基于订阅者/发布者的微内核操作系统进程间通信的形式化验证

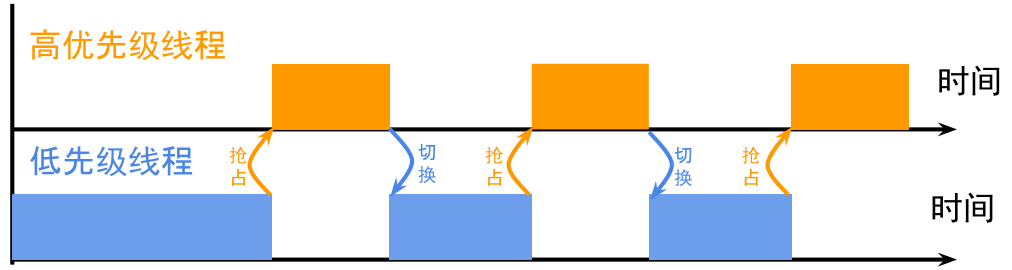
郭　建
jguo@sei.ecnu.edu.cn
华东师范大学

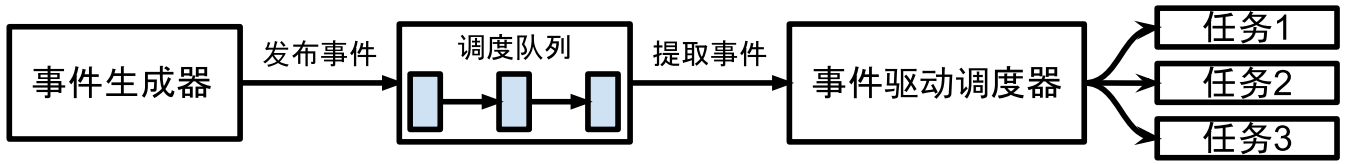
大纲

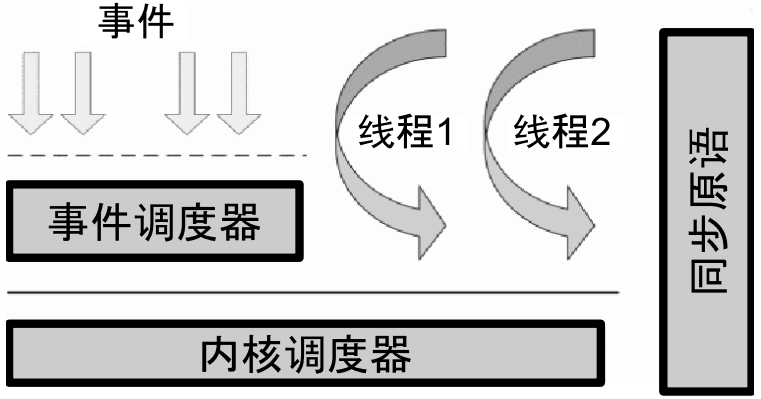
- 基于事件总线的微内核混合编程模型
- EventB 的形式化建模与验证框架
- 事件总线的重写需求与精化策略
- 事件总线的抽象模型
- 事件总线的精化模型
- 证明与结果分析

