FIMS: A view from the Trenches

Sean O'Halpin

EBU MDN Workshop 2016-06-08

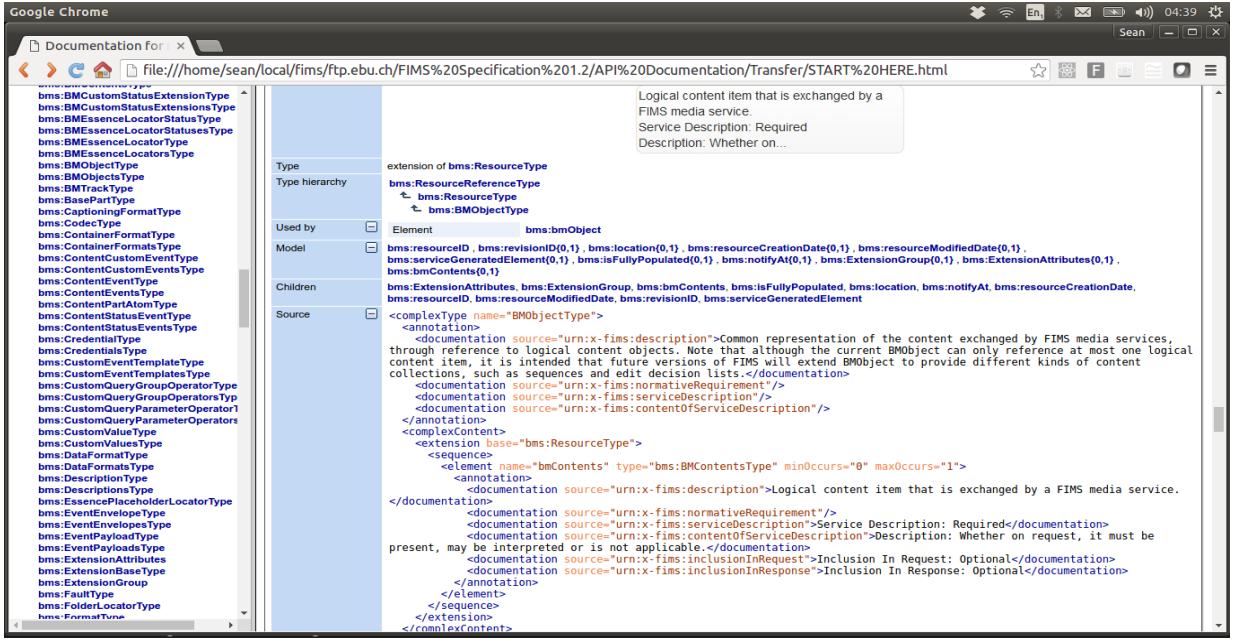
The Thesis

“FIMS is not being adopted because:
programmers are ignorant and lazy”

Ignorant

- I'm certainly ignorant
  - I didn't know about FIMS before I started here at the EBU
- My media background is in advertising, the internet and radio
  - No TV post-production or distribution
  - This is all new to me

Lazy



This looks like too much work to figure out

What this talk is about

- what we're doing in this area
- mapping some of our core concepts onto FIMS core concepts
- pain points & suggestions for making FIMS more accessible to developers

