

AUTOSAR™

软件定义汽车 面向服务架构的应用迁移



龚小平

2023-03-16



BMW Group



BOSCH



DAIMLER



TOYOTA

VOLKSWAGEN
AKTIENGESELLSCHAFT

背景 - 软件定义汽车

汽车行业正在采用面向服务的架构（SOA）作为设计软件定义汽车（SDV）等现代应用的新范式

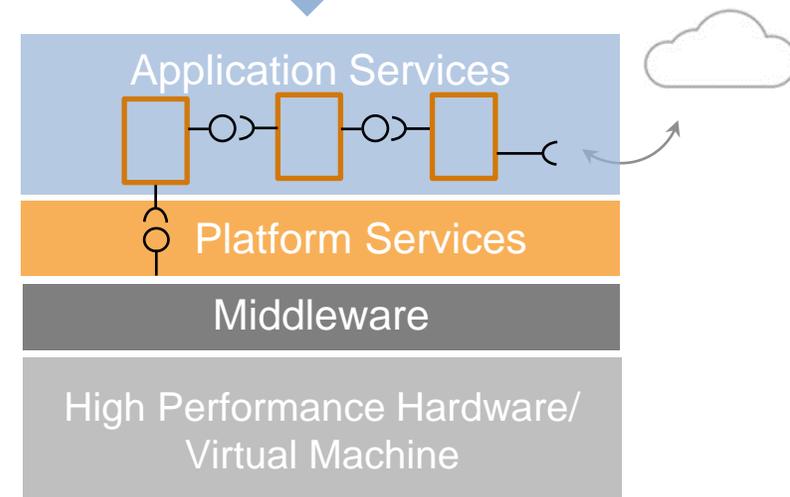


100110
001010
1001100010
001010
010010
100110
001010
010010



软件更新需求

- 频繁
- 可选
- OTA



更高的抽象:面向服务的架构

趋势 - 主机厂加大SOA自研力度

软件定义汽车



智能
网联
AI

基于中间件的SOA

C++

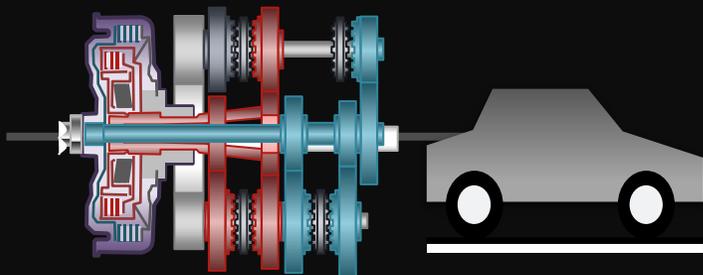


ROS

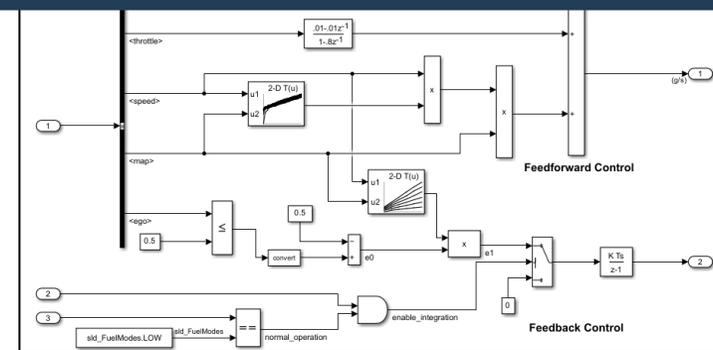


如何将传统的汽车软件应用迁移到基于 SOA 的服务?

基础车辆平台



控制
实时
CAN



内容

- 将传统汽车应用转换为 SOA 应用有哪些挑战?
- 如何将传统的应用软件组合分解为SDV的服务?
- 基于模型设计如何帮助将已有应用迁移到 AP服务?

内容

- 将传统汽车应用转换为 SOA 应用有哪些挑战?
- 如何将传统的应用软件组合分解为SDV的服务?
- 基于模型设计如何帮助将已有应用迁移到 AP服务?

