Real Digital Threads

Digital Threads must be Real Artifacts,
not be only Marketing concepts

I’m Going to Tell you About

- This Agenda
- Who I am
- Business Pains
- What to demand in a Digital Thread Platform
- Some Terms
  - Implications
- Some Graph Concepts
- Some Digital Thread Examples
- Leaving time for dialog…

To help you understand that an *effective* digital thread service offers both the curation of the digital thread as well as the capability to explore disorganized data to discover what should be, for business or mission purposes, part of a digital thread.

Who am I to tell you about Digital Threads?

SPEAKER CREDENTIALS

Lonnie VanZandt, Principal Solutions Architect, Intercax

- University of Illinois Computer & Electrical Engineer ”CompSci” MS and BS
- Bell Laboratories Unix, Telecomm, and Data Comm MTS
- Northrop, CACI Systems Engineer and Business Analyst
- C, C++, C#, Java, Scala, Python, Ruby, Javascript Developer
- Former NoMagic Chief Architect
- UML, SysML, UPDM, UAF, BPMN Standards and Modeler
- Systems Integrator
- Data Wrangler
- Solutions Architect
- Hobbies:
  - Economics, Philosophy, Fatherhood, Personal Lumberjack

- Weaver of Digital Fabric for over 30 years

Why industry needs Real Digital Threads, managed in a Platform

BUSINESS PAIN

Worker and Data in Motion Waste

Knowledge Engineers waste an average of **4 hours each day** on menial data entry.
(https://simplyflows.com/time-wasted-on-repetitive-tasks-is-40-percent)

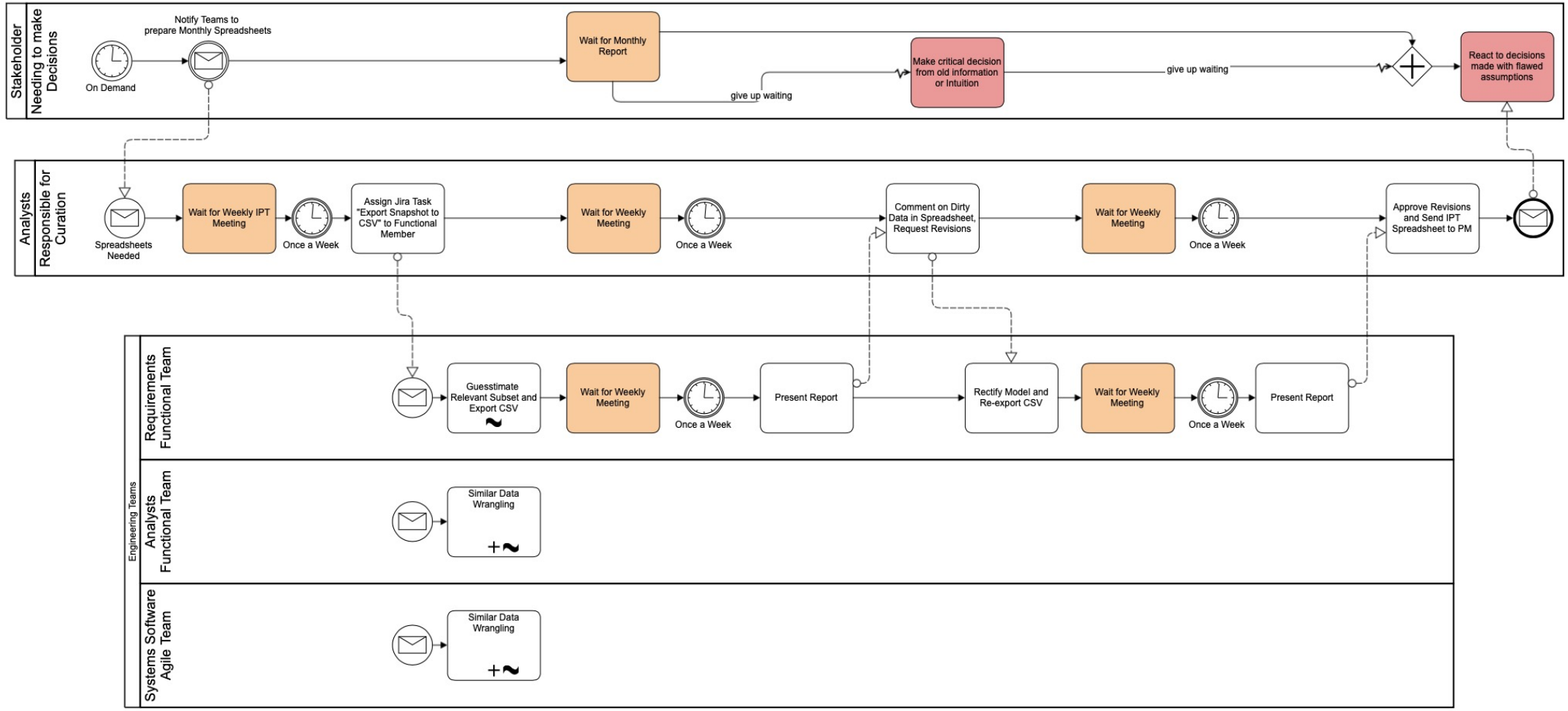
Paper-based processes are **75% Waste** compared to Digital Engineering (*Hedberg et al., 2016*)

- Organizations have isolated “silos of excellence”.
- Team members cannot find and share the information.
- Data collection for review, analysis and documentation is slow.
- Capability Gaps and Design Errors are detected too late.
- Projects are behind schedule and over budget.
- Team members unaware of changes
  - even that affect their own work.

