深向

28, 05, 2024 | 北京

基于模型设计满足汽车软件质量和快速交付的挑战

柳少康, 深向科技有限公司

盖亮, 深向科技有限公司

MATLAB EXPO

深向

品牌介绍

我们是一家科技公司，专注于智能新能源卡车的研发和制造，

致力于开创一个全新的智慧货运时代

中国首家实现量产交付电动重卡新势力

由自动驾驶技术领跑者百度与公路干线物流产业互联网头部企业狮桥联手打造

Chapter 1：汽车软件质量属性的实践

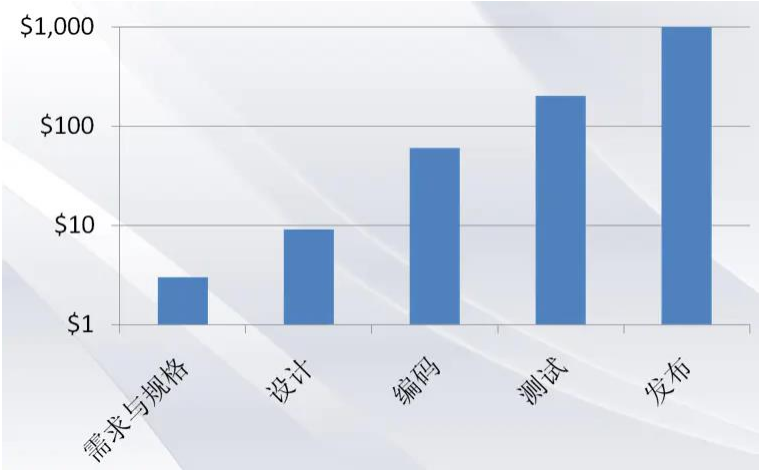
Chapter 2：汽车软件快速交付的挑战

Chapter 1：汽车软件质量属性的实践

Chapter 2：汽车软件快速交付的挑战

引入软件质量属性的目的

- 软件开发过程中，修复缺陷的成本随开发环节推进而呈**指数上升**
- 不注重软件质量属性，会产生技术负债，未来不得不花费**更大的代价**重构软件
- 不注重软件质量属性，会导致软件反复修改，**团队产出效率低**，不利于快速交付



技术负债			不良后果
架构负债		代码冗长	代码缺陷
文档负债	编程风格混乱	代码命名混乱	代码可维护性差
测试负债		代码异味	团队产出效率低

汽车软件质量属性的实践

- **基本类**

  可用性、安全性、合理性/可行性、鲁棒性/容错性

- **维护类**

  平台化/兼容性、继承性、追溯性、KISS原则、易修改、可读性

- **标定类**

  标定类要求

- **调试类**

  观测量设置

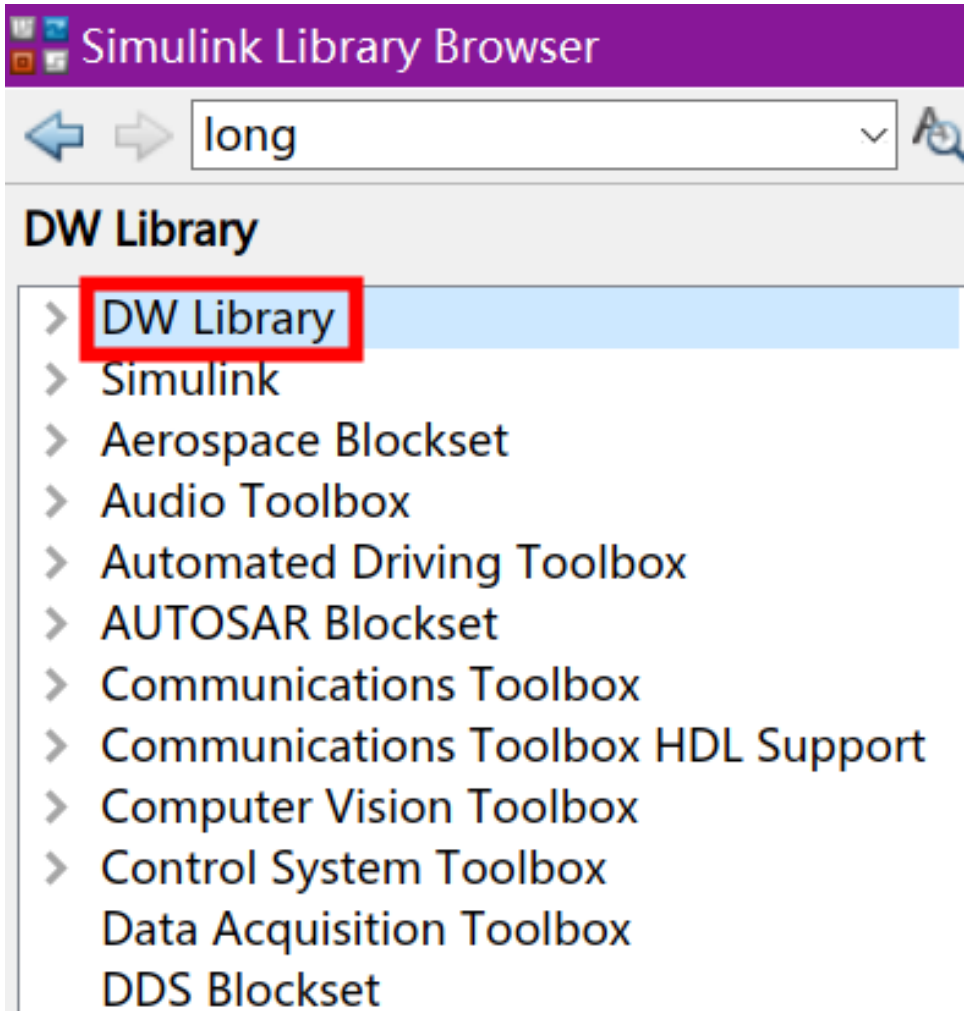
实践1：可用性

- 统一的建模工具、建模配置、代码生成配置
- 统一的自定义模块库
- 统一管理的SWC库、SWC名称
- 统一的SWC模板、数据字典模板
- 统一的建模规范、命名规范、缩写表；检查工具
- 统一的单元测试工具
- 软件开发过程List：

  Model Review

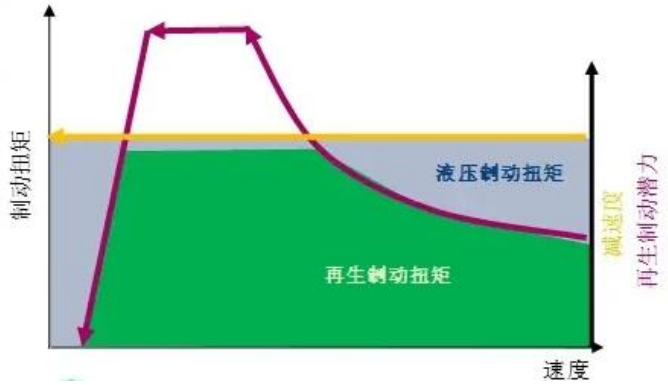
  建模规范检查、命名规范检查

  单元测试、背靠背测试



实践2：合理性/可行性

- 使用Simulink设计数据流、控制算法
- 使用Stateflow设计状态机、流程图、调度器、组合逻辑
- 使用Embedded MATLAB做矩阵运算
- 使用一阶惯性滤波处理传感器信号噪声
- **使用卡尔曼滤波做传感器融合、估算仅可间接测量的物理量**
- 使用PID做单输入-单输出、(近似)线性系统的闭环控制
- 使用MPC做多输入-多输出、 (近似)线性系统的闭环控制
- **使用Soft-run控制概念分配各并联执行器的任务：对于能量密度高、响应速度慢的执行器，负责稳态响应；对于功率密度高、响应速度快的执行器，负责动态响应**



