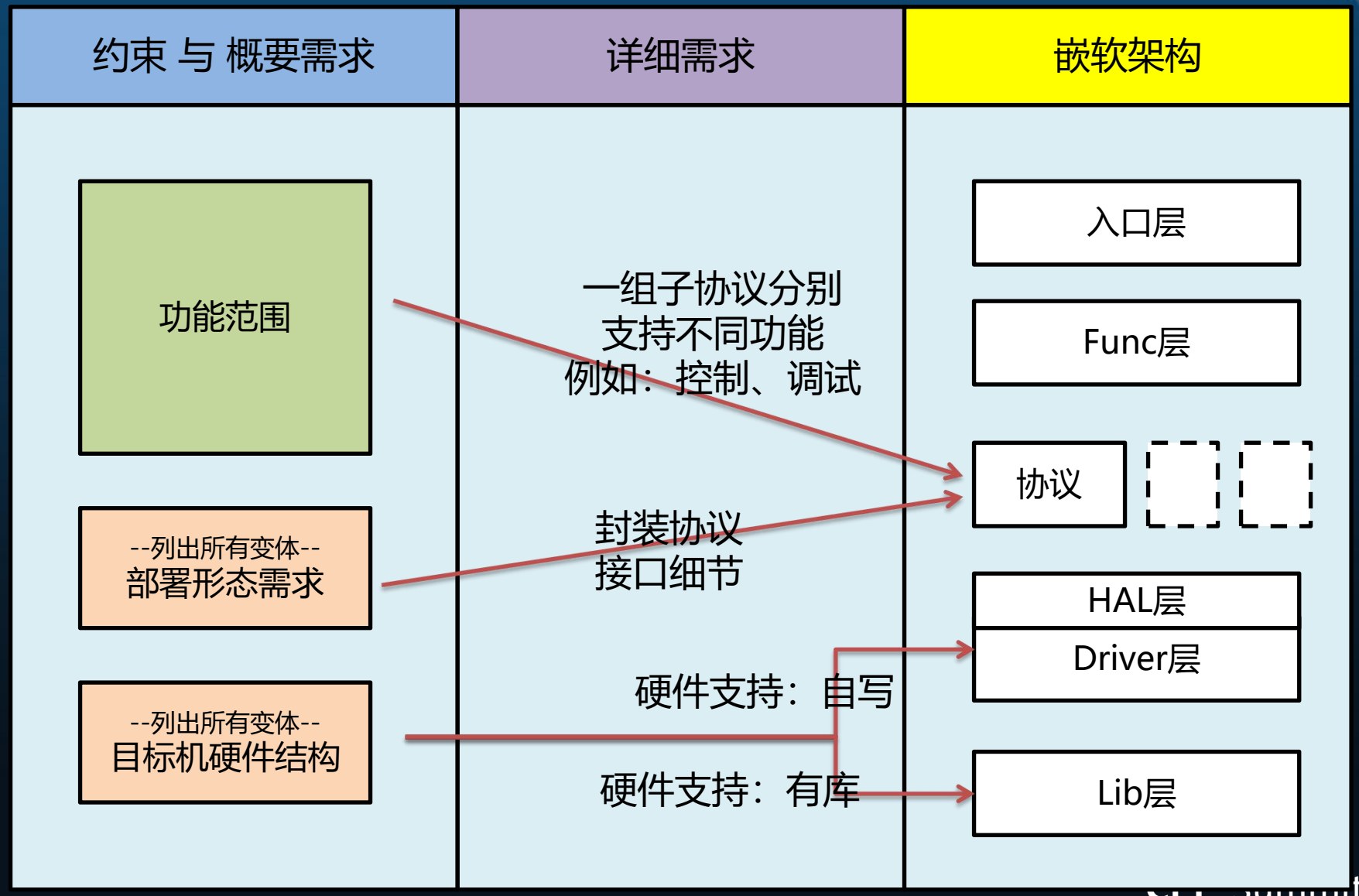
企业问：

产品系列，需求变更和型号变化怎么应对

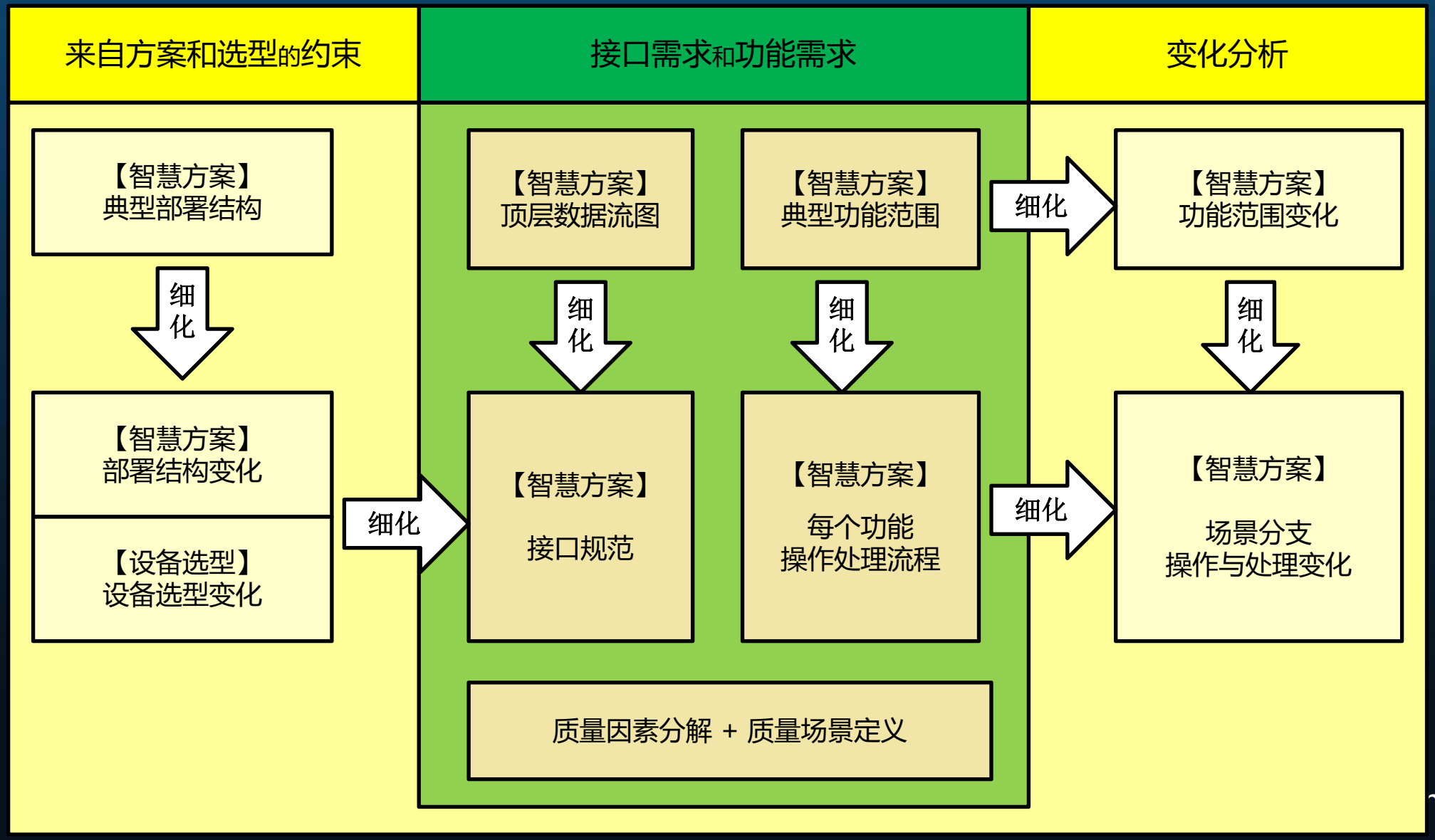
嵌软需求——分析内容框架


来自方案设计的约束
来自硬件设计的约束
接口需求
功能需求
部署结构草图
顶层数据流图
功能范围
细化
细化
细化
细化
--列出所有变体--
部署形态需求
--列出所有变体--
目标机硬件结构
细化
细化
接口规范
界面原型
+菜单导航
+界面转换流
每个功能
定义成
操作与处理流程
质量因素分解树 + 质量场景定义表