为什么需要对已有应用实施迁移

重用经过验证的已有算法

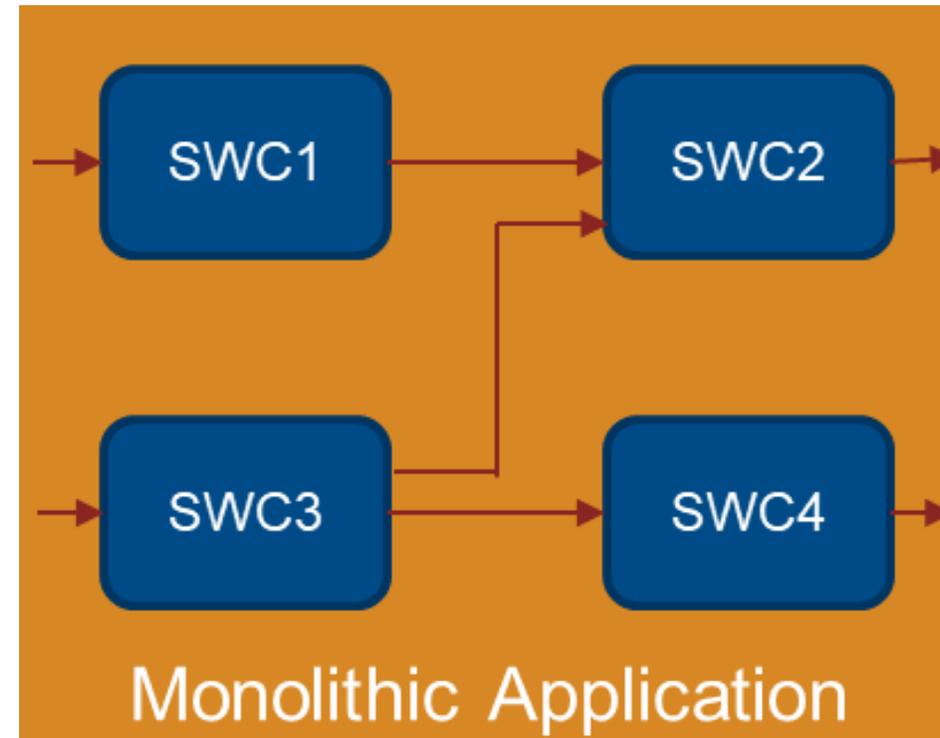
- 知识资产
- 算法库
- 作为扩展到高级功能的基线



已有应用本质上是单体式的

由各种软件组件组成

- 作为单个可执行文件部署在微控制器上
- 对不断变化的需求灵活性低



Reference: Software Architecture Patterns: Understanding Common Architecture Patterns and When to Use Them by Mark Richards

将传统汽车应用转换为SOA应用有哪些挑战

打破单体式架构

- 挑战 - 将旧应用程序的单体架构分解为更小、更模块化的服务。
- 要求 - 深入了解现有的应用程序架构以及服务化的设计原则。

识别和定义服务

- 挑战 - 确定应用需要提供的服务并清晰定义它们之间的接口。
- 要求 - 了解应用提供的功能以及定义服务化的工作流程。

确保兼容性

- 挑战 - 将旧应用程序转换为 SOA 应用程序可能会导致兼容性问题。
- 要求 - 确保新服务与现有服务兼容，并且应用程序仍可与生态系统中的其他系统通信。

管理数据

- 挑战 - 在旧应用程序中，数据可能与应用程序紧密耦合，因此难以作为单独的服务进行提取和管理。
- 要求 - 全面分析数据模型以及如何将其与应用程序逻辑分离。

性能和可扩展性

- 挑战 - 性能和可扩展性是迁移到 SOA 体系结构时的一个关键属性。
- 要求 - 确保新服务可以处理与现有应用程序相同的负载，并且可以根据需要进行扩展或缩减。

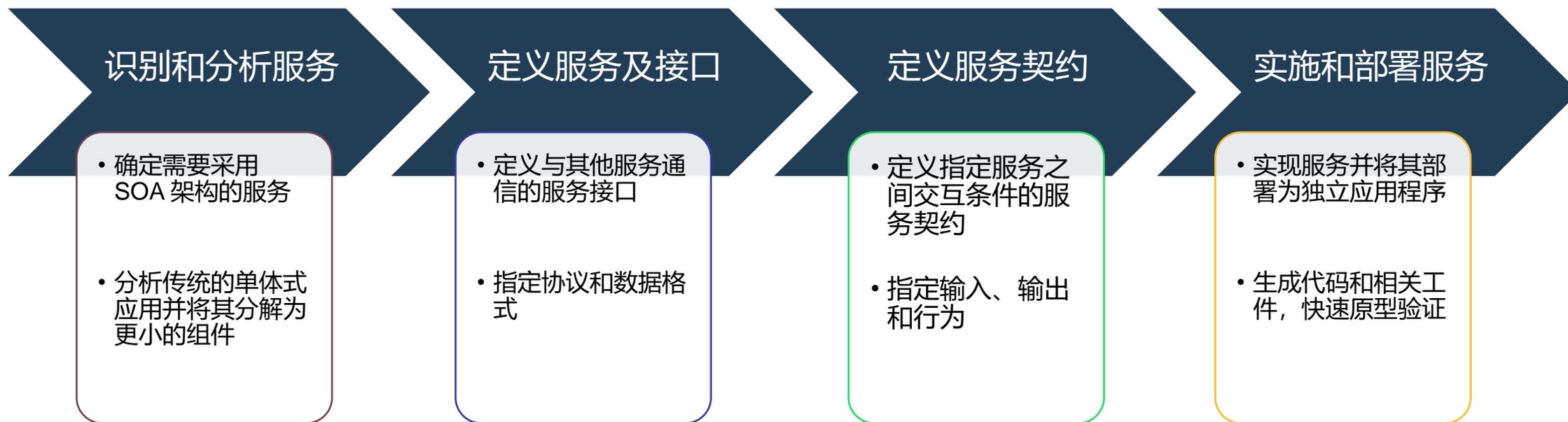
文化挑战

- 挑战 - 开发人员可能习惯于使用单体架构。
- 要求 - 需要学习新的架构和开发流程。

内容

- 将传统汽车应用转换为 SOA 应用有哪些挑战?
- 如何将传统的应用软件组合分解为SDV的服务?
- 基于模型设计如何帮助将已有应用迁移到 AP服务?