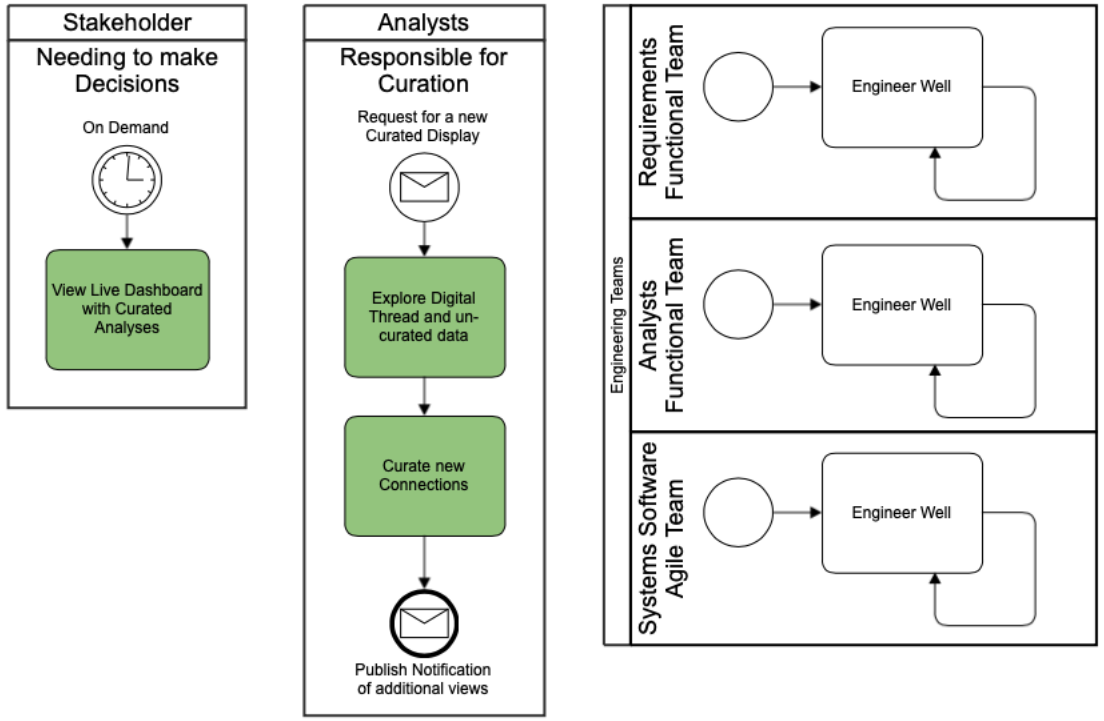
Managing digital data streams through models, seamless transmission of digital information, advances in analyzing data and trends, and efficiently communicating information to Decision-makers would **SAVE** manufacturers **$37 Billion** annually (*Anderson, 2016*).

Traditional *inefficient* Workflow

Global Product Data Interoperability Summit | **2022**



Digital Thread Workflow



Economic efficiency comes from concurrency and the reduction of wasteful staff and data movements, and from the making of decisions based on accurate (fresh) information.

Good Sources for Economic Analyses

- Return on Investment
  - Cost of Capital (e.g. 5.89%, for US Aerospace 2022)
    - https://pages.stern.nyu.edu/~adamodar/New_Home_Page/datafile/wacc.html
- Labor Statistics
  - Salaries (e.g. $115,167 for GS-14, Step 7, 2022)
    - https://www.federalpay.org/gs/2022
- Business Process Reengineering
  - DoD-specific: TFMMS and BAT
    - Total Force Manpower Management System
    - Billet Analysis Tool
  - BEA Business Enterprise Architecture (DoDAF model)
  - Provide effort-estimated business process models for numerous IT workflows

Relationships Between the Internal Rate of Return, Net Present Value and the Cost of Capital

If	Then	Capital Budgeting Decision
NPV < 0	IRR < Cost of Capital	Reject the investment from the cash flow perspective. Other factors could be important.
NPV = 0	IRR = Cost of Capital	Provides the minimum return. Probably reject from the cash flow perspective. Others factors could be important.
NPV > 0	IRR > Cost of Capital	Screen in for further analysis. Other investments may provide better returns and capital should be rationed, i.e., go to the most profitable projects. Others factors could be important.

https://maaw.info/IRRNPVandCostofCapital.htm

That you should Demand of your Digital Thread Platform

CAPABILITIES

Digital Thread Platform Capabilities

- Offer user-friendly SSO while honoring information assurance for API-to-API calls
- Provide provenance of who did what when
- Create real, tangible Digital Thread artifacts
- Store real Digital Thread artifacts
- Alter/Delete real Digital Thread artifacts
- Explore dis-organized engineering data across the organization
  - Discover what might *need* to be curated before it has been curated
- Offer a Query Language that can express Business-relevant concerns
  - Execute queries in a business-responsive timeframe
- Visualize query results

- But wait, there's more…

Digital Thread Platform Capabilities

- Offer services through a modern REST API that returns JSON results
  - to maximize integration into Industry's best ESBs, BI Dashboards, CI/CD engines, and Workflow Automation technologies

"Technologies like REST and SOAP APIs facilitate the ability to federate access to important information across systems. Because digital threads cover the full end-to-end lifecycle of an asset, regardless of the many platforms that host information about the asset, using technologies like REST and SOAP can tap into this 'dark data' to give additional context for the asset through its extended lifecycle."
– G. Coleman, VP, ENOVIA Advocacy, Dassault

- Be a loosely-coupled federation of microservices

"Microservices can improve the flexibility and scalability of cross-platform interactions and data exchange. The autonomous and streamlined nature of microservices allows developers from different application areas to work independently, increasing productivity and creating more easily accessible data for the digital thread."
– C. Longwell, Product Marketing Manager, Aras

Business Queries answerable by real Digital Threads

- How many Requirements Repositories do we have?
- What is the URL for each of our Jira Repositories?
- Which of our Jama Repositories are offline?
- Which Requirements Project has the most disorganized (non-curated) Requirements?
- Which Jama Project has the most curated Requirements?
- What percentage of a Requirements Project is curated?
- What are the 5 most recently modified Requirements in all Requirements Repositories?
- Who modified the 5 most recently modified Requirements?
- Which disorganized Requirements mention the phrase “Acme Widget”?
- Which curated Requirements mention the phrase “Stanley Sprocket”?