实践2：合理性/可行性

- 不能过多穷举场景
- 不能除零
- 不能出现代数环
- 不能使用浮点量做等于比较
- 不能使用布尔量做数学运算
- 不能使用非布尔量做与、或、非判断
- 不能使用非布尔量做开关选择
- 不能使用非布尔量做使能条件

- 对于查表：选择合适的内插、外插方法；各输入信号、各轴标定量数据类型要完全一致
- 对于Dead Zone、Rate Limiter、Saturation等模块：up值要大于等于lo值
- 使用合理的舍入模式
- 使用合理的溢出处理方法
- 使用合理的**变量范围**

  统一规定车速、开度、扭矩、温度、转速、电压、电流等常见变量的范围

实践3：平台化/兼容性

- 在各项目中，若实现同一功能的核心控制策略基本相同，用**一个软件组件**实现
- 平台化软件组件中，对于在某个项目中单独适用的控制策略，使用**配置字**做适配
- 变量名称**避免使用经常变动的**网段名称、节点名称、报文名称、报文ID、信号值描述的数值
- 对于只有0、1两个信号值的变量，若可能发生功能扩展，**可定义成枚举量**

动力性能等级	0:ECO 1:Sport

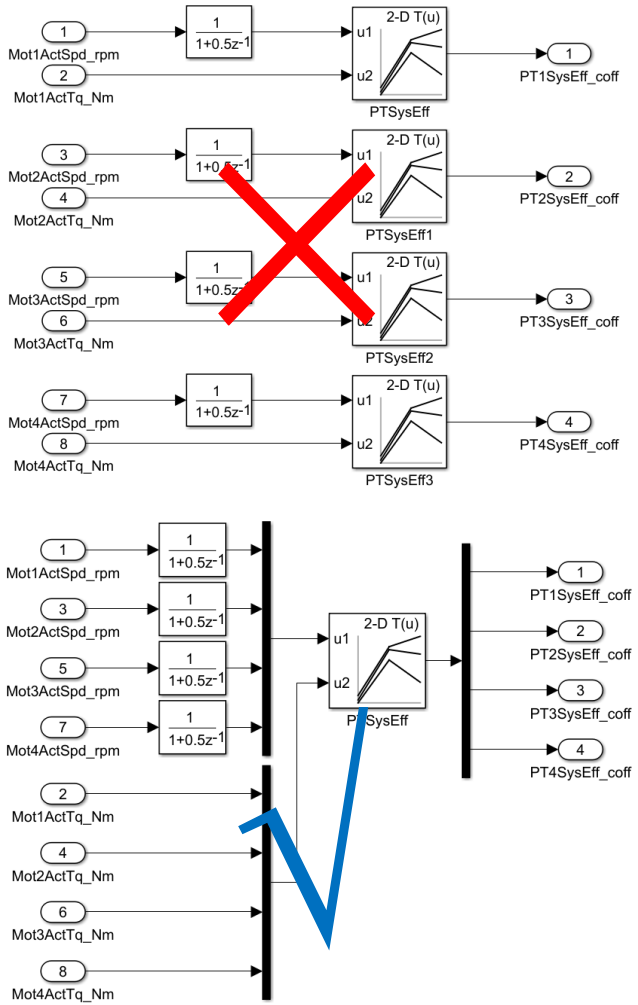
实践4：KISS原则

- 使用尽可能少的模块和代码来实现需求
- 采用简单的模块和算法来实现需求，避免使用高深算
- 设置并复用全局常量：

  重力加速度、π、km/h-m/s换算系数……
- 设置并复用全局标定量：

  传动比、传动效率、轮距、轴距、滑阻曲线……
- 复用数据流上游组件的常量、标定量
- 算法的统一处理：

  变速箱传动比计算、变速箱传动效率计算……

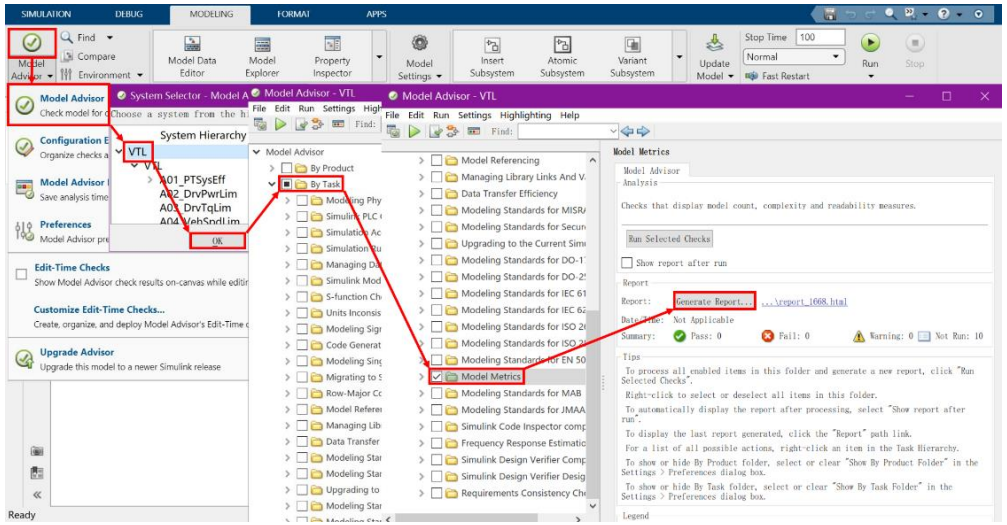


实践4：KISS原则

- 设置并复用自定义模块库：

  Debounce、有效性检测、范围检测、滤波、位操作、

  进制转换、计时器、PID……



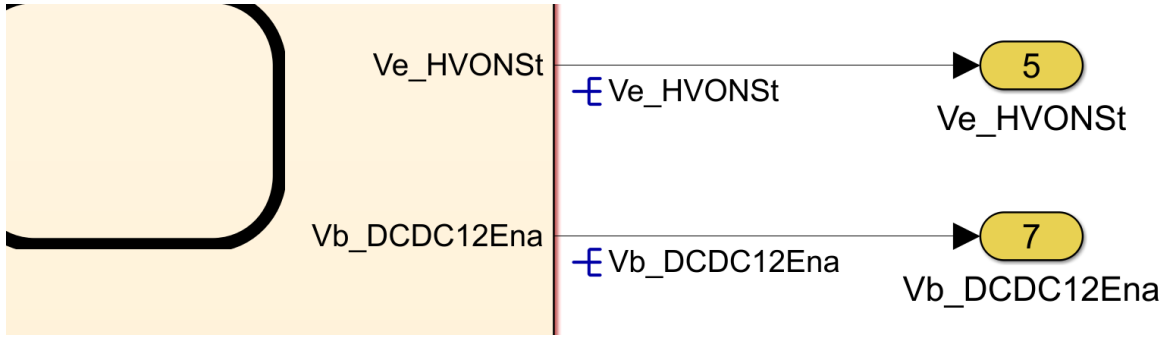
- SWC复杂度

	圈复杂度	输入接口数量	输出接口数量	子系统层数	Stateflow层数	状态机数量	与、或、非层数
良好	≤300	≤25	≤10	≤4	≤2	≤7	≤2
可接受	300~700	25~50	10~20	4~8	2~3	7~14	2~4
不可接受	＞700	＞50	＞20	＞8	＞3	＞14	＞4

注：基于一个SWC中只有一个Runable的评价方法

实践5：易修改

- 按照**自顶向下、高内聚、低耦合、原子服务**的思想划分SWC、 SWC中的子系统
- SWC中各数据流之间应相对独立，尽可能减少交互
- 尽可能减少闭环数据流
- 实现特定功能的信号与状态机的值区分开



