

# Forward and backward compatibility design techniques applying the HL7 FHIR standard

Igor Bossenko<sup>1</sup>, Gunnar Piho<sup>2</sup> and Peeter Ross<sup>1</sup>

<sup>1</sup>Department of Health Technologies, TalTech, Akadeemia Str 15A, 12618 Tallinn, Estonia

<sup>2</sup>Department of Software Science, TalTech, Akadeemia Str 15A, 12618 Tallinn, Estonia

## Abstract

Interoperability issues are acute not only in heterogeneous environments where different standards are used, but also in homogeneous environments where the same standard is used differently for individual organisations. Interoperability is also important within one organisation and especially arises when transferring systems from one version of the standard to another. This article discusses the compatibility of the various versions of the HL7 FHIR standard, focusing on the current R4 version at the time of writing. It also describes the individual features of earlier versions of the HL7 FHIR standard. The article analyses the principles of forward and backward compatibility and proposes recommendations for how to develop a sustainable and balanced interoperability system.

## Keywords

HL7 FHIR, EHR, electronic health record, interoperability, compatibility design techniques

## 1. Overview of HL7 FHIR interoperability principles

HL7 FHIR [1] is an international healthcare information exchange standard that provides a range of predefined resources, the possibility to expand these resources and a framework for the exchange of these resources between interested parties. The HL7 FHIR community continuously develops the standard. The first draft of the HL7 FHIR standard was published in 2012 [2], and since then, almost every year, a new version of the standard or an intermediate version (ballot) has been released. During the development of the standard, both the HL7 FHIR framework as well as HL7 FHIR resources have been modified. For example, DSTU1 used an rss/atom feed-based architecture [3], but this was replaced by an internal Bundle in DSTU2.

HL7 uses a specific development process for the development of FHIR resources. Resources are divided into logical domains, where a specific working group, consisting of both clinical and IT professionals, is responsible for the development of a specific logical domain. The working group assesses the needs of the domain and creates information models that describe the data elements, both the mandatory and recommended terminology and the constraints applicable to the data model. The resource-related deployment guide describes the purpose and use of the resource and its relationships with other business resources and provides illustrative examples that simplify the use of resources.

---

HEDA-2022: The International Health Data Workshop, June 19-24, 2022, Bergen, Norway


✉ igor.bossenko@taltech.ee (I. Bossenko); gunnar.piho@taltech.ee (G. Piho); peeter.ross@taltech.ee (P. Ross)

🌐 taltech.ee/en/emed-lab (I. Bossenko)

🆔 0000-0003-1163-5522 (I. Bossenko); 0000-0003-4488-3389 (G. Piho); 0000-0003-1072-7249 (P. Ross)



© 2022 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

Resource development is evolutionary. It undergoes several specification reviews and community connectathons (community testing) and feedback from early implementers. As a result, each resource is assigned a level of maturity according to the FMM (FHIR Maturity Model) [4] scale, which in turn is based on the CMMI [5] scale. FMM maturity levels are numbered from 0 to 5. For example, maturity level 0 means a 'draft' that was recently published, and level 5 becomes a resource that has been published in at least two releases with a maturity level other than zero and has been implemented in at least five independent production systems in more than one country. The final level is 'normative' or stable. Maturity levels are directly related to the stability of resources: the higher the number the more stable the resource and the lower the chance of non-backward changes occurring.

Forward compatibility means that content compatible with the old version of the standard is also compatible with the new version. For normative resources, FHIR seeks to ensure, but does not guarantee, the compatibility of resources. Backward compatibility means that instances created against future versions of a specification work with older versions of the specification. The development of the resource information model is based on the '80/20' [6] principle of reuse and composability – focusing on 20% of the requirements that satisfy 80% of the interoperability needs. For this purpose, resources are created in such a way that they meet the general or common data requirements of many use cases in order to avoid the proliferation of numerous, overlapping and redundant resources. Use cases not covered by the base resource can be implemented using extensions (FHIR extensions [7]) and customisation (FHIR profiles [8]).