- Among the curated Parts, which curated Parts for a given Project weigh the most?
- Among the curated Parts, which curated Parts for a given Project have the most volume?

- Among the curated Parts, which curated Parts for a given Project cost the most?
- Which curated Parts of a Project have a mass that exceeds the limits of its related Requirements?
- If a given Requirement is altered, which Parts have to be reverified and what were the verification plans for those Parts?

Questions about the existence of and relationships between cross-disciplinary content and provenance.

Digital Artifact, Digital Thread, Digital Thread Platform

DEFINITION OF TERMS

Digital Engineering Artifact

- A digital engineering artifact is
- a fact, claim, assertion, or axiom
- that is held in a definitive source of truth
- in digital (encoded, transmissible) format.

«requirement»
REQ-123 Payload Volume

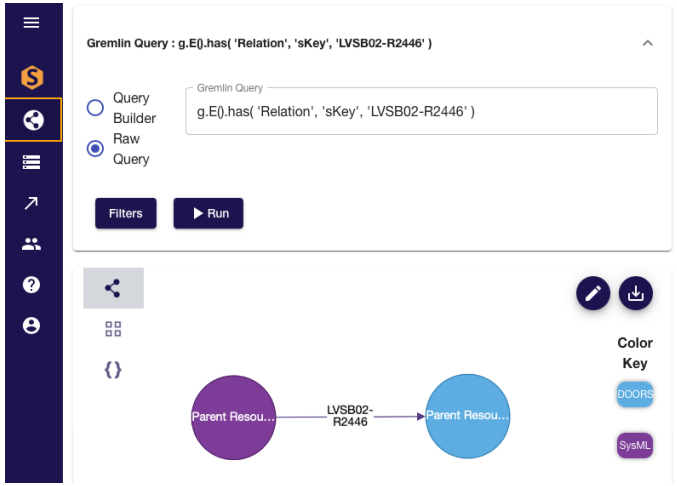
Id = "REQ-123"
Text = "Any suitable carrier MUST have a storage hold whose Volume meets or exceeds 500 cubic meters."

- Uniform Resource Identifier – uniquely identifies the Artifact
- Uniform Resource Locator – provides retrieval access to the Artifact

```
http://localhost.com/repositories/sysml
?project=PROJECT-66b78f33-a6c5-4004-9dfb-59397ccde514
&element=_19_0_4_bee02fc_1660761250615_803589_16231
```

Digital Thread

- A digital thread is
- a tangible, analytical, accessible graph
- of relationships
- between digital engineering artifacts.




```
curl 'http://syndeia35.yoursite.com:9000/relations/LVSB02-R2446' \
-H 'accept: application/json' \
-H 'X-Auth-Token: HASHEDdeadbeefMhNk'
```

Digital Thread Platform

- A digital thread platform is
- a service that provides curation for
- the creation, reporting, update, and deletion of these digital threads
- and which provides visualization and analytics
- over the digital threads and to the connected digital artifacts.

(Falcon 9 with its 96% Successful Launch Rate)

- An *effective* service offers both curation of the digital thread
- as well as
- the capability to explore disorganized data
- to discover what should be, for business or mission purposes, part of a digital thread.

A platform is not the final destination – it enables access to the goal.

Logically Formal, Analytic, Visual, Queriable

IMPLICATIONS

Digital Thread

$$\mathcal{G} \equiv (V, E)$$

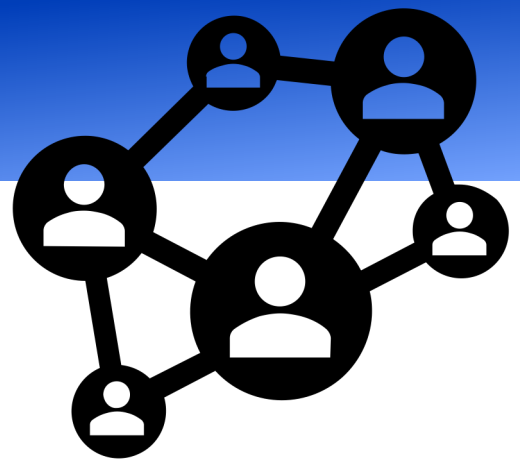
- A digital thread is a
- logically formal graph
- composed of references to the numerous engineering artifacts
- and the structural, semantic, and provenance relationships
- between these artifacts.

- A *business-useful* digital thread is one that is open, dynamic, scalable, secure, and analytical.

  - Analytic: Reasoning or acting from a perception of the *parts and interrelations* of a subject.