实践6：可读性

- 软件组件内部结构要求

  数据流方向自左向右、自上到下

  全屏浏览时能看清本层控制策略

  各模块、状态机、流程图之间不可相互遮挡

- 子系统要求

  子系统、Stateflow关闭内部预览

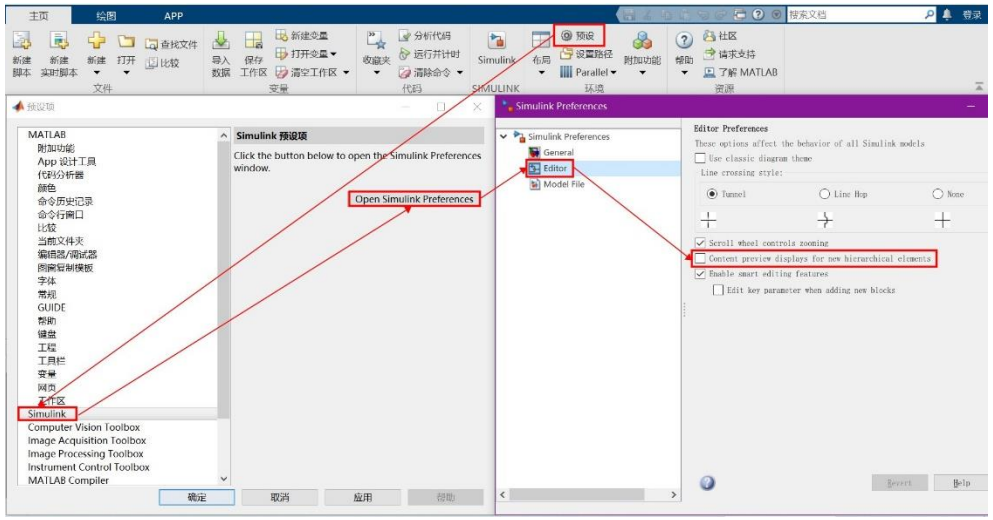
- Stateflow要求

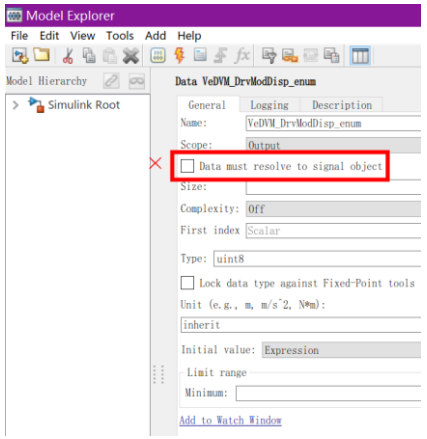
  多个条件的与或非判断，在Stateflow外部处理成布尔量

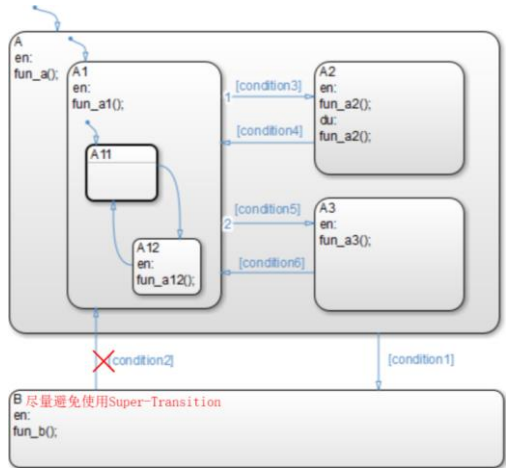
  在Stateflow外部添加观测量

  避免使用Super-Transition

  Flowchart使用模板，水平线放置条件，垂直线放置动作







实践6：可读性

- 信号传输要求

  避免大规模的信号线交叉

  同一个信号名称应该一致

  Goto模块的Tag visibility设置为local

  使用Signal Conversion分割观测量名称

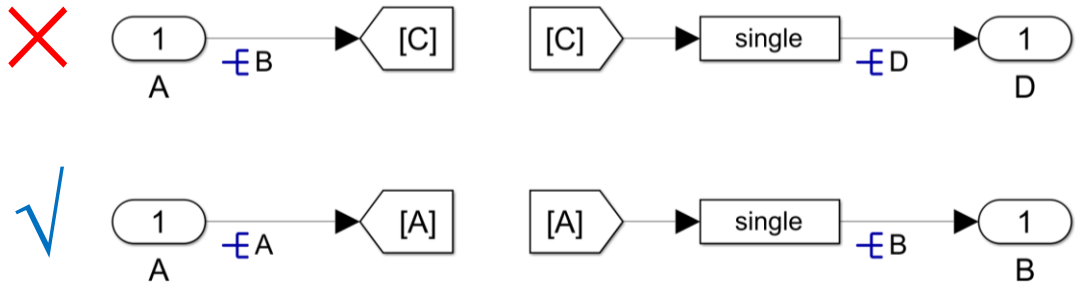
  使用Data Type Conversion转换数据类型

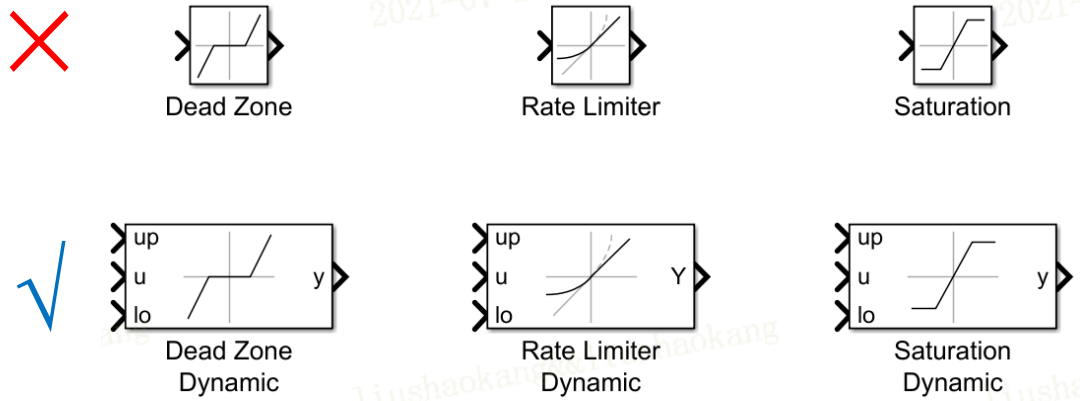
- 模块使用要求

  使用易查看标定量的模块

  避免在Gain、Compare To Constant中填写标定量

  隐藏可通过图标识别的模块名称





实践6：可读性

- 命名要求

  子系统、Stateflow、信号名称要通俗易懂

  子系统、 Stateflow、信号名称要代表其物理含义

- 算法要求

  模型要尽可能体现出公式、算法

  使用Simulink搭建简单公式，避免使用Embedded MATLAB

- 单位要求

  使用常用单位，如s、min；kg；V；A；Ω；℃；kW……

  避免使用不常用单位，如ms、h；g、t；mV、kV……

- 注释要求

  在复杂算法、复杂状态机、关键信号处添加注释

DrvrTqReqA_Nm↵	DrvrTqReqEco_Nm↵
DrvrTqReqB_Nm↵	DrvrTqReqCmft_Nm↵
DrvrTqReqC_Nm↵	DrvrTqReqSpt_Nm↵
↵	↵
Filt15_null↵	EcoPosLrgDecFilt_null↵
Grdt11_Nmps↵	CrpNegMedIncGrdt_Nmps↵
Grdt64_Nmps↵	SptPosSmlDecGrdt_Nmps↵

