# Inter-Organizational Workflows in FHIR

Oliver Krauss, Andreas Schuler,
Anna Lackerbauer

November 17 , 2016

# Who am I?

- **Name:** Oliver Krauss
- **Company:** University of Applied Sciences Upper Austria
- **Background:**
  - Researcher in Medical Informatics since 2012
  - Lecturer for Healthcare Standards
  - Main Focus on Interoperability & Automation
  - (lazy) e-Health Blogger (http://ehealth.fh-hagenberg.at)

# KIMBO – COLLABORATIVE INTERDISCIPLINARY MEDICAL BOARDS

# The project

- Collaboration between



- Promoted by FFG

# The goal


- A standardized collaboration between different organizations (hospital, family physician, radiologist, …)
- Using the hottest standard in healthcare
- Enabling partial automation
- Use Case
  - Tumor Board (also called *Multi Disciplinary Team Meeting (MDTM)*).

# WORKFLOWS IN FHIR

# About Workflows

- Are a way to define how business processes (should) look
- Using existing standards such as BPMN, CPMN, Petri-Networks, State-Machines, …
- Workflow Engines can run defined processes, allowing partial automation
- We use BPMN & FHIR

# What do we need?

- Requires a Definition

BPMN Process

PlanDefinition

- Requires Information about it's execution

BPMN ProcessInstance

CarePlan

# What do we need?

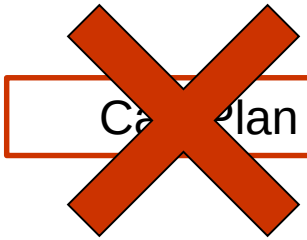
- Requires a Definition

BPMN Process

PlanDefinition

- Requires Information about it's execution

BPMN ProcessInstance



Composition

# Why not CarePlan?

- A CarePlan defines what *should* be done for the treatment of a *specific patient*
  - Primary focus - medical workflows
  - Does not need to be followed
- Interdisciplinary Workflows require:
  - exact specifications for all participants
  - these specifications to be followed!
  - may not be specific to one single patient
    - Tumor Board focuses on several patients at once

# The ProcessInstance Extension

- Reference(PlanDefinition) process 1..1 The Process that is being executed
- Reference(Task) tasks 0..* The tasks that will be executed / were already executed
  - Generated from actionDefinition by Transform during Task execution

# Defining your Workflow

- Using PlanDefinition to define the business process
- PlanDefinition.actionDefinition equals a User Task to be executed
  - ActivityDefinition & Transform can be used to generate Requests that are performed by someone else (see http://build.fhir.org/workflow.html#commpatterns)
  - We are missing Dataflow!
  - We are missing Automation!

# The Dataflow Extension

- The PlanDefinition and each actionDefinition get their own *DataPool*:
  - Endpoint endpoints 0..* List of Endpoints to get data for the task and/or the expected outcome of a task
  - PlanDefinition -> Global DataPool for entire process
  - PlanDefinition.actionDefinition -> Local DataPool for single task
  - As FHIR Endpoint:
    - GET & search for data provided by one organization to execute a task
    - PUT & POST for data the requesting organization expects as a result