A digital thread from Syndeia:

```
g.E().has( 'Relation', 'container', 'MY-THREAD' )
```

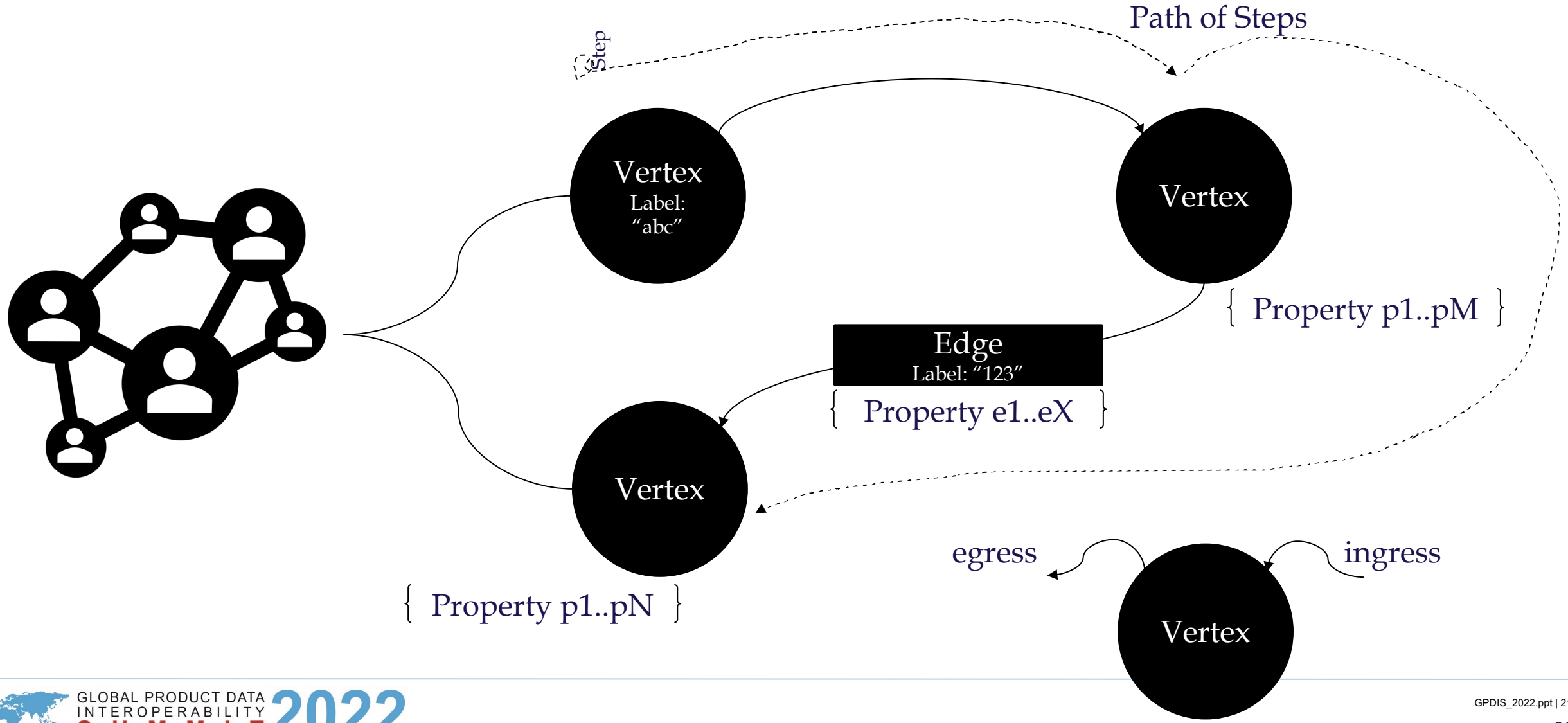


For Digital Threads and Graph Queries

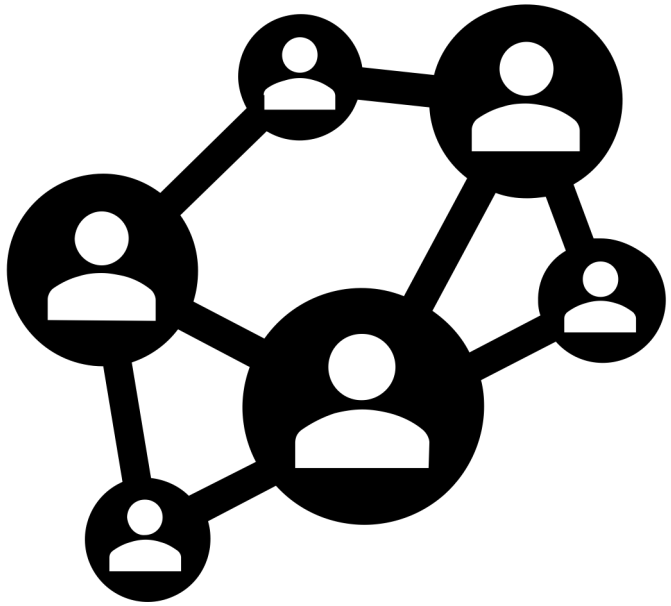
GRAPH CONCEPTS

Copyright 2008-2022 Intercax. All rights reserved.

Graph Concepts



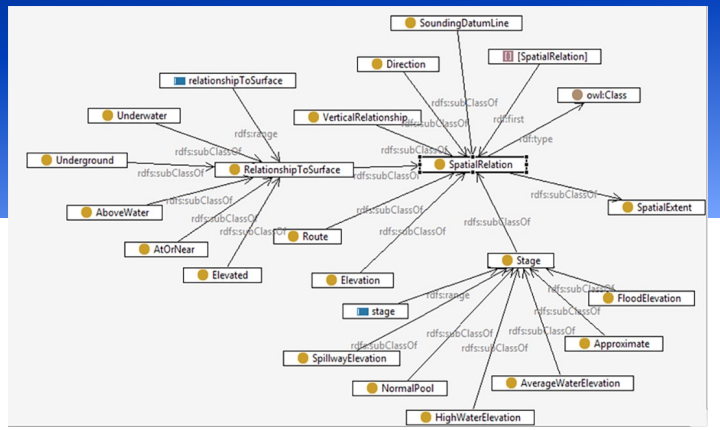
Graph Variations



Syndeia's graph is a
Directed, Typed Graph
where both Vertices and Edges
have Labels and Properties