嵌软需求——分析内容框架：价值管窥



智慧方案——需求分析内容框架

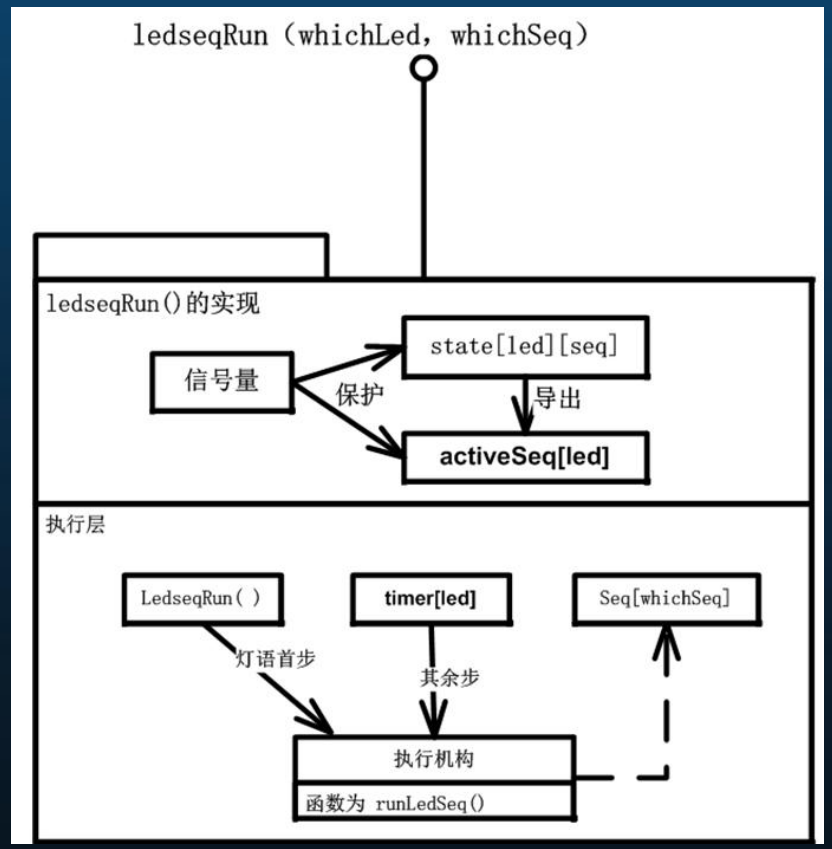


产品化
功能深度定义一例 5分钟

两多一少目标定律

功能多，接口少，参数值多，

能做到两多一少，才叫抽象。

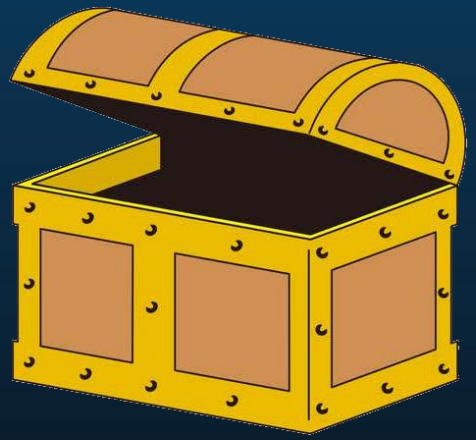



```
//TODO: Change, right now is called so fast it looks like seq_lowbat
const ledseq_t seq_altHold[] = {
  { true, LEDSEQ_WAITMS(1)},
  {false, LEDSEQ_WAITMS(50)},
  {    0, LEDSEQ_STOP},
};

const ledseq_t seq_linkup[] = {
  { true, LEDSEQ_WAITMS(1)},
  {false, LEDSEQ_WAITMS(0)},
  {    0, LEDSEQ_STOP},
};


const ledseq_t seq_charged[] = {
  { true, LEDSEQ_WAITMS(1000)},
  {    0, LEDSEQ_LOOP},
};

ledseq_t seq_charging[] = {
  { true, LEDSEQ_WAITMS(200)},
  {false, LEDSEQ_WAITMS(800)},
  {    0, LEDSEQ_LOOP},
};

ledseq_t seq_chargingMax[] = {
  { true, LEDSEQ_WAITMS(100)},
  {false, LEDSEQ_WAITMS(400)},
  {    0, LEDSEQ_LOOP},
};
```

强力手段：Type通配

① 强制类型转换、RTTI运行时类型识别

② C/C++的void *、union



③ Java Flag接口、反射机制

④ Java/JS的instanceof()、typeof()

⑤ C++的xxxx_cast...

⑥ C/C++的宏，C++模板，Java泛型

都是重要的接口实现手段。

TYPE通配—— C/C++ UNION

TYPE通配—— C/C++ UNION

```
typedef struct CvMat
{
    int type;
    int step;

    /* for internal use only */
    int* refcount;
    int hdr_refcount;

    union
    {
        uchar* ptr;
        short* s;
        int* i;
        float* fl;
        double* db;
    } data;

    #ifdef __cplusplus
        union
        {
            int rows;
            int height;
        };

        union
        {
            int cols;
            int width;
        };
    #else
        int rows;
        int cols;
    #endif

}
CvMat;
```