编程模型

- 多线程模型
- 事件驱动模型
- 混合模型





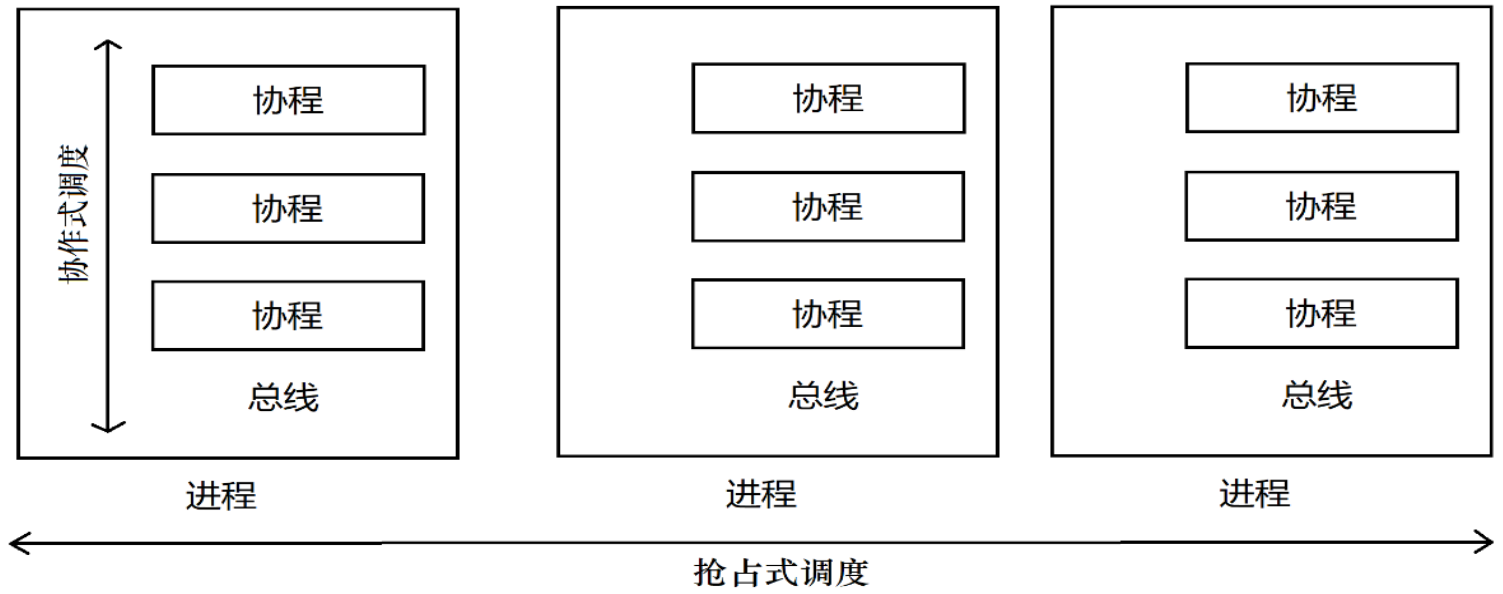


线程的分类

- 用户级线程（类似于协程）
  - 内核不知线程存在，只负责调度进程
  - 每一时刻，进程内部只有一个线程在运行
  - 线程之间以协作方式调度：线程主动将控制权交给其他线程
- 内核级线程（轻量级进程）