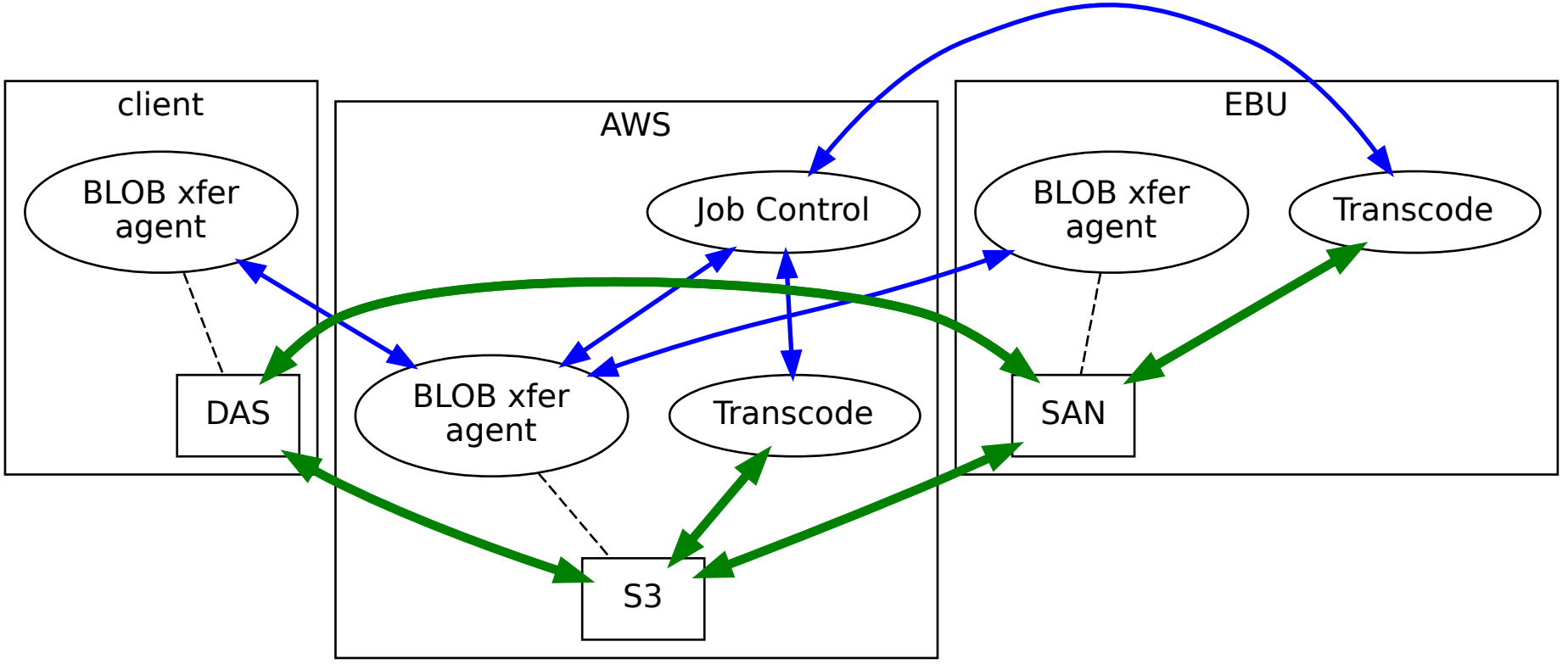
Who am I?

- Sean O'Halpin
- Senior Engineer in BBC R&D
- currently on secondment to the EBU
- working on investigating applications of IMF

What are we making?

- We are exploring applications of IMF
- We need a system to handle
  - File transfer (to get content into our system)
  - Transcoding (from broadcast formats to IMF)
  - Transforms of IMF packages (e.g. adding 'subs and dubs')

A simple transcoding system



Differences from the FIMS domain

- This is a prototype of limited scope
- All package manipulations happen within the system
  - So no need for external APIs
- File transfer is only for getting content into and out of the system
  - It is not a user-level service as such

Basic use cases

- Import package
- Convert to IMF
- Import related assets
- Apply transformations to create a new package, e.g.
  - add localized audio and subtitles
  - skip scenes which are unacceptable to a local market
- Export package

Specific functions

- BLOB transfer
  - Accelerated multi-part upload/download to/from S3
  - And to a SAN here at the EBU
- Transcoding
  - Using Windows-based transcoding software
  - Would like to use cloud-based transcoding
    - But transcoding to IMF not yet available

Implementation

- Elixir - highly concurrent language
  - Erlang Online Telecom Platform (OTP)
- control / data bus separation
- command / event
- resource pools
- message queues

Our model

- **Essence**: content blobs (AV, subtitles, images, etc.) acted upon by the system
- **Asset**: metadata representing the business value of content
- **Package**: structured bundle of Assets treated as a unit
- Services
  - **Analyse**: determine what kind of blob we're dealing with
  - **Validate**: validate blob against schema
  - **Transfer**: move blobs around
  - **Transcode**: convert Essence from one format to another
  - **Transform**: generate new assets and essences from existing ones
- **Job**: track work done on Assets by Services