FHIR extensions are a fundamental part of FHIR architecture. Each element expanded according to the needs. Each extension can be repeated an unlimited number of times. With the extension mechanism, it is possible to solve every existing and potential use case. Where extensions allow new elements to be added, profiles regulate the use of resources in a specific user manual. FHIR profiles describe what specific elements, including extensions, must be used, whether they are mandatory and what terminology and additional restrictions must be used. The FHIR profile allows you to describe the exact rules for, say, the transmission of blood pressure data, the admission of an unknown patient to the emergency department or the registration of a death.

## **1.1. Related works**

The evolution of services requires the provision of support, the creation of new features, which leads to many versions that need to be maintained [9]. Service versioning typically means versioning the implementation or an interface of a service [10] [11]. In the context of service evolution, which regards the integration between customers and providers, the versioning of the service interface is often addressed, particularly the service interface description. Since the FHIR standard uses the latest web standards [1] - such as web services and REST, the most interesting relevant work is in the field of web services. Many authors over the decades have analysed versioning in the software design, and in Web Services and Service Oriented Architecture (SOA) particularly. The research identifies forward [12] and backward [13] compatibility service design techniques that facilitate a good balance between requirements in multiple environments. Often, providers publish new versions with release notes that should help clients apply changes (Google, Amazon, etc.). Typically, release notes describe the explicit changes but fail to identify

how they affect the entire service and compatibility with previous versions [14]. Service change management requires mechanism for the identification and classification of changes and mechanisms for the resource transformations between versions. These topics highlighted in different works, for accurate recognition of changes [14] and usage oriented compatibility assessments [15] [16]. Compatibility is central issue of evolution, because it provides valuable information regarding the changes in the services [10]. Several works describe solutions for compatibility issues, such as version-aware approach for Web Service Client Application [17], framework that supports the evolution of services [18] and backward compatibility study [19]. In summary, despite attempts to create a universal approach for service versioning, service stakeholders lack a proper mechanism to recognize changes and their impact on evolving services quickly.

## **1.2. Challenges of implementing changes in HL7 FHIR**

The FHIR information model, along with extensions and profiling, allows the creation of the necessary implementation guides for virtually every case in health care. This flexibility has obvious advantages and disadvantages. On the one hand, there is the possibility for rapid adoption in the changing world of health care and the provision of the most suitable information services for healthcare companies. On the other hand, constant change leads to a situation where we encounter a lot of data created on the basis of different versions of FHIR standards and profiles and their data differ to one degree or another. The multiplicity of versions raises several questions:

- Backward-compatibility ensures that consumers using old services need not be upgraded to use the modified service. That is, when service providers upgrade their services to offer the latest functionalities, service consumers need not be aware of the changes and can continue to use the services as before. On the other hand, forward-compatibility guarantees that new consumers need not be downgraded to work with an old service. That is, when service consumers deploy a new client application that conforms to the modified Web service interface, it should be able to work with the old Web service as well, without having to downgrade [12].
- How should 'old' data, i.e. data created according to previous standards, be processed in software that only supports the last valid version of the standard? How should deleted attributes, changed cardinality, expired concepts in terminology, etc. be handled?
- How should software be designed so that it can function in an environment where a new standard has been introduced, about which it knows nothing at the time of design? How should the software handle new elements still unknown to it, the disappearance of elements, new concepts in terminology, etc.?
- Can software that create data using different versions of the standard work in one information space? Do they cause one to another problems with continuous cross-version conversions?

Further work will try to answer these questions.

### 1.3. Methodology

In this paper, forward and backward compatibility design methods and individual properties in the context of the HL7 FHIR standard are compared. In analysis, the SWOT methodology is used. The SWOT methodology evaluates various parameters of the solution: strengths, weaknesses, opportunities and threats.

## 2. Design techniques and their application

### 2.1. Definitions

The FHIR specification thoroughly describes the principles of change management and versioning [20]. Based on this description, the following can be argued:

- Versions are forward compatible if the content created with the old standard is also compatible with the new version of the standard.
- Versions are not forward compatible if at least one of the requirements is not satisfied.
- Backward compatibility means that instances created against future versions of a specification will work with older versions of the specification.
- Versions are not backward compatible if at least one of the requirements is not satisfied.
- Versions are forward and backward compatible when both forward and backward compatibility conditions apply simultaneously.

