TCVN **T I Ê U C H U Ẩ N Q U Ố C G I A**

TCVN xxx:2017

ISO/IEC 18384:2016

Xuất bản lần 1

CÔNG NGHỆ THÔNG TIN – KIẾN TRÚC THAM CHIẾU CHO CÁC GIẢI PHÁP SOA (SOA RA) – PHẦN 3: TẬP CÁC KHÁI NIỆM VÀ PHÂN LOẠI MỞ RỘNG CỦA SOA

Information technology – Reference Architecture fr Service Oriented Architecture (SOA RA) – Part 3: Service Oriented Architecture bản thể luận

HÀ NỘI – 2017

TIÊU CHUẨN QUỐC GIA **TCVN xxx : 2017**

Công nghệ thông tin – Kiến trúc tham chiếu cho Kiến trúc hướng dịch vụ (SOA RA)

Phần 3:

Tập các khái niệm và phân loại mở rộng của SOA (Service Oriented Architectue Bản thể luận)

1. Phạm vi

Phần này của tiêu chuẩn ISO/IEC 18384 định nghĩa tập các khái niệm và phân loại mở rộng chính thức cho SOA, một kiểu kiến trúc hỗ trợ hướng dịch vụ. Các thuật ngữ được định nghĩa trong tập các khái niệm này đều là các thuật ngữ chính trong từ điển thuộc Phần ISO/IEC 18384-1.

2. Tài liệu tham khảo

Tài liệu sau đây (một phần hoặc toàn bộ) được tham chiếu một cách hợp lệ trong tài liệu này và không thể thiếu khi áp dụng. Đối với các tài liệu có nhiều phiên bản, chỉ tham chiếu đến phiên bản được nêu. Đối với các tài liệu không ghi phiên bản, chỉ áp dụng phiên bản gần nhất.

ISO/IEC 18384-1, *Information technology — Reference Architecture for Service Oriented Architecture (SOA RA) — Part 1 Terminology and concepts for SOA*.

3. Thuật ngữ, định nghĩa và từ viết tắt

3.1. Thuật ngữ và định nghĩa

Đối với mục đích của tài liệu này, các thuật ngữ và định nghĩa được đưa ra trong tập ISO/IEC 18384-1 và áp dụng như dưới đây.

3.1.1.

Không trong suốt

Cấu trúc bên trong không thể nhìn thấy được bởi người quan sát bên ngoài.

3.1.2.

Bản thể luận

Mô hình biểu diễn một miền và được sử dụng để bàn luận về các đối tượng trong miền đó và mối quan hệ giữa chúng.

Lưu ý 1: Phần này của tiêu chuẩn ISO/IEC 18384 ở mức cao và không có nghĩa là được sử dụng cho các kết luận chính thức.

[SOURCE: ISO/IEC/TR 24800-1:2007, 2.1.9]

3.2. Các từ viết tắt

Đối với mục đích của tài liệu này, có các từ viết tắt sau đây

ABB	Khối kiến trúc thành phần
BPMN	Mô hình hóa và Ký hiệu quy trình nghiệp vụ
EA	Kiến trúc tổng thể
ESB	Trục dịch vụ tổng thể
IT	Công nghệ thông tin
OWL	Ngôn ngữ bản thể luận Web
RA	Kiến trúc tham chiếu
RDF	Khung định nghĩa tài nguyên
SLA	Thỏa thuận mức độ dịch vụ
SOA	Kiến trúc hướng dịch vụ
UML	Ngôn ngữ mô hình hóa thống nhất

4. Ký hiệu

Bản thể luận được biểu diễn trong ngôn ngữ bản thể luận Web định nghĩa bởi Hiệp hội Web toàn cầu (W3C). OWL có 3 ngôn ngữ biểu diễn thành phần gồm: OWL-Lite, OWL-DL và OWL-Full (xem Tài liệu tham khảo [10] để biết định nghĩa về 3 ngôn ngữ của OWL này). Bản thể luận này sử dụng OWL-DL, ngôn ngữ thành phần có khả năng diễn đạt lớn nhất và vẫn duy trì được tính đầy đủ và khả năng quyết định.

Bản thể luận bao gồm các lớp (class) và các đặc tính (properties) tương ứng với các khái niệm về SOA. Các định nghĩa chính thức của OWL được bổ sung bằng các mô tả ngôn ngữ tự nhiên về khái niệm, với nhiều hình ảnh minh họa về mối quan hệ giữa chúng và các ví dụ về việc sử dụng chúng. Với mục đích trình bày, bản thể luận cũng bao gồm UML (xem Tài liệu tham khảo [8] minh họa hình ảnh về các lớp và đặc tính. Các định nghĩa OWL và ngôn ngữ tự nhiên trong Phần này của tiêu chuẩn ISO/IEC 18384 thiết

lập định nghĩa bản thể luận có giá trị; các sơ đồ chỉ dành cho các mục đích giải thích. Một số thuật ngữ ngôn ngữ tự nhiên được sử dụng để mô tả các khái niệm không được biểu diễn chính thức trong bản thể luận; các thuật ngữ đó mang ý nghĩa theo đúng ngôn ngữ tự nhiên của chúng.

Tính sẵn có của OWL biểu diễn định dạng RDF tiêu chuẩn cho phép dễ dàng tải về các công cụ của kiến trúc sư và người phát triển và cũng cho phép xác nhận giá trị của chúng.

Phần này của tiêu chuẩn ISO/IEC 18384 sử dụng các ví dụ minh họa bản thể luận. Một trong số các ví dụ này, ví dụ rửa xe, được sử dụng thống nhất trong toàn bộ tài liệu để minh họa các khái niệm chính (xem Phụ lục A để thấy ví dụ hoàn chỉnh). Các ví dụ khác được sử dụng rời rạc trong từng phần để minh họa các điểm cụ thể khác nhau.

5. Các quy ước

Phông chữ **in đậm** được sử dụng cho lớp, thuộc tính và tên một thể hiện của OWL khi chúng xuất hiện trong câu.

Chữ *in nghiêng* được sử dụng để nhấn mạnh và để xác định thể hiện đầu tiên của một từ yêu cầu định nghĩa.

Các định nghĩa và cú pháp OWL được hiển thị bằng phông chữ có chiều rộng cố định.

Mũi tên không gắn nhãn trong các biểu đồ UML có nghĩa là lớp thành phần.

Các ví dụ trong Phần này của tiêu chuẩn ISO/IEC 18384 là các thông tin chính thức và nhằm mục đích minh họa.

6. Sự phù hợp

ISO/IEC 18384 gồm 3 phần phù hợp với các yêu cầu khác nhau:

1. Thuật ngữ và khái niệm – chỉ phù hợp với các thuật ngữ và gắn với ngữ nghĩa trong các định nghĩa;

2. Kiến trúc tham chiếu cho các giải pháp SOA – chỉ phù hợp với ngữ nghĩa của mô hình đặc tả và các lớp, các ABB hay các khả năng được sử dụng

3. Bản thể luận về SOA – phù hợp cho các ứng dụng OWL hoặc các ứng dụng khác.

Sự phù hợp với Phần này của tiêu chuẩn ISO/IEC 18384 được định nghĩa như dưới đây.

Có hai loại ứng dụng dụng có thể phù hợp với bản thể luận này. Một là các bản thể luận dựa trên OWL (thường là các mở rộng của bản thể luận về SOA); Hai là ứng dụng không dựa trên OWL, ví dụ như một mô hình đặc tả hoặc một phần của phần mềm (xem Khoản 2 cho phiên bản OWL được yêu cầu)

Một ứng dụng OWL phù hợp (bắt nguồn từ bản thể luận dựa trên OWL)

- Phù hợp với tiêu chuẩn OWL được định nghĩa trong Khoản 2,

- Chứa toàn bộ bản thể luận trong Phụ lục C,

- Có thể thêm các cấu trúc OWL khác, bao gồm các định nghĩa lớp và đặc tính,

- Có thể có ý nghĩa với các bản thể luận khác ngoài bản thể luận về SOA.

Phần này của tiêu chuẩn ISO/IEC 18384 không sử dụng cấu trúc OWL 2 (xem Tài liệu tham khảo [15]); tuy nhiên, các ứng dụng phù hợp có thể lựa chọn sử dụng OWL hoặc OWL 2.

Một ứng dụng dụng không dựa trên OWL phù hợp

- Bao gồm chuyển đổi đã được định nghĩa và thống nhất (ít nhất là về ánh xạ ngữ nghĩa) thành một tập con của bản thể luận trong Phụ lục C,