  - 内核知晓线程存在，以线程为基本调度单位
  - 进程成为线程的容器/分区
  - 多处理器上，同一进程中可以有多个线程同时运行

抢占式调度

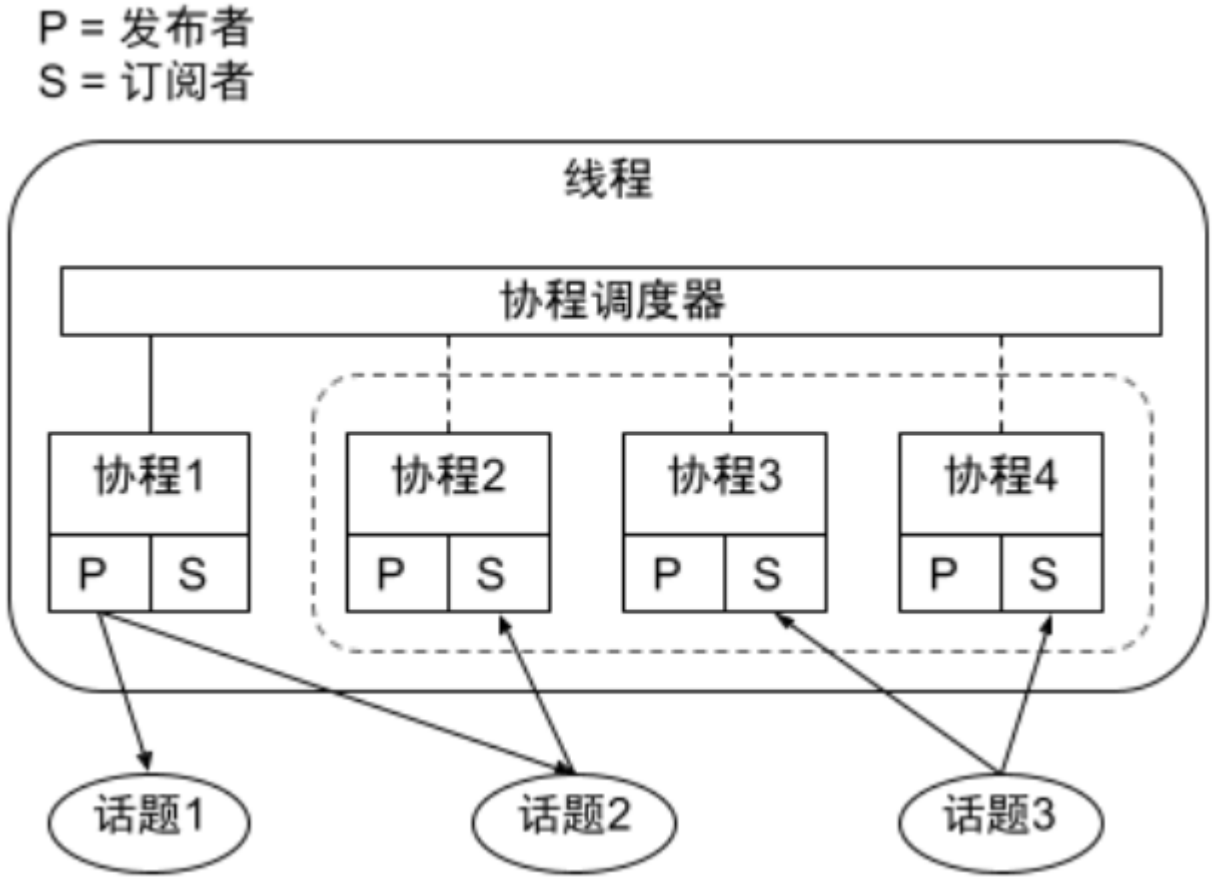
- 进程内包含若干协程
- 管理协程称为事件总线，管理事件的收发和协程的调度
- 协程运行到切换点，暂停运行，跳转至事件总线，由事件总线调度协程