实践7：标定类要求

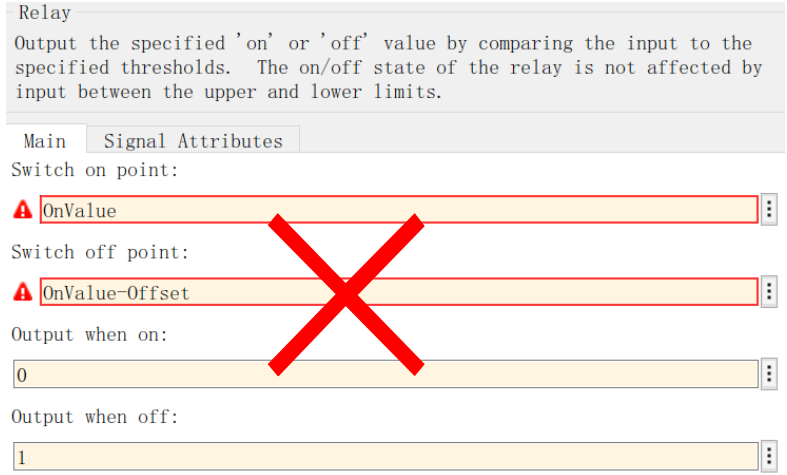
- 所有应用层输入，设置覆写标定量
- 所有SWC输出，设置覆写标定量
- 重新上下电才恢复的故障，设置Reset标定量
- 统一滞环标定量设置方式

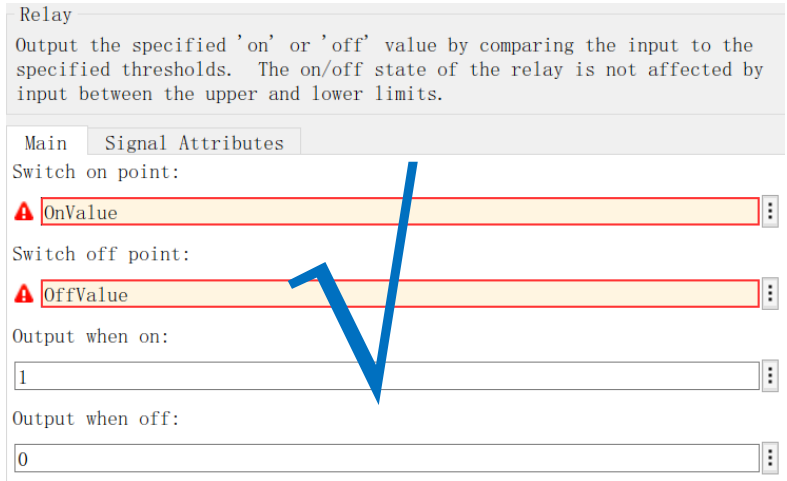
  如使用On/Off标定量的方式；禁用On/On-Offset标定量的方式

  On时输出1，Off时输出0；禁止On时输出0，Off时输出1
- 对于部分数学模型，使用查表算法而非公式

  如滑行阻力模型、制动力模型、I曲线模型等
- 查表断点数

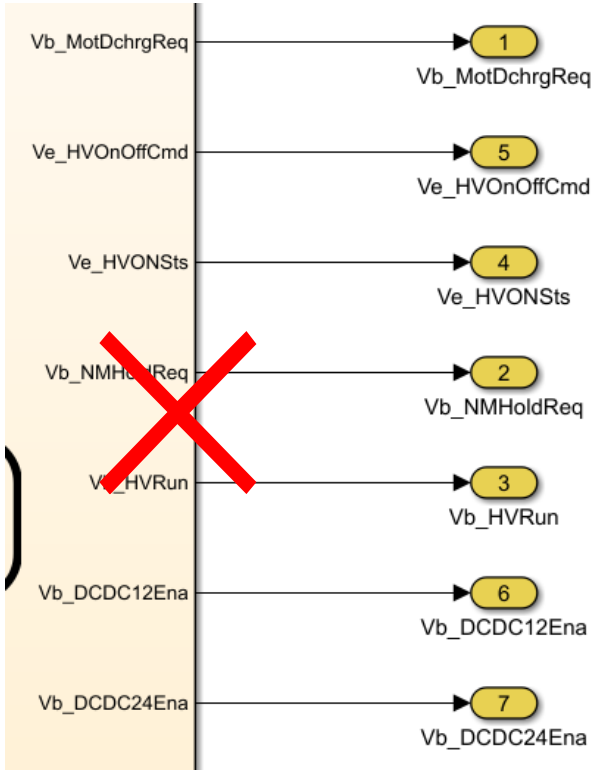
  有5、10、20等个数标准，可适配平台化软件，可适配标定工具

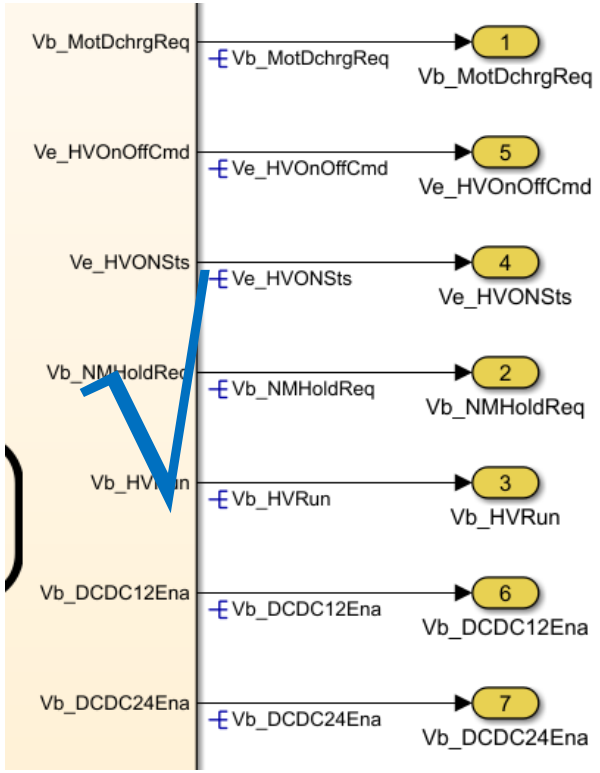




实践8：观测量设置

- 关键子系统输出
- Stateflow的状态、其他输出
- If Else、 Switch Case输出
- 多个条件取与、或、非处
- 多层与、或、非处
- （二维、三维）查表输出
- PID输出
- Embedded MATLAB输出
- 被多处策略引用的变量





Chapter 1：汽车软件质量属性的实践

Chapter 2：汽车软件快速交付的挑战

1. 我们面临的挑战

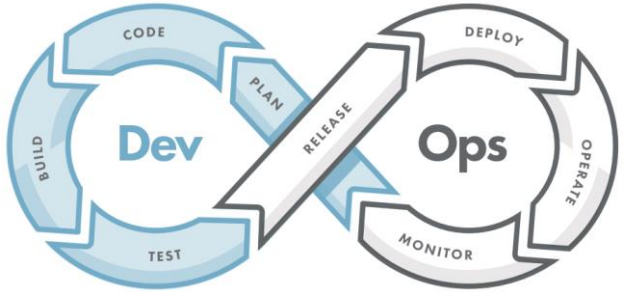
智驾系统

图1 智驾系统功能示意图

挑战：保质保量、快速交付、快速迭代

面临的问题：

- 软件集成效率如何提高？
- 如何做好多模块协同开发？
- 何种方式快速验证和迭代？



MATLAB
SIMULINK

图2 面对挑战的“武器”：核心流程和工具

2. 我们应对挑战的“武器”

