

Evaluating FAIR Digital Object as a distributed object system

This manuscript ([permalink](#)) was automatically generated from [stain/2022-fdo-paper@6ff8ad5](#) on May 6, 2022.

Authors

- **Stian Soiland-Reyes**

 [0000-0001-9842-9718](#) ·  [stain](#) ·  [soilandreyes](#)

Department of Computer Science, The University of Manchester, UK; Informatics Institute, Faculty of Science, University of Amsterdam, NL · Funded by [BioExcel-2](#) (European Commission [H2020-INFRAEDI-2018-1 823830](#)); [BY-COVID](#) (European Commission [HORIZON-INFRA-2021-EMERGENCY-01 101046203](#))

- **Carole Goble**

 [0000-0003-1219-2137](#) ·  [carolegoble](#) ·  [CaroleAnneGoble](#)

Department of Computer Science, The University of Manchester, UK · Funded by [BioExcel-2](#) (European Commission [H2020-INFRAEDI-2018-1 823830](#)); [EOSC-Life](#) (European Commission [H2020-INFRAEOSC-2018-2 824087](#)); [BY-COVID](#) (European Commission [HORIZON-INFRA-2021-EMERGENCY-01 101046203](#))

- **Paul Groth**

 [0000-0003-0183-6910](#) ·  [pgroth](#) ·  [pgroth](#)

Informatics Institute, Faculty of Science, University of Amsterdam, NL

Abstract

Interoperability Framework for Fast Data

Quotes from [1]:

- **Symbiotic** (purpose and intent): Motivations to have the interaction, with varying levels of mutual knowledge of governance, strategy and goals.
- **Pragmatic** (reaction and effects): Management of the effects of the interaction at the levels of choreography, process and service e
- **Semantic** (meaning of content): *Inference Rule base*. Interpretation of a message in context, at the levels of rule, known application components and relations and definition of concepts
- **Syntactic** (notation of representation): _Representation of application components, in terms of composition, primitive components and their serialization format in messages
- **Connective** (transfer protocol): Lower-level formats and network protocols involved in transferring a message from the context of the sender to that of the receiver
- **Environmental** (deployment and migration) Environment in which each application is deployed and managed, including the portability problems raised by migrations

Symbiotic: Expresses the purpose and intent of two interacting applications to engage in a mutually beneficial agreement. Enterprise engineering is usually the topmost level in application interaction complexity, since it goes up to the human level, with governance and strategy heavily involved. Therefore, it maps mainly onto the symbiotic category, although the same principles apply (in a more rudimentary fashion) to simpler subsystems. This can entail a tight coordination under a common governance (if the applications are controlled by the same entity), a joint venture agreement (if the two applications are substantially aligned), a collaboration involving a partnership agreement (if some goals are shared) or a mere value chain cooperation (an outsourcing contract).

Pragmatic: The effect of an interaction between a consumer and a provider is the outcome of a contract, which is implemented by a choreography that coordinates processes, which in turn implement workflow behaviour by orchestrating service invocations. Languages such as Business Process Execution Language (BPEL) [31] support the implementation of processes and Web Services Choreography Description Language (WS-CDL) is an example of a language that allows choreographies to be specified.

Semantic: Both interacting applications must be able to understand the meaning of the content of the messages exchanged: both requests and responses. This implies interoperability in rules, knowledge and ontologies, so that meaning is not lost when transferring a message from the context of the sender to that of the receiver. Semantic languages and specifications such as Web Ontology Language (OWL) and Resource Description Framework (RDF), map onto this category.

Syntactic: This deals mainly with form, rather than content. Each message has a structure composed of data (primitive applications) according to some structural definition (its schema). Data need to be serialized to be sent over the network as messages using representations such as XML or JSON.

Connective: The main objective is to transfer a message from the context of one application to the other regardless of its content. This usually involves enclosing that content in another message with control information and implementing a message protocol (such as SOAP or HTTP) over a communications network according to its own protocol (such as TCP/IP) and possibly resorting to routing gateways if different networks are involved.

Environmental: Each application also interacts with the environment (e.g. a cloud or a server) in which it is deployed, anewed or by migration. The environment's management application programming interface (API) and the infrastructure level that the application requires will most likely have impact on the way applications interact, particularly if they are deployed in (or migrate between) different environments, from different vendors. Interoperability between an application and the environment in which it is deployed usually known as portability.