事件总线混合编程模型

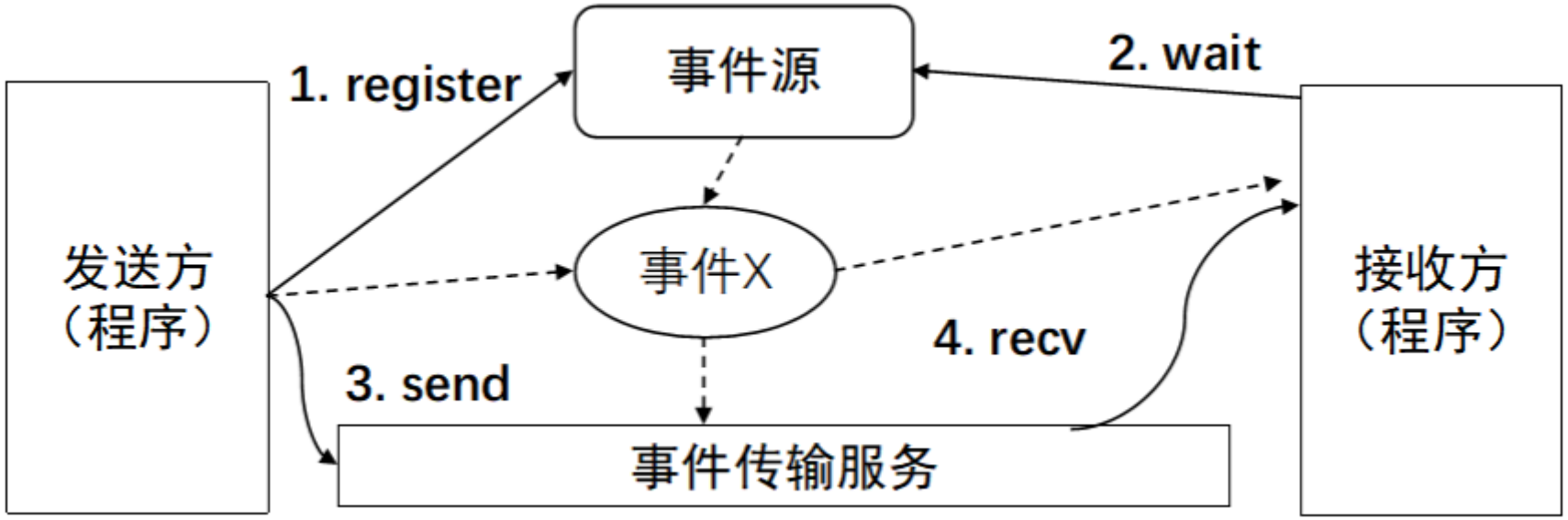
- 管理协程称为事件总线，

  管理事件收发和协程调度
- 集成模块之间的通信
- 自定义的事件调度方法
- 事件驱动程序的简化



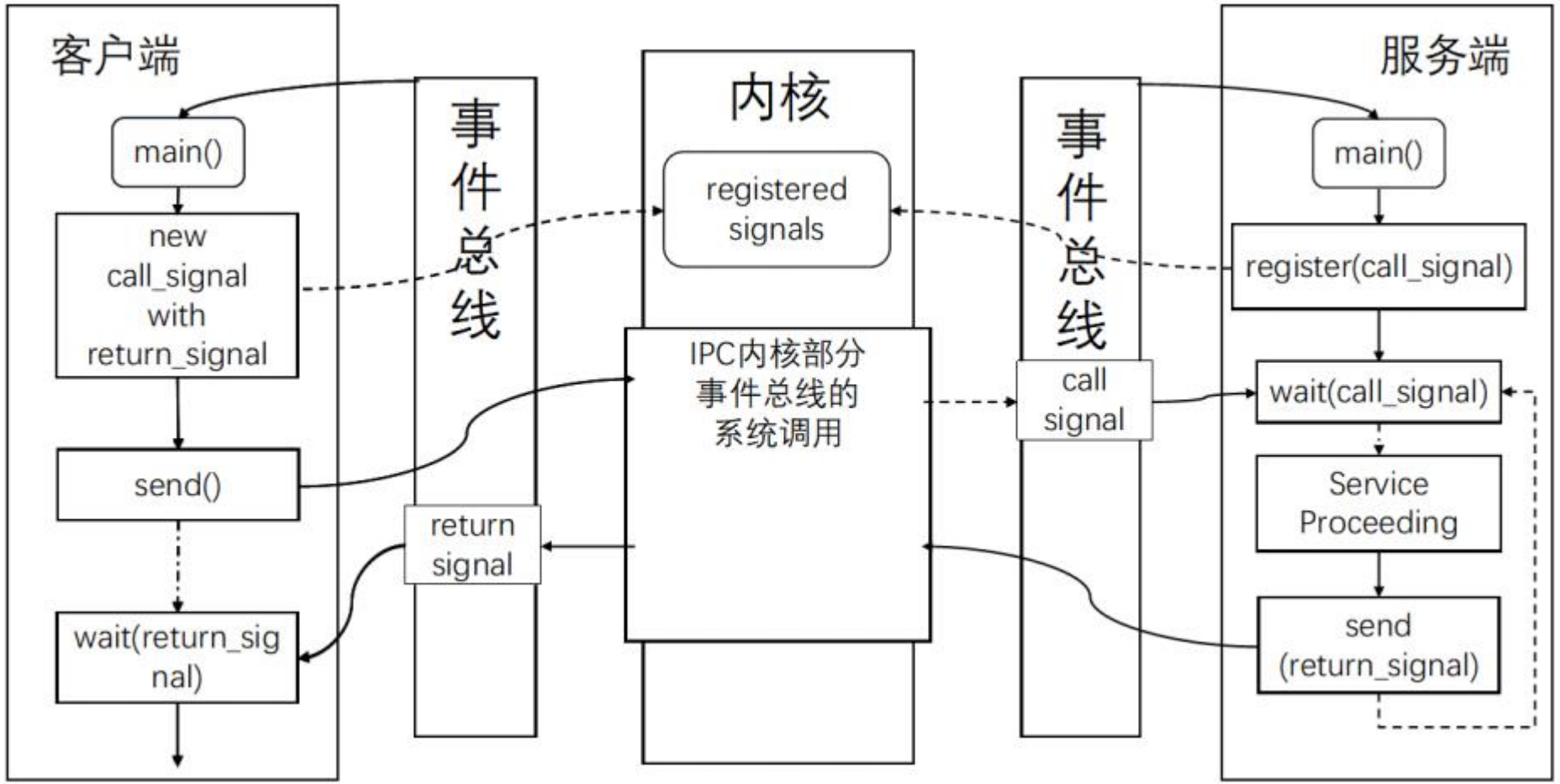
事件总线中事件的传递

- 事件总线的内部对象可以抽象为发布订阅模式，事件源对应话题、 事件对应消息、程序对应订阅者或者发布者。

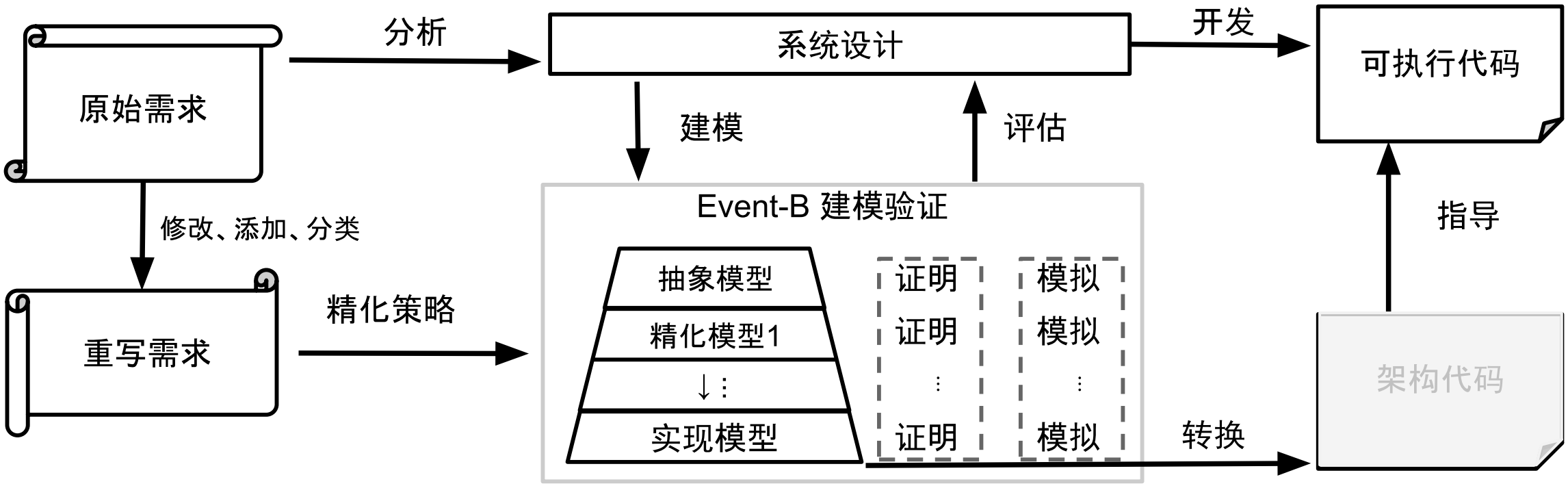


事件总线的同步服务调度

- 事件总线
  - 同步服务调用
  - 异步服务调用
- 区别是在进行服务调用后是否立即等待调用结果。



形式化建模与验证框架


原始需求
分析
系统设计
开发
可执行代码
建模
评估
修改、添加、分类
Event-B 建模验证
抽象模型
精化模型1
实现模型
证明
证明
证明
模拟
模拟
模拟
指导
重写需求
精化策略
架构代码
转换

重新需求

- 修改
  - 初始需求不清楚或不合理的需求
- 分解与添加
  - 对复杂的需求进行分解，以方便建模实现
  - 增加描述系统属性的需求
- 分类
  - **FUN**：功能需求
  - **ENV**：环境需求
  - **SAF**：性质需求
