

# Windows Workflow Foundation



Константин Красноперов  
Team lead



Артем Шумков  
Разработчик



НПО «Компьютер»  
Информационные системы управления

# Workflow

- ▶ Библиотека\компонент
- ▶ Описание операций\действий в рамках процесса
- ▶ История развития:
  - ▶ Версия 3.5
  - ▶ Версия 4.0
  - ▶ Версия 4.5
- ▶ Используется в Sharepoint
- ▶ Исходники на [referencesource.microsoft.com](https://referencesource.microsoft.com)

# Что такое workflow?

- ▶ Персистенность
  - ▶ SQL хранилище
  - ▶ PostgreSQL
- ▶ Активности
- ▶ Bookmarks
- ▶ Обновление схем
- ▶ Транзакционность

# Когда нужно использовать workflow?

- ▶ Процесс/алгоритм имеет длительный срок выполнения. (распределённый, несколько участников)
- ▶ Требуется настройка без участия программиста

Пример



# Хостинг Workflow

## Workflow-сервис

**WorkflowServiceHost** - хост для размещения службы

### **(!) Обработка нескольких рабочих процессов**

- Создание новых экземпляров рабочих процессов
- Загрузка существующих экземпляров из хранилища
- Выполнение
- Персистентность
- Закладки
- Обработка входящих запросов от сервиса

# Хостинг Workflow

## Workflow-сервис

### Варианты размещения:

- Обычное управляемое .NET-приложение
- Служба Windows
- Приложение на IIS

### Транспорт:

http, net.tcp, net.pipe, MSMQ

# Хостинг Workflow

## Workflow-сервис

### Типы хостинга:

- **Статический**

Определение маршрута в xamlx-файле, находится в каталоге приложения

- **Динамический**

Определение маршрута в сторонней сборке в bin-каталоге. В каталоге приложения svc-файл со ссылкой на определение маршрута

```
<%@ServiceHost
```

```
Factory="System.ServiceModel.Activation.WorkflowServiceHostFactory"
```