- Có thể thêm các cấu trúc khác, chứa các định nghĩa lớp và đặc tính,

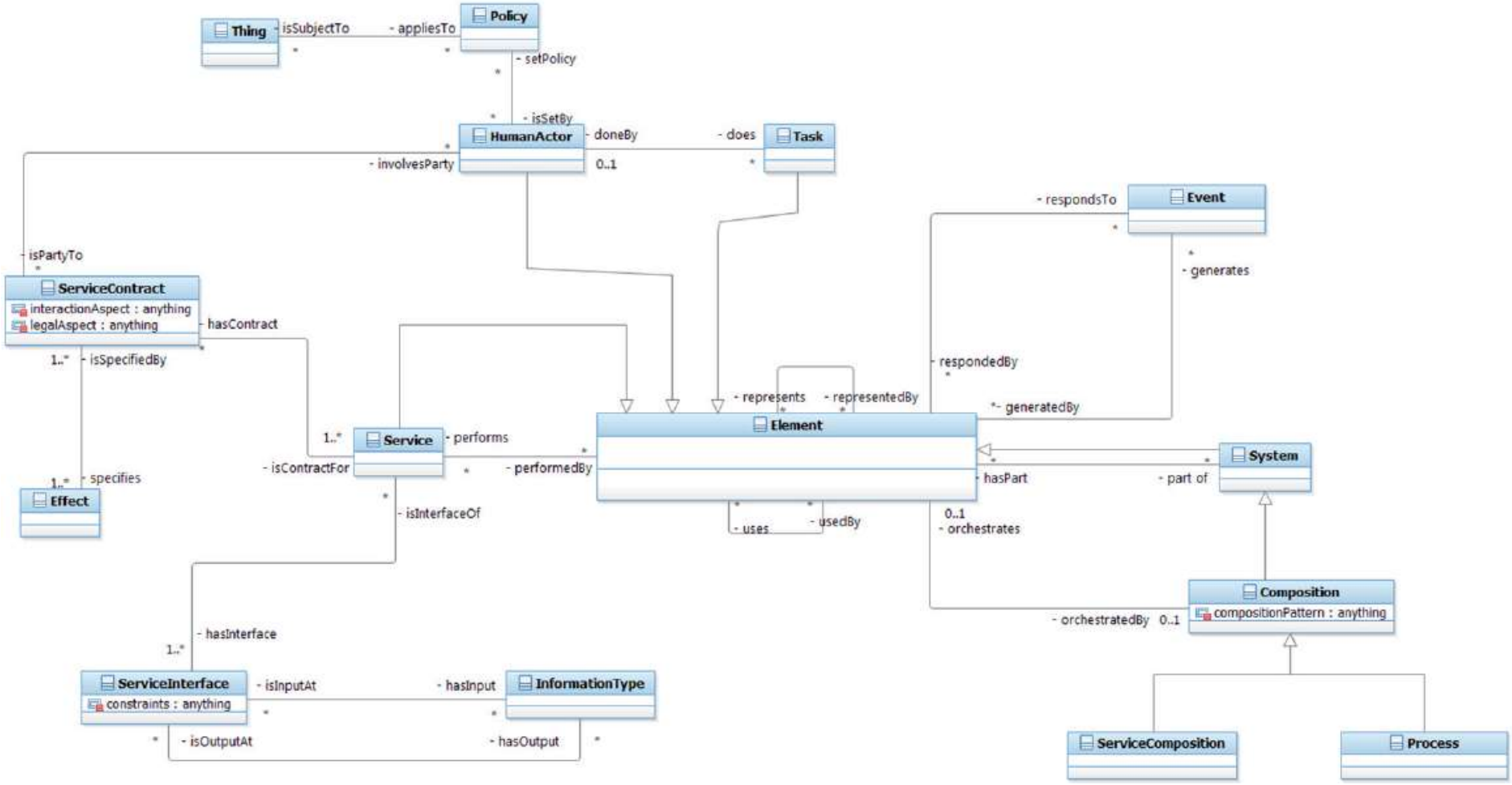
- Có thể có ý nghĩa và/hoặc sử dụng các bản thể luận khác ngoài bản thể luận về SOA

7. Tổng quan về Bản thể luận về SOA

7.1. Tóm tắt

Hình ảnh trực quan về toàn bộ bản thể luận được biểu diễn trong Hình 1. Các khái niệm trong Hình 1 được mô tả trong nội dung bên dưới.

Phần này của tiêu chuẩn ISO/IEC 18384 bắt đầu bằng việc giải thích các khái niệm nền tảng cơ bản nhất về các phần tử và các hệ thống được theo dõi bằng việc giải thích các phần tử tác nhân con người (human actor) và tác vụ (task) SOA, sau đó, đưa ra các khái niệm, mô tả và hợp đồng cho các dịch vụ và phục vụ xây dựng trên đó nhằm giải thích các hợp thành (compositions) của các dịch vụ. Cuối cùng, Phần này của tiêu chuẩn ISO/IEC 18384 gồm cả các Chính sách (Policies) và Sự kiện (Events) có liên quan đến tất cả các phần tử của SOA.



Hình 1: Bản thể luận về SOA – Sơ đồ tổng quan

7.2. Sử dụng có mục đích

Mục này đưa ra các khuyến cáo và các giả định về cách thức để hiểu bản thể luận.

- Bản thể luận này được sử dụng cho việc biểu diễn các khái niệm ở mức cao và không hướng đến sử dụng cho việc bàn luận chính thức.

- Phần này của tiêu chuẩn ISO/IEC 18384 được thiết kế dành cho người chủ nghiệp vụ, kiến trúc sư và người thiết kế hệ thống, phần mềm, hỗ trợ cho người chủ nghiệp vụ và người phụ trách kỹ thuật có thể hiểu nhau.

- Phần này của tiêu chuẩn ISO/IEC 18384 tập trung vào một tập tối thiểu các thuật ngữ SOA, mô hình hóa các thuật ngữ đó một cách chi tiết.

- Phần này của tiêu chuẩn ISO/IEC 18384 giải thích mối quan hệ với các khái niệm quan trọng khác nhưng không cùng mức độ chi tiết như các thuật ngữ SOA. Ví dụ, chính sách được mô hình hóa nhưng không thể chi tiết được.

- Phần này của tiêu chuẩn ISO/IEC 18384 giới hạn đối với các cấu trúc OWL, không sử dụng các cấu trúc đó trong OWL 2 (xem Tài liệu tham khảo [15]), bởi vì các cấu trúc OWL thuộc phạm vi Phần này của tiêu chuẩn ISO/IEC 18384. Điều này nhất quán với OWL 2 và không cản trở các cấu trúc khác sử dụng cùng với OWL 2.

- Phần này của tiêu chuẩn ISO/IEC 18384 nghiên cứu các thuật ngữ và mối quan hệ SOA trong Phần tiêu chuẩn ISO/IEC 18384-1 và Phần tiêu chuẩn ISO/IEC 18384-2. Mô hình đặc tả riêng trong ISO/IEC 18384-2 cung cấp cơ sở cho việc mô hình hóa trong ISO/IEC 18384-2 và được sử dụng để mô tả và hiểu hơn về kiến trúc tham chiếu.

- Phần này của tiêu chuẩn ISO/IEC 18384 đưa ra các khái niệm, thuật ngữ và ngữ nghĩa SOA cả về mặt nghiệp vụ và kỹ thuật, từ đó, tạo ra nền tảng cho các công việc tiếp theo trong các lĩnh vực cụ thể.

- Phần này của tiêu chuẩn ISO/IEC 18384 cung cấp phương tiện để giải quyết các vấn đề và đưa ra các cơ hội rõ ràng để thúc đẩy sự hiểu biết lẫn nhau.

- Phần này của tiêu chuẩn ISO/IEC 18384 có thể đưa ra điểm khởi đầu cho sự phát triển các giải pháp SOA theo hướng mô hình.

7.3. Việc áp dụng

Bản thể luận về SOA được phát triển để nâng cao hiểu biết và dễ dàng đọc hiểu.

Nó cũng có thể được coi là điểm khởi đầu cho sự phát triển theo hướng mô hình, bằng việc áp dụng vào các miền ứng dụng cụ thể.

Bản thể luận được áp dụng cho một miền sử dụng cụ thể bằng việc thêm các thể hiện (instances) lớp OWL SOA của các sự vật thuộc miền đó. Điều này đôi được gọi là "phổ biến bản thể luận". Ngoài ra, một ứng dụng có thể bổ sung thêm định nghĩa của các lớp và thuộc tính mới, có thể nhập các bản thể luận khác và có thể nhập thể hiện OWL bản thể luận vào các bản thể luận khác.

Bản thể luận định nghĩa mối quan hệ giữa các thuật ngữ, nhưng không quy định chính xác cách chúng được áp dụng. Để giải thích về bản thể luận là gì và tại sao chúng lại cần thiết, xem Tài liệu tham khảo [11] và [14]. Các ví dụ được cung cấp trong phần này của ISO/IEC 18384 đang mô tả cách mà bản thể luận có thể được áp dụng trong các tình huống thực tế. Các ứng dụng khác nhau của bản thể luận cho các tình huống tương tự vẫn có thể xảy ra. Sự khởi tạo chính xác bản thể học trong các tình huống thực tế cụ thể là một vấn đề đối với người dùng bản thể luận, miễn là các khái niệm và ràng buộc được xác định bởi bản thể luận được áp dụng đúng, sự khởi tạo là hợp lệ.

8. Hệ thống và các phần tử

8.1. Tổng quan

System (hệ thống) và *element* (phần tử) là hai khái niệm của bản thể luận này. Cả hai khái niệm đều được sử dụng bởi những người triển khai thực tế, bao gồm cả khái niệm về các hệ thống với các phần tử và các hệ thống kết hợp với nhau (hệ thống gồm các hệ thống). Sự khác nhau giữa miền này với miền kia là ở bản chất cụ thể của các hệ thống và các phần tử, ví dụ, một hệ thống điện có các loại phần tử khác với một hệ thống SOA.

Trong bản thể luận, chỉ xem xét các phần tử và các hệ thống trong cùng một miền SOA. Một số miền SOA thành phần sử dụng thuật ngữ *component* (thành phần) thay cho thuật ngữ phần tử. Điều này không bị mâu thuẫn vì bất kỳ cấu phần nào của một hệ thống SOA cũng có thể là một phần tử của hệ thống đó.

Khoản này mô tả các lớp dưới đây của bản thể luận:

Element

System

Ngoài ra, nó định nghĩa các đặc tính như sau:

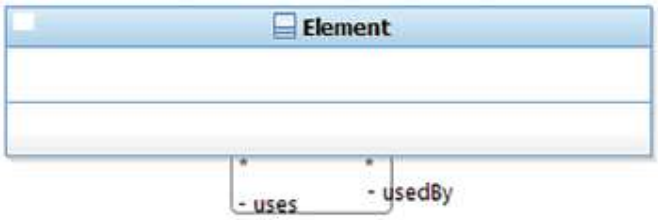
use và **usedBy**

represents và **representedBy**

8.2. Lớp Element

```
<owl:Class rdf:about="#Element">
</owl:Class>
```

Một *element* (phần tử) là một thực thể không rõ ràng và không thể chia cắt ở mức trừu tượng. Phần tử có ranh giới được xác định rõ ràng. Khái niệm phần tử được thể hiện bởi lớp **Element** OWL**,** như minh họa trong Hình 2



Hình 2 – Lớp phần tử

Trong ngữ cảnh bản thể luận về SOA, chỉ xét chi tiết các phần tử chức năng thuộc miền SOA. Có hơn 4 loại phần tử khác nhau thuộc 4 lớp thành phần (System, HumanActor, Task và Service) được mô tả trong phần dưới đây. Các ví dụ về các loại phần tử như thành phần phần mềm hay thành phần công nghệ (chẳng hạn như việc thực hiện Trục dịch vụ tổng thể ESB,…).

8.3. Các đặc tính uses và usedBy

```
<owl:ObjectProperty rdf:about="#uses">
   <rdfs:domain rdf:resource="#Element"/>
   <rdfs:range rdf:resource="#Element"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="#usedBy">
   <owl:inverseOf>
    <owl:ObjectProperty rdf:about="#uses"/>
   </owl:inverseOf>
</owl:ObjectProperty>
```

Các phần tử có thể sử dụng các phần tử khác theo nhiều cách khác nhau. Nhìn chung, khái niệm về một vài phần tử sử dụng phần tử khác được áp dụng bởi những người triển khai thực tế tất cả các mô hình, việc thực thi và các đối tượng vật lý. Sự khác nhau giữa miền này với miền kia phụ thuộc vào cách nhìn của người sử dụng.