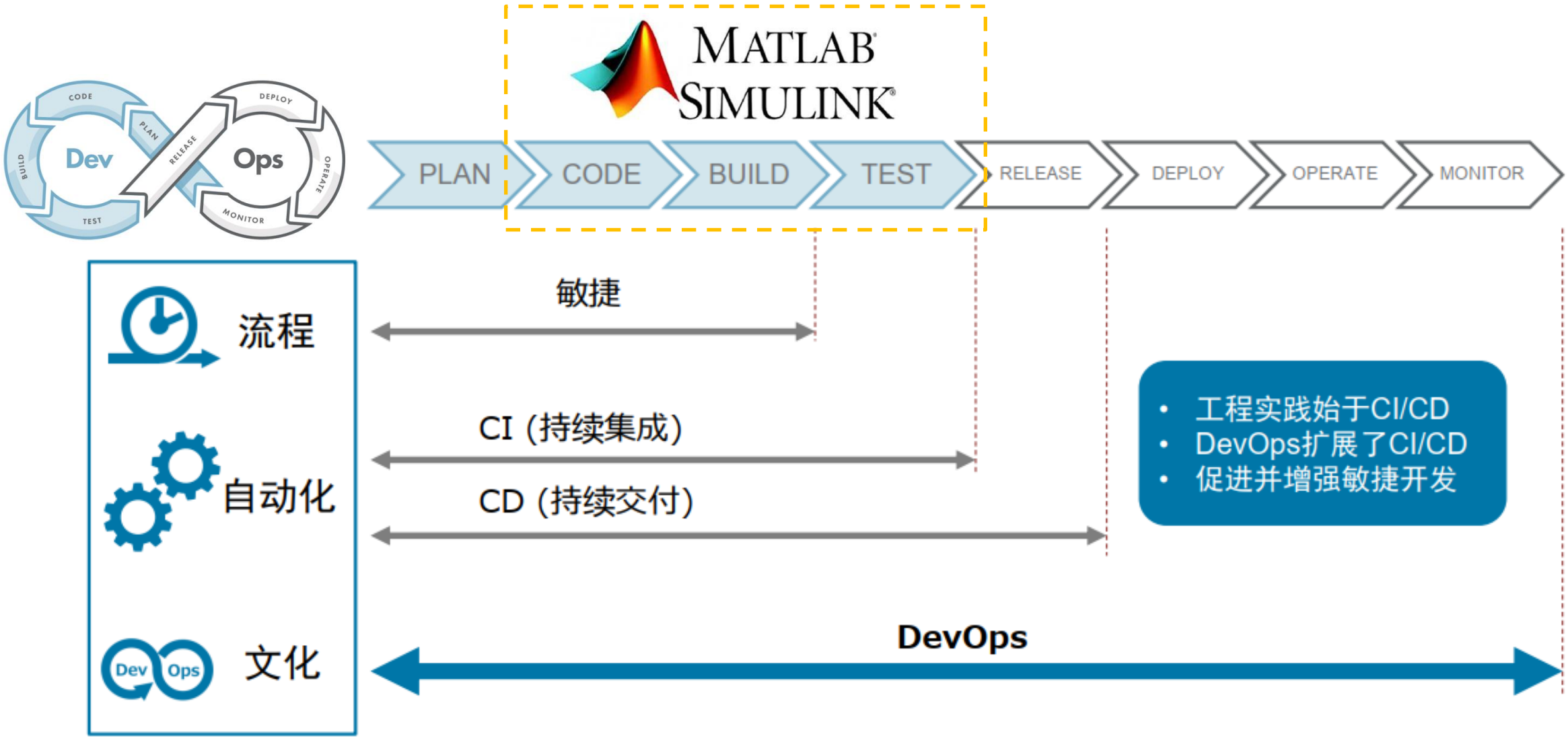


图3 敏捷开发的指导流程：CICD和DevOps

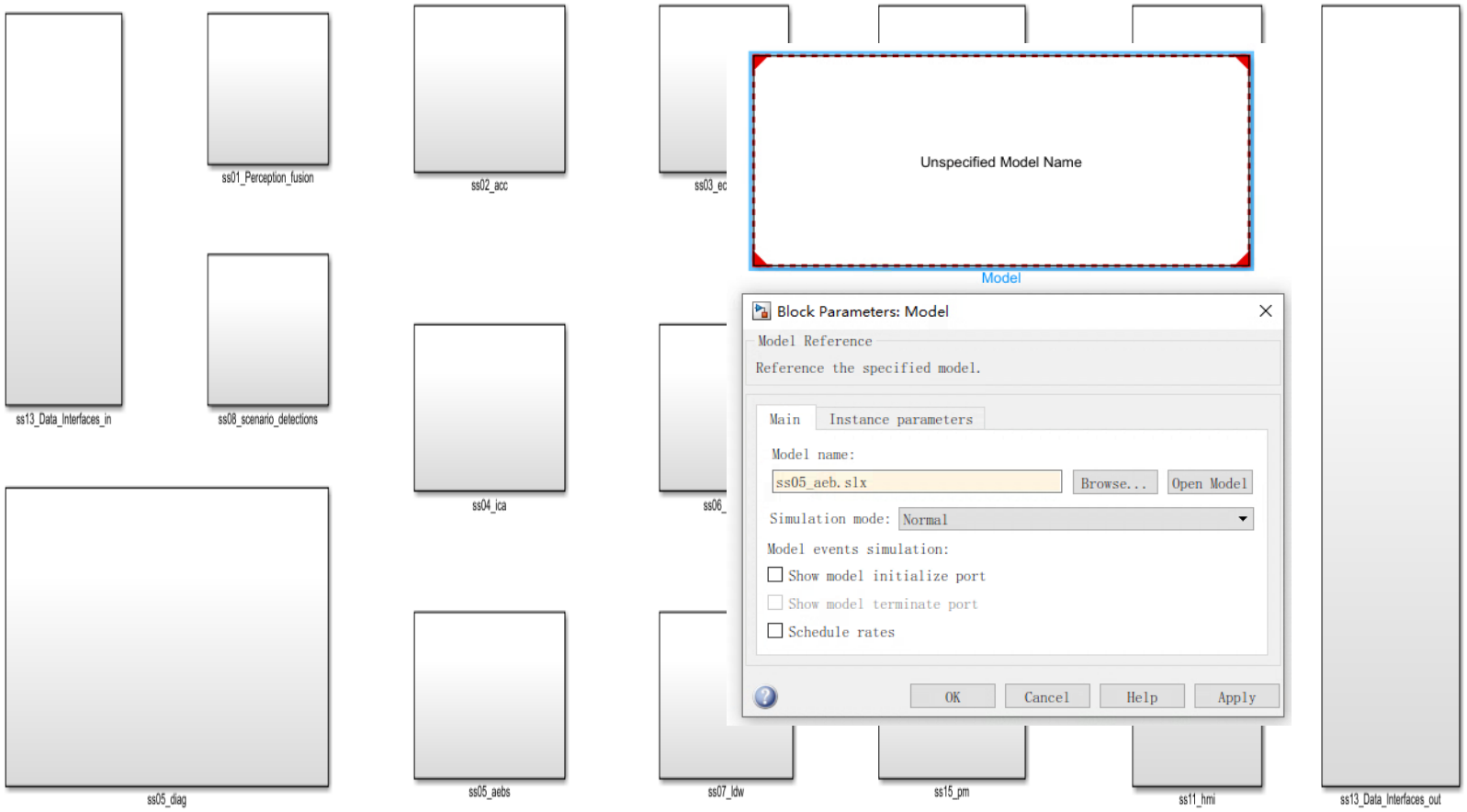
3. 指导思想的“落地页”

图4 CICD在开发中的简化实践（侧重于MCU软件）

4. “落地页”的拆解实践——使用Model Reference进行多模块软件集成

Model Reference的使用

- ✓ 模型迭代修改量大幅降低
- ✓ 便于子模型版本管理和测试
- ✓ 便于CICD自动化更新



注：每个子模块通过model reference关联

图5 软件架构和Model Reference设置示意图

4. “落地页”的拆解实践——使用git进行子模块版本和权限管理

- 每个子模型分别设置git仓库作为子仓库
- 主模型中所在的git仓库设置git submodules对子仓库进行管理
- git 权限区分集成人员、模块开发人员、仿真人员和测试人员等