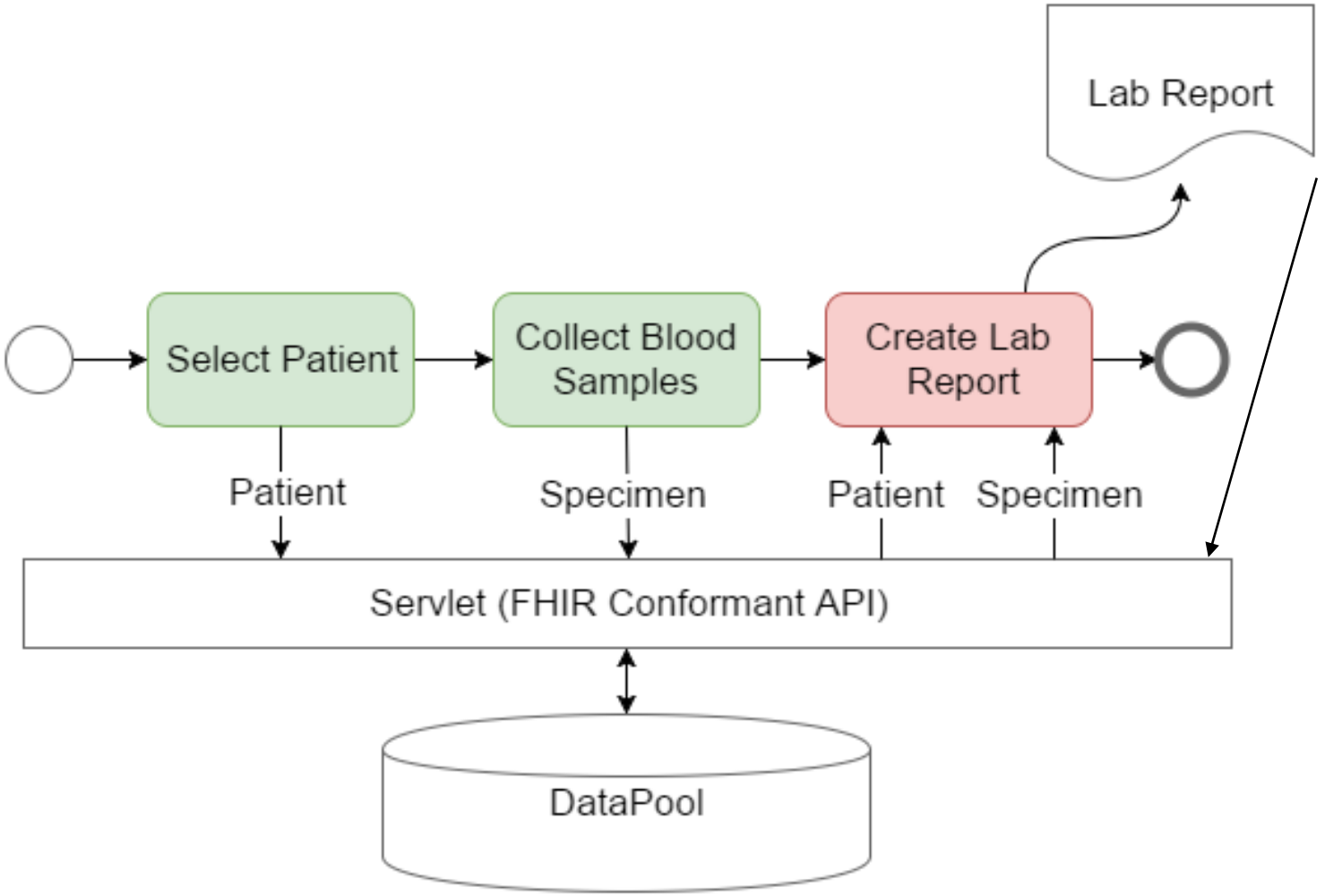
# The Dataflow Extension

- At Runtime:
  - A Transformer transforms the List of Endpoints into a Conformance Statement
  - [base]/Composition/{processInstanceId}/{actionDefinitionId/metadata} -> Points to a Capability (former Conformance) Statement
    - Everything behind this address = DataPool for the process
    - GET = load from pool | PUT, POST = push to pool

# The Dataflow Extension

- Are you insane?
  - Please refrain from asking inappropriate questions
- The Reason for the DataPool(s):
  - Tell partner-organizations in a machine-readable way *exactly* where to get input from, and how to give results back to you
  - Instant FHIR Server only for the information you want to give / need to have. Including all Profiles
  - After Execution -> DataPools = instant protocol of everything that happened (similar to IHE-XDW)

# The Dataflow Extension



# The Automation Extension

- Why automate the workflow?
  - Anonymization of data provided to other organizations
  - Standard tasks such as sending mails, notifications, …
  - Transporting information from A to B

# The Automation Extension

- (another) Extension on actionDefinition:
  - automated (Bool) (0..1)
  - If exists, and true this task will be handled by a computer system
- 4 Types of automated tasks:
  - Adding information to the Data Pool
  - Pushing information from the Data Pool to a (FHIR) Server
  - Handling the Request – Event pattern
  - *True* automation, calling a FHIR Operation

# THE WORKFLOW OF WORKFLOWS IN FHIR

# How to start a process?

1. Load all PlanDefinitions from the (Workflow) FHIR Server
2. Select the PlanDefinition you want to execute
3. FHIR Operation `$process-start`
   - IN: PlanDefinition
   - OUT: Composition with ProcessInstance Extension (containing info on the PlanDefinition)

- Done!