Một phần tử sử dụng một phần tử khác nếu nó tương tác (interacts) với nó theo một cách nào đó. Tương tác ở đây được giải thích là rất khác nhau, ví dụ, một phần tử chỉ đơn giản là một thành phần (được sử dụng bởi) của một vài hệ thống (xem phần định nghĩa chính thức về lớp **System**), một phần tử tương tác với (sử dụng) một phần tử khác (chẳng hạn như một dịch vụ; xem phần định nghĩa chính thức về lớp **Service**) theo hình thức *ad hoc* (rời rạc) hoặc thậm chí bị phụ thuộc chặt chẽ vào một hợp thành (xem phần định nghĩa chính thức về lớp **Composition**). Đặc tính **uses** (sử dụng) và **usedBy** (được sử dụng bởi), là các quan niệm trừu tượng của một phần tử sử dụng phần tử khác. Các đặc tính này không chỉ biểu diễn mối quan hệ tạm thời. Các thể hiện của đặc tính có thể bao gồm “đang sử dụng tại thời điểm

hiện tại" (uses at this instant), "đã và đang sử dụng" (has used) và "có thể sử dụng trong tương lai" (may in future use).

Đối với mục đích của bản thể luận này, còn có rất nhiều ý nghĩa có thể hiểu khác nhau về quan hệ *uses* chưa được liệt kê và định nghĩa chính thức. Các giải thích ý nghĩa được trình bày đối với cách tiếp cận miền thành phần, ứng dụng hay thậm chí là thiết kế cụ thể.

8.4. Element - Ví dụ về tổ chức

Sử dụng ví dụ về tổ chức, một **Element** điển hình là các đơn vị tổ chức và con người của tổ chức đó. Việc xem xét một phần nhất định của một tổ chức là một đơn vị hay là một nhóm người trong một đơn vị là một lựa chọn quan trọng mang tính trừu tượng.

Trong nội bộ một đơn vị tổ chức, đơn vị đó thực tế có thể sử dụng con người như là các thành viên. Lưu ý rằng, một người có thể là một thành viên của (được sử dụng bởi) nhiều đơn vị tổ chức.

Đối với bên ngoài, cơ cấu nội bộ của một đơn vị là chưa rõ ràng, vì tổ chức có thể thay đổi con người trong một đơn vị mà không cần thay đổi định nghĩa về đơn vị tổ chức đó.

Ví dụ đơn giản này thể hiện rằng một số phần tử có cấu trúc bên trong. Thực tế, nhìn trong nội bộ, chúng là một tập có cấu trúc các vật đơn giản khác (theo định nghĩa lớp Hệ thống phần 8.5)

8.5. Lớp System

```
<owl:Class rdf:about="#System">
   <owl:disjointWith>
    <owl:Class rdf:about="#Task"/>
   </owl:disjointWith>
   <owl:disjointWith>
    <owl:Class rdf:about="#Service"/>
   </owl:disjointWith>
   <rdfs:subClassOf>
    <owl:Class rdf:about="#Element"/>
   </rdfs:subClassOf>
</owl:Class>
```

Một *system* là một tập có tổ chức các vật khác. Cụ thể, các vật trong tập hệ thống là các thể hiện của **Element** và mỗi thể hiện như vậy được sử dụng bởi hệ thống. Khái niệm *system* được thể hiện bởi lớp **System** OWL, minh họa trong Hình 3.

Hình 3 – Lớp Hệ thống

Định nghĩa hệ thống này bị ảnh hưởng lớn từ tiêu chuẩn ISO/IEC 42010:2011 (Tài liệu tham khảo [13]).

Trong ngữ cảnh bản thể luận về SOA, chỉ xem xét chi tiết các hệ thống chức năng thuộc miền SOA. Lưu ý rằng một thể hiện của **System** được mô tả đầy đủ nên bao gồm các nội dung cơ bản (như là một tập) *một phần* quan hệ với thể hiện ít nhất một của **Element.**

Do System là một lớp thành phần của **Element** nên tất cả các hệ thống đều có ranh giới và không có tính rõ ràng (quan điểm hộp đen) đối với bên ngoài, loại trừ các cấu trúc lớp **System** vì cấu trúc của nó không có ranh giới rõ ràng. Theo quan điểm SOA, điều này thật sự không phải là nhược điểm do tất cả các hệ thống SOA đáng quan tâm đều có các đặc điểm có thể nhận ra đối với người bên ngoài (người sử dụng). Hơn nữa, lớp **System** là một lớp thành phần của **Element** cho phép chúng ta biểu diễn một cách tự nhiên quan niệm về hệ thống của các hệ thống – các hệ thống mức thấp hơn chỉ đơn giản là các phần tử được sử dụng bởi hệ thống mức cao hơn.

Đồng thời khi hỗ trợ quan điểm nhìn từ bên ngoài (quan điểm hộp đen), tất cả các hệ thống cũng hỗ trợ quan điểm nhìn từ bên trong (quan điểm hộp trắng) cho thấy cách thức biểu diễn chúng là một tập có tổ chức. Ví dụ, đối với quan niệm về một dịch vụ, điều này thường tương ứng với quan điểm đặc tả kỹ thuật dịch vụ *so với* quan điểm thực hiện dịch vụ (tương tự như cách mà SoaML[9] định nghĩa các dịch vụ khi có cả phần đặc tả kỹ thuật/quan điểm hộp đen và phần thực hiện/quan điểm hộp trắng).

Điều quan trọng phải nhận ra rằng ngay cả khi các hệ thống sử dụng các phần tử biểu diễn một khía cạnh quan trọng về đặc tính **uses**, thì không cần thiết phải "tạo ra" một hệ thống chỉ để biểu diễn một vài phần tử sử dụng phần tử khác. Thực tế, thậm chí đối với các hệ thống, có thể biểu diễn rằng chúng có sử dụng các phần tử bên ngoài – mặc dù điều này trong nhiều trường hợp sẽ không được biểu diễn ở mức hệ thống, thay vào đó là bởi một phần tử của hệ thống sử dụng thể hiện của **Element** bên ngoài đó.

System được định nghĩa tách rời với các lớp *Service* và *Task*. Các thể hiện của các lớp này được coi là các tập chứa các vật khác. **System** đặc biệt không được định nghĩa tách rời với lớp *HumanActor* do thực tế trong nhiều thể hiện, một tổ chức chỉ là một loại hệ thống cụ thể. Tuy nhiên, không có định nghĩa cụ thể về lớp giao cắt này.

8.6. System – Các ví dụ

8.6.1. Ví dụ về tổ chức

Tiếp tục ví dụ về tổ chức từ phần 8.5, một đơn vị tổ chức có thể được biểu diễn như một thể hiện của System bao gồm nhiều người đóng vai trò là các thành viên (và các thể hiện của phần tử).

8.6.2. Ví dụ về hợp thành dịch vụ

Sử dụng ví dụ hợp thành dịch vụ, dịch vụ A và B là các thể hiện của **Element** và hợp thành của A và B là một thể hiện của **System** (hệ thống sử dụng A và B). Điều quan trọng là phải nhận ra hành động hợp thành khác với sự hợp thành như là một vật - ở ngữ cảnh này, thuật ngữ hợp thành được sử dụng với nghĩa thứ hai.

Xem thêm Khoản 11 đưa ra một định nghĩa chính thức về dịch vụ và hợp thành dịch vụ (và nhắc lại ví dụ trong ngữ cảnh rõ ràng hơn).

8.6.3. Ví dụ về rửa xe ô tô

Xem xét một doanh nghiệp rửa xe ô tô. Toàn bộ công ty là một đơn vị tổ chức và có thể là nhiều thể hiện trong bản thể luận theo cách sau:

- *CarWashBusiness* là một thể hiện của **System**.

- *Joe* (người chủ) là một thể hiện của **Element** và được sử dụng bởi (chủ sở hữu của) *CarWashBusiness*.

- *Mary* (thư ký) là một thể hiện của **Element** và được sử dụng bởi (nhân viên của) *CarWashBusiness*.

- *John* (người rửa xe) là một thể hiện của **Element** và được sử dụng bởi (nhân viên của) *CarWashBusiness*.

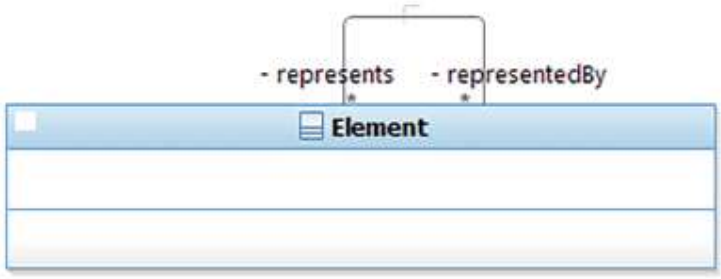
- *Jack* (người quản lý và vận hành rửa xe) là một thể hiện của **Element** và được sử dụng bởi (nhân viên của) *CarWashBusiness.*

8.7. Đặc tính represents và representedBy.

```
<owl:ObjectProperty rdf:about="#represents">
   <rdfs:domain rdf:resource="#Element"/>
    <rdfs:range rdf:resource="#Element"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="#representedBy">
   <owl:inverseOf>
    <owl:ObjectProperty rdf:about="#represents"/>
   </owl:inverseOf>
</owl:ObjectProperty>
```

Môi trường được mô tả bởi một Kiến trúc SOA là hợp thành có thứ bậc về mặt bản chất (xem phần 11.2 về định nghĩa lớp *Composition*); nói một cách khác, các phần tử của các hệ thống SOA có thể được lặp đi lặp lại gồm cả các mức trừu tượng cao hơn. Một điểm quan trọng là điều này vừa có thể được xử lý bởi đặc tính **uses** và **usedBy**, trong đó, thể hiện được quan niệm hệ thống gồm các hệ thống. Đây vẫn là một quan hệ rất bền vững mặc dù không thể hiện khái niệm trừu tượng về mặt kiến trúc. Nhu cầu về khái niệm trừu tượng của kiến trúc được tìm thấy ở nhiều nơi, chẳng hạn như một vai trò đại diện cho con người đóng vai trò đó, một đơn vị tổ chức đại diện cho những con người của đơn vị đó (khác với cùng một đơn vị tổ chức sử dụng những người của đơn vị đó, khi đó mối quan hệ **represents** cho thấy đơn vị là một điểm tương tác thay thế cho con người), một khối xây dựng kiến trúc đại diện cho một cấu trúc cơ bản (ví dụ, rất quan trọng đối với kiến trúc sư của tổ chức khi muốn phân biệt rõ ràng giữa các khối cấu trúc và khối xây dựng kiến trúc), và một trục dịch vụ tổng thể (ESB) đại diện cho các dịch vụ có thể truy cập thông qua trục (ví dụ, thích hợp khi tồn tại sự tương tác và sự phụ thuộc vận hành mô hình hóa rõ ràng). Khái niệm về những điểm thay đổi rõ ràng như vậy hay mức độ trừu tượng, được chứa bởi đặc tính *reprensents* (biểu diễn) và *representedBy* (được biểu diễn bởi) như minh họa trong Hình 4.