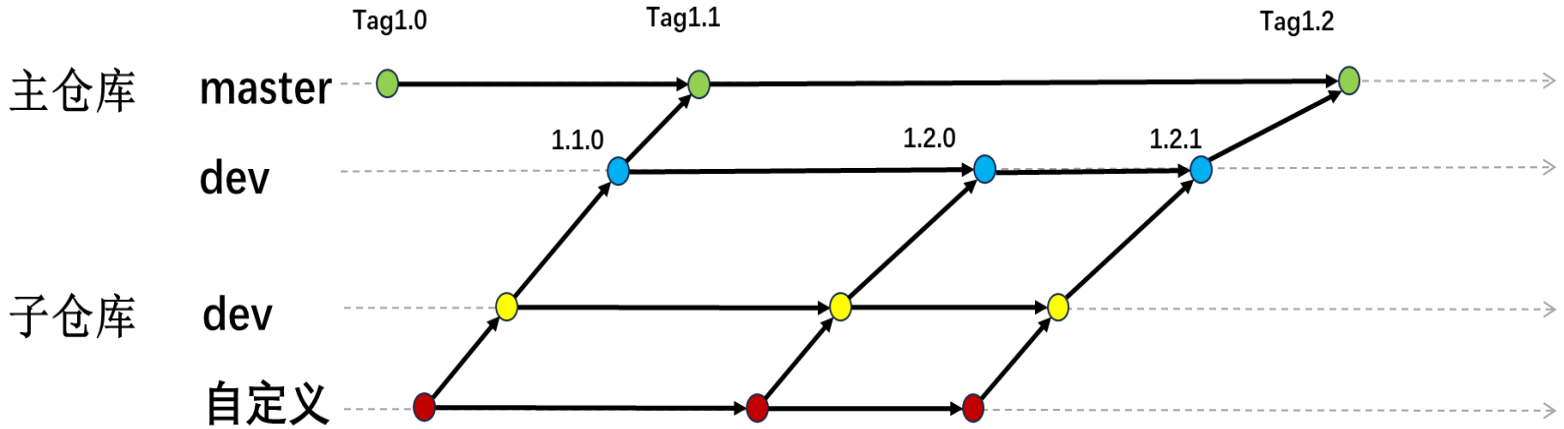
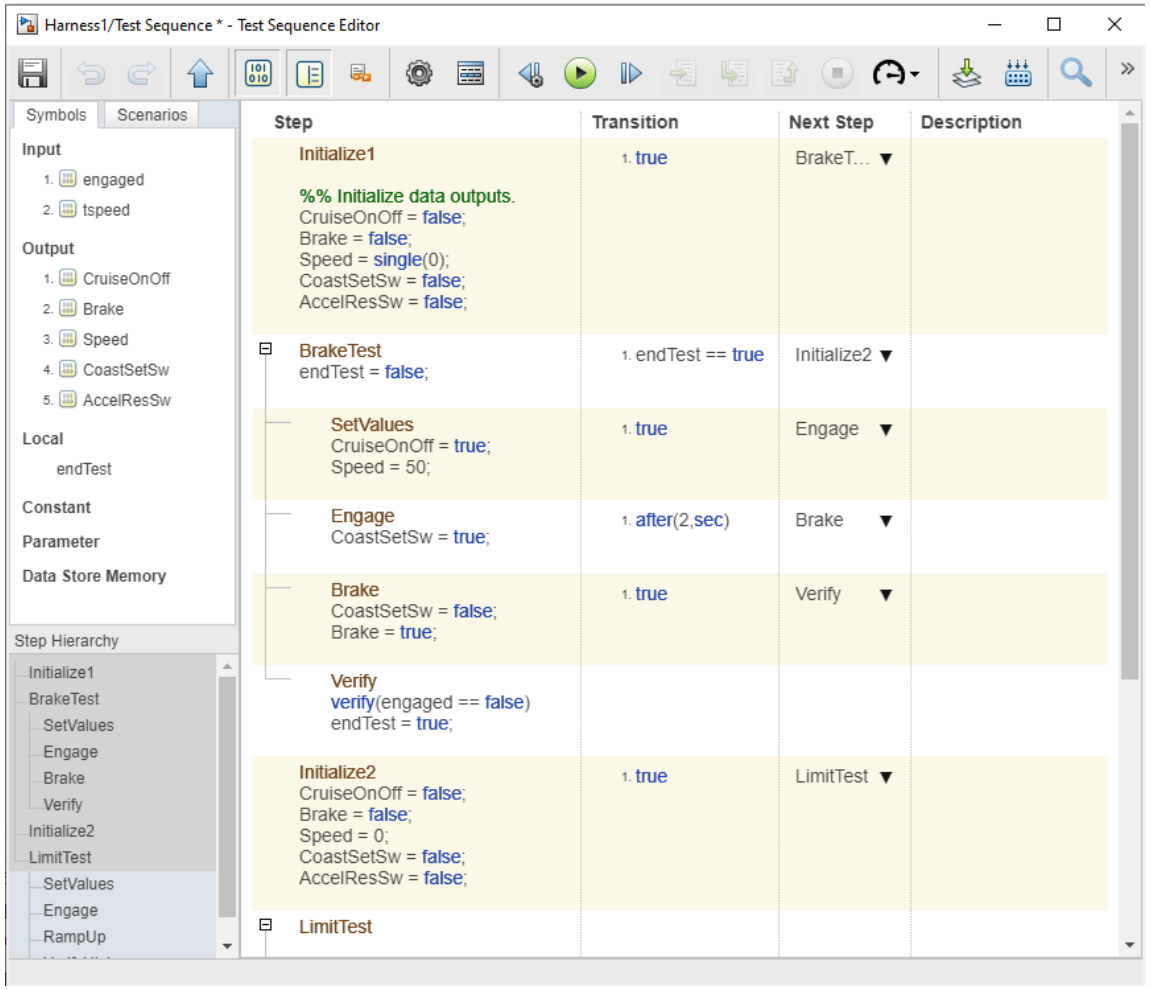