Metamodel: Resource, Service, Transaction, Process, Response, Operation, Request, Channel, Protocol, Link

A comparison framework for middleware infrastructures

Quotes from [2]:

- **Openness:** The middleware infrastructure should enable extending the applications built on top of it in various ways. (e.g., adding, removing, upgrading, composing services, etc.).
- **Scalability:** The middleware infrastructure should facilitate the effective operation of the applications at many different scales.
- **Performance:** The middleware infrastructure should enable the efficient and predictable, if needed, execution of the applications that are built on top of it.
- **Distribution transparency:** is the property that determines if the application is perceived by users, or developers as a whole rather than as a collection of independent constituent elements. The requirement for distribution transparency is quite generic and it is usually refined into a number of more specific transparencies including:
 - **Access transparency:** the infrastructure should enable accessing local and remote application elements in the same way.
 - **Location transparency:** the infrastructure should enable accessing the application elements without knowledge of their physical location.
 - **Concurrency transparency:** the infrastructure should allow concurrent processing on resources, without interference.
 - **Failure transparency:** the infrastructure should enable service provisioning despite the occurrence of failures.
 - **Migration transparency:** the infrastructure should provide means for changing the location of elements of the application without compromising the application's correct operation, i.e. without affecting the elements that depend on the migrated elements.
 - **Persistence transparency:** the infrastructure should provide means for concealing the deactivation and reactivation of elements from other elements that are using them.
 - **Transaction transparency:** the infrastructure should provide means for coordinating the execution of atomic and isolated transactions.

Modularity: The application should consist of a collection of elements, each one providing services, used by the others. Modularity enables the identification of dependencies between the elements that make up the system. Consequently, it allows determining, which elements are affected by the eventual addition, removal or upgrade of services.

Encapsulation: For each constituent element, there is a clear separation between the element's interface and implementation. The interface is a well-defined specification of the provided services, the contract between the element and the entities using it. The implementation is the realization of the provided services. In general, it is safe to change the implementation of an element as long as the element's interface is preserved. Changing an element's interface without compromising the overall application integrity requires that the rest of the application does not depend on this particular interface, at the time of the change.

Inheritance: An interface specification (resp. implementation) may be derived from another one. The derived interface (resp. implementation) provides at least the services of the base interface (resp. implementation). Inheritance enables the vertical and horizontal composition of services.

Signal interfaces: defining asynchronous stimuli that can be handled by instances of engineering objects, providing these interfaces.

Operation interfaces: defining operations that can be invoked on instances, providing these interfaces. Invoking an operation causes a request message to be sent by the invoker to the invoked instance. Invoking an operation may further result in a reply sent from the invoked instance to the invoking instance.

Stream interfaces: defining operations that can be invoked on instances, providing these interfaces. The result of invoking a stream operation is the continuous conveyance of information from the invoked instance to the invoking instance.

Comparing FDO and Web as middleware infrastructures

DOIP [\[3\]](#)

Table 1: Comparing FAIR Digital Object (with the DOIP 2.0 protocol [\[3\]](#)) and Web technologies (using Linked Data) as middleware infrastructures [\[2\]](#)

Quality	FDO w/ DOIP	Web w/ Linked Data
Openness		
Scalability		
Performance		
Distribution transparency		
Access transparency	FDOs always accessed through PID indirection, but this means difficult to make private test setup.	Global HTTP protocol frequently used locally and behind firewalls, but at risk of non-global URIs (e.g. <code>http://localhost/object/1</code>) and SSL issues (e.g. self-signed certificates, local CAs)
Location transparency	FDOs always accessed through PIDs. Multiple locations possible in Handle system, can expose geo-info.	PIDs and URL redirects. DNS aliases and IP routing can hide location. Geo-localized servers common for large cloud deployments.
Concurrency transparency	No explicit concurrency measures. FDO kernel metadata can include checksum and date.	Operations are classified as stateless and idempotent. Cache control, ETag (~checksum) and date in HTTP headers. Limited transaction capabilities (e.g. If-Not-Modified) on same resource.
Failure transparency	DOIP status codes, e.g. <code>0.DOIP/Status.104</code> , additional codes can be added as custom attributes	HTTP status codes e.g. 404 Not Found, structured error documents in Open API (??)
Migration transparency		HTTP 300 status can provide temporary or permanent redirections.
Persistence transparency		
Transaction transparency		
Modularity		

Quality	FDO w/ DOIP	Web w/ Linked Data
Encapsulation	FDO principles are protocol independent. Indirection by PID, unclear which protocol to use for which FDO (e.g. CODRA supports native DOIP , DOIP over HTTP and CODRA REST API)	HTTP 1 semantics seamlessly upgrades to HTTP 2. <code>http</code> vs <code>https</code> exposes encryption detail 1 . Need URI Design [5] to avoid application dependence, e.g. use of PURL services.
Inheritance	Type system currently undefined for FDO and DOIP, can piggyback of FDO type's schema (e.g. CORDRA <code>\$ref</code> use of JSON Schema references [6])	Media Type with multiple suffixes [7] , multiple profiles (RFC6906) [8] , Semantic type systems (RDFS [9] , OWL2 [10] , SKOS [11]), OpenAPI 3 [12] inheritance and Polymorphism
Signal interfaces	DOIP 2.0 is synchronous, in FDO async operations undefined. Could be handled as custom jobs/futures FDOs	HTTP/2 multiplexed streams [13] , Web Sockets [14] , Linked Data Notifications [15] , custom jobs/futures REST resources
Operation interfaces	CRUD predefined in DOIP, custom operations through <code>Op.ListOperations</code> (<i>Operation FDO</i> currently undefined)	CRUD predefined in HTTP methods [@{doi:10.17487/RFC7231}] , URI Templates [16] , OpenAPI operations [12] , HATEOAS incl. schema.org Actions , JSON HAL [17] & Link headers (RFC8288) [18]
Stream interfaces	Undefined in FDO, DOIP can support multiple byte stream elements with custom FDO type to determine their combination	HTTP 1.1 [19] chunked transfer , HLS (RFC8216) [20] , MPEG-DASH (ISO/IEC 23009-1:2019) [21]