- 重写需求共 36 个：15 个功能需求，19 个环境需求和 2 个性质需求

重写需求与精化策略

模型名称	上下文	抽象机	需求	描述
模型 0（抽象模型）	C0	M0	1 到 9	建立发布-订阅模式模型。
模型 1	C0	M1	10 到 16	区分消息处理方法。
模型 2	C1	M2	17 到 21	引入线程的概念。
模型 3	C2	M3	22 到 25	引入协程的概念。
模型 4	C2	M4	26 到 27	分解消息创建。
模型 5	C2	M5	28	替换模型参数。
模型 6	C3	M6	29 到 35	介绍协程状态。
模型 7（实现模型）	C3	M7	36	验证无死锁属性。

模型分类：五类

- **发布订阅模式** 模型 0（被称为抽象模型）对事件总线的最抽象机制（发布订阅模式）的建模。该模型描述了发布订阅模式的基本功能。
- **消息处理** 模型 1，表达了消息处理的具体实现，添加了发布队列，调 度队列和接收队列的概念。此处对非法和合法消息的判断和处理有所区别。
- **线程和协程** 模型 2 到模型 5，引入线程和协程的概念。发布者和订阅者的两个角色逐渐被线程和协程取代。
- **运行状态** 模型 6，引入了协程的运行状态，对系统的调度进行了建模。
- **辅助证明** 模型 7，也被称为实现模型，事件总线的最后一级模型，它的作用是帮助证明模型 6 的无死锁属性，它不涉及任何逻辑与实现层面的修改。