Hình 4 – Đặc tính represents và representedBy

Điều quan trọng để hiểu chính xác bản chất sự khác biệt giữa việc sử dụng một phần tử (E1) và việc sử dụng một phần tử (E2) để biểu diễn E1. Nếu E1 thay đổi, thì bất kỳ phần tử nào sử dụng trực tiếp E1 đều có sự thay đổi theo nhưng phần tử sử dụng E2 thì không có sự thay đổi nào.

Khi áp dụng khái niệm trừu tượng kiến trúc thông qua đặc tính **represents**, có 3 lựa chọn kiến trúc khác nhau có thể được tạo ra gồm:

Một phần tử đại diện cho một phần tử khác theo nghĩa đen, chỉ đơn giản bằng cách ẩn đi sự tồn tại của phần tử đó và bất kỳ thay đổi nào đối với nó. Sẽ có mối quan hệ một-một giữa thể hiện của **Element** và thể hiện (khác) của **Element** mà nó đại diện. Một ví dụ thực tế đơn giản là khái niệm của một nhà môi giới hoạt động như một trung gian giữa một người bán (mà không muốn được biết đến) và một người mua.

Một phần tử đại diện cho một phần tử khác. Tồn tại quan hệ nhiều-một giữa các thể hiện của **Element** (mỗi một thể hiện biểu diễn một khía cạnh khác) và một thể hiện (khác) của **Element**. Ví dụ thực tế đơn giản như quan niệm rằng một người có thể đóng (được thể hiện bởi) nhiều vai trò khác nhau.

Một phần tử là khái niệm trừu tượng có thể biểu diễn rất nhiều phần tử khác. Tồn tại quan hệ một-nhiều giữa một thể hiện của **Element** (như một khái niệm trừu tượng) với nhiều thể hiện của **Element** khác. Ví dụ thực tế đơn giản là quan niệm về bản thiết kế kiến trúc biểu diễn khái niệm trừu tượng của rất nhiều khối khác nhau được xây dựng theo bản thiết kế đó.

Nên nhớ rằng trong thực tế, một thể hiện của **Element** sẽ chỉ biểu diễn một loại sự vật. Đặc biệt, một thể hiện của **Element** thông thường đại diện cho các thể hiện thuộc một trong các lớp **System**, **Service**, Human **Actor** và **Task** (trừ trường hợp một sự vật vừa là thể hiện thuộc **System** và vừa thuộc **Actor**). Tham khảo các khoản dưới đây về định nghĩa các lớp **Service**, Human **Actor** và **Task**.

8.8. Các ví dụ về represents và representedBy

8.8.1. Ví dụ về tổ chức

Mở rộng hơn ví dụ về tổ chức, giả sử rằng một công ty mong muốn thành lập một đơn vị tổ chức O1. Có 2 cách để thực hiện việc này.

Xác định tổ chức mới trực tiếp như là một nhóm người P1, P2, P3 và P4. Điều này có nghĩa là tổ chức mới được coi là một nút lá trong phân cấp tổ chức và do đó, bất kỳ sự thay đổi nào về nhân sự cũng có nghĩa là tổ chức đó cần phải thay đổi theo.

Xác định tổ chức mới là một cấu trúc tổ chức mức cao hơn, kết hợp với 2 tổ chức đã có là O3 và O4. Thật trùng hợp, sự kết hợp O3 và O4 có thể vẫn gồm 4 người P1, P2, P3 và P4, nhưng khi bất cứ thành viên nào của O3 hoặc O4 thay đổi thì cũng không gây ảnh hưởng đến tổ chức mới. Hơn nữa, các thành viên của O3 thực chất không làm việc trong cùng một tổ chức với các thành viên của O4 (trong thực thế chúng ta không cần phải biết điều này) – ngược lại với lựa chọn đầu tiên khi P1, P2, P3 và P4 đều là đồng nghiệp trong cùng một tổ chức mới.

Theo cách này, khía cạnh trừu tượng của đặc tính **represents** tạo ra sự khác biệt ngữ nghĩa của việc xác định tổ chức mới. Bất kỳ sự tạo ra thể hiện (instantiation) bản thể luận nào cũng có thể sử dụng các đặc tính **represents** và **representedBy** để xác định một cách rõ ràng ngữ nghĩa và những thay đổi.

8.8.2. Ví dụ về rửa xe ô tô

Joe lựa chọn tổ chức doanh nghiệp thành 2 đơn vị, một đơn vị phục vụ quản lý và một đơn vị thực hiện rửa xe. Việc này có thể được thể hiện trong bản thể luận như sau:

- *CarWashBusiness* là một thể hiện của **System**.

- *AdministrativeSystem* là một thể hiện của **System**.

- *Administration* là một thể hiện của Element đại diện cho *AdministrativeSystem* (khía cạnh đơn vị tổ chức là không “trong suốt”, nghĩa là bỏ qua tất cả các vật liên quan đến *AdministrativeSystem*).

- *CarwashBusiness* sử dụng (có đơn vị tổ chức) *Administration*.

- *CarWashSystem* là một thể hiện của **System**.

- *CarWash* là một thể hiện của **Element** đại diện cho *CarWashSystem* (khía cạnh đơn vị tổ chức là không "trong suốt", nghĩa là bỏ qua tất cả các vật liên quan đến *CarWashSystem*).

- *CarWash* là thành viên của *CarWashBusiness*.

- *Joe* (chủ sở hữu) là một thể hiện của **Element** và hiện tại được sử dụng bởi *AdministrationSystem*.

- *Mary* (thư ký) là một thể hiện của **Element** và hiện tại được sử dụng bởi *AdministrationSystem*.

- *John* (người rửa xe) là một thể hiện của **Element** và hiện tại được sử dụng bởi *CarWashSystem*

- *Jack* (Người quản lý và vận hành rửa xe) là một thể hiện của **Element** và hiện tại được sử dụng bởi *CarWashSystem*

9. HumanActor và Task

9.1. Tổng quan

Con người, tổ chức và các vật mà chúng thực hiện là khía cạnh quan trọng của các hệ thống SOA. *HumanActor* (tác nhân con người) và *Task* (tác vụ) nắm bắt điều này như là một tập hợp gồm các khái niệm cốt lõi khác về bản thể luận. Cả hai đều là những khái niệm chung và có liên quan đến phạm vi bên ngoài miền SOA. Với mục đích của bản thể luận về SOA này, phạm vi cụ thể trong các tác vụ là cơ bản (automic - nguyên tử) [tương ứng với, ví dụ như định nghĩa ký hiệu mô hình hóa quy trình nghiệp vụ (BPMN) 2.0 về Task (tham khảo tài liệu tham chiếu [4]) và các tác nhân hạn chế đối với con người và tổ chức.

Khoản này mô tả các lớp sau đây của bản thể luận:

HumanActor

Task

Ngoài ra, nó còn định nghĩa các đặc tính:

Does và **doneBy**

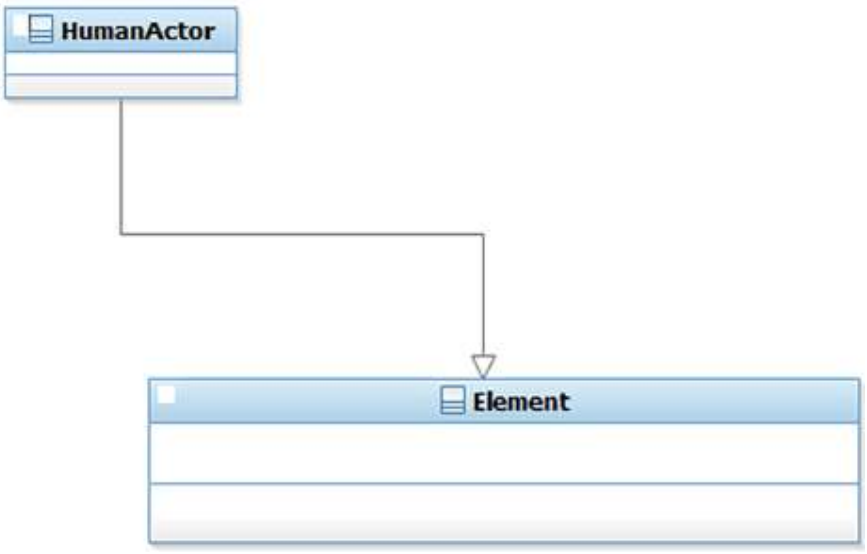
9.2. Lớp **HumanActor**

```
<owl:Class rdf:about="#HumanActor">
   <rdfs:subClassOf>
    <owl:Class rdf:about="#Element"/>
   </rdfs:subClassOf>
```

```
   <owl:disjointWith>
    <owl:Class rdf:about="#Task"/>
   </owl:disjointWith>
   <owl:disjointWith>
   <owl:Class rdf:about="#Service"/>
    </owl:disjointWith>
   </owl:Class>

<owl:Class rdf:about="#ServiceContract"/>
   </owl:disjointWith>
   <owl:disjointWith>
    <owl:Class rdf:about="#ServiceInterface"/>
   </owl:disjointWith>
</owl:Class>
```

Một *human actor* (tác nhân con người) là một người hoặc một tổ chức. Khái niệm về *human actor* được thể hiện bởi lớp **HumanActor** OWL, minh họa trong Hình 5



Hình 5 – Lớp HumanActor

HumanActor được định nghĩa tách rời với lớp *Service* và *Task*. Các thể hiện của các lớp này được coi như không phải là con người hoặc tổ chức. **HumanActor** cụ thể không được định nghĩa tách rời với lớp **System** vì thực tế trong nhiều trường hợp, một tổ chức chỉ là một loại hệ thống cụ thể. Tuy nhiên, không có định nghĩa cụ thể về lớp giao cắt này.