图6 git主仓库与子仓库关系图

4. “落地页”的拆解实践——使用Simulink Test进行单元测试



- ✓ 单元测试用例编写
- ✓ TestHarness测试
- ✓ 决策覆盖度/条件覆盖度报告

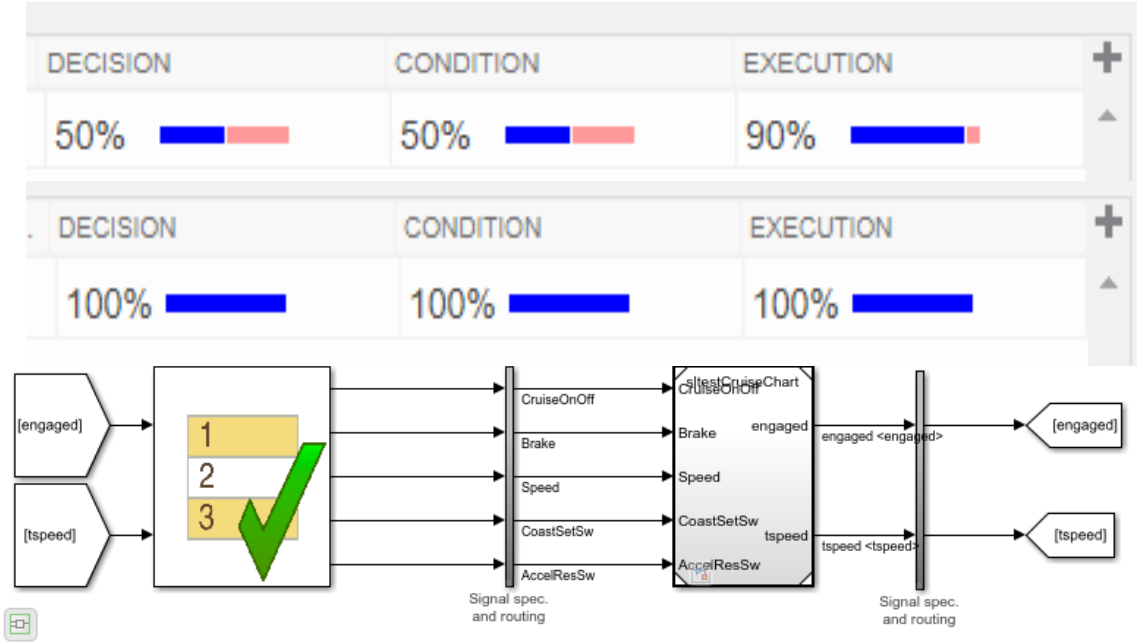
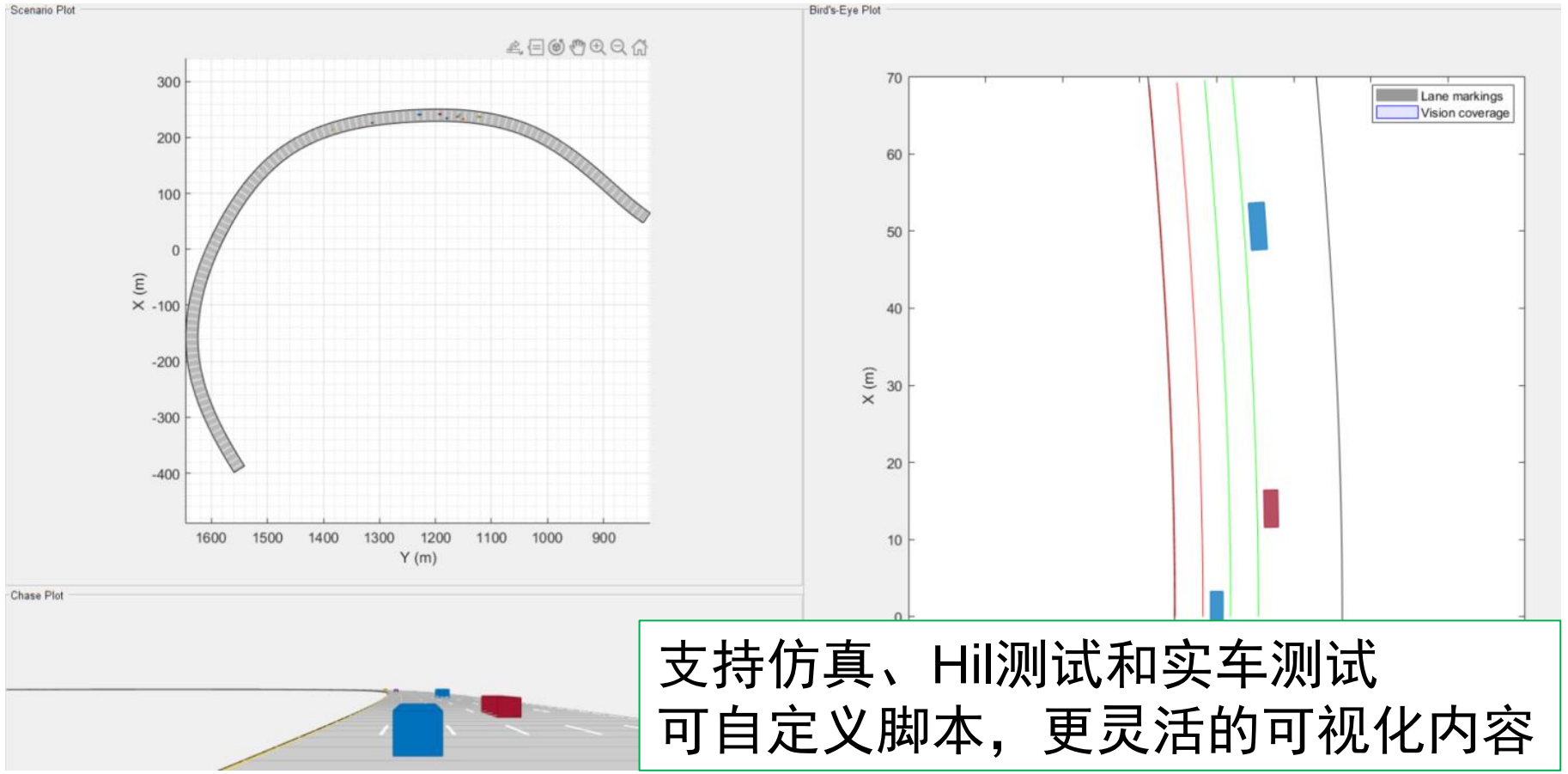


图7 单元测试示意图

4. “落地页”的拆解实践——使用driving scenario designer进行可视化



支持仿真、Hil测试和实车测试
可自定义脚本，更灵活的可视化内容

图8 MATLAB可视化仿真调试工具

4. “落地页”的拆解实践——使用Simulink.project进行模型工程管理

MATLAB“组合拳”

Simulink.project

- 工程文件管理
- git版本管理
- 脚本启动关闭管理
- 数据管理
- 模型和代码库管理
- 过程文件和临时文件管理
- 输出文件管理

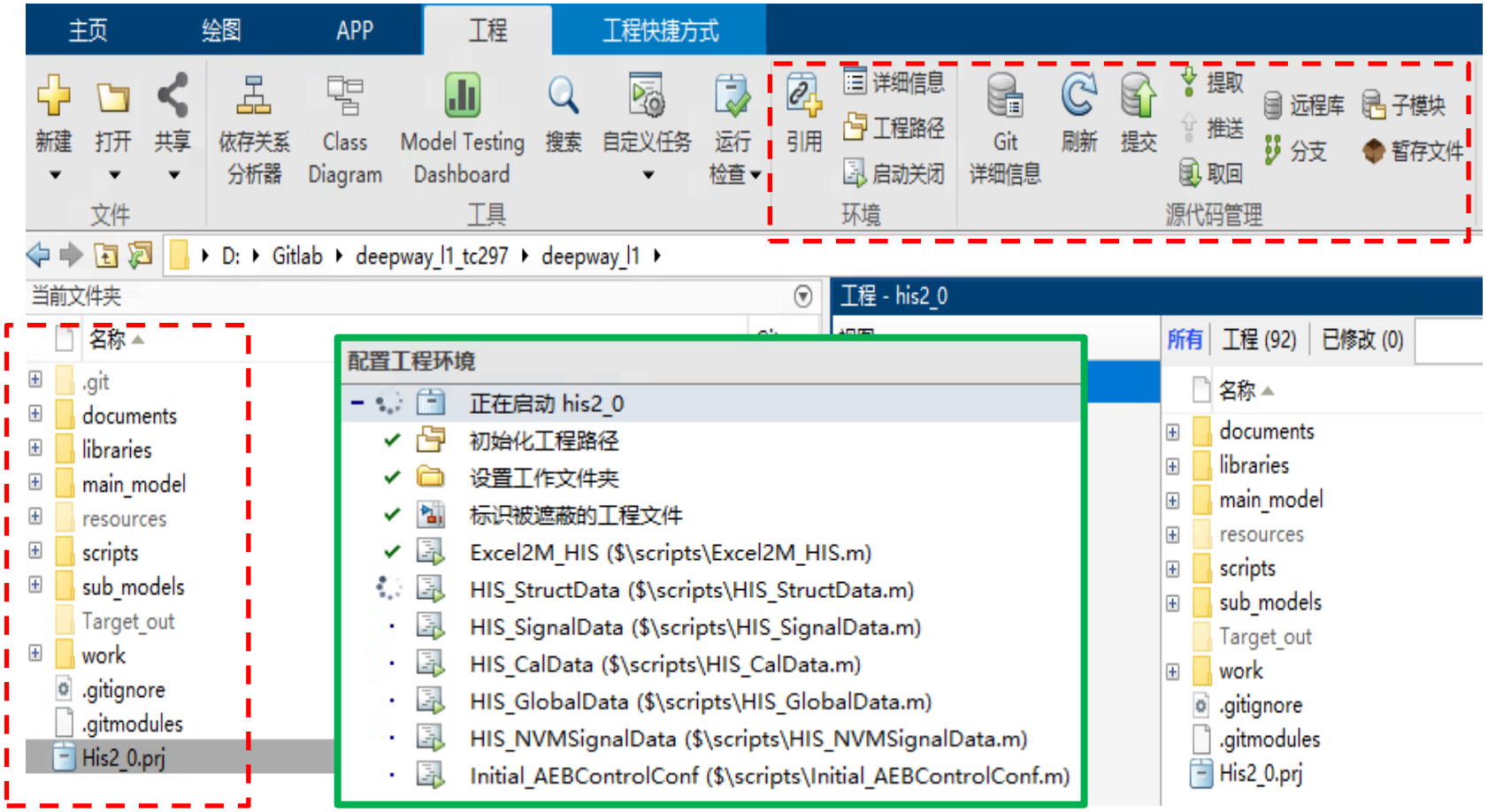
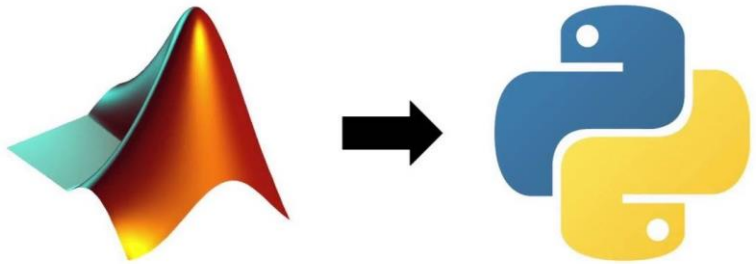