TYPE通配—— C/C++ VOID *

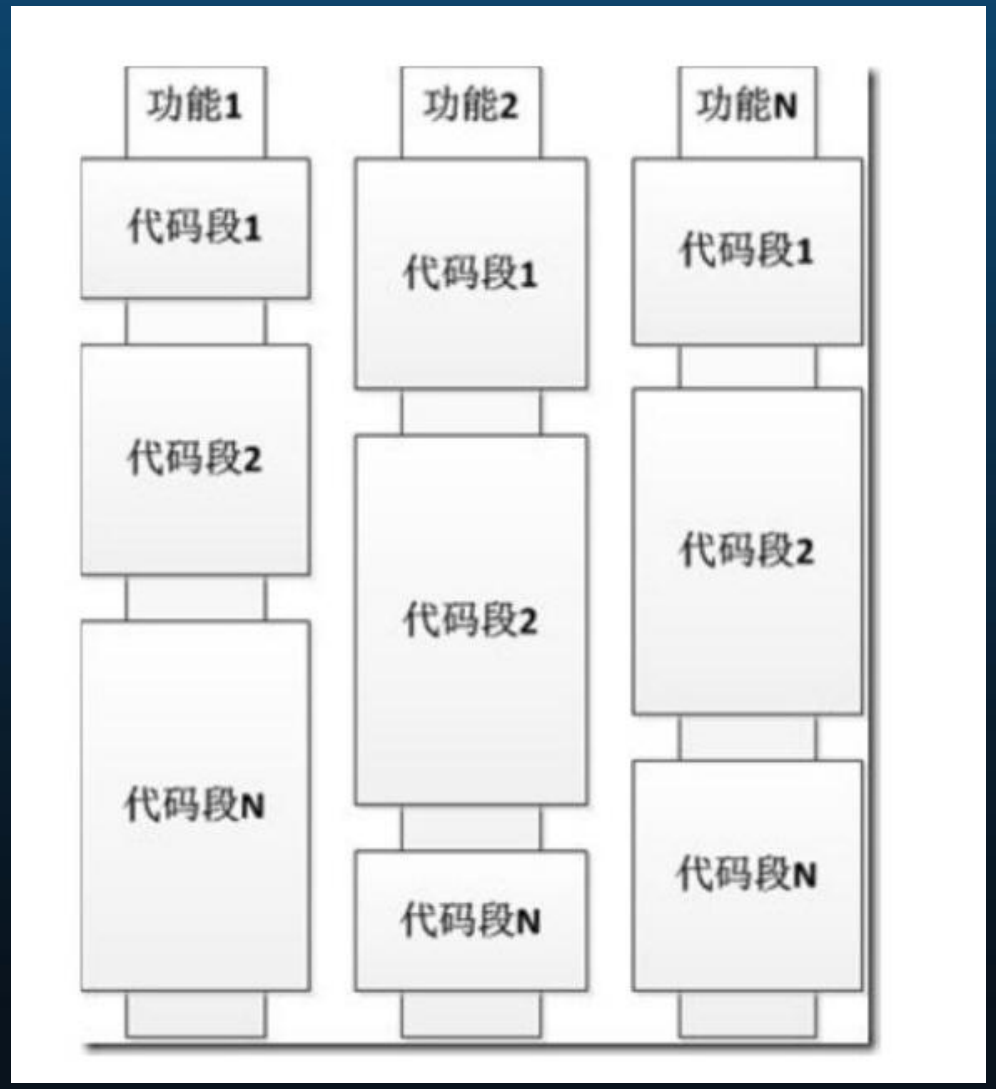
```
void qsort(
    void *base,
    size_t num,
    size_t width,
    int (__cdecl *compare )(const void *elem1, const void *elem2 )   );
```