### 2.2. Forward-compatible solutions

Forward compatibility can easily be achieved by the developer of the standard if the principles of the forward-compatible solution are followed. In the new version of the standard for the forward-compatible solution:

1. all elements that existed in the previous version are present;
2. new added elements are not mandatory;
3. data types are the same or have a wider range;
4. cardinality is weaker, e.g. from mandatory [1..1] becomes optional [0..1];
5. all values from the previous version are present in the related terminology list.

Consider that in version X we have resource R and in version X+1 the same resource is updated to version R' with the data structures, as illustrated in Figure 1. The most important factor is the decision of the developer of the standard on whether to support the principle of compatibility or not. If compatibility is supported, it makes sense to develop checklists to use for each modified resource, verifying that all compatibility principles have been implemented.

Resource R in version X

Element	Datatype	Cardinality
a1	t1	1..1
a2	markdown	0..1

Resource R ' in version X+1

Element	Datatype	Cardinality
<b>a1</b>	t1	<b>0..1</b>
<b>a2</b>	<b>string</b>	0..1
a3	t3	<b>0..1</b>

**Figure 1:** Forward compatibility in HL7 FHIR resources**Table 1**

Forward and non-forward compatible changes

Change in the new version	Forward Compatible
Add new artifacts (resources, elements, operations)	Yes
Add new optional element in resource	Yes
Add new artifact to the Implementation Guide	Yes
Reduce cardinality, incl. an element becomes an optional	Yes
Depricate artifact	Yes
Remove artifact (resource, element, operation, profile)	No
Modify artifact name or path	No
Modify element data type (except string to markdown)	No
Modify "Is Modifier" and "Is Summary" flags	No
Modify slicing and aggregation rules	No
Remove search criteria in profiles	No
Change the list of global profiles in Implementation Guide	No

Table 2.2 summarize the guidelines for forward compatible changes.

The developers of the FHIR standard try to adhere to the principle of compatibility with normative resources [21]. This is an indicative way of developing a standard because it eliminates many of the problems inherent in non-forward-compatible solutions. However, this doesn't guarantee that all old systems will interact with future systems.

### 2.3. Non-forward-compatible solutions

If the developer of the standard does not proceed from the compatibility principle, the content created by the old standard will not be compatible with the new version by default. In this case, two software, one with the old version and the other with the new version, start creating conflicting content, which leads to the failure of one application or another. This problem is easiest to solve organisationally, requiring the transition of all software to a new version from a certain moment. However, with the abundance of software and varying lengths of the release cycles, this is very problematic, if not impossible.

On the other hand, data that were created with an earlier version and that no longer meet the new standard will remain in the data repository. This leads to the idea that every implementer

should support not only one version of the latest standard, but all versions of the standard supported by the data repository keeper. It is widespread practice to support the last two or three versions of the standard and to archive earlier versions by preventing the adoption of data in formats which are too old.

### 2.3.1. Data conversion

Supporting two or three versions of each software significantly increases the price of the software being developed and the associated support price. In addition, archived versions are emerging, i.e. version support that may not be available on new software. Consider that the patient has several diagnoses transmitted to the data repository on an accrual basis using different versions of DSTU R2 (1.0), STU R3 (3.0) and R4 (4.0) in force at the time. Consider that the data repository keeper is archiving version DSTU R2, that is, new software no longer needs R2 support. However, the patient diagnoses query returns all resources, including those that were created in archived versions. The sample query illustrated in the following script asks for all patient 'P1' diagnoses.

```
GET /fhir/Condition?patient=P1
```