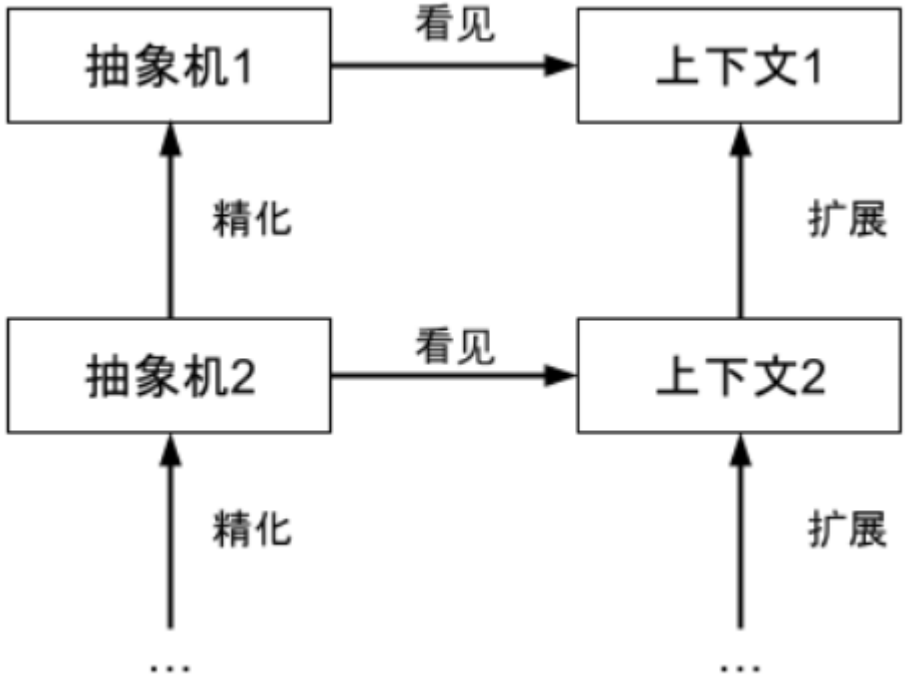
抽象模型：重新需求

- 抽象模型重写需求共 9个
  - 4个功能需求
  - 4 个环境需求
  - 1 个性质需求

事件总线包含一些可以发送一些消息的发布者。	ENV-1
事件总线包含一些可以接收一些消息的接收者。	ENV-2
事件总线定义了一组主题。	ENV-3
发布者发送的每条消息都与一个唯一的主题相关联。	ENV-4
事件总线可以创建新的发布者、接收者、主题。	FUN-1
订阅者可以对主题进行订阅，将主题添加到其订阅集合中进行记录。	FUN-2
订阅者都可以通过从其订阅集合中删除该主题来取消订阅主题。	FUN-3
订阅者可以取消对某个它感兴趣主题的订阅，之后将不会收到与该主题相关的任何消息。	FUN-4
当发布者发送消息时，对消息主题感兴趣的订阅者最终将收到消息。	SAF-1

基于event-B的建模

- EventB 是基于精化技术的形式化方法
  - 采用谓词逻辑和集合论进行系统建模
  - 精化模型之间的一致性检查用于验证不同层次模型的一致性
  - EventB 语言有两个组成部分：抽象机和上下文
    - 抽象机描述其动态规范。一个抽象机可以精化另一个抽象机
    - 上下文描述静态规范，包括全局集合、常量和公理。一个上下文可以扩展另一个上下文



抽象模型—上下文

- 四个集合：主题集合、发布者集合、订阅者集合和消息集合

```
CONTEXT C0
    SETS
        TOP     \\主题集合
        PUB     \\发布者集合
        SUB     \\订阅者集合
        MSG     \\消息集合
    END
```

抽象模型—抽象机

- 抽象模型的抽象机是 M0，描述发布订阅模式的动态规范，包含一系列变量、不变量和事件。
- 定义了 10 个变量和 11 个不变式
- main_1，称为“主属性”，它描述了系统的正确运行状况
  - 对消息主题感兴趣的订阅者最终能收到发布者发送的消息

VARIABLES:

top

sub

pub

msg

tms

pms

rms

era

erm

stp

INVARIANTS:

inv1_1 : $top \subseteq TOP$

inv1_2 : $sub \subseteq SUB$

inv1_3 : $pub \subseteq PUB$

inv1_4 : $msg \subseteq MSG$

inv1_5 : $tms \in msg \rightarrow top$

inv1_6 : $pms \in msg \rightarrow pub$

inv1_7 : $rms \in sub \rightarrow \mathbb{P}(msg)$

inv1_8 : $erm \in msg \rightarrow 1 \,..\, era$

inv1_9 : $era \in \mathbb{N}_1$

inv1_10 : $stp \in (1 \,..\, era) \rightarrow (top \leftrightarrow sub)$

main_1: $\forall s, m \cdot s \in sub \wedge m \in rms(s) \Rightarrow tms(m) \mapsto s \in stp(erm(m))$