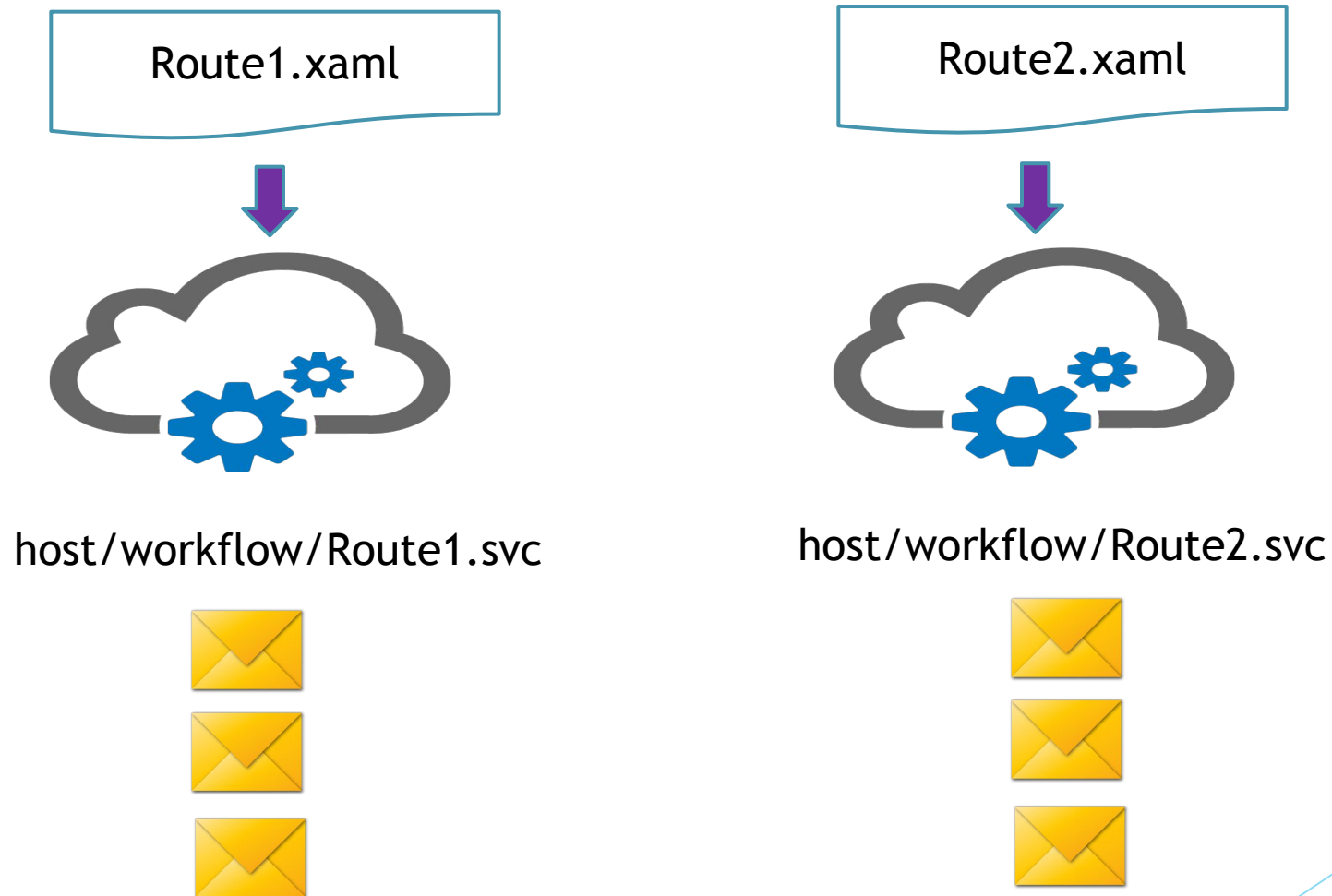
```
Service="SampleService" %>
```

- **Явный подъём хоста**



# Workflow в DirectumRX

## Архитектура Workflow-сервиса



# Workflow-сервис

## Workflow Instance Management

### ***WorkflowControlEndpoint:***

стандартная конечная точка с фиксированным контрактом  
*IWorkflowInstanceManagement*

### **Операции:**

- **Terminate**, **Cancel** - принудительно завершить процесс
- **Suspend** - приостановить процесс
- **Unsuspend** - возобновить приостановленный процесс

# Workflow-сервис

// Создать хост для схемы маршрута.

```
CreateServiceHost(Uri baseAddress, RouteDescription route)
{
```

    // Поднимаем хост

```
    activity = LoadRouteActivity(route);
```

```
    routeService = new WorkflowService()
```

```
    {
```

```
        Body = activity,
```

```
        ConfigurationName = route.Name
```

```
    };
```

```
    host = new DynamicServiceHost(routeService, new[] { baseAddress });
```

    // Добавляем конечную точку для прослушивания msmq-очереди

```
    uri = new Uri($"net.msmq://localhost/private\\workflow/{route.Name}.svc");
```

```
    config = new ServiceElement(route.Name);
```

```
    endpoint = new ServiceEndpoint("netMsmqBinding", uri, contract: "ISampleRoute")
```

```
    config.Endpoints.Add(endpoint);
```

```
    host.LoadConfigurationSection(config);
```

```
}
```

# Workflow-сервис

## Интеграция с WCF

(!) Контракт сервиса выводится автоматически на основе **Receive** и **SendReply**