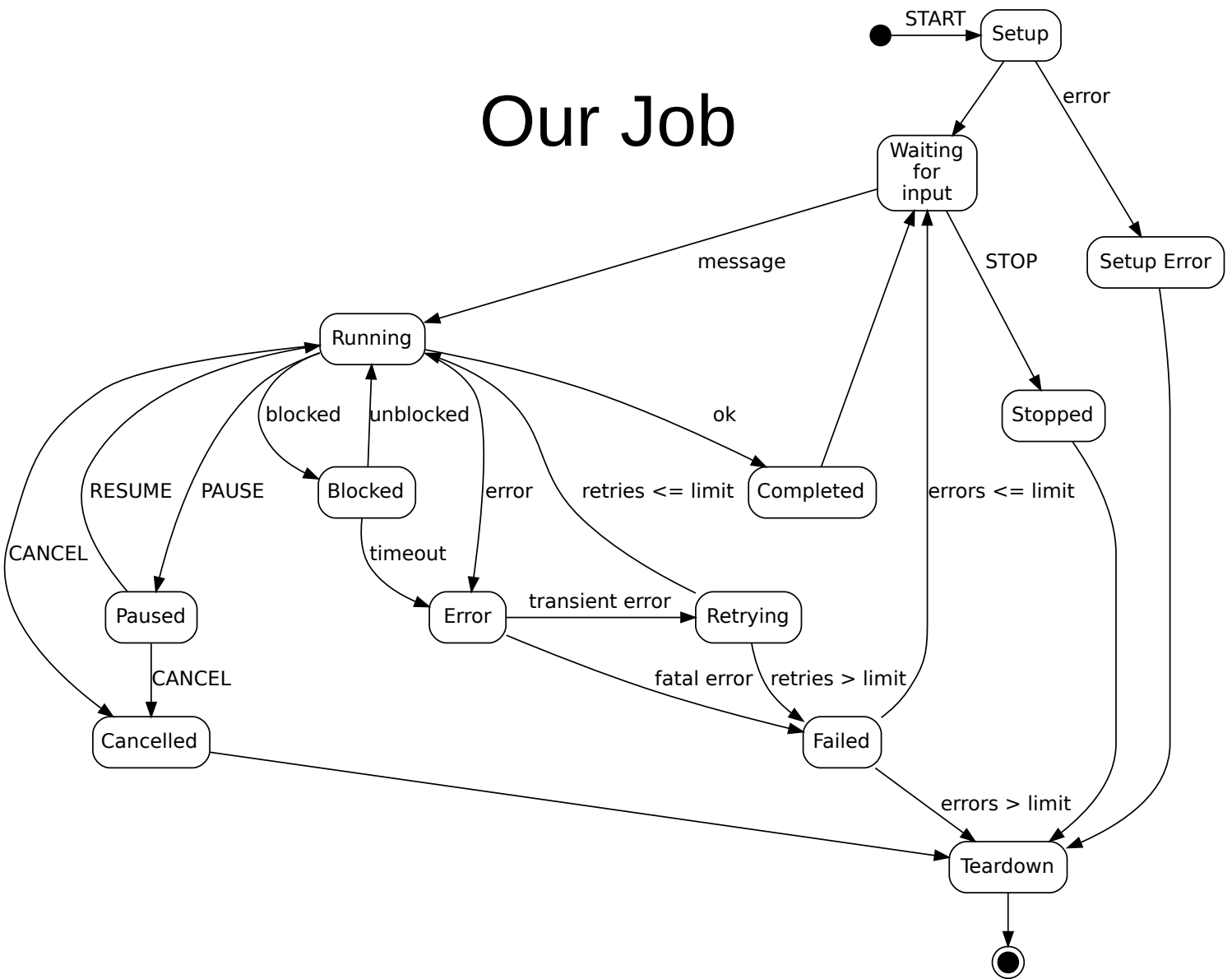
Overlap with FIMS 1.2

- ~~capture~~
- transfer
- transform

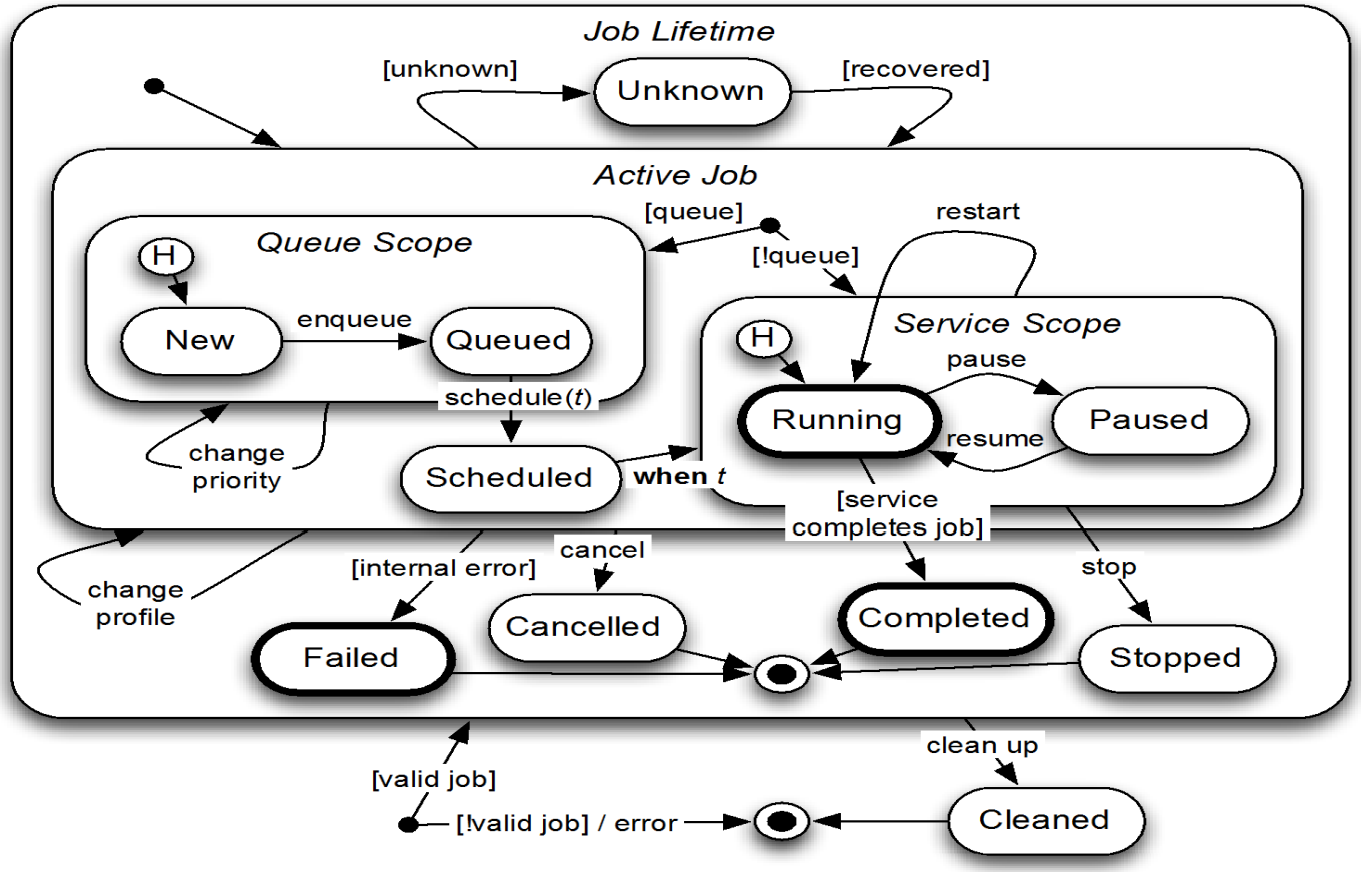
Our model compared to FIMS

Our model	FIMS
Job	Job
Application	Service
Asset	BMObject

Our Job


START
Setup
error
Waiting for input
message
STOP
Setup Error
Running
Stopped
blocked
unblocked
ok
RESUME
PAUSE
Blocked
error
retries <= limit
Completed
errors <= limit
CANCEL
timeout
Paused
Error
transient error
Retrying
CANCEL
fatal error
retries > limit
Cancelled
Failed
errors > limit
Teardown

FIMS Job


Job Lifetime
[unknown]
Unknown
[recovered]
Active Job
[queue]
[!queue]
restart
Queue Scope
H
New
enqueue
Queued
schedule(t)
change
priority
Scheduled
when t
Service Scope
H
pause
Running
resume
Paused
[service
completes job]
change
profile
[internal error]
cancel
Failed
Cancelled
Completed
stop
Stopped
clean up
[valid job]
[!valid job] / error
Cleaned

Applications vs Services

- It appears we are using different definitions of 'service'
- In our system, the artefacts that get built and deployed are called 'applications'
- One or more applications together provide the 'services' offered to clients.
- This is slightly unfortunate as it clashes with an important IT industry definition of Service