- Graph of unadorned Vertices and Edges
- Property Graph
  - Vertices have Properties
  - Edges may have Properties
- Directed Graph
  - Edges have Direction
- Typed Graph
  - Vertices have labels that assert the Type of a Vertex
  - Edges may have Types
- Partitioned Graphs
  - Graphs have namespaces, separation
- Knowledge Graphs
  - A Description Logic holds and unasserted claims may be inferred or implied from asserted claims

A Remark on Ontologies



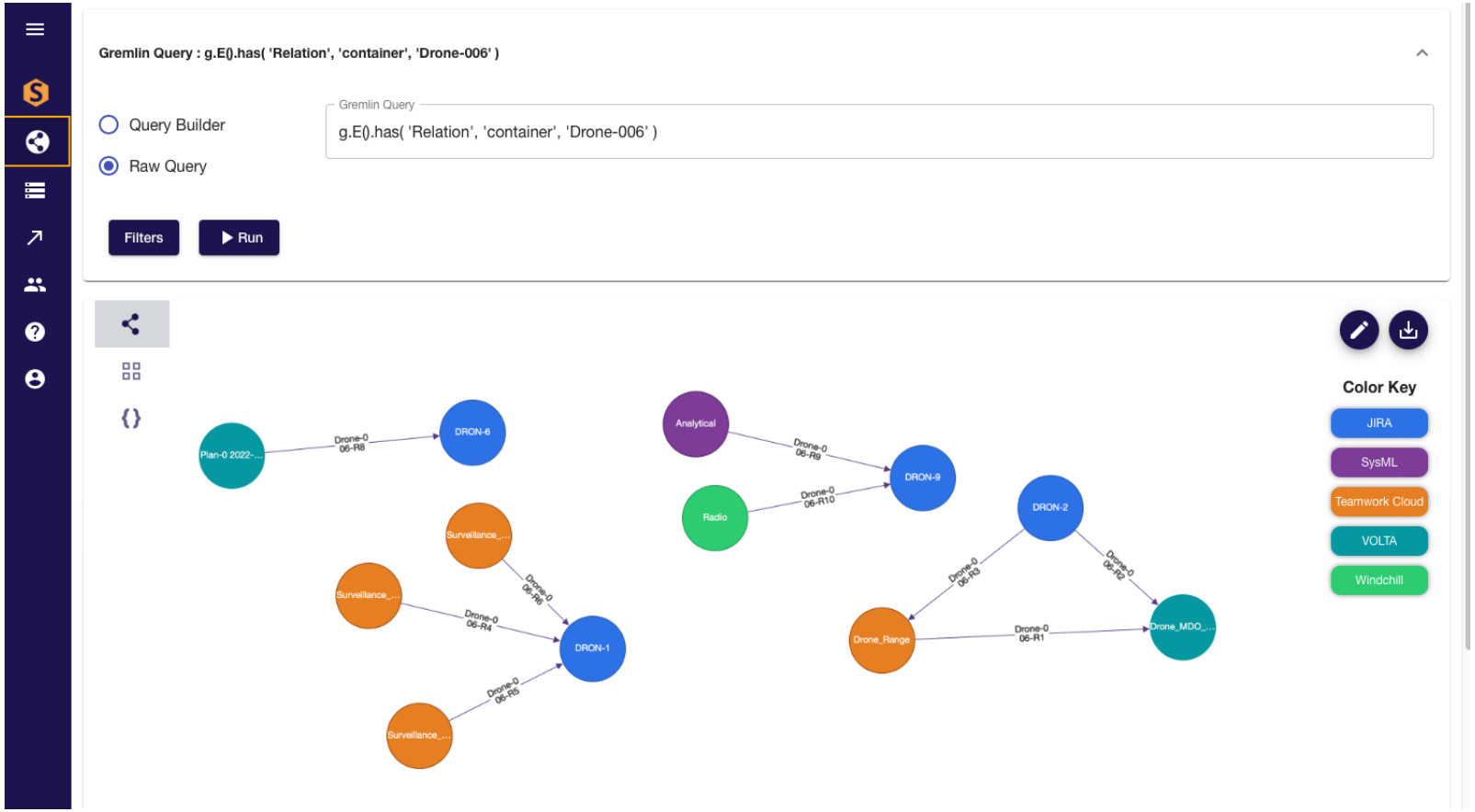
- Ontologies (*or, rather, OWL RDF triple stores*)
  - Can be curated by a few specialists
  - Are good for specifying Concepts
    - When relatively limited in scope and size
  - And greatly enable Reasoning from given Facts to implied Consequences

- Property Graphs
  - Can be populated by many generalists
  - Are good for recording Instances
    - Even when relatively wide in scope and size
  - And greatly enable storing, navigating, and querying millions of related real things.