图9 Simulink.project功能和界面示意图

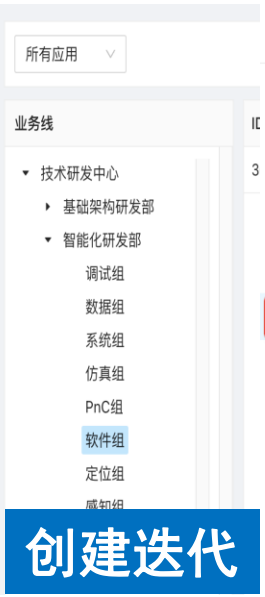
4. “落地页”的拆解实践——使用CICD服务器生成制品



Matlab具有很强的拓展性，使用兼容其他开发语言调用

```
PS D:\agent\dist> cd .\agent_engine\
PS D:\agent\dist\agent_engine>
PS D:\agent\dist\agent_engine>
PS D:\agent\dist\agent_engine> .\agent_engine.exe start
check all environments ...
check all environments pass
model name: HisMcuXT_ANP.slx
matlab: ()
```

创建应用



创建迭代

创建流水线

构建成功&生成制品

图10 CICD流程示意图

最后：一些思考

模型质量和开发规范

快速交付流程

各类工具链

Matlab/Simulink

Hitech、Ecocoder、git

JIRA、Pingcode 等

开发人员成长

实践经验积累

开发预算　使用场景

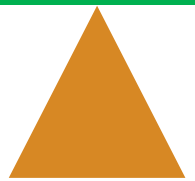
开发周期　验收标准

人力资源　。。。。。。

规范

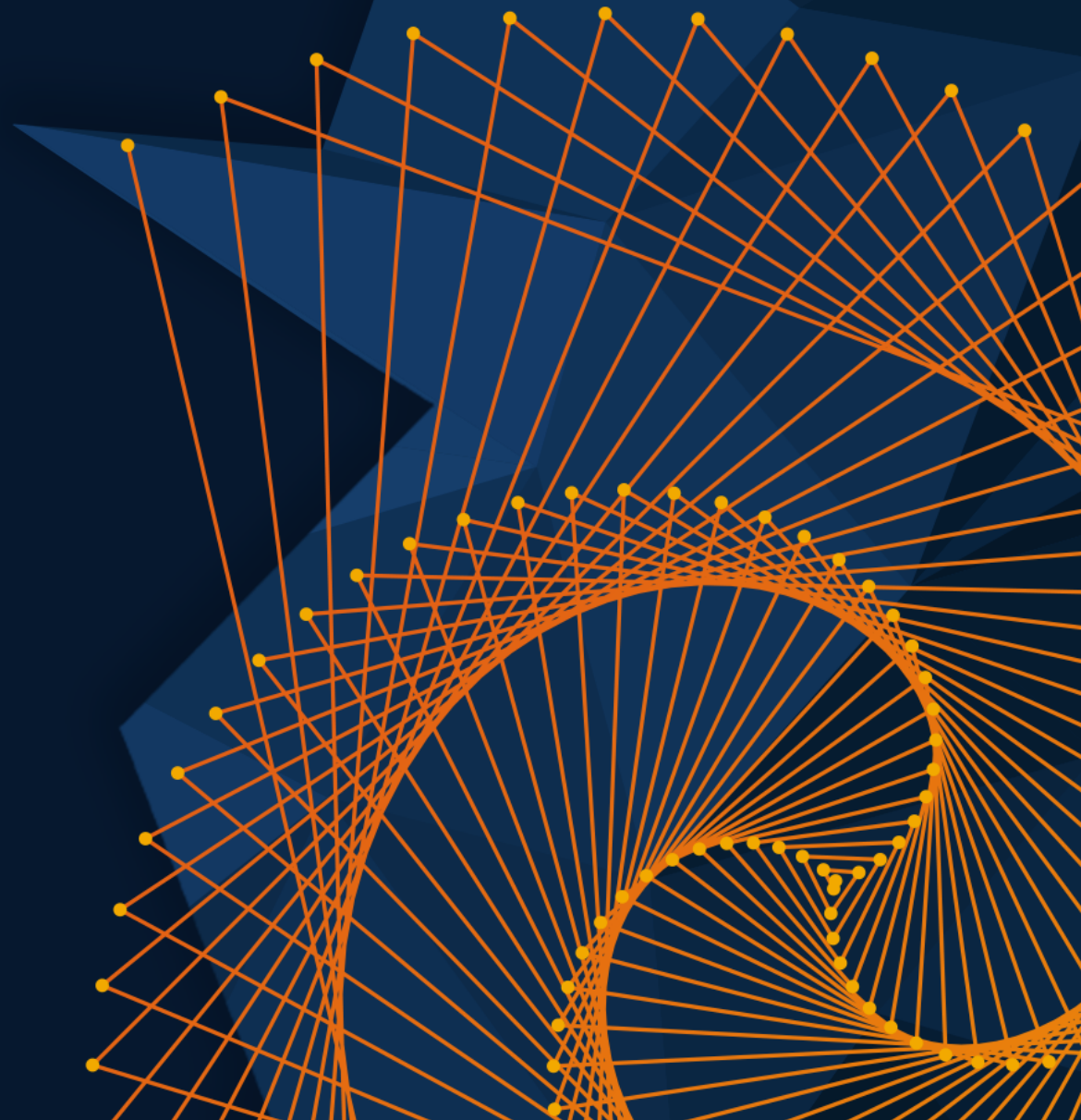
流程　工具

保质保量快速交付



MATLAB EXPO

Thank you



MathWorks®