9.3. HumanActor – Các ví dụ

9.3.1. Đặc tính uses and usedBy áp dụng cho HumanActor

Theo định hướng, một tác nhân con người có thể tự sử dụng các dịch vụ, các hệ thống và các tác nhân con người khác. Ngoài ra, một tác nhân con người đôi khi cũng có thể được sử dụng bởi một tác nhân

con người khác hoặc bởi một hệ thống (được coi là một phần tử trong hệ thống hoặc như một tác nhân con người trong một quy trình).

9.3.2. Đặc tính represents và representedBy áp dụng cho HumanActor

Như đã đề cập trong phần giới thiệu của khoản này, các tác nhân con người được coi là một phần thuộc các hệ thống thể hiện kiến trúc hướng dịch vụ. Tuy nhiên, trong nhiều trường hợp, khi là một phần tử thuộc hệ thống SOA, sẽ không thảo luận về con người hay tổ chức cụ thể, thay vào đó là đại diện có tính trừu tượng của chúng tham gia vào các quy trình, cung cấp các dịch vụ,… Nói một cách khác, chỉ đề cập đến các phần tử đại diện cho các tác nhân con người.

Ví dụ, một nhà môi giới (một thể hiện của **HumanActor**) có thể đại diện cho người bán hàng (một thể hiện của **HumanActor**) không muốn tiết lộ danh tính, một vai trò (một thể hiện của **Element**) có thể đại diện (khía cạnh vai trò của) cho nhiều thể hiện của **HumanActor**, và một đơn vị tổ chức (một thể hiện của **HumanActor**) có thể đại diện cho rất nhiều người (tất cả các thể hiện của **HumanActor**) thuộc đơn vị đó.

Nên nhớ rằng “lớp vai trò” không được định nghĩa trước, do đó, sử dụng **Element** với đặc tính **represents** là cách tiếp cận tổng quát hơn và không giới hạn khả năng để xác định các hệ thống dựa trên vai trò. Đối với mục đích triển khai thực tế, cách đơn giản là “lớp thành phần về vai trò” của **Element**, lớp thành phần này không được định nghĩa rõ ràng.

9.3.3. Ví dụ về tổ chức

Tiếp tục ví dụ về tổ chức từ phần 8.8.1, P1 (*John*), P2 (*Jack*), P3 (*Joe*) và P4 (*Mary*) có thể được thể hiện như các thể hiện của **Element** và thực tế là các thể hiện của HumanActor. Tất cả O1 (*CarWashBusiness*), O3 (*CarWash*) và O4 (*Administration*) cũng có thể được biểu diễn như (tổ chức) các tác nhân con người theo quan điểm hành động tại cùng một thời điểm mà chúng là các hệ thống theo quan điểm tập hợp/hợp thành (collection/composition).

9.3.4. Ví dụ về rửa xe

Tham khảo Phụ lục A về hoàn thiện khía cạnh tổ chức của ví dụ rửa xe

9.4. Lớp Task

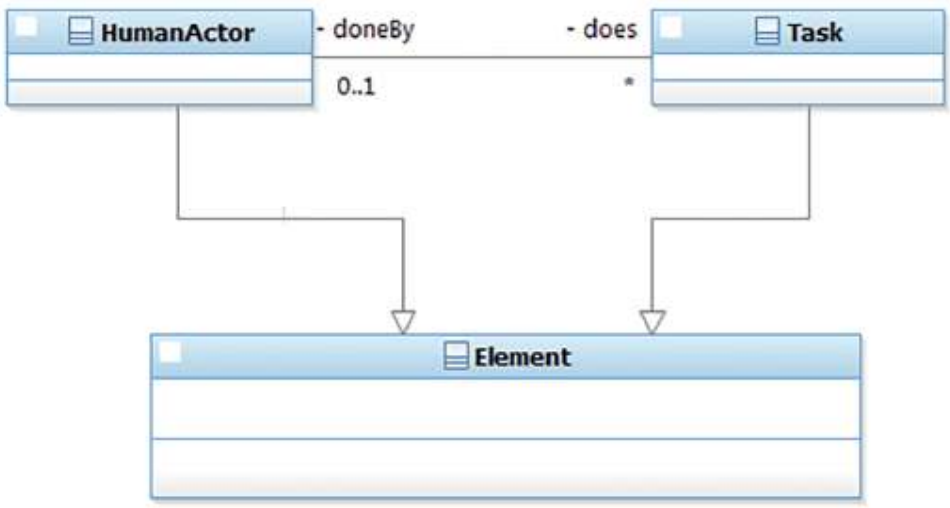
```
<owl:Class rdf:about="#Task">
   <owl:disjointWith>
    <owl:Class rdf:about="#System"/>
   </owl:disjointWith>
   <owl:disjointWith>
    <owl:Class rdf:about="#HumanActor"/>
   </owl:disjointWith>
   <owl:disjointWith>
    <owl:Class rdf:about="#Service"/>
```

```
   </owl:disjointWith>
   <rdfs:subClassOf>
    <owl:Class rdf:about="#Element"/>
   </rdfs:subClassOf>
</owl:Class>
```

Một *task* (tác vụ) là một hành động cơ bản để hoàn thành một kết quả xác định. Các tác vụ được làm bởi người hoặc tổ chức, cụ thể là bởi các thể hiện của **HumanActor**.

Ký hiệu mô hình hóa quy trình nghiệp vụ (BPMN) 2.0 định nghĩa tác vụ như sau: "Một Task là một Activity (hoạt động) cơ bản nhất trong một luồng Process (quy trình) (tham khảo tài liệu [4]). Một Tác vụ được sử dụng khi công việc trong Quy trình không thể chia nhỏ ở mức chi tiết hơn nữa. Tổng quát, người dùng cuối (end-user) và/hoặc các ứng dụng được sử dụng để thực hiện Tác vụ khi nó được thực thi". Với mục đích của bản thể luận, cần bổ sung độ chính xác bằng cách phân biệt ký hiệu làm (doing) với ký hiệu thực hiện (performing). Các Tác vụ được (tùy chọn) làm bởi các tác nhân con người, ngoài ra (khi là thể hiện của Element), các tác vụ có thể sử dụng các dịch vụ được thực hiện bởi các cấu phần công nghệ (xem chi tiết tại phần 10.3 và ví dụ trong Phụ lục A).

Khái niệm về task được thể hiện bởi lớp **Task** OWL, minh họa trong Hình 6.



Hình 6 – Lớp Task

Task được định nghĩa là tách rời với các lớp *System*, *Service* và *HumanActor*. Các thể hiện của các lớp này không được coi là các hành động cơ bản.

9.5. Đặc tính does và doneBy

```
<owl:ObjectProperty rdf:about="#doneBy">
   <rdfs:domain rdf:resource="#Task"/>
   <rdfs:range rdf:resource="#HumanActor"/>
</owl:ObjectProperty>
```

```
<owl:ObjectProperty rdf:about="#does">
   <owl:inverseOf>
    <owl:ObjectProperty rdf:about="#doneBy"/>
   </owl:inverseOf>
</owl:ObjectProperty>

<owl:Class rdf:about="#Task">
   <rdfs:subClassOf>
    <owl:Restriction>
     <owl:onProperty>
      <owl:ObjectProperty rdf:about="#doneBy"/>
     </owl:onProperty>
     <owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
>0</owl:minCardinality>
    </owl:Restriction>
   </rdfs:subClassOf>
   <rdfs:subClassOf>
    <owl:Restriction>
     <owl:maxCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
>1</owl:maxCardinality>
     <owl:onProperty>
      <owl:ObjectProperty rdf:about="#doneBy"/>
     </owl:onProperty>
    </owl:Restriction>
   </rdfs:subClassOf>
</owl:Class>
```

Các tác vụ bản chất là ý định được làm bởi con người hoặc tổ chức. Nếu nghĩ rằng tác vụ là những sự vật thực tế được làm, thì điều cốt lõi là từng thể hiện của **Task** sẽ được làm chủ yếu bởi một thể hiện của **HumanActor**. Do tính cơ bản các thể hiện của **Task**, nên được loại trừ khi thể hiện đó được làm bởi nhiều thể hiện Human**Actor**. Điều cốt lõi sẽ không tồn tại nếu một người nào đó lựa chọn không thể hiện tất cả các tác nhân con người có thể có. Mặt khác, thể hiện **HumanActor** giống nhau có thể (theo thời gian) dễ dàng thực hiện nhiều thể hiện **Task**. Đặc tính **does** và **doneBy**, bao gồm cả quan hệ giữa tác nhân con người và các tác vụ mà nó thực hiện.

9.6. Task – Các ví dụ

9.6.1. Đặc tính **uses** và **usedBy** áp dụng cho **Task**

Theo định hướng, trường hợp phổ biến nhất về một tác vụ sử dụng một phần tử khác là khi một tác vụ tự động (trong một quy trình được điều phối; tham khảo Khoản 11 về định nghĩa *process* (quy trình) và *orchestration* (điều phối)) sử dụng một dịch vụ như việc thực hiện của nó. Ngoài ra, một tác vụ đôi khi có thể được sử dụng bởi một hệ thống (như là một phần tử trong hệ thống đó, ví dụ một tác vụ trong một quy trình)

9.6.2. Đặc tính represents và representedBy áp dụng cho Task

Như đã đề cập trong phần giới thiệu về Khoản này, các tác vụ về bản chất là thành phần thuộc các hệ thống SOA. Tuy nhiên, trong nhiều trường hợp, khi một phần tử của hệ thống SOA, sẽ không đề cập đến việc làm thực tế, thay vào đó là thể hiện trừu tượng của nó được sử dụng như một phần tử thuộc các hệ thống, các quy trình,… Nói cách khác, ở đây, chỉ thảo luận các phần tử đại diện cho tác vụ.

Một ví dụ đơn giản, một hoạt động có tính trừu tượng trong một mô hình quy trình (liên quan đến một vai trò) có thể đại diện cho một tác vụ cụ thể (được làm bởi một người giữ vai trò đó). Lưu ý rằng do tính cơ bản của một tác vụ nên nó không có ý nghĩa khi nói về việc nhiều phần tử đại diện cho các khía cạnh khác nhau của nó.