抽象模型—抽象机（续）

- 基本操作：7个事件
  - 创建类
    - 创建主题事件 create_top、创建发布者事件 create_pub 和创建订阅者事件 create_sub
  - 主题类
    - 订阅主题subscrib_topic
    - 取消主题 rmv_topic
  - 发布类
    - send_msg 指示发布者 p 发布主题为 t 的消息 m
    - handle_msg 检测新发布的消息，并将其分配给对该消息主题感兴趣的每个订阅者
- 证明义务的证明：40个自动证明

EVENT create_top

ANY t

WHERE

grd1: $t \notin top$

THEN

act1: $top := top \cup \{t\}$

END

EVENT subscrib_topic

ANY s, t

WHERE

grd1: $s \in sub$

grd2: $t \in top$

THEN

act1: $stp(era) := stp(era) \cup \{t \mapsto s\}$

END

EVENT send_msg

ANY p , m , t

WHERE

grd1: $p \in pub$

grd2: $m \notin msg$

grd3: $t \in top$

THEN

act1: $msg := msg \cup \{m\}$

act2: $tms(m) := t$

act3: $pms(m) := p$

act4: $erm(m) := era$

END

精化模型

- 消息分类处理的逻辑（模型1）
- 引入线程和协程概念（模型2~5）
- 添加系统的运行状态（模型6）
- 辅助证明无死锁属性（模型7）

消息分类处理的逻辑（模型1）

订阅者的接收队列中的消息被该订阅者处理后，该消息将从其订阅者的接收队列中删除。	FUN-5
发送给订阅者的消息是订阅者订阅的，该消息被订阅者接收。	FUN-6
发送给订阅者的消息不是订阅者订阅的，该消息不被订阅者接收。	FUN-7

- 表达了消息处理的具体实现，添加
  - 发布队列的概念
  - 调度队列的概念
  - 接收队列的概念
  - 对非法和合法消息的判断和处理

引入线程和协程概念（模型2~5）

- 从第二层模型到第五层模型
- 引入线程和协程的概念。发布者和订阅者的两个角色被线程和协程取代。
- 线程和协程的关系如下：操作系统中有一个或多个线程，而一个线程中有一个或多个协程。
- 事件总线中的每个协程都可以视为与发布者和订阅者结合

添加系统的运行状态（模型6）

- 引入了协程的运行状态
  - 协程有两个状态，活动状态和睡眠状态
  - 协程进入睡眠状态，只能执行接收消息事件
  - 线程中最多有一个协程正在运行，两个事件 wait_msg 和 recv_msg 定义了协程状态的转变
    - 事件 wait_msg 将运行状态从活动更改为睡眠
    - 事件 recv_msg 将运行状态从更改为从 sleeping 到 active。
- 系统的协程调度进行了建模。

辅助证明无死锁属性（模型7）

- 事件总线的最后一级模型，
  - 帮助证明模型6的无死锁属性，它不涉及任何逻辑与实现层面的修改
  - 交互式证明

性质：证明义务

- 上下文：集合 s 和常量集 c 构成上下文，
- v 是执行事件之前的变量集，而 v′ 是执行事件之后的变量集。
- 公理：不需要证明就成立的性质，公理集用记为 P(s, c)
- 不变式 ： 系统在运行期间始终满足某些特性，不变式集记为 I(s, c, v)
  - 胶合不变式证明义务：保证两个精化模型的一致性
- 卫兵条件增强证明义务：保证精化模型中的卫兵条件是不断增强的
- 仿真证明义务：保证精化模型能够对抽象模型的所有行为进行仿真

性质：证明义务（续）

- 定理证明义务：定理证明义务有自动创建，以确保可以从公理和不变式证明所陈述的上下文和抽象机中定义的定理，分为2类：T(s, c, v) 和 T(s, c)。
  - P(s, c) ⇒ T(s, c)
  - P(s, c) ∧ I(s, c, v) ⇒ T(s, c, v)

性质：关键属性

- 无死锁性：每个模型在任何状态下至少有一个要执行的事件
  - 在每个模型级别分别证明其无死锁属性。
  - 较低级别模型的证明结果可以用作定理，以帮助证明较高精化的模型。
  - 令 $G_i(1 < i < n)$ 为第 i 个事件的所有卫兵条件的并，抽象模型无死锁性质为：
  - **dlf_0:**

$$P(s,c) \wedge I(s,c,v)$$

$$\Rightarrow$$

$$G_1(s,c,v) \vee G_2(s,c,v) \vee \cdots \vee G_n(s,c,v)$$

性质：关键属性(续)