The answer is a bundle, where the Profile section specifies the version of the resource, as illustrated in Listing 1.

```
Body: { "resourceType": "Bundle",
  "type": "searchset", "total": 3, "entry": [ {
    "fullUrl": "https://example.com/base/Condition/201",
    "resource": {
      "resourceType": "Condition",
      "id": "201",
      "meta": {
        "lastUpdated": "2013-05-25T22:12:21Z"
        "profile" : ["http://hl7.org/fhir/1.0/
          StructureDefinition/Condition"]
      }
    }, ... }, {
    "fullUrl": "https://example.com/base/Condition/301",
    "resource": {
      "resourceType": "Condition",
      "id": "301",
      "meta": {
        "lastUpdated": "2016-06-26T22:12:21Z"
        "profile" : ["http://hl7.org/fhir/3.0/
          StructureDefinition/Condition"]
      }
    }, ... }, {
    "fullUrl": "https://example.com/base/Condition/401",
    "resource": {
      "resourceType": "Condition",
      "id": "401",
```

```
"meta": {
  "lastUpdated": "2019-09-29T22:12:21Z"
  "profile": [ "http://hl7.org/fhir/4.0/
                StructureDefinition/Condition" ]
}, ... }
```

Listing 1: Patient diagnoses query which returns data generated in three FHIR versions

In this situation, all resource entries must be converted from version R2 to a newer version; in our example, preferably to the latest version R4. This conversion requires precise instructions on how to perform said conversion. The following situations must be considered:

1. The element in R2 is removed from the new version – in this case, in the new version, this element must be represented as an extension.
2. The data type of the element changes between the two versions – a formula is drawn up to convert the data and the original element is presented as an extension.
3. The new version of ValueSet has no value from version R2 – a map of the concepts (ConceptMap) must be composed between versions R2 and R4.
4. In the new version, a new mandatory element is added – it is necessary to make a formula for how the value of the new element can be calculated from the data in R2.

The fact that the data in the versions withdrawn from use, as a rule, are associated with completed operations, narrowing the required number of mappings, should be taken into consideration. In a situation where it is not possible to create a formula or mapping table, or where the clinical value may change as a result of this activity, special values such as 'unknown' or 'data-in-error' must be added to the list to avoid changing the clinical content. Such conversion can be done either using the FHIR server custom adapter, which would convert R2 resources to R4 based on the described mappings, or by creating a physical entry in the data repository with reference to the newer version profile. Creating such inter-versioning mappings for resources and terminology is a serious analytical task that deserves its own article.

The solution with dynamical transformation is preferred, as it allows you to easily repeat activities without changing data, ensuring the best quality. It is important to mention that the resource conversion between versions is not part of the FHIR specification, has no agreed syntax and is not supported by any FHIR server by default. On the other hand, for large amounts of data, continuous conversion can place a heavy load on the CPU.

The same or other adapters can help make permanent transformations in data. In this case, a new version of the resource with a reference to the new standard is generated in the data repository. In this regard, it is important to proceed from the principle of not changing the original data, including, as a result of the transformation, to create a new version of the resource, with the original version remaining unchanged. An alternative is a combined solution, where for a sufficiently long period of time (6-12 months), a dynamical converter is used. After which, in the absence of any claim to clinical content, a single physical conversion from the old standard to the new one is carried out using the same adapter. The converter is then withdrawn from use and the old standard is archived.

### 3. Backward-compatible solutions

For a backward-compatible solution, the old application must be able to process copies of the new version without knowing the changes. Such a situation cannot be guaranteed by the FHIR specification, nor by the local standard developer. Software design and readiness to adopt the new version of the standard play a bigger role. A software developer can develop systems that follow backward compatibility strategies and increase their interoperability capabilities [22]. In the simplest case, the FHIR specification describes a series of actions that end-applications must use to increase backward compatibility when processing [23] a resource generated with the new standard:

1. Ignore unknown (defined in the new version) elements
2. Ignore references to an unknown resource;
3. Ignore unknown codes
4. Ignore unknown search criteria
5. Respond within the prescribed error codes when addressing an unknown URL

Unfortunately, in reality, few implementers observe these rules due to the technical limitations of the software or the potential clinical risk that arises from ignoring certain elements. Another considerable alternative is the use of narrative. Narrative [24] is a human-readable summary of the resource. In a situation where the oldest software is not able to process a resource created according to a newer standard, it:

1. does not focus on processing the content part and displays only the narrative or;
2. combines the processing of the elements it understands and the displaying of the narrative.