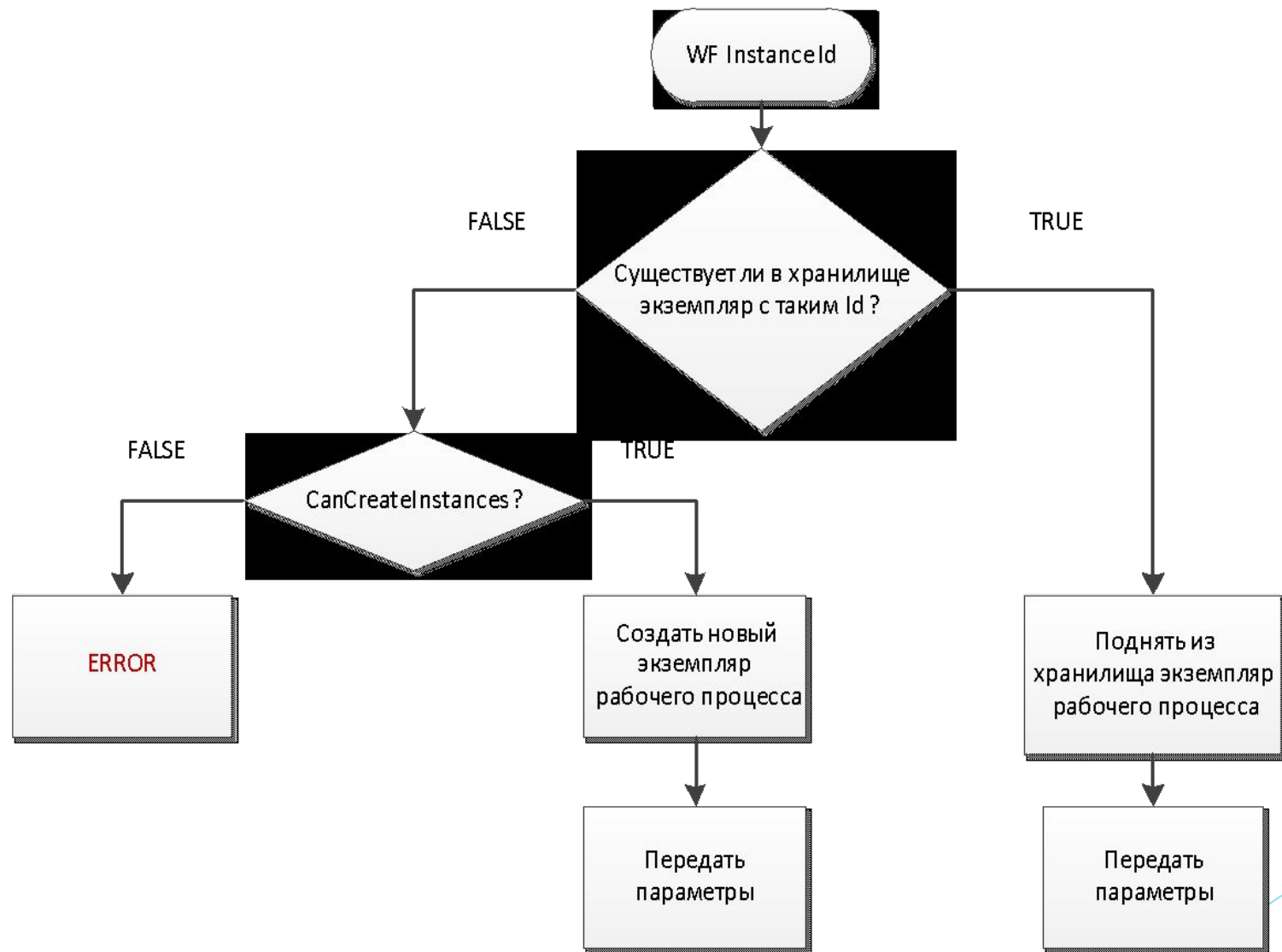
```
/* Точка входа */  
ReceiveRequest  
{  
    // Имя контракта сервиса  
    ServiceContract  
    // Метод сервиса  
    Operation  
    // Параметры  
    Parameters: [...]  
    // Создать новый процесс ?  
    CanCreateInstances  
}
```



```
/* Точка выхода */  
SendReply  
{  
    // Запрос  
    Request  
    // Ответные данные  
    Response: [...]  
}
```

# Workflow-сервис

## Корреляция



# Workflow-сервис

## Виды корреляций (1)

### Корреляция через контекст

двухсторонний протокол (например, http)

передача через контекст (в заголовках или cookie-файлах)

- **Неявная**

работает автоматом в парах `Receive/SendReply` и `Send/ReceiveReply`

- **Явная**

отправка и получение связаны через дескриптор `CorrelationHandle`

`Receive`: `RequestReplyCorrelationInitializer (handle)`

`SendReply (handle)`

# Workflow-сервис

## Виды корреляций (2)

### Корреляция на основе содержимого

протокол не поддерживает обмен контекстом

некоторые данные в сообщениях уникально идентифицируют нужный экземпляр рабочего процесса

```
QueryCorrelationInitializer (handle, MessageQuery(k1, k2))
```