工作流程



设计原则 - SOLID

- **Single-Responsibility principle** (单一职责原则)
- Open-Close principle (开放闭合原则)
- Liskov substitution principle (里氏替换原则)
- Interface segregation principle (接口隔离原则)
- **Dependency inversion principle** (依赖倒置原则)

- *Explanation of Adaptive Platform Software Architecture R22-11*
- *Agile Software Development: Principles, Patterns, and Practices*

- 一个组件只负责功能的一个独立部分
- 减少接口变更需要, 提升可维护性

- 客户端不应该依赖于用不到的方法
- 将复杂接口按需拆分为简单接口

- 高层模块应该依赖于抽象而非低层模块
- 降低耦合, 提升复用
- 避免循环依赖

架构平台

组件层级

仿真

行为建模

函数定义

代码和架构文件生成

接口

The screenshot displays the AUTOSAR architecture platform interface. At the top, a toolbar includes various tool icons and a 'SIMULATE' section with 'Stop Time 10.0', 'Normal', 'Fast Restart', 'Run', and 'Stop' buttons. A red box highlights the 'Export' menu, which is open to show options like 'Configure XML Options', 'Generate Code and ARXML', and 'Export Composition'. The 'Export Composition' dialog is also visible, showing 'Exported ARXML folder' and 'Package code and ARXML' checked. The main workspace shows a component-level diagram with 'Sensors', 'Ctrl', and 'Actuator' components. A red box highlights the 'Sensors' component. Below the diagram, a 'ThrottleControlComposition: Schedule Editor' window shows a sequence diagram with function calls like 'TPS_Secondary_read_5ms' and 'TP_Monitor_D1'. A 'Requirement Editor' window is open on the left, showing a list of requirements. The AUTOSAR logo is at the bottom left.

内容

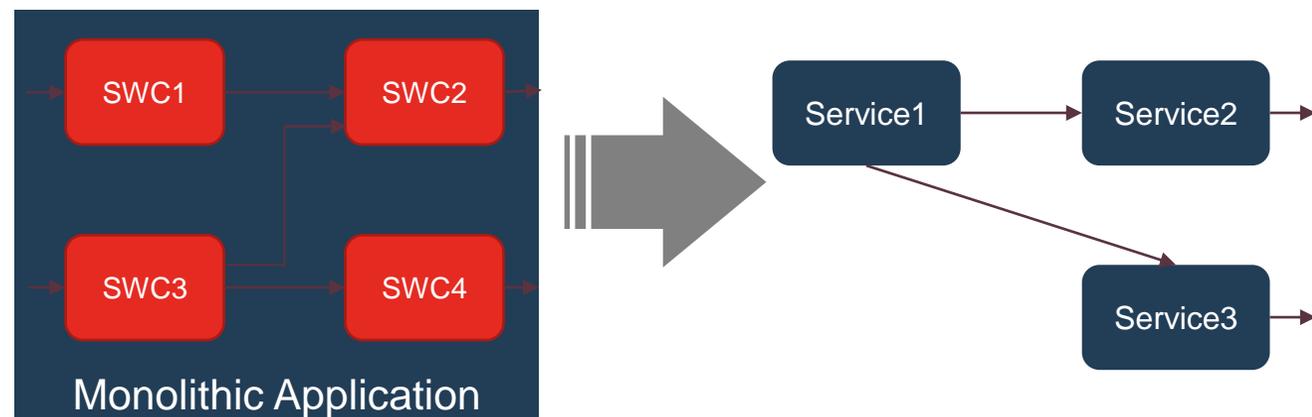
- 将传统汽车应用转换为 SOA 应用有哪些挑战?
- 如何将传统的应用软件组合分解为SDV的服务?
- 基于模型设计如何帮助将已有应用迁移到 AP服务?

识别和分析服务



要将单体式应用程序组件分解为服务，我们需要：

- 识别不同的组件、功能和依赖关系
- 了解组件交互和组件的执行顺序



Reference - <https://chrisrichardson.net/post/refactoring/2020/08/21/ten-principles-for-refactoring-to-microservices.html>