In these cases, the application must be designed to identify the Backward Compatibility problem and display the narrative. The sample query is as follows:

```
GET /Observation/example
Accept: application/fhir+json; fhirVersion=3.0
```

A sample response to this is illustrated in Listing 2.

```
Content-Type: application/fhir+json; fhirVersion=3.0 {
  "resourceType": "Observation",
  "id": "example",
  "text": {
    "status": "generated",
    "div": "<p><b>id </b>: example </p>
          <p><b>status </b>: final </p>
          <p><b>code </b>: Body Weight </p>
          <p><b>subject </b>: <a>Patient/example </a></p>
```



```

    <p><b>effective </b>: 28/03/2016 </p>
    <p><b>value </b>: 100 kg</p>
  }, "status": "final",
  "code": {
    "coding": [ {
      "system": "http://loinc.org",
      "code": "29463-7",
      "display": "Body Weight"
    } ] },
  "subject": {
    "reference": "Patient/example"
  },
  "effectiveDateTime": "2016-03-28",
  "valueQuantity": {
    "value": 100,
    "unit": "kg",
    "system": "http://unitsofmeasure.org",
    "code": "kg"
  } }

```

Listing 2: HL7 FHIR Narrative example

and the related visual presentation in any web browser is as follows:

```

id : example
status : final
code : BodyWeight
subject : Patient/example
effective : 28 – 03 – 2016
value: 100 kg

```

Health Information System records are often subject to commercial and legal requirements for their storage, sometimes up to 100 years, which can lead to a situation in which formalised data is not processed and is not displayed in all applications and devices for the required period of time. In order to ensure this kind of backward compatibility for each FHIR producer, it is reasonable to think about a narrative generation system that takes into account the peculiarities of local medicine, legalisation and jurisdiction and generates a narrative by which resources are received on the FHIR server if the narrative is not predefined by the record creator.

#### 4. Use of extensions in applications with backward and forward compatibility

The processing of unknown elements must be considered as a special case. Unknown elements can occur when elements are removed, renamed or added in a newer version of a specification. Some of approaches enforce forward compatibility through extensions [25]. According to the

FHIR specification, each element has an extension URL that uniquely identifies said element. An extension URL can be automatically derived in the form:

```
http://hl7.org/fhir/[version]/StructureDefinition/  
extension-[Path]
```

In this case, the FHIR server could return the elements changed between versions as extensions. The example below demonstrates this approach on the example of an animal patient, where it was possible to determine animal characteristics in the STU3 version.

```
GET /Patient/animal  
Accept: application/fhir+json; fhirVersion=3.0
```

This returns the resource with elements 'animal.species' and 'animal.genderStatus' as shown in Listing 3

```
{ "resourceType": "Patient",  
  "id": "animal",  
  "active": true,  
  "name": [ {  
    "use": "usual",  
    "given": ["Kenzi"]  
  } ],  
  "animal": {  
    "species": {  
      "coding": [ {  
        "system": "http://hl7.org/fhir/animal-species",  
        "code": "canislf",  
        "display": "Dog"  
      } ] },  
    "genderStatus": {  
      "coding": [ {  
        "system": "http://hl7.org/fhir/animal-genderstatus",  
        "code": "neutered"  
      } ] } } } }
```

Listing 3: FHIR R3 Animal species

However, the same query in the next release R4...

```
GET /Patient/animal  
Accept: application/fhir+json; fhirVersion=4.0
```

...returns resource elements (Listing 4) as animal-species and animal-genderStatus extensions, since these elements were removed from the standard in R4.

```
{ "resourceType": "Patient",  
  "id": "animal",
```

```

"active": true ,
"name": [ {
  "use": "usual" ,
  "given": ["Kenzi"]
} ] ,
"extension": [ {
  "url": "http://hl7.org/fhir/3.0/StructureDefinition/
extension-Patient.animal.species" ,
  "valueCodeableConcept": {
    "coding": [ {
      "system": "http://hl7.org/fhir/animal-species" ,
      "code": "canislf" ,
      "display": "Dog"
    } ] } } ] ,
"extension": [ {
  "url": "http://hl7.org/fhir/3.0/StructureDefinition/
extension-Patient.animal.genderstatus" ,
  "valueCodeableConcept": {
    "coding": [ {
      "system": "http://hl7.org/fhir/animal-genderstatus" ,
      "code": "neutered"
    } ] } } ] }

```

Listing 4: FHIR R4 Animal species

As you can see from the example, in the FHIR resource of normative content, a developer who creates an application in accordance with this Standard may encounter extensions they may not have considered, which is why well-designed applications that display all unknown extensions ensure better clinical quality and insight into the data.