- 无死锁性：每个模型在任何状态下至少有一个要执行的事件
  - 抽象模型的第一个精化模型的无死锁性：
    - dlf_1: $dlf\_0 \vdash P(s, c) \wedge I(s,c,v) \Rightarrow G_1(s, c, v) \vee G_2(s, c, v) \vee \cdots \vee G_n(s, c, v)$
  - 实现模型（最后一层精化模型）的无死锁性：

dlf: $dlf\_0, \ldots, dlf\_6 \vdash P(s, c) \wedge I(s, c, v) \Rightarrow$

$$G_1(s, c, v) \vee G_2(s, c, v) \vee \cdots \vee G_{16}(s, c, v)$$

性质：关键属性(续)

- 主属性：正确传递系统中的所有消息
  - 抽象模型对主属性的描述：如果订阅者 s 收到消息 m，则发送 m 时，s 必须对该主题感兴趣

main_1: $\forall s, m \cdot s \in sub \wedge m \in rms(s) \Rightarrow tms(m) \mapsto s \in stp(erm(m))$

- 随着模型的完善，主属性的描述和证明也会发生变化。在第一层、第五层精化模型中重写了主属性。

性质：关键属性(续)

- 主属性：正确传递系统中的所有消息
  - .....

➢随着模型的完善，主属性的描述和证明也会发生变化。在第一层、第五层精化模型中重写了主属性。

➢在第一层精化模型中，事件 **handle_msg** 分为必需事件 **recv_msg** 和拒绝事件 **refu_msg**。需要修改主属性以匹配本层模型的精化结果，具体表现为通过添加的变量 **e** 描述事件以确保消息的及时性。

main_2: $\forall s, m \cdot s \in sub \land m \in rms(s) \Rightarrow$

$(\exists e \cdot e \in erm(m) \mathinner{..} era \land tms(m) \mapsto s \in stp(e))$

性质：关键属性(续)

- 主属性：正确传递系统中的所有消息
  - .....

➢随着模型的完善，主属性的描述和证明也会发生变化。在第一层、第五层精化模型中重写了主属性。

➢.....

➢在第五层精化模型中引入了线程和协程以分别替换发布者和订阅者。协程 c 替换了订阅者 s。

main_3: $\forall c, m \cdot c \in crt \wedge m \in crms(c) \Rightarrow$

$(\exists e \cdot e \in erm(m) \mathinner{..} era \wedge tms(m) \mapsto c \in cstp(e))$

性质证明

- 不变式建立证明义务
  - 抽象机的初始事件运行后全部不变式需要得到保持。
    - *K(s, c, v')* 是初始事件的后谓词集

$$P(s,c) \wedge K(s,c,v')$$

$$\Rightarrow$$

$$I(s,c,v')$$

性质证明(续)

- 不变式保持证明义务
  - 要求系统在运行期间始终满足某些特性
    - *G(s, c, v)* 为执行的事件的卫兵条件集，*R(s, c, v, v')* 为抽象机事件的前后谓词集

$$P(s,c) \wedge I(s,c,v) \wedge G(s,c,v) \wedge R(s,c,v,v')$$

$$\Rightarrow$$

$$I(s,c,v')$$

性质证明(续)

- 定理证明义务
  - 是自动创建的，以确保可以从公理和不变式证明所陈述的上下文和抽象机定理
  - 定理可以分为 *T(s, c, v)* 和 *T(s, c)*，

$$P(s,c) \Rightarrow T(s,c)$$

$$P(s,c) \wedge I(s,c,v) \Rightarrow T(s,c,v)$$

证明方法

- Rodin 是 EventB 方法的基于 Eclipse 的集成开发平台
- 利用 Atelier B，AnimB 和 ProB 插件进行辅助建模验证
  - Atelier B 被用来自动生成与证明证明义务，大多数证明义务都在无需用户干预的情况下可以完成
  - ProB 对系统进行约束求解和模型检查，利用 ProB 的约束解决功能进行生成测试用例、死锁检查、模型检测
  - AnimB可以对模型进行模拟和仿真

结果统计与分析

- 共有634 个证明义务，其中577个为自动证明，其余 57 个需要交互式证明，自动证明率为 91%。
  - 其中每个精化模型中的无死锁属性都需要进行交互式证明

模型名称	总的证明义务数量	自动证明数量	交互证明数量
M0	40	40(100%)	0(0%)
M1	65	64(98%)	1(2%)
M2	88	74(84%)	14(16%)
M3	101	94(93%)	7(7%)
M4	22	21(95%)	1(5%)
M5	216	195(90%)	21(10%)
M6	78	78(100%)	0(0%)
M7	24	11(46%)	13(54%)
总计	**634**	**577(91%)**	**57(9%)**

谢谢