识别和分析服务

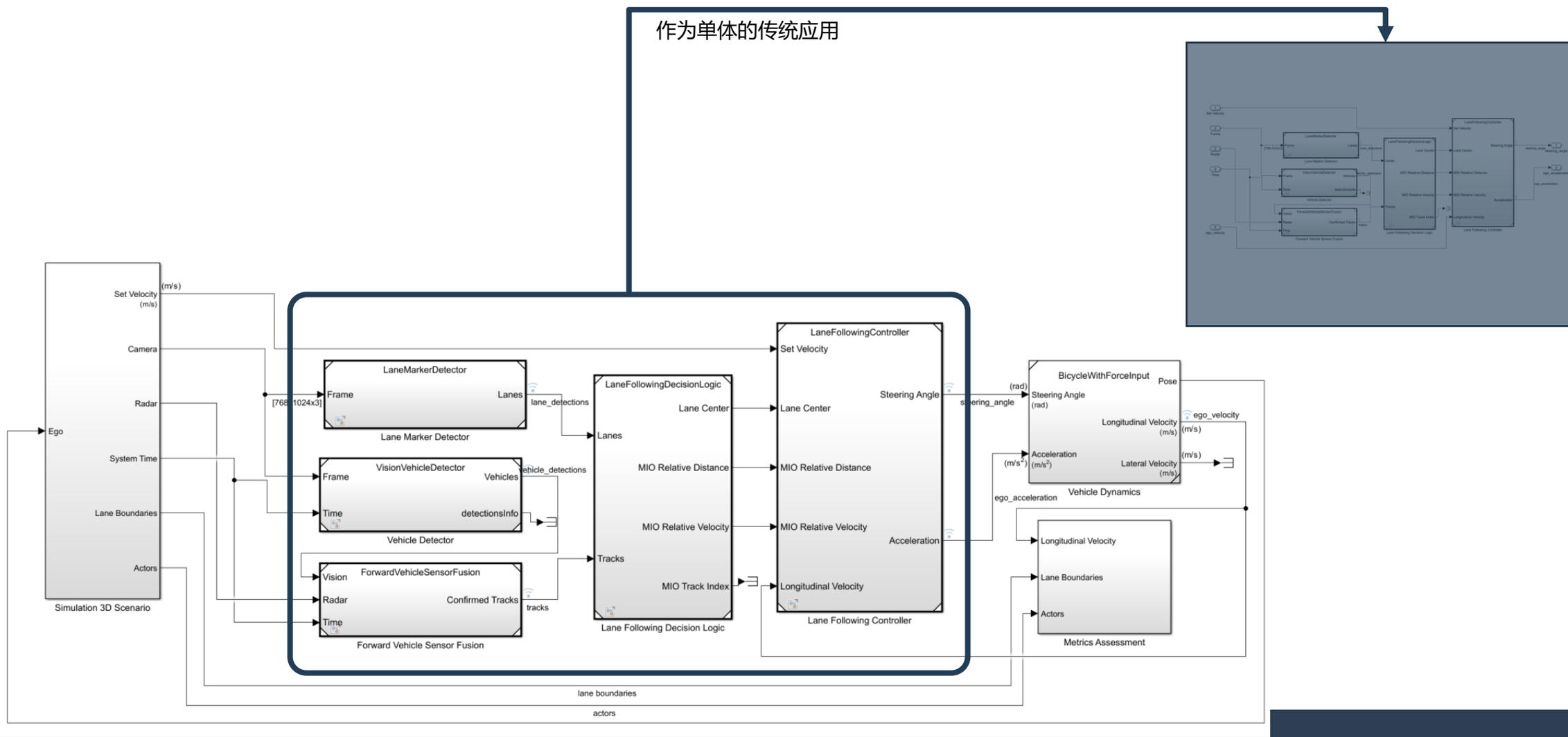
识别和分析服务

定义服务及接口

定义服务契约

实施和部署服务

作为单体的传统应用



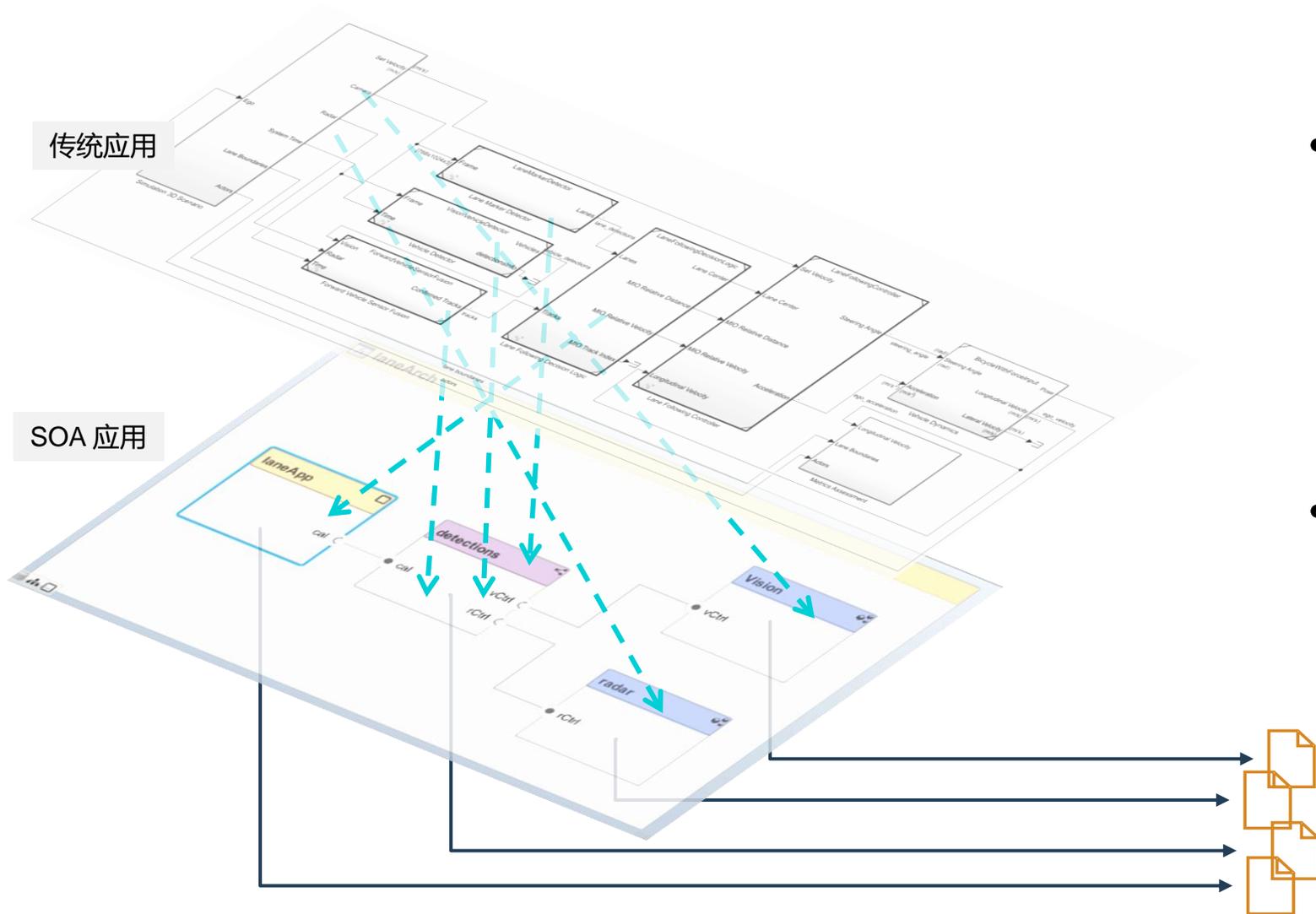
识别和分析服务

识别和分析服务

定义服务及接口

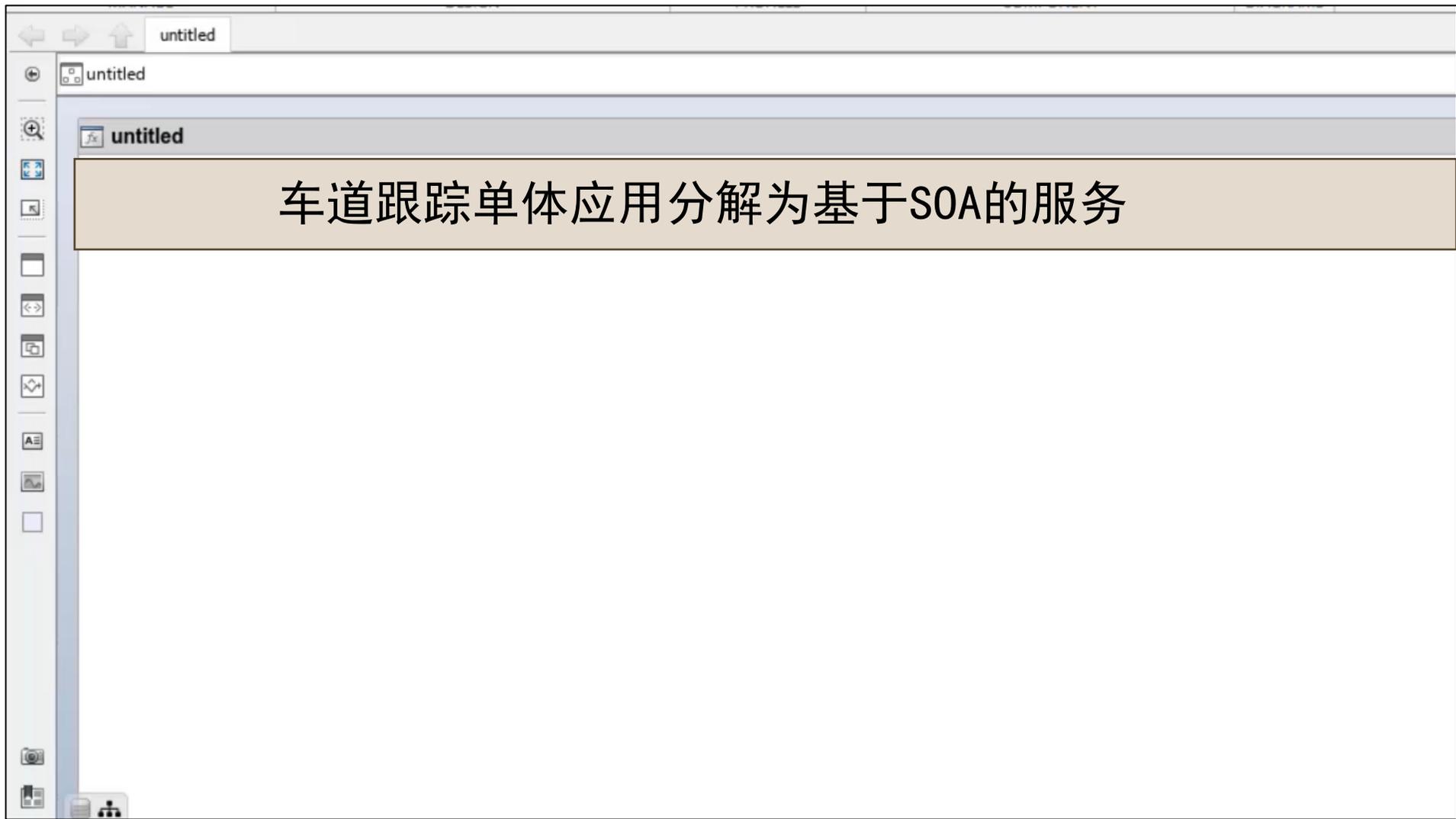
定义服务契约

实施和部署服务



- 确定可以单独部署为服务的组件，这些服务封装了一个功能，并从单体式应用程序模型的其余部分抽象出来
- 将每个已识别的组件部署成作为服务的可执行文件

识别和分析服务



定义服务接口



- 包含事件的数据元素和服务提供方提供的操作
- 作为服务的抽象与外部进行交互
- 具有功能性

The screenshot shows the laneArch software interface. The main workspace displays a block diagram with components: laneApp, detections, Vision, and radar. laneApp has a 'cal' port connected to a 'cal' port on the detections block. The detections block has 'vCtrl' and 'rCtrl' ports connected to the Vision block. The radar block is also visible. The Property Inspector on the right shows the configuration for the 'cal' port, including Name, Action (Client), and Stereotype. The Interfaces table at the bottom lists the interfaces defined in the model.

Interfaces	Type	Dimensions	Units	Complexity	Minimum	Maximum
laneArchDD.sidd						
Adjustments						
status = Adjust(opts)						
status = Calibrate(opts)						
Calibrations						