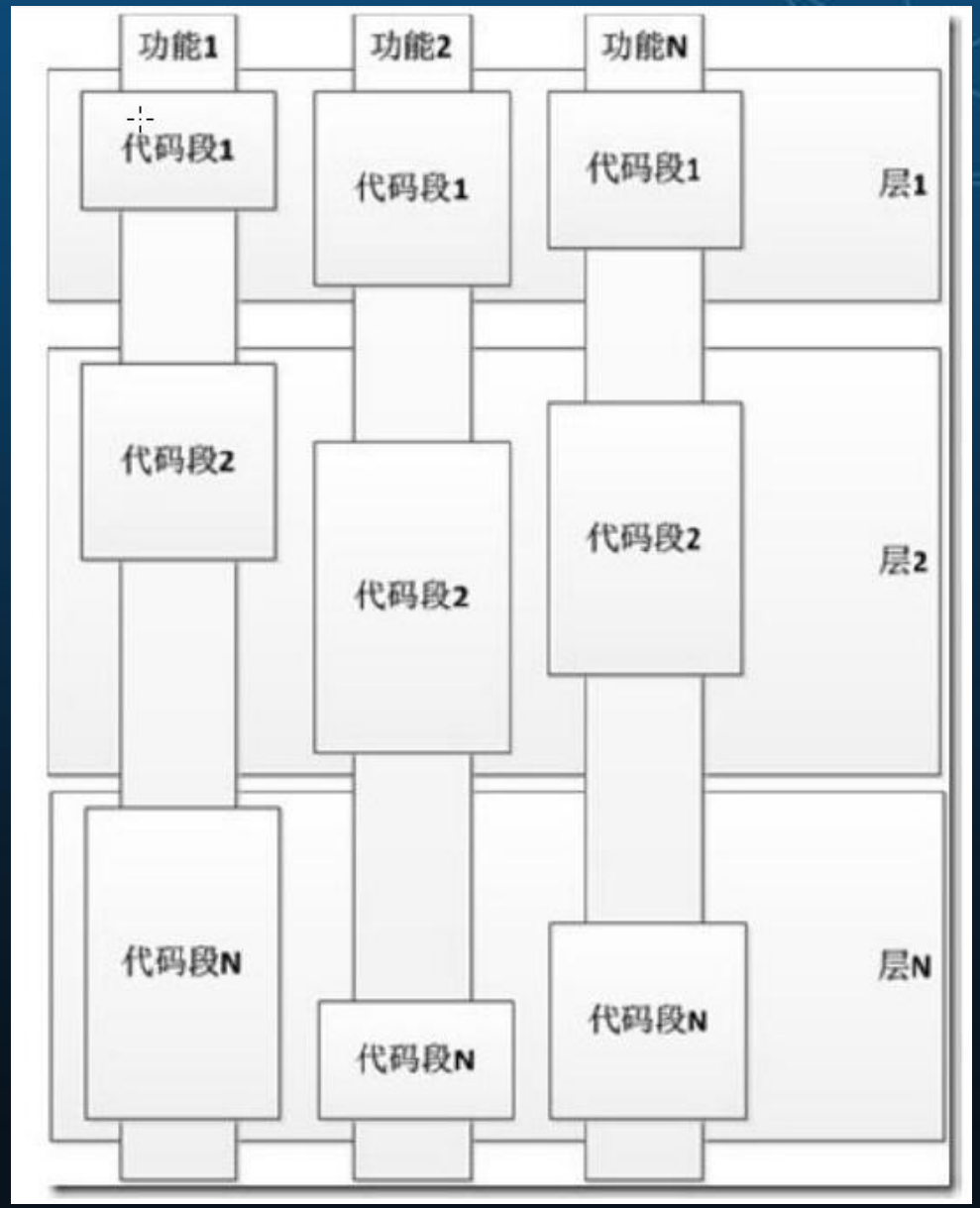
```
struct employee {
  int employee_num;
  char employee_name[100];
  } employees[100];

int cmp( const void *a , const void *b )
{
   struct employee *c = (employee *)a;
   struct employee *d = (employee *)b;
   return c-> employee_num
            - d-> employee_num;
}
qsort (employees,100, sizeof(employees [0]),cmp);
```

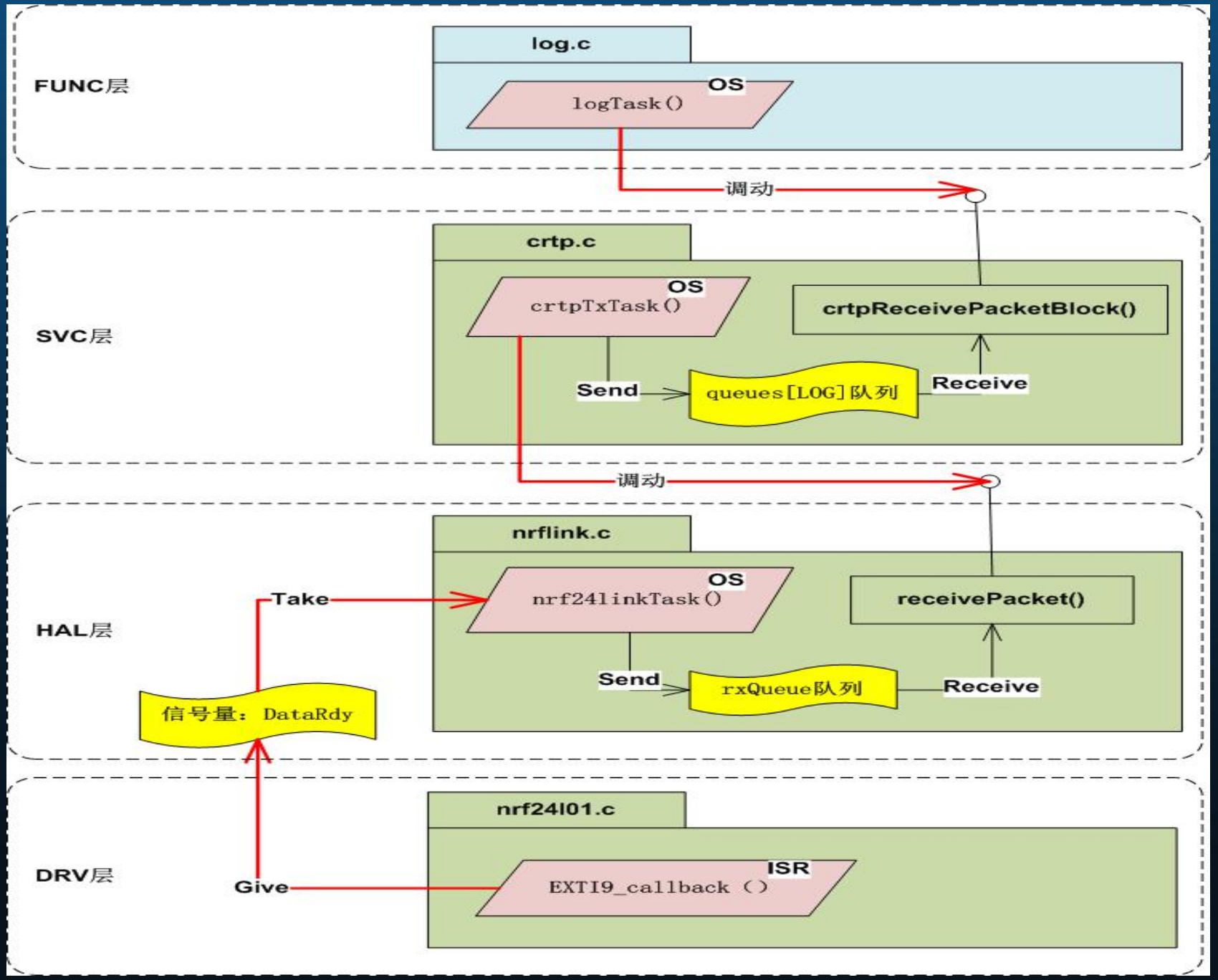
```
int num[100];

int cmp ( const void * i , const void * j )
{
    return *(int *)i - *(int *)j;
}
qsort (num,100,sizeof(num[0]),cmp);
```