Receive:

```
CorrelatesWith (handle)
```

```
CorrelatesOn: MessageQuery(k1, k2)
```

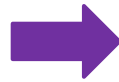
# Workflow-сервис

## Клиент-серверные транзакции

Клиент:

**TransactionScope**

```
{  
  Send (request, params)  
  ...  
  ReceiveReply (request)  
}
```



Сервер:

**TransactedReceiveScope**

```
{  
  Request:  
    Receive (operation, params)  
  Body:  
    ...  
    SendReply(message)  
}
```



# Workflow-сервис

## Логгирование

```
/// <summary>
/// Записать трассировочное сообщение.
/// </summary>
public class Track : WorkflowCodeActivity
{
    protected override void InternalExecute(CodeActivityContext context)
    {
        context.Track(new TaskTrackingRecord(tenant, taskId, message, args));
    }
}

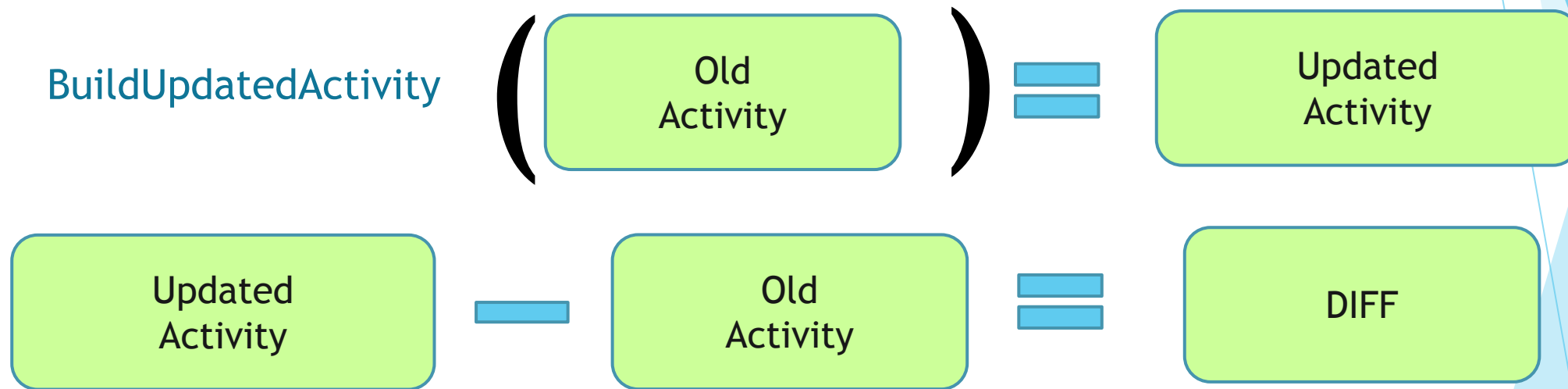
public sealed class LoggerTrackingParticipant : TrackingParticipant
{
    protected override void Track(TrackingRecord record, TimeSpan timeout)
    {
        this.logger.Log(record);
    }
}

<behaviors>
  <serviceBehaviors>
    <behavior>
      <workflowExtensions>
        <add extension="Sungero.Workflow.Service.Management.LoggerTrackingParticipant, Sungero.Workflow.Service"/>
      </workflowExtensions>
    </behavior>
  </serviceBehaviors>
</behaviors>
```

# Workflow-сервис

## Обновление маршрутов

### (1) Построение DIFF:



```
OldActivity;  
DynamicUpdateServices.PrepareForUpdate(OldActivity);  
UpdatedActivity = BuildUpdatedActivity(OldActivity);  
DIFF = DynamicUpdateServices.CreateUpdateMap(updatedActivity);
```

# Workflow-сервис

## Обновление маршрутов

### (2) Обновление экземпляров:



```
Instance = WorkflowApplication.GetInstance(InstanceId, Store);  
wfApplication = new WorkflowApplication(UpdatedActivity, RouteName);  
wfApplication.Load(Instance, DIFF);  
wfApplication.Unload();
```