配置服务接口

定义服务边界

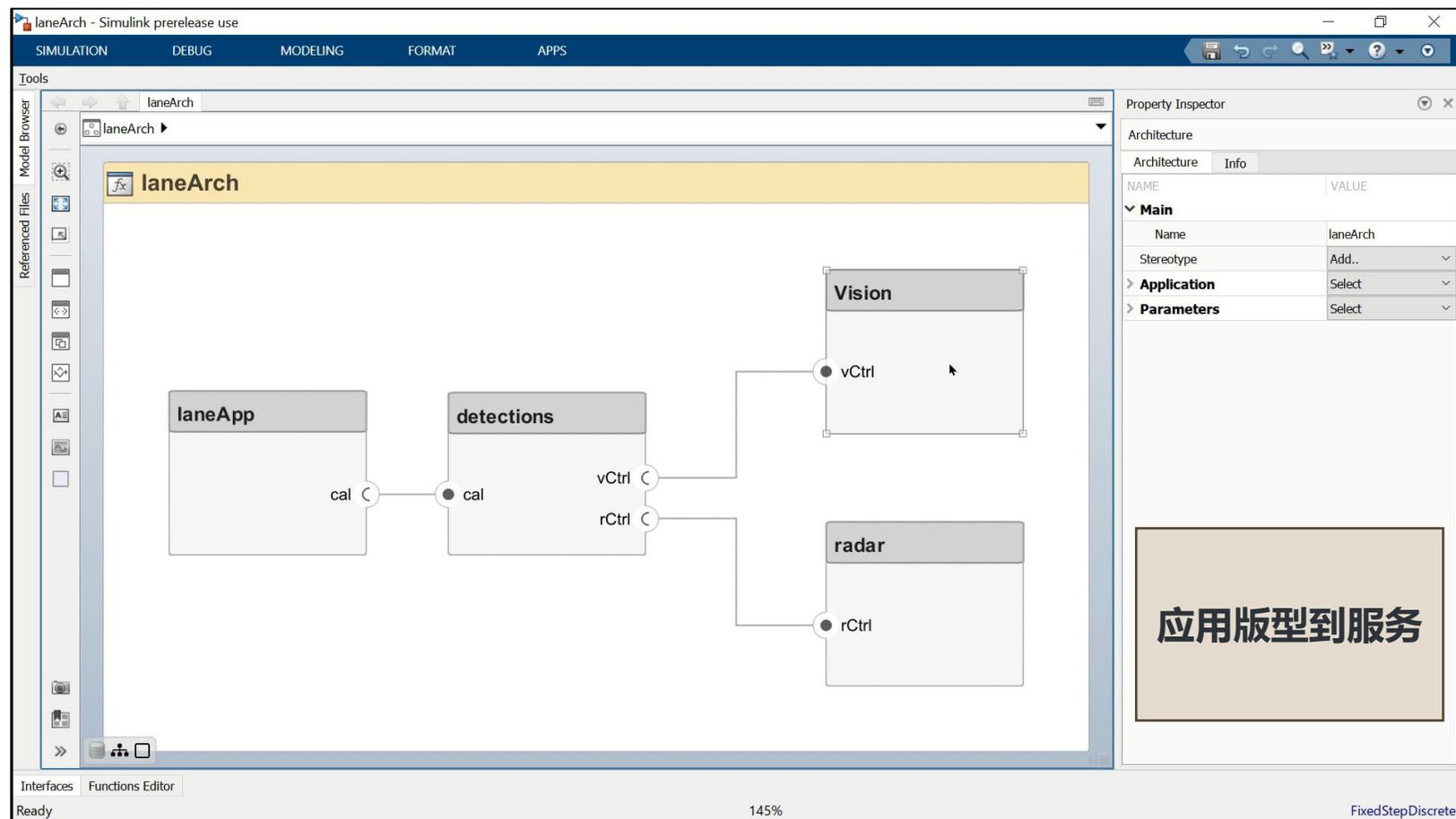
识别和分析服务

定义服务及接口

定义服务契约

实施和部署服务

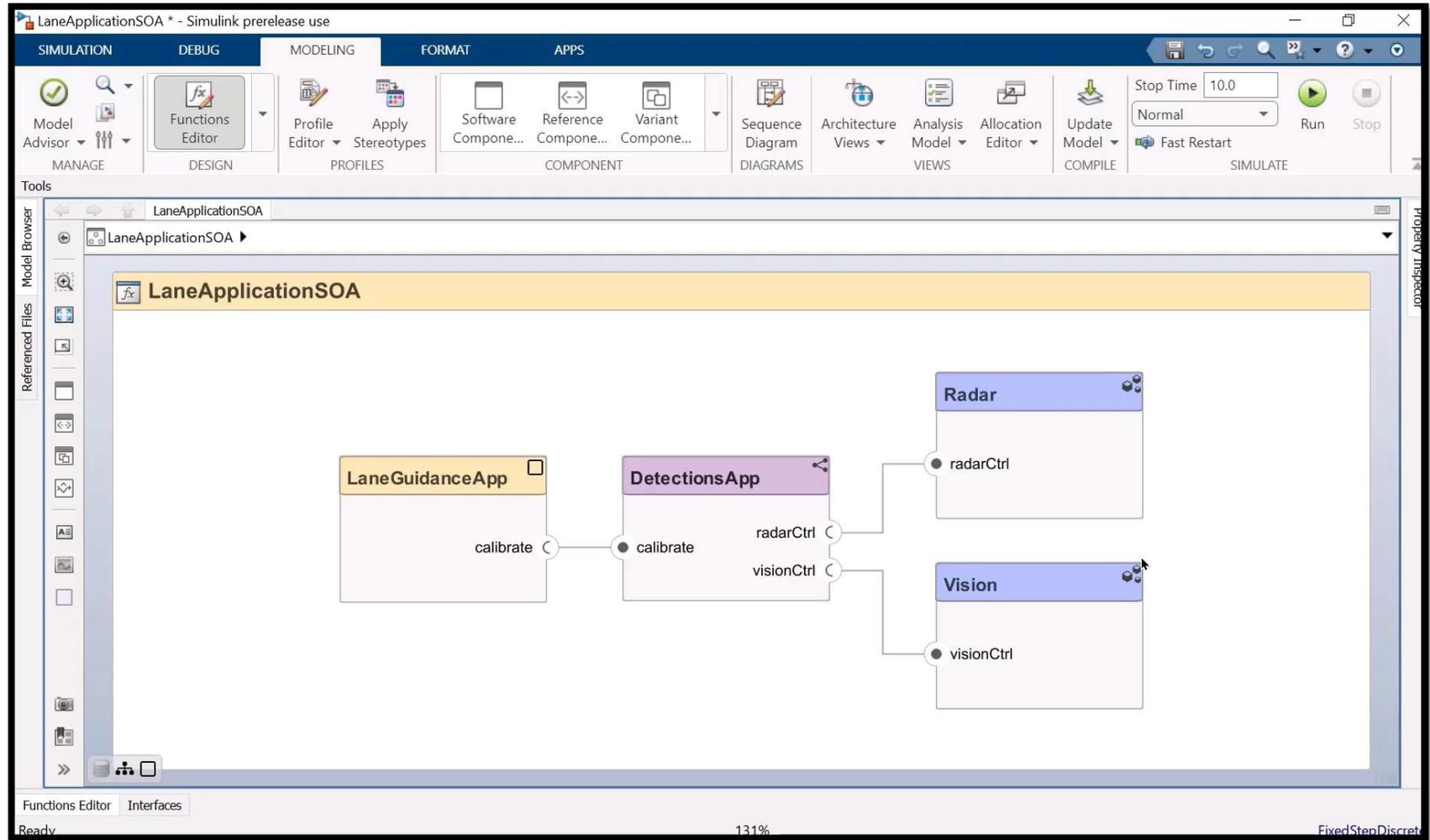
- 定义服务的职责、范围和依赖关系等
- 作为服务接口的补充
- 包括非功能属性



定义服务契约



- 指定服务的输入、输出和行为
- 可以看作服务原型
- 服务双方并行的前提



实现和部署服务

识别和分析服务

定义服务及接口

定义服务契约

实施和部署服务

- 配置中间件 (AP) 属性
- 检查映射关系
- 生成C++代码

The screenshot shows the Simulink environment with the following components and connections:

- Model Browser:** LaneApplicationSOA
- Referenced Files:** LaneApplicationSOA
- Model Diagram:**
 - LaneGuidanceApp** (containing **calibrate**)
 - DetectionsApp** (containing **calibrate** and **visionCtrl**)
 - Radar** (containing **radarCtrl**)
 - Vision** (containing **visionCtrl**)
- Property Inspector:**

NAME	VALUE
Main	
Name	LaneApplicationSOA
Exported Composition Name	LaneApplicationSOA
Stereotype	Add..
Parameters	
	Select

实现和部署服务

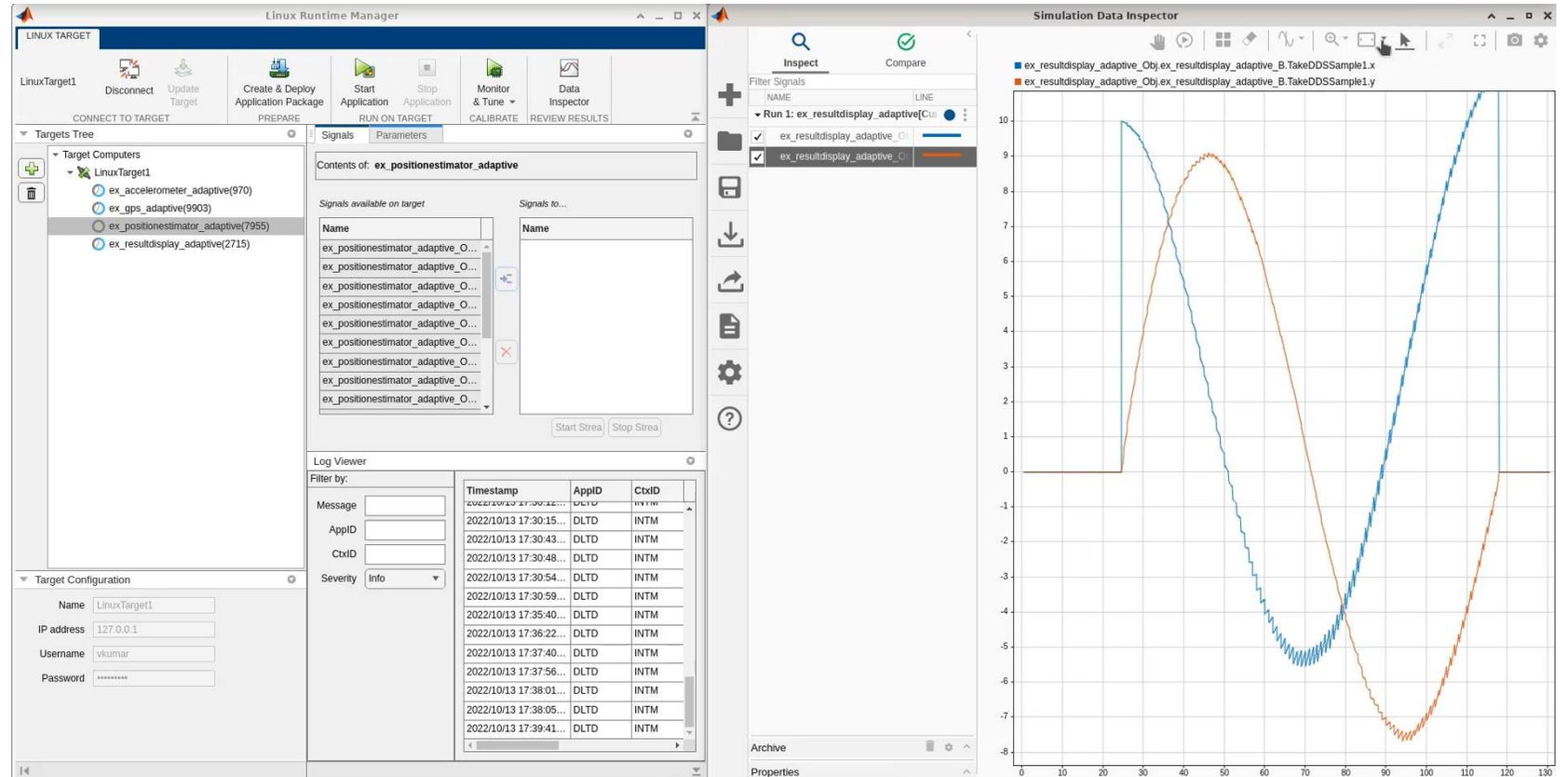
识别和分析服务

定义服务及接口

定义服务契约

实施和部署服务

- 快速原型部署
- 服务运行控制
- 测量和标定



总结

- 将传统汽车应用转换为 SOA 应用所面临着诸多挑战
- 需要结合设计原则和工程经验开展服务的识别和分析
- 选择合适的流程、方法和工具平台实施迁移工作
- 基于模型设计提供了SOA迁移的关键能力
 - 服务的定义和建模
 - 自动生成C++代码
 - 快速原型部署和验证

MATLAB EXPO

中国

5月30日 | 上海

6月06日 | 北京

Register at matlabexpo.cn



© 2023 The MathWorks, Inc.



获取更多MATLAB & Simulink在AUTOSAR的应用