嵌软架构核心：运行架构

- 前台任务 & 后台任务
- Task & Task/ISR
- Thread & Thread


功能1
代码段1
代码段2
代码段N
功能2
代码段1
代码段2
代码段N
功能N
代码段1
代码段2
代码段N


功能1
功能2
功能N
代码段1
代码段1
代码段1
层1
代码段2
代码段2
代码段2
层2
代码段N
代码段N
代码段N
层N


FUNC层
log.c
OS
logTask()
调动
SVC层
crtp.c
OS
crtpTxTask()
crtpReceivePacketBlock()
Send
queues[LOG]队列
Receive
调动
HAL层
nrflink.c
OS
nrf24linkTask()
receivePacket()
Take
Send
rxQueue队列
Receive
信号量：DataRdy
DRV层
nrf24l01.c
ISR
EXTI9_callback ()
Give

```
////////////////////////////////////////////////////
//            programming by config
////////////////////////////////////////////////////

int judgeMotorStatus()
{
    return 1;
}

void doBraking()
{
    printf("\n  this is a job");
}

void reportResult()
{
    printf("\n  this is a job");
}

JobDefine  brakeFrontendTask[] = {
  {"brake job",    judgeMotorStatus,  doBraking,    reportResult},
  {"only report", judgeMotorStatus,  NULL,         reportResult},
  {"only do",      judgeMotorStatus,  doBraking,    NULL},
  {NULL,           NULL,              NULL,         NULL},
};

interrupt CpuTimer0ISR()
{
    MULTI_JOB_RUN(brakeFrontendTask);
    return 0;
}
```

```
////////////////////////////////////////////////////
//            job framework
////////////////////////////////////////////////////

typedef int  (*JudgeFunc) (void);
typedef void (*LocalJob)  (void);
typedef void (*RemoteJob) (void);

typedef struct {
    char*     name;
    JudgeFunc judge;
    LocalJob  localJob;
    RemoteJob remoteJob;
} JobDefine;

void JobRun(JobDefine jobs[], int single)
{
    for (int i=0; NULL != jobs[i].name; i++)
    {
        if( jobs[i].judge() )
        {
            if(NULL != jobs[i].localJob)  jobs[i].localJob();
            if(NULL != jobs[i].remoteJob) jobs[i].remoteJob();
            if(single) return;
        }
    }
}

#define SINGLE_JOB_RUN(jobs) JobRun(jobs, 1);

#define MULTI_JOB_RUN(jobs) JobRun(jobs, 0);
```