9.6.3. Ví dụ về tổ chức

Tiếp tục ví dụ về tổ chức trong phần 8.8.1, chúng ta thể hiện các tác vụ được làm bởi các tác nhân (con người) P1, P2, P3 và P4 và cách thức thể hiện các tác vụ đó là các phần tử trong các hệ thống lớn hơn mô tả các sự vật như các quy trình về tổ chức. Khoản 11 sẽ chính thức liên quan đến khái niệm *composition* (hợp thành), bao gồm các việc đưa ra khái niệm về một *process* như một loại *compostion* cụ thể.

9.6.4. Ví dụ về rửa xe

Là một thành phần quan trọng của hệ thống rửa xe, John và Jack thực hiện các tác vụ thủ công nhằm rửa một chiếc xe ô tô.

- *Jack* và *John* là các thể hiện của **HumanActor**.

- *WashWindow* là một thể hiện của **Task** và được làm bởi *John*.

- *PushWashButton* là một thể hiện của **Task** và được làm bởi *Jack*.

10. Service, ServiceContract và ServiceInterface

10.1. Tổng quan

Service (dịch vụ) là một khái niệm cốt lõi khác của bản thể luận này. Đây là một khái niệm cơ bản đối với SOA và luôn luôn được sử dụng trong thực tế khi mô tả hoặc kỹ thuật phần mềm các hệ thống SOA, tuy nhiên, không dễ để đưa ra định nghĩa một cách chính thức. Bản thể luận được dựa trên định nghĩa về *service* sau đây:

Một service là một "biểu diễn logic của một tập các hoạt động có đầu ra cụ thể, có tính khép kín (self-contained), có thể bao gồm các dịch vụ khác và là một "hộp đen" đối với người sử dụng dịch vụ.

Định nghĩa này khá tương đồng với định nghĩa trước đó về thuật ngữ này trong Phần Kiến trúc tham chiếu SOA, ISO/IEC 18384-1.

Hoạt động trong định nghĩa service được sử dụng theo nghĩa chung, không mang ý nghĩa cụ thể với quy trình nào. Bản thể luận bỏ qua "nghiệp vụ như là một thành phần mang tính bản chất trong định nghĩa service. Lý do là quan niệm về nghiệp vụ liên quan đến quan điểm của một người, ví dụ, một quan niệm về CNTT của một người là một quan niệm về nghiệp vụ của người khác (nghiệp vụ của CNTT). Service được định nghĩa bởi bản thể luận này không thể biết được liệu khái niệm đó được áp dụng đối với quan niệm về miền nghiệp vụ hay quan niệm về miền CNTT.

Một số định nghĩa khác về thuật ngữ service trong SOA như sau:

- *"Một cơ chế cho phép truy cập đến một hoặc nhiều khả năng, trong đó, việc truy cập được cung cấp sử dụng một giao diện đã được quy định trước và được thực hiện phù hợp với các ràng buộc và chính sách bởi mô tả dịch vụ.* (Tham khảo Tài liệu [3]).

- *"Một khả năng được cung cấp bởi một hoặc nhiều thực thể (entity) đến các thực thể khác sử dụng các "thuật ngữ và điều kiện" và các giao diện đã được định nghĩa trước.* (Tham khảo Tài liệu [9]).

Trong mức độ chính xác thông thường của ngôn ngữ tiếng Anh, các định nghĩa này không mâu thuẫn; chúng đều nhấn mạnh đến các khía cạnh khác nhau của cùng một nội dung. Tất cả 3 định nghĩa đều thuộc về SOA và giải thích cụ thể được thuật ngữ dịch vụ theo ngôn ngữ tiếng Anh chung.

Khoản này mô tả các lớp sau đây của bản thể luận:

- **Service**;

- **ServiceContract**;

- **ServiceInterface**;

- **InformationType**.

Ngoài ra, khoản còn xác định các đặc tính sau:

- **performs** (thực hiện) và **performedBy** (được thực hiện bởi);

- **hasContract** (có hợp đồng) và **isContractFor** (là hợp đồng cho);

- **involvesParty** (gồm các bên tham gia) và **isPartyTo** (là bên tham gia);

- **specifies** (xác định) và **isSpecifiedBy** (được xác định bởi);

- **hasInterface** (có giao diện) và **isInterfaceOf** (là giao diện của);

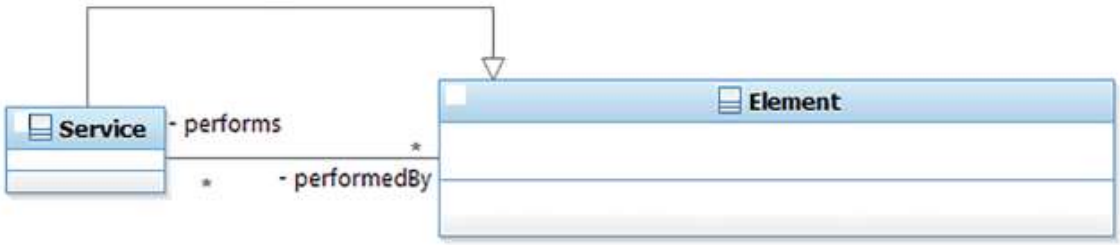
- **hasInput** (có đầu vào) và **isInputAt** (là đầu vào tại);

- **hasOutput** (có kết quả) và **isOutputAt** (là kết quả tại).

10.2. Lớp Service

```
<owl:Class rdf:about="#Service">
   <owl:disjointWith>
    <owl:Class rdf:about="#System"/>
   </owl:disjointWith>
   <owl:disjointWith>
    <owl:Class rdf:about="#Task"/>
   </owl:disjointWith>
   <owl:disjointWith>
    <owl:Class rdf:about="#HumanActor"/>
   </owl:disjointWith>
   <rdfs:subClassOf>
    <owl:Class rdf:about="#Element"/>
   </rdfs:subClassOf>
</owl:Class>
```

Một *service* (dịch vụ) là biểu diễn trừu tượng của một tập các hoạt động đã xác định được kết quả, có tính khép kín (self-contained), có thể bao gồm các dịch vụ khác và là một "hộp đen" đối với người sử dụng dịch vụ. Khái niệm về service được thể hiện bởi lớp **Service** OWL, minh họa trong Hình 7.



Hình 7 – Lớp Service

Trong bản thể luận về SOA này, chỉ xem xét các dịch vụ dựa trên SOA. Các miền khác, chẳng hạn như quản lý dịch vụ tích hợp, có thể có các dịch vụ không phải dựa trên SOA sẽ nằm ngoài phạm vi của bản thể luận về SOA này.

Service được định nghĩa tách rời với các lớp *System*, *Task* và *HumanActor*. Các thể hiện của các lớp này bản thân không được coi là các dịch vụ ngay cả khi chúng có thể cung cấp các khả năng như các dịch vụ.

10.3. Đặc tính performs và performedBy

```
<owl:ObjectProperty rdf:about="#performs">
   <rdfs:domain rdf:resource="#Element"/>
   <rdfs:range rdf:resource="#Service"/>
</owl:ObjectProperty>
```

```
<owl:ObjectProperty rdf:about="#performedBy">
   <owl:inverseOf>
    <owl:ObjectProperty rdf:about="#performs"/>
   </owl:inverseOf>
</owl:ObjectProperty>
```

Do bản thân một dịch vụ chỉ là biểu diễn logic, nên bất kỳ dịch vụ nào cũng có thể *performedBy* (được thực hiện bởi) bởi một vật. Vật này *performs* (thực hiện) một dịch vụ mà những người tương tác với nó không thể nhìn thấy rõ được, tính không rõ ràng là bản chất của lớp **Element**. Khái niệm này được thể hiện bởi đặc tính *performs* và *performedBy* như minh họa trong Lớp dịch vụ ở Hình 7.

Điều này cũng có thể thấy trong thực tế, khi các dịch vụ có thể được thực hiện bởi các phần tử của các loại khác nhau thay vì các hệ thống. Điều này bao gồm các phần tử như thành phần phần mềm, các tác nhân con người và các tác vụ.

Lưu ý rằng một thể hiện của **Service** có thể được thực hiện bởi nhiều thể hiện của **Element** khác nhau. Chỉ cần dịch vụ được thực hiện là một, và một người quan sát bên ngoài không thể nhận thấy sự khác nhau (về các quy định theo hợp đồng, các thỏa thuận mức độ dịch vụ SLA,…tham khảo định nghĩa về lớp *ServiceContract* trong 10.6). Ngược lại, một thể hiện bất kỳ về **Element** có thể không thực hiện hoặc thực hiện một hoặc nhiều dịch vụ.

Trong khi một dịch vụ có thể được thực hiện bởi các phần tử khác nhau, bản thân dịch vụ (do là biểu diễn logic) không thực hiện các dịch vụ khác. Tham khảo Ví dụ hợp thành dịch vụ đơn giản (11.7.1) về một ví dụ cách thức biểu diễn hợp thành dịch vụ chính thức trong bản thể luận.

10.4. Service Consumers và Service Providers

Các thuật ngữ được sử dụng trong SOA thường bao gồm các quan niệm về Consumer (Nhà sử dụng dịch vụ) và Service Providers (Nhà cung cấp dịch vụ). Có hai thách thức với thuật ngữ này:

- Nó không phân biệt giữa mặt quy định về hợp đồng của việc sử dụng/cung cấp với mặt tương tác của việc sử dụng/cung cấp. Một quy định về hợp đồng không cần thiết phải chuyển thành sự phù thuộc về tương tác nếu không có lý do nào khác vì việc thực hiện quy định hợp đồng đó có thể được bắt nguồn từ một bên thứ ba.

- Sử dụng hoặc cung cấp một dịch vụ chỉ có ý nghĩa trong ngữ cảnh, gồm ngữ cảnh hợp đồng hoặc ngữ cảnh tương tác. Các thuật ngữ này, do đó, không phù hợp với việc tạo ra các tuyên bố về các phần tử và các dịch vụ một cách độc lập.