## 5. Conclusion

Although versioning of web servers, in general, is fairly well researched, versioning in the FHIR standard is not covered in the scientific literature. The author considered the design principles of forward and backward compatibility and methods for solving problems in particular cases. Several solutions are aggregated in Table 2. A good standard developer creates implementation guides with both forward and backward compatibility principles in mind. The FHIR specification and current study prescribes a series of forward and backward compatibility rules.

Each implementer should create a checklist with the forward and backward compatibility rules specified in the FHIR specification and upgrade them based on their own deployment experience and that of others. If backward compatibility is difficult to achieve, the developer of the standard should at least ensure that the forward compatibility of the standard. In the author's opinion, failure to ensure forward compatibility is equivalent with non-versioning or with a situation where there are no versioning principles or design.

**Table 2**  
Comparison of design principles

Strengths	Weaknesses
Forward-compatible and backward-compatible solutions Forward compatible solutions	Non-backward compatible solutions
Opportunities	Threats
Backward-compatible solutions	Non-forward-compatible solutions No versioning principles

The FHIR specification does not solve all of the forward and backward compatibility issues mentioned, and given that at the time of writing the article, the share of normative resources was less than 10% of the total specification, every developer of the standard must take the versioning of the system design very seriously. On the other hand, it is equally important to enlighten educational services software developers about forward and backward compatibility, carry out FHIR Connectathons in order to increase interoperability between versions and offer certification and require software developers to comply with it.

The FHIR specification describes quite a few measures that help ensure forward and backward compatibility. The FHIR extensions mechanism helps convert resources between versions if there is appropriate support on the FHIR server. In work proposed, converters allow the content of resources between versions to be transformed dynamically. The narrative ensures the readability of clinical content by a specialist even after decades have passed. Table 3 describes the key aspects that need to be addressed in the field of forward and backward compatibility for each standard developer. This article is the first in its series and discusses the design principles that can be applied by the standard developer in their work. Further research should look at the following:

- The syntax describing transformations, which includes both conversion of resource elements and conversion of terminology concepts, is not standardised by FHIR and requires an article of its own.
- FHIR data retention logic and endpoint logic of FHIR servers, which handle different relics (e.g. R2 and R5) and serve responses using different headers, paths and even resources that were valid in the respective version.

### **Authors' contribution**

I.B. designed the idea and wrote the manuscript with support from G.P. All authors contributed to the final version of the manuscript. G.P. and P.R. supervised the project.

### **Acknowledgments**

We thank the TEHIK (Health and Welfare Information Systems Centre, Estonia) <https://www.tehik.ee/> and the Kodality <https://www.kodality.com/> teams who validated the methods de-

**Table 3**

Most important aspects of forward and backward compatibility

Strengths	Weaknesses
Generate a narrative based on uniform rules on the FHIR server to provide human-readable output	Complexity of inter-version mappings of resources and terminology
When storing resources, store the version of the resource alongside the resource	High server load, complexity and support cost for inter-version, dynamical resource converter between versions
Store resources at least in the original/received form	Ir-reversibility of on-demand transformations
Maintain forward compatibility of resources by standard design	
Opportunities	Threats
Maintain forward and backward conversions for the parts of resources that matter to the application	Changing of clinical meaning during inter-version transformation
Dynamical resource converter between versions	
On-demand data transformer from older version to current version	
Store resources in transformed state	
Store resources in both received and transformed state	
Host multiple end-points for a variety of purposes	

scribed in adopting the FHIR standard for the National Health Service of Estonia. This work in the project 'ICT programme' was supported by the European Union through the European Social Fund.