Assessing DOIP against FDO

Assessing FDO against FAIR

References

1. **An Interoperability Framework and Distributed Platform for Fast Data Applications**
José Carlos Martins Delgado
Data Science and Big Data Computing (2016) <https://doi.org/gp3rds>
DOI: [10.1007/978-3-319-31861-5_1](https://doi.org/10.1007/978-3-319-31861-5_1)
2. **A Comparison Framework for Middleware Infrastructures.**
Apostolos Zarras
The Journal of Object Technology (2004) <https://doi.org/cj5q8r>
DOI: [10.5381/jot.2004.3.5.a2](https://doi.org/10.5381/jot.2004.3.5.a2)
3. **Digital object interface protocol specification, version 2.0**
DONA Foundation
DONA foundation (2018-11-12) <https://hdl.handle.net/0.DOIP/DOIPV2.0>
4. **Upgrading to TLS Within HTTP/1.1**
R Khare, S Lawrence
RFC Editor (2000-05) <https://doi.org/gp33dv>
DOI: [10.17487/rfc2817](https://doi.org/10.17487/rfc2817)
5. **Hypertext Style: Cool URIs don't change.** <https://www.w3.org/Provider/Style/URI.html>
6. **draft-bhutton-json-schema-00** <https://datatracker.ietf.org/doc/html/draft-bhutton-json-schema-00>
7. **draft-ietf-mediamaan-suffixes-00 - Media Types with Multiple Suffixes**
<https://datatracker.ietf.org/doc/draft-ietf-mediamaan-suffixes/00/>
8. **The 'profile' Link Relation Type**
E Wilde
RFC Editor (2013-03) <https://doi.org/gp32qZ>
DOI: [10.17487/rfc6906](https://doi.org/10.17487/rfc6906)
9. **RDF Schema 1.1** <http://www.w3.org/TR/rdf-schema/>
10. **OWL 2 Web Ontology Language Document Overview (Second Edition)**
<http://www.w3.org/TR/owl2-overview/>
11. **SKOS Simple Knowledge Organization System Reference** <http://www.w3.org/TR/skos-reference/>
12. **OpenAPI Specification v3.1.0 | Introduction, Definitions, & More**
<https://spec.openapis.org/oas/v3.1.0.html>
13. **Hypertext Transfer Protocol Version 2 (HTTP/2)**
M Belshe, R Peon
RFC Editor (2015-05) <https://doi.org/gp32q9>
DOI: [10.17487/rfc7540](https://doi.org/10.17487/rfc7540)
14. **WebSockets Standard** <https://websockets.spec.whatwg.org/>
15. **Linked Data Notifications** <https://www.w3.org/TR/ldn/>
16. **URI Template**

J Gregorio, R Fielding, M Hadley, M Nottingham, D Orchard
RFC Editor (2012-03) <https://doi.org/gp33dw>
DOI: [10.17487/rfc6570](https://doi.org/10.17487/rfc6570)

17. **draft-kelly-json-hal-08** <https://datatracker.ietf.org/doc/html/draft-kelly-json-hal-08>
 18. **Web Linking**
M Nottingham
RFC Editor (2017-10) <https://doi.org/gf8jcd>
DOI: [10.17487/rfc8288](https://doi.org/10.17487/rfc8288)
 19. **Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing**
R Fielding, J Reschke (editors)
RFC Editor (2014-06) <https://doi.org/gp32q8>
DOI: [10.17487/rfc7230](https://doi.org/10.17487/rfc7230)
 20. **HTTP Live Streaming**
W May
RFC Editor (2017-08) <https://doi.org/gp32rc>
DOI: [10.17487/rfc8216](https://doi.org/10.17487/rfc8216)
 21. **ISO/IEC 23009-1:2019**
14:00-17:00
ISO
<https://www.iso.org/cms/render/live/en/sites/isoorg/contents/data/standard/07/93/79329.html>
-

1. The `http` protocol (port 80) can in theory also upgrade [4] to TLS encryption, as used by [Internet Printing Protocol](#) for `ipp` URLs, but on the web, best practice is explicit `https` (port 443) URLs to ensure followed links stay secure.↵