Đây là những lý do tại sao bản thể luận lựa chọn không thông qua các khái niệm về sử dụng và cung cấp, thay vào đó cho phép sử dụng các thuật ngữ sử dụng và cung cấp với các quy định về hợp đồng và/hoặc các quy tắc tương tác mô tả bởi các hợp đồng dịch vụ; tham khảo định nghĩa lớp *ServiceContract* trong phần 10.6. Theo cách thức đơn giản nhất, ngoài ngữ cảnh về một hợp đồng dịch vụ chính thức,

khía cạnh tương tác giữa các dịch vụ sử dụng và cung cấp, thậm chí có thể được hiểu đơn giản bằng cách nói rằng một phần tử sử dụng một dịch vụ hoặc một phần tử thực hiện (hay cung cấp) một dịch vụ; tham khảo các ví dụ trong phần 10.5.

10.5. Service - Các ví dụ

10.5.1. Đặc tính uses và usedBy áp dụng cho Service

Theo định hướng, không thật sự có ý nghĩa khi nói về một dịch vụ sử dụng một phần tử khác. Trong khi vật thực hiện dịch vụ đó có thể bao gồm cả việc sử dụng các phần tử khác (và chắc chắn thuộc trường hợp của *Service Composition*), bản thân dịch vụ đó (khi là biểu diễn logic đơn thuần) không sử dụng các phần tử khác.

Theo định hướng khác, phổ biến nhất của tất cả các tương tác được tìm thấy trong SOA: quan niệm rằng một phần tử sử dụng một dịch vụ bằng việc tương tác với nó. Lưu ý rằng theo quan điểm vận hành, thì tương tác này thực chất là việc đạt được một điều gì đó ngoài dịch vụ bằng cách thực hiện các bước điển hình như sau:

- Chọn dịch vụ để tương tác (điều này không thể biết trước cho dù nó được làm tại thời gian chạy hoặc tĩnh tại giai đoạn thiết kế và/hoặc xây dựng).

- Chọn một phần tử thực hiện dịch vụ đó [trong môi trường SOA, điều này thường được thực hiện “bên trong” một trục dịch vụ tổng thể (ESB)];

- Tương tác với dịch vụ phần tử đã lựa chọn (thường cũng qua ESB).

10.5.2. Đặc tính represents và representedBy áp dụng cho Service

Khái niệm thành phần trung gian dịch vụ (service mediations), các proxy dịch vụ, trục dịch vụ tổng thể (ESB),...là những khái niệm cơ bản đối với những người triển khai thực tế khi mô tả và thực hiện các khía cạnh về vận hành các hệ thống SOA. Theo quan điểm của bản thể luận, tất cả những khái niệm này có thể được nắm bắt bởi một phần tử khác đại diện cho dịch vụ đó, mức độ gián tiếp có tính quan trọng khi nó không muốn kết nối về hoạt động đến một điểm dịch vụ (service endpoint) cụ thể, thay vào đó là duy trình kết nối lỏng và khả năng chuyển đổi linh hoạt khi cần thiết. Lưu ý rằng, bằng việc tận dùng đặc tính *represents* và *representedBy* theo cách này thì mô hình (pattern) tương tác vận hành tương đối phức tạp mô tả trong phần 9.6.2 (chọn dịch vụ, chọn một phần tử thực hiện dịch vụ và tương tác với phần tử lựa chọn) sẽ được đóng gói.

Trong khi một dịch vụ được đại diện bởi một sự vật là khá bình thường, nhưng lại rất khó khăn để tưởng tượng dịch vụ có thể biểu diễn cái gì. Ở một mức độ thực tế nào đó, một dịch vụ biểu diễn một sự biểu hiện bất kỳ của chính nó, chỉ có đặc tính *performs* và *performedBy* được lựa chọn để mô tả điều này thay vì sử dụng các đặc tính *represents* và *representedBy*. Kết quả là các ứng dụng thực tế của bản thể luận có thể có các dịch vụ biểu diễn sự vật bất kỳ và không mong muốn.

10.5.3. Minh họa sự khác biệt giữa làm một Task và thực hiện một Service

Sự khác biệt giữa một tác nhân con người làm một tác vụ và một phần tử (công nghệ, tác nhân con người,…) thực hiện một dịch vụ là rất quan trọng. Tác nhân con người thực hiện tác vụ có trách nhiệm làm tác vụ đó, tuy nhiên, trong thực tế, nhiều trường hợp có thể sử dụng một dịch vụ để đạt được kết quả đó.

- *John* là một thể hiện của **HumanActor**.

- *WashWindows* là một thể hiện của **Task** được làm bởi *John*.

- *SoapWater* là một thể hiện của **Service**.

- *WaterTap* là một thể hiện của *Element*.

- *WaterTap* thực hiện *SoapWater*.

- *John* sử dụng *SoapWater* (để làm *WashWindows*).

Lưu ý rằng *SoapWater* rõ ràng không trực tiếp làm *WashWindow* cũng không phải *WaterTap* làm *WashWindows*.

10.5.4. Ví dụ về rửa xe

Joe cung cấp 2 dịch vụ khác nhau cho khách hàng: dịch vụ rửa xe cơ bản (basic wash) và dịch vụ rửa xe chất lượng cao (gold wash). Điều này có thể được thể hiện trong bản thể luận theo cách sau (tập con đối với thành phần tương ứng với 2 dịch vụ này):

- *GoldWash* là một thể hiện của **Service**.

- *BasicWash* là một thể hiện của **Service**.

- *CarWash* thực hiện cả *BasicWash* và *GoldWash*.

- *WashManager* thể hiện cả *BasicWash* và *GoldWash* (có nghĩa là điểm tương tác khi khách hàng yêu cầu dịch vụ cũng như trả tiền).

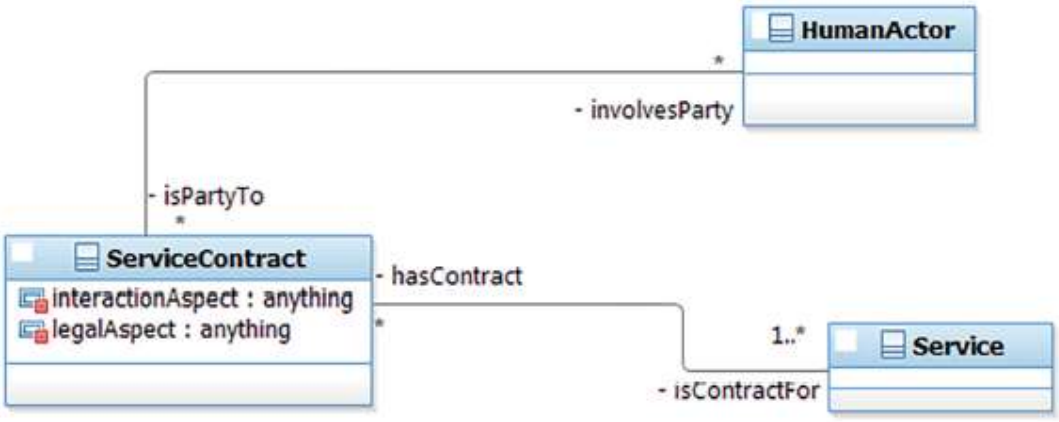
Lưu ý rằng việc sử dụng có mục đích *WashManager* đại diện cho cả 2 dịch vụ. Điều này phụ thuộc vào quyết định của Joe khi khách hàng rửa xe không tương tác trực tiếp với bộ phận rửa xe, thay vào đó là tương tác với người (tác nhân con người) có vai trò là quản lý việc rửa xe.

10.6. Lớp ServiceContract

```
<owl:Class rdf:about="#ServiceContract">
  <owl:disjointWith>
    <owl:Class rdf:about="#HumanActor"/>
```

```
   </owl:disjointWith>
   <owl:disjointWith>
    <owl:Class rdf:about="#Task"/>
   </owl:disjointWith>
</owl:Class>
```

Trong nhiều trường hợp, cần phải có các thỏa thuận cụ thể để định nghĩa cách thức sử dụng một dịch vụ. Điều này có thể là do mong muốn quy định về việc sử dụng phải như vậy hoặc chỉ đơn giản là vì dịch vụ sẽ không hoạt động đúng chức năng trừ khi các tương tác với nó được thực hiện theo một trình tự nhất định. Một *service contract* (hợp đồng dịch vụ) định nghĩa các thuật ngữ, các điều kiện và các nguyên tắc tương tác mà người tham gia đồng ý (trực tiếp hoặc gián tiếp). Một *service contract* có tính ràng buộc với tất cả những người tham gia tương tác, bao gồm cả chính dịch vụ đó và phần tử cung cấp dịch vụ cho một tương tác cụ thể. Khái niệm về *service contract* được thể hiện bởi lớp **ServiceContract** OWL, được minh họa trong Hình 8.



Hình 8 – Lớp ServiceContract

10.7. Đặc tính interactionAspect và legalAspect Datatype

```
<owl:DatatypeProperty rdf:about="#interactionAspect">
   <rdfs:domain rdf:resource="#ServiceContract"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:about="#legalAspect">
   <rdfs:domain rdf:resource="#ServiceContract"/>
</owl:DatatypeProperty>

<owl:Class rdf:about="#ServiceContract">
   <rdfs:subClassOf>
    <owl:Restriction>
     <owl:onProperty>
      <owl:DatatypeProperty rdf:about="#legalAspect"/>
     </owl:onProperty>
     <owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
     >1</owl:minCardinality>
```

```
    </owl:Restriction>
   </rdfs:subClassOf>
   <rdfs:subClassOf>
    <owl:Restriction>
     <owl:maxCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
     >1</owl:maxCardinality>
     <owl:onProperty>
      <owl:DatatypeProperty rdf:about="#legalAspect"/>
     </owl:onProperty>
    </owl:Restriction>
</rdfs:subClassOf>
   <rdfs:subClassOf>
    <owl:Restriction>
     <owl:onProperty>
      <owl:DatatypeProperty rdf:about="#interactionAspect"/>
     </owl:onProperty>
     <owl:maxCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
     >1</owl:maxCardinality>
    </owl:Restriction>
   </rdfs:subClassOf>
   <rdfs:subClassOf>
    <owl:Restriction>
     <owl:onProperty>
      <owl:DatatypeProperty rdf:about="#interactionAspect"/>
     </owl:onProperty>
     <owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
     >1</owl:minCardinality>
    </owl:Restriction>
   </rdfs:subClassOf>
</owl:Class>
```

Hợp đồng dịch vụ quy định rõ ràng cả khía cạnh về tương tác (xem đặc tính *hasContract* và *isContractFor*) và khía cạnh về thỏa thuận pháp lý (xem đặc tính *involvedParty* và *isPartyTo*) khi sử dụng một dịch vụ. Hai khía cạnh này được bao gồm khi định nghĩa đặc tính **interactionAspect** và đặc tính kiểu dữ liệu **legalAspect** trong lớp **ServiceContract**. Lưu ý rằng thuộc tính thứ hai, khía cạnh thỏa thuận pháp lý, bao gồm các khái niệm như thỏa thuận mức độ dịch vụ (SLAs).