- A difference of Theoretical vs Applied, perhaps

Digital Thread Platform Example: Visualize Query Results

WEB UX

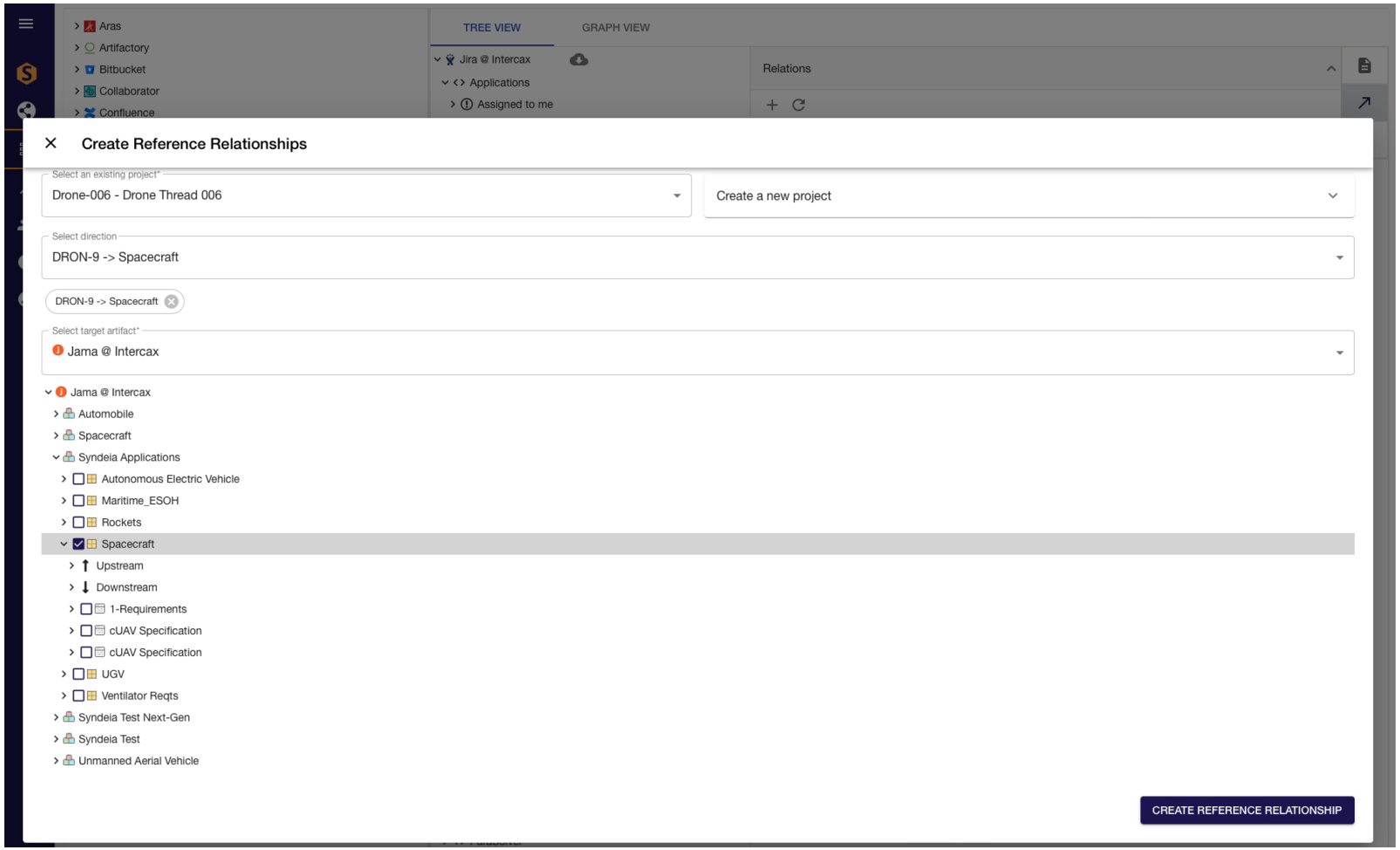


The Digital Thread Platform's web User Experience offers convenient user forms to enter and view queries by a user on demand.

Nevertheless, the Platform's feature-rich modern RESTful JSON-format API also enables programmatic automation of data-entry drag-n-drop human-in-the-loop workflow.

Digital Thread Platform Example: Create a new Digital Thread

WEB UX



The Digital Thread Platform's web User Experience offers convenient user forms to create relations by a user on demand.

Nevertheless, the Platform's feature-rich modern RESTful JSON-format API enables programmatic automation of data-entry drag-n-drop human-in-the-loop workflow.