# $process-start

- IN: PlanDefinition
- OUT: Composition with ProcessInstance Extension
- Generates the DataPools (servlets) and initializes the Composition
- Composition contains reference to PlanDefinition + current and past Tasks (no future Tasks)
- Starts automated Tasks

# How to execute the process?

1. Composition contains all active tasks
   - Task references Practicioner that should execute it
   - Automated Tasks don‘t reference practitioner
2. Execute User-Tasks in your software
   - (optionally) store information on them in the Composition
3. When a Task is finished execute `$finish-task`
4. `$finish-task` will load the next Tasks

# $finish-task

- IN: Composition, TaskId
- OUT: updated Task
- Checks if all Task-requirements (outputs as defined in PlanDefinition) have been met
  - If not, Task will not be finished, error escalated to user
  - If yes, task will be closed, next Task(s) will be selected from referenced PlanDefinition
- Starts automated Tasks if there are any

# $finish-task Alternative

- Task resource currently has 11 defined Operations ()
  - Including Setting In & Output
  - Also `$finish`
  - Why not use it?
    - Not sure if implementation can be extended to check validity & next tasks selected from the PlanDefinition without breaking off from FHIR STU3

# How to finish the process?

- Also handled by `$finish-task` if no further Tasks are left in the PlanDefinition
- A user can only cancel a process using `$process-cancel`
- A user can restart a process from it‘s cancellation point using `$process-resume`

# $process-cancel

- IN: Composition
- OUT: updated Composition
- Marks Composition either as obsolete or cancelled
- Also cancels all Tasks and Requests
  - If no Task was started -> entered in error
  - If any Task was started -> process cancelled
  - Request only cancelled if not started by fullfiller
  - If fullfiller started Request -> escalated to user

# $process-resume

- IN: Composition
- OUT: updated Composition
- Marks Composition as resumed
- Finished Tasks will not be changed
- Cancelled Tasks will be restarted as new Tasks/Requests
  - Old Tasks still marked as cancelled, so history doesn‘t have to be searched

# CONCLUSION & OUTLOOK

# When to use this workflow systems?

- To execute business processes with more than 1 organization involved
  - Especially those w.o. close technical collaboration
- NOT to execute workflows for your users
  - Forcing BPMN on a user generally doesn't work
- Not all tasks need to be specific
  - Only those that other organizations need to handle

# Outlook

- Next steps:
  - Continue implementation concept; First results will be ready in January WGM San Antonio
  - Publish required Profiles & Extensions
  - Publish required OperationDefinitions
- Long Term:
  - Showcase Tumor Board
  - Auditing & Security Concerns

# How you can be involved


- Anyone is welcome to collaborate
- We will be at the next 3 WGM & FHIRDevDays 2017
  - Anyone interested in a Workflow Track?
- Join the workflow topic in Zulip
- Join the workflow calls
- Contact: oliver.krauss@fh-hagenberg.at, andreas.schuler@fh-hagenberg.at
- You can also find us on Zulip

# THE WORKFLOW OF WORKFLOWS OF WORKFLOWS IN FHIR

# JUST KIDDING

# AUTOMATION DETAILS

# The Extension

- Extensions on actionDefinition:
  - Bool automated 0..1 If exists and true, this Task will be handled by a computer system
    - The following slides show what happens if automated is set to true
  - Endpoint operation 0..1 If exists and true, this operation will be called, using the *endpoints* Extension to provide input and direct output.

# In-N-Out Automation

- Invariant A) only one PUT/POST, 1..* GET
- Invariant B) only one GET, 1..* PUT/POST
- M:N doesn't work because which GET goes to which PUT/POST?
- GET/PUT/POST Endpoints can be:
  - http://example/fhir-stu3/* -> Target any FHIR Server
  - /fhir-stu3/* -> Target local server that runs the process
  - /{process}/{actionDefinitionId} -> target data pool of activity
  - /{process}/ -> target data pool of process itself

# Operation Automation

- Invariant: operation is set
- When Task becomes active system calls the operation
- When operation is finished, Task is finished as well
- GET/PUT/POST Endpoints will be used as In/Output for the Operation
  - Endpoint names must match parameter names of operation!

# Request/Event Pattern Automation


- Invariant: both ActivityDefinition and Transformer are set in the ActionDefinition
- When Task becomes active system sets off the Request
- Handling response may be done by a user
  - Invariant: Operation is set as well
  - Operation with Boolean Out will be used as validator, ending Task on success or escalating failure to user
- GET/PUT/POST Endpoints can still be used
  - Information about the task is added to the request

# Questions?

- Also available at
  - oliver.krauss@fh-hagenberg.at,
  - andreas.schuler@fh-hagenberg.at or
  - in the Coffee-Breaks