# Workflow-сервис

## Версионирование

Поднимаем хост как обычно:

```
// Актуальная версия маршрута
WorkflowService service_V2 = new WorkflowService
{
    Name = "SampleService",
    Body = new SampleActivity_V2(),
    DefinitionIdentity = new WorkflowIdentity
    {
        Name = "SampleService",
        Version = new Version(2, 0, 0, 0)
    }
};

WorkflowServiceHost host = new WorkflowServiceHost(service_V2,
    new Uri("http://localhost:8080/SampleService"));
```

# Workflow-сервис

## Версионирование

Добавляем старые версии для совместимости:

```
// Предыдущая версия маршрута.
```

```
WorkflowService service_V1 = new WorkflowService  
{  
    Name = "SampleService",  
    Body = new SampleService_V1(),  
    DefinitionIdentity = new WorkflowIdentity  
    {  
        Name = "SampleService",  
        Version = new Version(1, 0, 0, 0)  
    }  
};
```

```
// Добавляем старую версию в конечную точку службы  
host.SupportedVersions.Add(service_V1);
```

# Workflow-сервис

## Версионирование

- Несколько версий маршрута на одной конечной точке
- Новые экземпляры рабочих процессов стартуют по актуальной версии, а старые экземпляры дорабатывают по старым версиям
- Все версии имеют одинаковое имя сервиса и уникальные номера
- Контракт всех версий должен быть одинаковым

# Хостинг Workflow

## WorkflowApplication

Контейнер (среда выполнения) для WorkflowInstance:

### **(!) Только один экземпляр рабочего процесса**

- Создание новых экземпляров рабочих процессов
- Загрузка существующих экземпляров из хранилища
- Выполнение
- Персистентность
- Приостановка на закладке и возобновление
- Прекращение
- Входные и выходные параметры
- Передача параметров в закладки при возобновлении
- События жизненного цикла процесса

# Хостинг Workflow

## WorkflowApplication

События жизненного цикла процесса:

- Успешное завершение (Completed)
- Принудительное прекращение (Aborted)
- Приостановка (Idle)
- Выгрузка из памяти в хранилище (Unloaded)
- Ошибка (OnUnhandledException)