Digital Thread Platform Example: Create a new Digital Thread

REST API

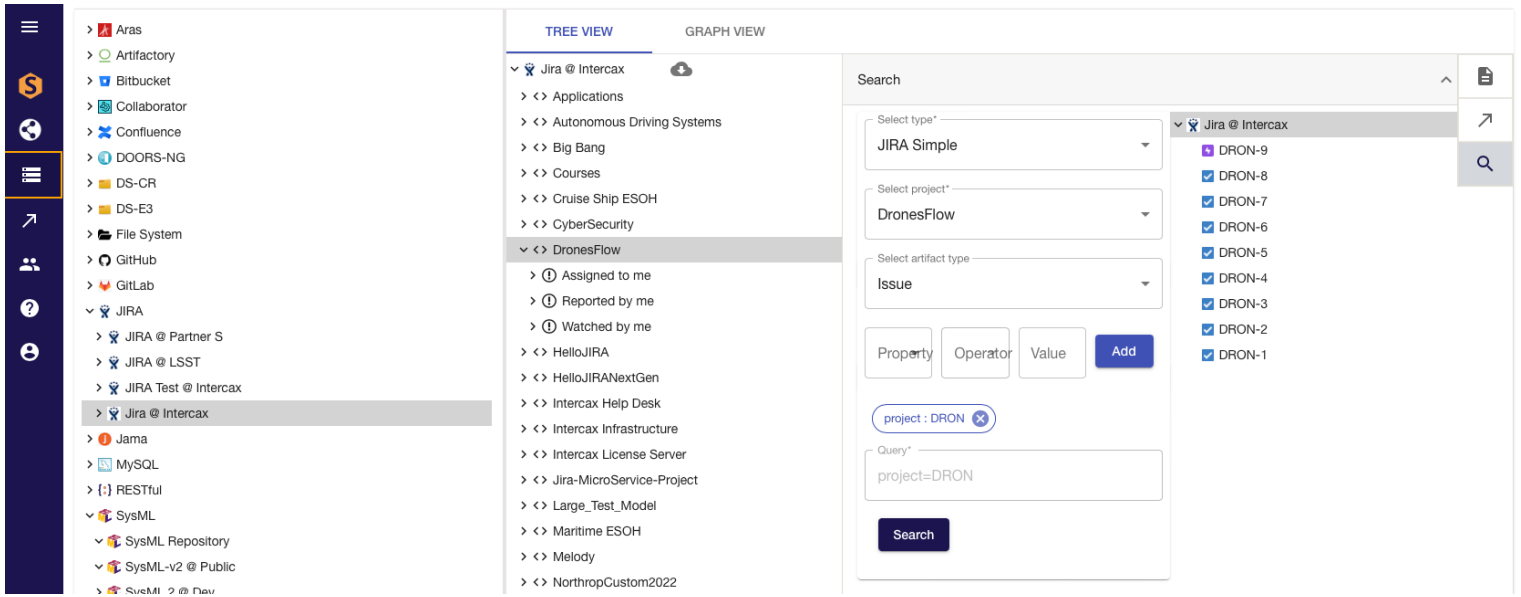

```
def findOrCreateReferenceConnectionFromSessionToTask( api_client: sc.ApiClient, syndeiaProject:
sc.Container, voltaRepo: sc.Repository, sessionArtifact: sc.Artifact,
jiraRepo: sc.Repository, taskArtifact: sc.Artifact ) -> Union[ None, sc.Relation ]:
relationName = f"({sessionArtifact.name})-:REFERENCES:->({taskArtifact.name})"
emptyInternalReference = sc.TypeReferenceCreate( internal = sc.InternalReferenceForm( key = "" ) )
relationDetails = sc.RelationCreate(
name= relationName,
description= relationName,
type = emptyInternalReference,
container= sc.ContainerReferenceCreate( internal = syndeiaProject.internal ),
source= sessionArtifact,
target= taskArtifact,
source_artifact= emptyInternalReference,
target_artifact= emptyInternalReference,
source_service_provider= { "name": "VOLTA", "repoKey": voltaRepo.internal.key },
target_service_provider= sc.ServiceProvider( name="JIRA", repo_key= jiraRepo.internal.key )
)
voltaUsername = secrets[ 'volta' ][ 'userid' ]
jiraUsername = secrets[ 'jira' ][ 'userid' ]
jiraPassword = secrets[ 'jira' ][ 'token' ]
relationApi = sc.RelationApi( api_client )
try:
authToken = voltaSignIn( api_client, voltaRepo.internal.key )

if ( authToken is None ):
logger.error( "Unexpected None token from Volta sign_in" )
raise ApiException

creationResponse: sc.RelationCreatedMessage = relationApi.create_reference_relation(
src_user_id = voltaUsername, src_ext_auth_token = authToken,
tgt_user_id = jiraUsername, tgt_ext_auth_token = jiraPassword,
relation_create = relationDetails )

return creationResponse.resources
…
```

The Digital Thread Platform's feature-rich modern RESTful JSON-format API enables programmatic automation of data-entry drag-n-drop human-in-the-loop workflow.

Nevertheless, the Platform's web User Experience also offers convenient user forms to create relations by a user on demand.

Digital Thread Platform Example: Explore ASoTs for content to Curate

WEB UX



The Digital Thread Platform's web User Experience offers users convenient user forms to find raw content on demand.

Nevertheless, the Platform's feature-rich modern RESTful JSON-format API also enables programmatic automation of clerical human-in-the-loop workflow.

Digital Thread Platform Example: Explore ASoTs for content to Curate

REST API


```
# get the Jira Issues for a particular Jira Project
def getJiraIssuesForProject( api_client: sc.ApiClient,
jiraProjectName: str, query: str ) -> Union[ None, list[sc.ArtifactObjMessage] ]:
jira_api = sc.JIRAApi( api_client )
jiraRepoName = secrets['jira']['esteco']['repo']
jiraUsername = secrets['jira']['userid']
jiraPassword = secrets['jira']['token']

jiraRepo = findJiraRepositoryByName( api_client, jiraRepoName )

if ( jiraRepo is None ):
logger.error( f"Unable to find Jira Repository {jiraRepoName}" )
return None

strippedQuery = query.strip() if len(query) > 0 else ""

filter = f" and ({strippedQuery})" if len(strippedQuery) > 0 else ""


jqlQuery = sc.models.ArtifactSearch(
query = f"project = {jiraProjectName}{filter}" )

queryResponse = jira_api.search_jira_artifacts(
repo_key = jiraRepo.internal.key,
user_id = jiraUsername, ext_auth_token = jiraPassword,
artifact_search = jqlQuery )


issues = queryResponse.resources

logger.info( f"Found {len( issues )} issues for {jiraProjectName}" )
return issues
```

The Digital Thread Platform's feature-rich modern RESTful JSON-format API enables programmatic automation of clerical human-in-the-loop workflow.

Nevertheless, the Platform's web User Experience also offers users convenient user forms to find raw content on demand.