## References

- [1] HL7, Fhir is a standard for health care data exchange, published by hl7®, <http://hl7.org/fhir/>, 2022. Accessed: 2022-04-06.
- [2] HL7, History - fhir v4.6.0., <https://build.fhir.org/history.html>, 2011. Accessed: 2022-04-03.
- [3] HL7, Fhir dstu1, <http://hl7.org/fhir/DSTU1/http.html#paging>, 2011. Accessed: 2022-04-03.
- [4] HL7, Maturity - fhir v4.6.0., <http://hl7.org/fhir/versions.html#maturity>, 2011. Accessed: 2022-04-03.
- [5] M. C. Paulk, A history of the capability maturity model for software, ASQ Software Quality Professional 12 (2009) 5–19.
- [6] HL7, Fhir overview - architects., <https://www.hl7.org/fhir/overview-arch.html#principles>, 2011. Accessed: 2022-04-03.
- [7] HL7, Extensibility - fhir v4.0.1., <https://www.hl7.org/fhir/extensibility.html>, 2011. Accessed:

2022-04-03.

- [8] HL7, Profiling - fhir v4.0.1., <https://www.hl7.org/fhir/profiling.html>, 2011. Accessed: 2022-04-03.
- [9] M. P. Papazoglou, The challenges of service evolution, in: *International Conference on Advanced Information Systems Engineering*, Springer, 2008, pp. 1–15.
- [10] K. Becker, A. Lopes, D. S. Milojicic, J. Pruyne, S. Singhal, Automatically determining compatibility of evolving services, in: *2008 IEEE International Conference on Web Services*, IEEE, 2008, pp. 161–168.
- [11] D. Frank, L. Lam, L. Fong, R. Fang, M. Khangaonkar, Using an interface proxy to host versioned web services, in: *2008 IEEE International Conference on Services Computing*, volume 2, IEEE, 2008, pp. 325–332.
- [12] C. Armbruster, Design for evolution, <https://chrisarmbruster.com/documents/design-for-evolution-white-paper.pdf>, 1999. Accessed: 2022-04-03.
- [13] P. Kaminski, M. Litoiu, H. Müller, A design technique for evolving web services, in: *Proceedings of the 2006 conference of the Center for Advanced Studies on Collaborative research*, 2006, pp. 23–es. URL: <https://dl.acm.org/doi/abs/10.1145/1188966.1188997>, accessed: 2022-04-03.
- [14] M. Fokaefs, R. Mikhael, N. Tsantalis, E. Stroulia, A. Lau, An empirical study on web service evolution, in: *2011 IEEE International Conference on Web Services*, IEEE, 2011, pp. 49–56.
- [15] M. C. Yamashita, Service versioning and compatibility at feature level (2013).
- [16] M. Yamashita, B. Vollino, K. Becker, R. Galante, Measuring change impact based on usage profiles, in: *2012 IEEE 19th International Conference on Web Services*, IEEE, 2012, pp. 226–233.
- [17] R. Fang, L. Lam, L. Fong, D. Frank, C. Vignola, Y. Chen, N. Du, A version-aware approach for web service directory, in: *IEEE International Conference on Web Services (ICWS 2007)*, IEEE, 2007, pp. 406–413.
- [18] M. Yamashita, K. Becker, R. Galante, A feature-based versioning approach for assessing service compatibility, *Journal of Information and Data Management* 3 (2012) 120–120.
- [19] M. Endrei, M. Gaon, J. Graham, K. Hogg, N. Mulholland, Moving forward with web services backward compatibility, 2006.
- [20] HL7, Fhir change management and versioning, <http://hl7.org/fhir/versions.html>, 2011. Accessed: 2022-04-03.
- [21] HL7, Fhir rules for inter-version change, <http://hl7.org/fhir/versions.html#change>, 2011. Accessed: 2022-04-03.
- [22] T. Benson, G. Grieve, The fhir api, in: *Principles of Health Interoperability*, Springer, 2021, pp. 103–121.
- [23] HL7, Fhir backward compatibility rules, <http://hl7.org/fhir/versions.html#change>, 2011. Accessed: 2022-04-03.
- [24] HL7, Narrative - fhir v4.6.0, <https://build.fhir.org/narrative.html>, 2011. Accessed: 2022-04-03.
- [25] V. Andrikopoulos, S. Benbernou, M. P. Papazoglou, On the evolution of services, *IEEE Transactions on Software Engineering* 38 (2011) 609–628.