# WorkflowApplication

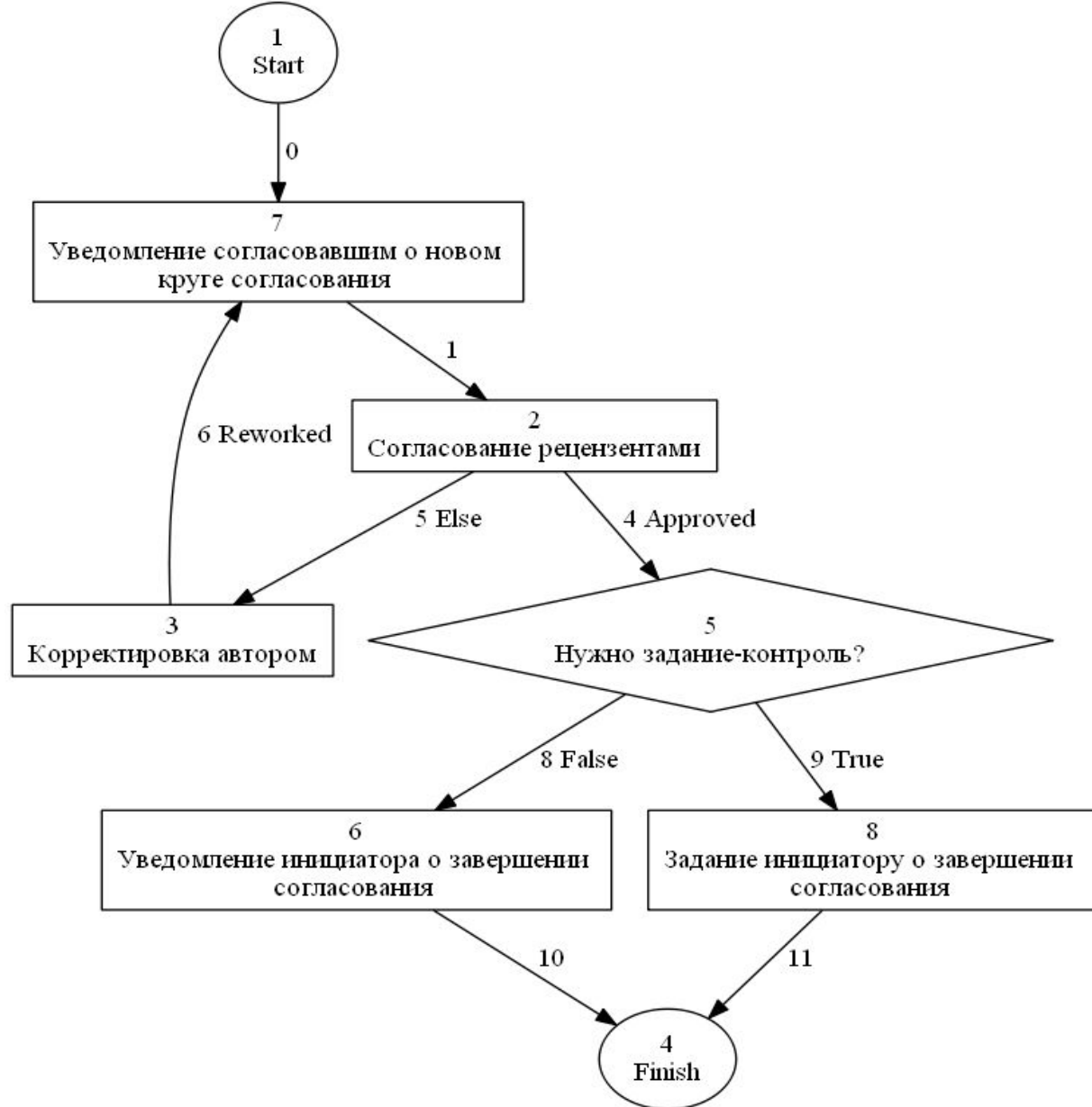
```
// Описание процесса
CodeActivity(InArgument, OutArgument)
{
    // Тело активности
    Execute(ActivityContext context)
    {
        context.CreateBookmark("BookmarkName", new BookmarkCallback(OnResumeBookmark));
    }

    // Вызывается после возобновления закладки
    OnResumeBookmark(ActivityContext context, Bookmark bookmark, object parameter)
    {
        OutArgument.Set(context, InArgument + parameter);
    }
}

// Входные параметры
inputs = new Dictionary { "InArgument" => "Hello, " };
wfApp = new WorkflowApplication(codeActivity, inputs);
// После завершения процесса получаем выходные параметры
wfApp.Completed += (s, e) => { WriteLine(e.Outputs["OutArgument"]) }
wfApp.Run();
// Возобновить процесс
wfApp.ResumeBookmark("BookmarkName", "Mr. Smith");
```