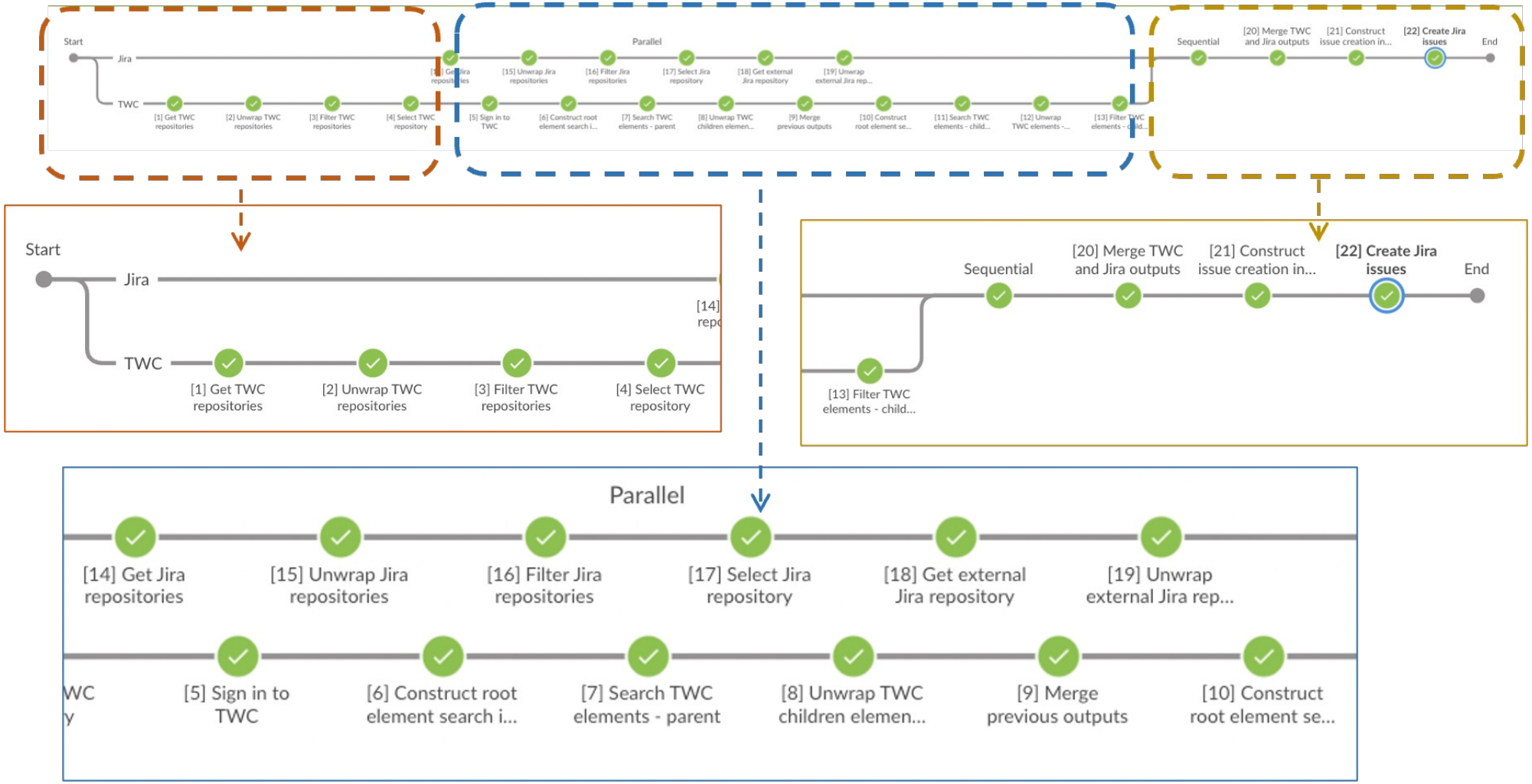
Digital Thread Platform Example: Who When

Gremlin Service


```
# Which Staff modified a Digital Thread When and how Often?
userName = { u,t -> userGet = new URL( "http://syndeia.company.com:9000/users/${u}" ).openConnection(); userGet.setRequestProperty( "Accept",
"application/json" ); userGet.setRequestProperty( "X-Auth-Token", t ); userGet.getResponseCode(); j = new groovy.json.JsonSlurper().parseText(
userGet.getInputStream().getText() ); j.resources.name }

signInToken = { -> signInPost = new URL( "http://syndeia.company.com:9000/signIn" ).openConnection(); message = '{ "username": "email@intercax.com",
"password": "pass-phrase", "rememberMe": false }'; signInPost.setDoOutput( true ); signInPost.setRequestProperty( "Content-Type", "application/json" );
signInPost.getOutputStream().write( message.getBytes( "UTF-8" ) ); signInPost.getResponseCode(); j = new groovy.json.JsonSlurper().parseText(
signInPost.getInputStream().getText() ); j.resources.token }

g.E().has( 'Relation', 'container', 'DZSB19' ) .group() .by( values( 'modifiedDate' ) .map{ ( new
java.text.SimpleDateFormat( "yyyy-MM-dd" ).parse( it.get() ) ).format( "YYYY MM/dd EEEE" ) } )
.by( groupCount().by( values( 'modifiedBy' ) ) ) .unfold() .order().by( keys ) .project( 'When',
'Who' ) .by( select( keys ) ) .by( select( values ).unfold() .map{ it -> "${userName(
it.get().getKey(), signInToken() )} (${it.get().getValue()} times)" }.fold() )

 ==>{When=2021 06/30 Wednesday, Who=[dirk zwemer (56 times), manas bajaj (2 times)]}
 ==>{When=2021 07/01 Thursday, Who=[lonnie vanzandt (2 times)]}
 ==>{When=2021 09/08 Wednesday, Who=[manas bajaj (5 times)]}
 ==>{When=2021 09/09 Thursday, Who=[lonnie vanzandt (1 times)]}
 ==>{When=2021 12/16 Thursday, Who=[lonnie vanzandt (16 times)]}
 ==>{When=2022 06/18 Saturday, Who=[lonnie vanzandt (2 times)]}
```

Digital Thread Platform Example: Pipeline Automation

Automation Service



Digital Thread Value Proposition

- An *effective* digital thread service offers
  - both the curation of the digital thread as well as
  - the capability to explore disorganized data
  - to discover what should be, for business or mission purposes, part of a digital thread

- With a real digital thread, one can
  - Discover information
  - Analyze information
  - Present information
  - and promptly Make Decisions based on facts

Syndeia Value Proposition

- Syndeia is the digital thread platform for model-based engineering.

- With Syndeia, engineering teams collaboratively and concurrently develop and manage real digital threads for complex systems
  - across federated models and data

- from definitive source modeling and simulation tools, enterprise applications, and data repositories.