ITIL Service Lifecycle

The Service Lifecycle is the central concept of the 5 volumes that define ITIL 3:

- Service Strategy
- Service Design
- Service Transition
- Service Operation
- Continual Service Improvement

http://itil.org/en/vomkennen/itil/ueberblick/index.php

ITIL definition of service

ITIL 3 defines 'service' as:

- “a means of delivering value to customers by facilitating outcomes customers want to achieve without the ownership of specific costs and risks”

http://wiki.en.it-processmaps.com/index.php/ITIL_Glossary/_ITIL_Terms_S#Service

FIMS Service = ITIL Application

The FIMS Service Lifecycle is similar to what ITIL calls the Application Management lifecycle

The shared aspects are highlighted in bold:

- Requirements
- Design
- **Build**
- **Deploy**
- **Operate**
- Optimize

Asset vs Business Media Object

- We think we're following common usage in the industry:

  As asset is “[a]ny file which contains essence or metadata that is part of a composition. Examples include track files and composition playlist files.”[1]

- Everyone we speak to calls these blobs 'assets'
- BMObject is a confusing term simply because it is so vague
  - Three generic words in a row

[1] http://www.cinecert.com/support/glossary/#a

The ugly

- The General Description document creates the unfortunate impression that FIMS is unfinished and incomplete
  - even though the parts actually specified seem pretty complete to me
  - too much talk about what FIMS will be in the future and not enough clarity on what FIMS is now
- ESB, SOA, SOAP
  - Much XML. Very Java. Most programmers don't wash, let alone use SOAP!
- REST API seems like an afterthought
  - and is presented as such
- BMObject (Business Media Object)
  - The term is too vague – it needs more explanation in the docs
  - Even the FIMS schema can't be bothered to write it out in full :)
- Service Lifecycle
  - an idiosyncratic definition - not the ITIL Service Lifecycle or anyone else's

The bad

- It's not obvious where to find the documentation
- No clear overview of how you put together the elements defined in the schemas
- The General Description document assumes you already know FIMS
  - e.g. it uses terms before defining them
- Very few examples (and only SOAP)
- The diagrams are confusing and not explained
- No structure and a lot of noise in the schema documentation
  - what you expect from automatically generated docs
  - OK as reference but unusable to learn from

The good

- It looks like we have similar ideas to FIMS about how to model this domain
  - In particular, it's clear that separation of the control and data channels is key
- For the three core FIMs concepts I examined (Jobs, Services and BMObjects) we have close analogues in our system (Jobs, Applications and Assets), which bodes well for making those parts FIMS compliant
  - There are differences in detail and approach, but nothing fundamental
- Studying FIMs has made me consider aspects of our system we hadn't fully thought about
  - e.g. abstracting the Essence locator (bmEssenceLocator) is a good idea
- FIMS appears to me a very useful resource
  - I just wish it were easier to learn

Conclusions

- Our 'naïve' analysis has come up with similar design elements to FIMS
  - Though there is much more to FIMS than our system will cover
- This makes me more confident that we understand the domain reasonably well (as FIMS is clearly the result of a lot of thought)
- The main barrier to entry for developers is the lack of hand-holding resources for learning FIMS
  - It seems to me there's nothing fundamentally wrong with FIMS per se
    - I found some of the terms chosen confusing at first but that is a minor quibble
  - But from the lazy and ignorant programmer's point of view, it is demanding to learn

Suggestions for a more developer-friendly FIMS

In order of effort

- link to specification docs on fims.tv landing page
- proper web page for the docs
- have all documentation available in HTML on the web
- introduction to core concepts early in the docs
- a clear description of what FIMS is now without all the confusing future plans
- put the REST API up front and centre
  - the SOAP fans are well-served by FIMS already
  - FIMS needs to win the REST crowd

Suggestions (cont.)

- examples of actual use
  - document samples don't tell you how they are used
- tutorials covering each of the major functional areas - Capture, Transfer and Transform
- better diagrams and more sequence diagrams showing the protocols at work
- a reference implementation
- a properly written specification - autodocs don't work on their own

Future possibilities?

- Will FIMS be extended to cover cloud-provided services?
  - i.e. compute, storage, transcoding, etc.
  - Creating vendor neutrality there would be difficult
    - As you are fighting the vendor's attempts to lock you in
  - But this may well be where FIMS could add most value
    - And we will have a platform well-suited to explore that