议程

1

改方法 接地气

分享点：上游接功能、中游抱接口、下游衔技术

2

嵌软架构 产出标准

分享点：文档等问题

3

需求变更/型号变化 应对

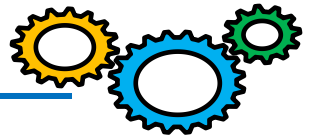
分享点：变更分析技巧、接口设计技巧

4

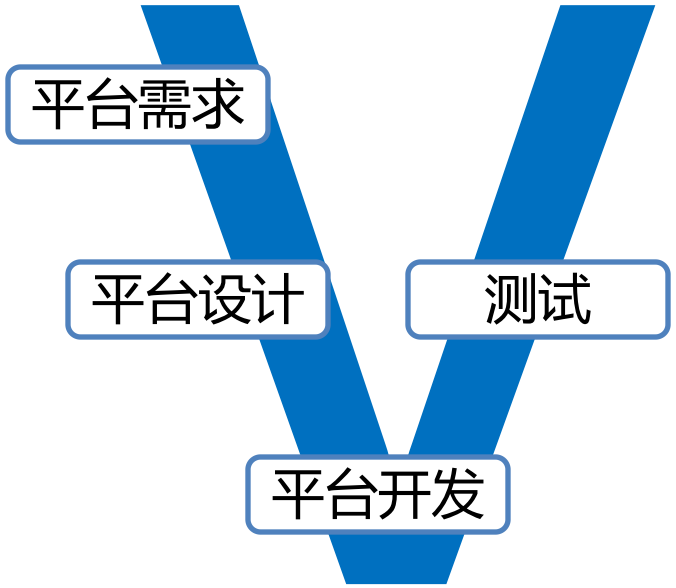
嵌软平台 部分思考

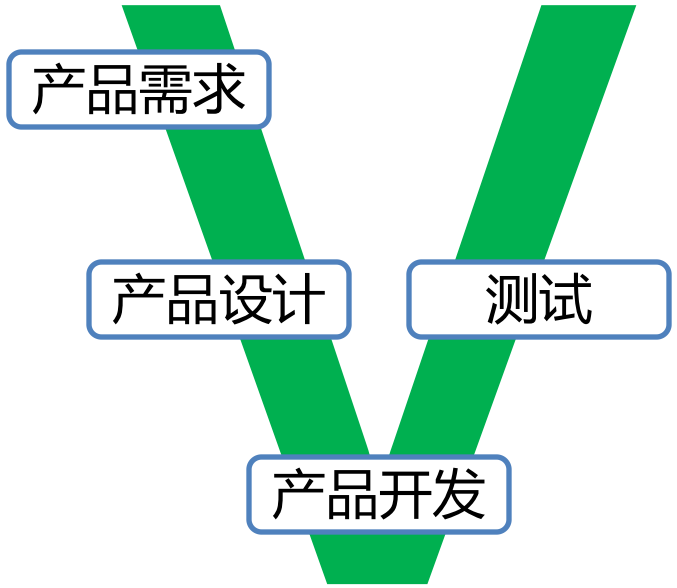
分享点：平台需求规范 与 平台架构设计[illegible]案

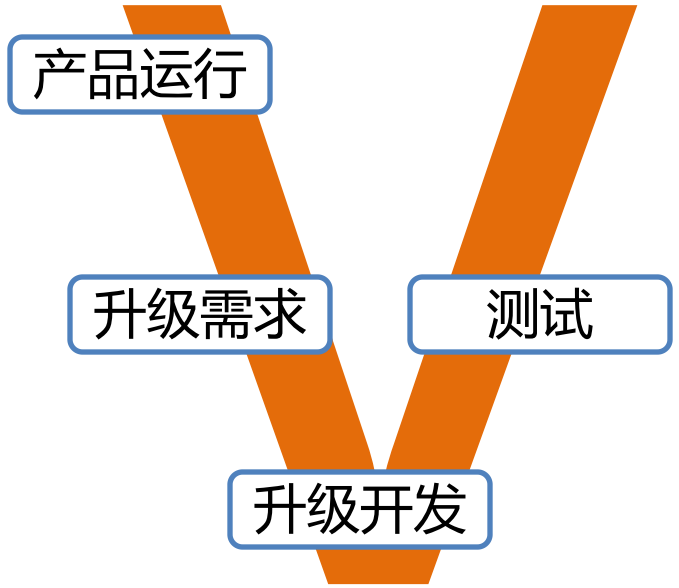
平台/产品生命周期全景



平台研发期 | 产品研发期 | 产品维护期

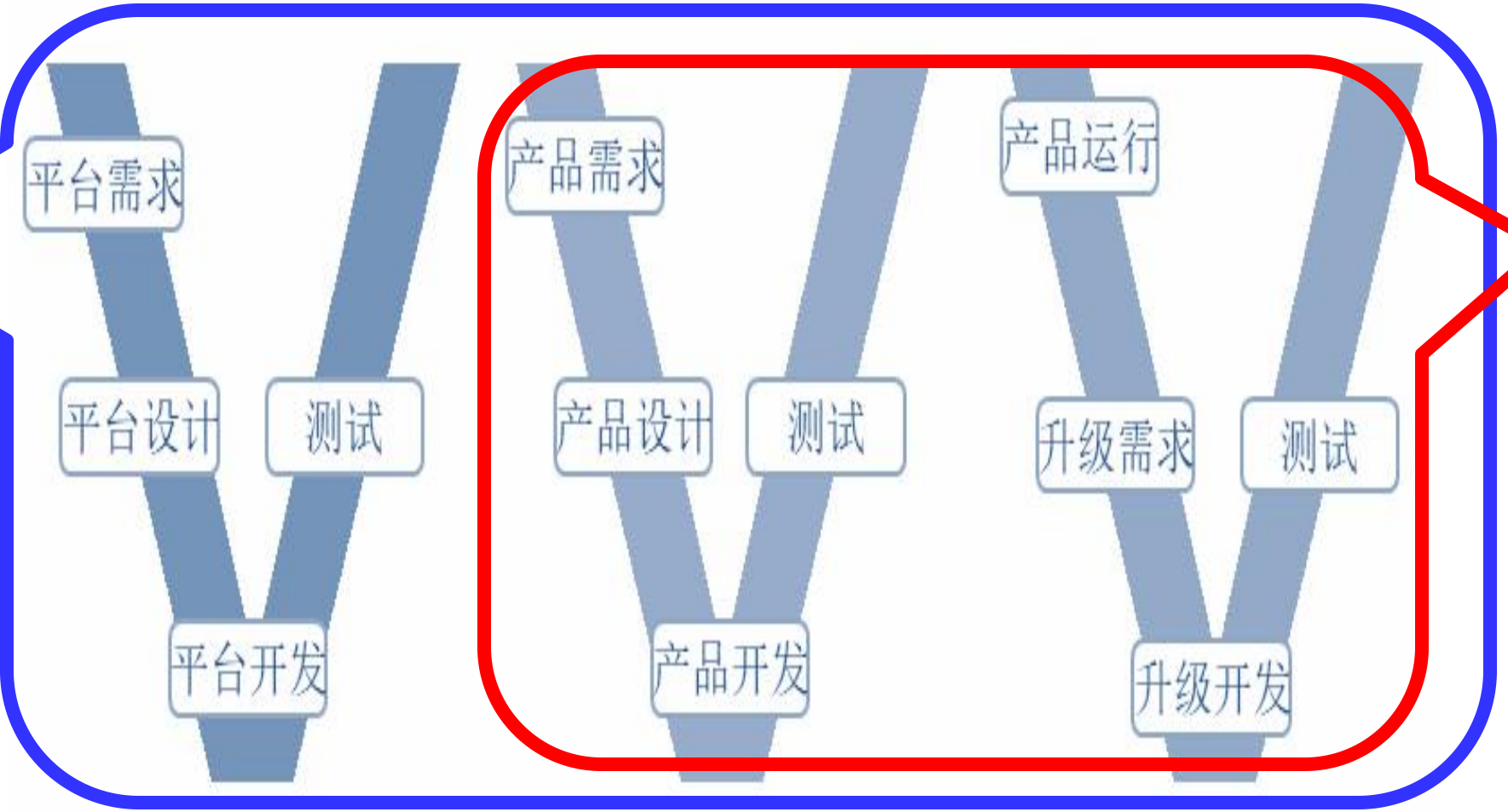






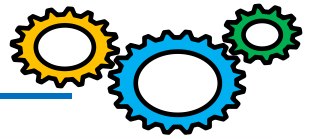
平台/产品架构师岗位的使命

平台架构师



产品架构师

嵌软平台架构——设计内容框架



需求视图

- 功能需求
- 集成需求
- 设备无关需求
- 部署灵活需求
- 性能需求
- 扩展需求
- 产品开发支持
- 调试诊断监控支持

逻辑视图

- 平台分层架构
- 平台子系统划分
- 子系统间协作接口

开发视图

- 平台组件划分
- 平台开发代码工程
- 平台开发涉及技术

运行视图

- 平台的总体多进程结构
- 平台进程间的通信

物理视图

- 多种部署模式
- 网络结构
- 软件运行环境
- 平台组件部署
- 产品组件部署

数据视图

- 持久化机制
- 数据模型
- 数据分布
- 数据同步

产品开发支撑视图

总览

- 基于平台的开发调试总体分层结构

产品设计规范

- 发布包结构规范
- 发布包mount路径规范
- 进程通信方式规范
- 其他规范……

编程式开发

- API开发接口
- OpenAPI开发接口
- 作为SDK的库与框架

图形化开发

- 模型与资源文件格式设计
- 模型与资源文件运行原理

谢谢