# Workflow в DirectumRX

# Разработка схем маршрутов



# Workflow в DirectumRX

## Разработка схем маршрутов

### RouteScheme.xml

#### Описание в виде графа

##### Активности:

(1) Задание/Уведомление

(2) Подзадача

(3) Условие

(4) Сценарий

(5) Мониторинг

##### Переходы:

1 → 2

2 → 3

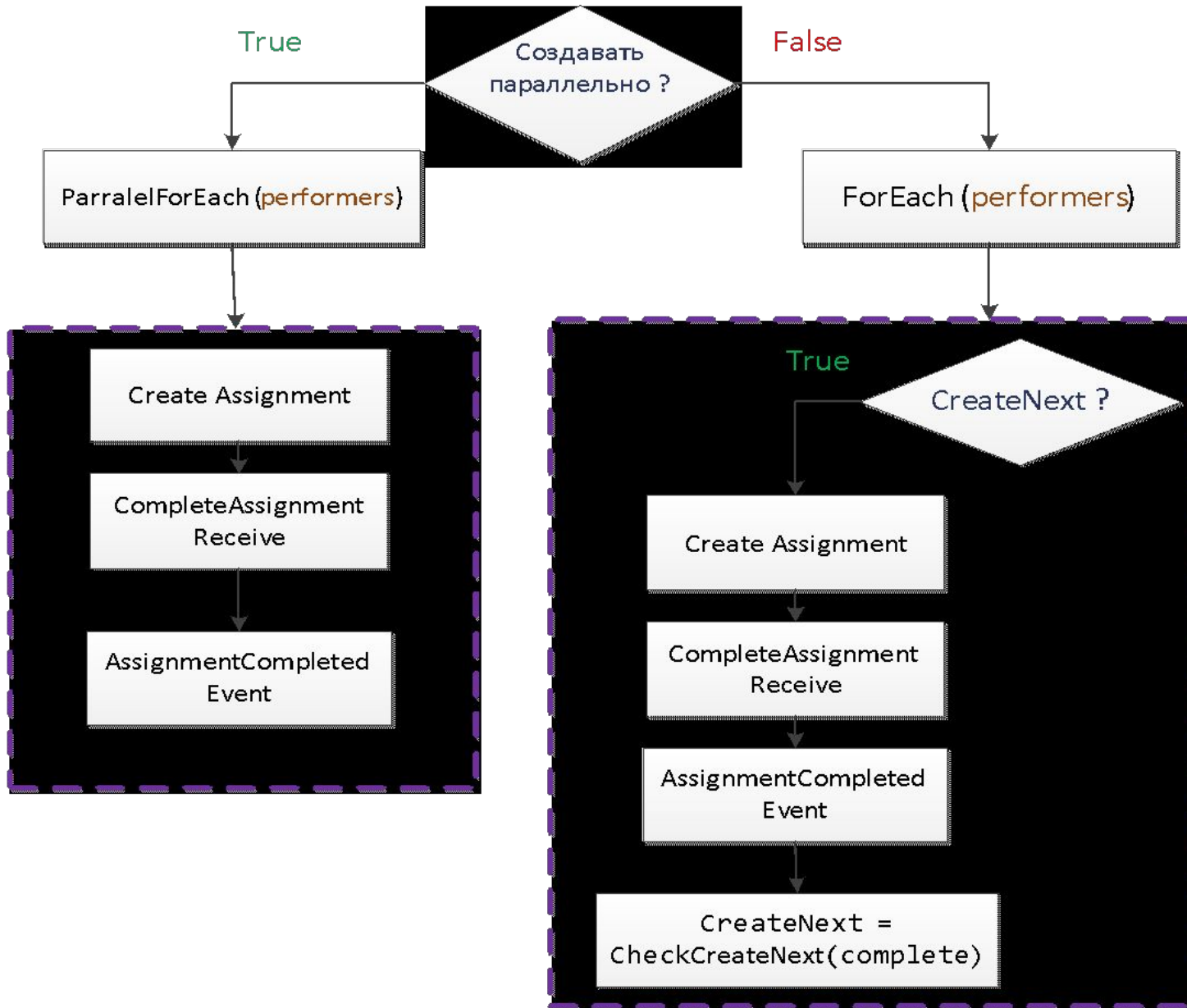
3 → 4 (True)

3 → 5 (False)

Параллельные ветки, OR/AND

# Преобразование из схемы в WF-маршрут

Задание



# Кастомные кодо-активности

// Активность для создания задания.

```
CreateAssignment: CodeActivity {
```

// Идентификатор задачи | Идентификатор исполнителя задания

```
InArgument<int> TaskId, PerformerId;
```

// Идентификатор созданного задания.

```
OutArgument<int> AssignmentId;
```

// Логика активности.

```
protected override void Execute(CodeActivityContext context) {
```

```
    using (var session = new Session()) {
```

// Получаем данные из контекста.

```
    performer = context.GetValue(this.PerformerId);
```

```
    mainTask = context.GetValue(this.TaskId);
```

// Создаём задание.

```
    assignment = session.Create<IAssignment>;
```

```
    assignment.Performer = performer;
```

```
    assignment.Status = AssignmentStatus.InProcess;
```

```
    ...
```

// Сохраняем данные в контекст.

```
    context.SetValue(this.AssignmentId, assignment.Id);
```

## Кастомные кодо-активности

- Хранение данных в контексте
- Доступ к закладкам
- Доступ к инфраструктуре
- Дочерние активности + операции с ними (Schedule, Idle, Abort)
- Максимум логики в коде

# Проблемы

- ▶ Есть ограничения на обновление схем workflow
- ▶ Мониторинги сделаны не оптимально
- ▶ Сломанный экземпляр просто так поднять не получится
- ▶ Сложно реализовать хранилище экземпляров workflow



Спасибо!

Остались вопросы?