Nếu muốn, có thể coi như một quy ước kiến trúc khi phân chia tương tác và các khía cạnh pháp lý thành hai hợp đồng dịch vụ khác nhau. Các lựa chọn như vậy sẽ áp dụng cho ứng dụng bất kỳ sử dụng bản thể luận này.

10.8. Đặc tính hasContract và isContractFor

```
<owl:ObjectProperty rdf:about="#isContractFor">
   <rdfs:domain rdf:resource="#ServiceContract"/>
   <rdfs:range rdf:resource="#Service"/>
</owl:ObjectProperty>
```

```
<owl:ObjectProperty rdf:about="#hasContract">
   <owl:inverseOf>
    <owl:ObjectProperty rdf:about="#isContractFor"/>
   </owl:inverseOf>
</owl:ObjectProperty>

<owl:Class rdf:about="#ServiceContract">
   <rdfs:subClassOf>
    <owl:Restriction>
     <owl:onProperty>
      <owl:ObjectProperty rdf:about="#isContractFor"/>
     </owl:onProperty>
     <owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
     >1</owl:minCardinality>
    </owl:Restriction>
   </rdfs:subClassOf>
</owl:Class>
```

Đặc tính **hasContract** (có hợp đồng) và **isContractFor** (là hợp đồng cho) bao gồm quan niệm trừu tượng về một dịch vụ có một hợp đồng dịch vụ. Bất kỳ ai muốn sử dụng dịch vụ đều phải tuân theo các khía cạnh tương tác (như định nghĩa trong đặc tính kiểu dữ liệu **interactionAspect**) của một hợp đồng dịch vụ bất kỳ áp dụng cho tương tác đó. Theo cách đó, các khía cạnh tương tác của một hợp đồng dịch vụ là độc lập với ngữ cảnh; chúng bao gồm các cách thức được xác định trước và cơ bản khi sử dụng dịch vụ .

Theo định nghĩa, một hợp đồng dịch vụ bất kỳ phải thuộc về ít nhất một dịch vụ. Cũng có thể một hợp đồng dịch vụ thuộc về nhiều dịch vụ; trong trường hợp này, một nhóm các dịch vụ sẽ chia sẻ cũng một mô hình (pattern) tương tác hoặc một hợp đồng dịch vụ (về pháp lý – xem đặc tính *involvesParty* và *isPartyTo* trong phần 10.9) quy định việc cung cấp và sử dụng nhiều dịch vụ.

10.9. Đặc tính involvesParty và isPartyTo

```
<owl:ObjectProperty rdf:about="#isPartyTo">
   <rdfs:domain rdf:resource="#HumanActor"/>
   <rdfs:range rdf:resource="#ServiceContract"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="#involvesParty">
   <owl:inverseOf>
    <owl:ObjectProperty rdf:about="#isPartyTo"/>
   </owl:inverseOf>
</owl:ObjectProperty>
```

Ngoài các quy tắc và quy định bên trong áp dụng cho tương tác bất kỳ với một dịch vụ (khía cạnh tương tác của hợp đồng dịch vụ được nắm bắt trong đặc tính kiểu dữ liệu **interactionAsspect**), có thể là các thỏa thuận pháp lý bổ sung áp dụng cho một số tác nhân con người và cách sử dụng dịch vụ của họ.

Đặc tính **involvesParty** (bao gồm các bên tham gia) và đặc tính **isPartyTo** (là bên tham gia), thể hiện quan niệm trừu tượng về hợp đồng dịch vụ cụ thể hóa các quy định pháp lý giữa các tác nhân con người trong việc sử dụng một hoặc nhiều dịch vụ chịu tác động của hợp đồng dịch vụ.

Trong khi các đặc tính **involvesParty** và **isPartyTo** định nghĩa mối quan hệ với các tác nhân con người liên quan đến hợp đồng dịch vụ, các quy định pháp lý thực tế đối với các tác nhân con người này được xác định trong đặc tính kiểu dữ liệu **legalAspect** thuộc hợp đồng dịch vụ. Điều này bao gồm cả khả năng xác định ai là nhà cung cấp và ai là nhà sử dụng dịch vụ theo quan điểm pháp lý.

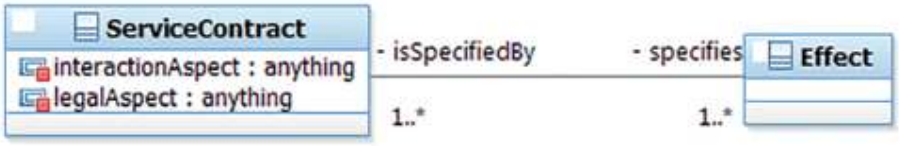
Tồn tại một mối quan hệ nhiều-nhiều giữa các hợp đồng dịch vụ với các tác nhân con người. Một tác nhân con người có thể là một bên tham gia không thuộc, hoặc thuộc một hay nhiều hợp đồng dịch vụ. Tương tự, một hợp đồng dịch vụ có thể không liên quan hoặc liên quan đến một hay nhiều tác nhân con người (không có trường hợp mà hợp đồng dịch vụ chỉ cụ thể hóa đặc tính kiểu dữ liệu **interactionAspect**). Nên nhớ rằng yếu tố rất quan trọng phải kể đến nguồn gốc hợp đồng khi mà tồn tại sự thỏa thuận về pháp lý giữa tác nhân con người A và tác nhân con người B (cả 2 đều là các bên tham gia thuộc hợp đồng dịch vụ), tuy nhiên, tác nhân B có nguồn gốc từ việc thực hiện dịch vụ với tác nhân con người C (có nghĩa là trong trường hợp này, tác nhân con người C thực hiện dịch vụ chứ không phải tác nhân con người B).

Đặc tính **involvesParty** cùng với đặc tính kiểu dữ liệu **legalAspect** thuộc **ServiceContract** không chỉ thể hiện các quy định tạm thời, chúng còn có khả năng thể hiện "tính bắt buộc ở thời điểm hiện tại", "đã có tính bắt buộc ở thời điểm đã qua" và "có thể bắt buộc tại thời điểm tương lai".

10.10. Lớp Effect

```
<owl:Class rdf:about="#Effect">
   <owl:disjointWith>
    <owl:Class rdf:about="#ServiceInterface"/>
   </owl:disjointWith>
</owl:Class>
```

Tương tác với một vật thực hiện một dịch vụ sẽ có những *effects* (tác động). Những tác động này bao gồm kết quả đầu ra của tương tác đó, và cách thức một dịch vụ (thông qua một phần tử thực hiện nó) cung cấp giá trị đến người sử dụng dịch vụ đó. Khái niệm *effect* được thể hiện bởi lớp **Effect** OWL, minh họa trong Hình 9.



Hình 9 – Lớp Effect

Lưu ý rằng lớp **Effect** thuần túy biểu diễn cách thức cung cấp kết quả hay giá trị đến người dùng có tương tác với một dịch vụ. Một tác động bên trong bất kỳ không được biểu diễn bởi lớp **Effect**.

Effect được định nghĩa tách rời với lớp *ServiceInterface*. (Lớp *ServiceInterface* được định nghĩa dưới đây thuộc phần này của tiêu chuẩn ISO/IEC 18384). Tương tác với một dịch vụ thông qua giao diện dịch vụ có thể có một kết quả đầu ra hoặc đưa ra một giá trị (một thể hiện của **Effect**) nhưng bản thân giao diện dịch vụ không thể tạo ra kết quả đầu ra hay giá trị đó.

10.11. Đặc tính specifies và isSpecifiedBy

```
<owl:ObjectProperty rdf:about="#specifies">
   <rdfs:domain rdf:resource="#ServiceContract"/>
   <rdfs:range rdf:resource="#Effect"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="#isSpecifiedBy">
   <owl:inverseOf>
    <owl:ObjectProperty rdf:about="#specifies"/>
   </owl:inverseOf>
</owl:ObjectProperty>

<owl:Class rdf: about="#Effect">
   <rdfs:subClassOf>
    <owl:Restriction>
     <owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
     >1</owl:minCardinality>
     <owl:onProperty>
      <owl:ObjectProperty rdf: about="#isSpecifiedBy"/>
     </owl:onProperty>
    </owl:Restriction>
   </rdfs:subClassOf>
</owl:Class>

<owl:Class rdf:about="#ServiceContract">
   <rdfs:subClassOf>
    <owl:Restriction>
     <owl:onProperty>
      <owl:ObjectProperty rdf: about="#specifies"/>
     </owl:onProperty>
     <owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
     >1</owl:minCardinality>
    </owl:Restriction>
   </rdfs:subClassOf>
</owl:Class>
```

Trong khi tự bản thân một dịch vụ sẽ có một tác động mỗi khi có một người dùng tương tác với nó, để thấy rõ tác động đối với một vật trong trường hợp cụ thể, tác động cần phải phải được xác định như một thành phần thuộc một hợp đồng dịch vụ. Đặc tính **specifies** (xác định) và **isSpecifiedBy** (được xác định

bởi) bao gồm quan niệm trừu tượng về một hợp đồng dịch vụ chỉ rõ tác động cụ thể như một thành phần của thỏa thuận đối với việc sử dụng một dịch vụ. Lưu ý rằng tác động cụ thể đó có thể ảnh hưởng đến cả đặc tính kiểu dữ liệu **interactionAspect** (đơn giản hóa việc xác định điều sẽ xảy ra khi tương tác với dịch vụ theo hợp đồng dịch vụ) và đặc tính kiểu dữ liệu **legalAspect** (xác định một tác động như đã có trong hợp đồng).

Bất kỳ một người dùng nào muốn có một tác động tương tác được đảm bảo với một dịch vụ cũng cần phải chắc chắn rằng tác động mong muốn đó được cụ thể hóa trong một hợp đồng dịch vụ áp dụng cho tương tác. Bằng định nghĩa, một hợp đồng dịch vụ bất kỳ xác định ít nhất một tác động. Theo một hướng khác, một tác động là một tác động của ít nhất một hợp đồng dịch vụ; điều này thể hiện rằng, thực tế, các tác động được xác định bởi hợp đồng dịch vụ đều được chính thức hóa (và không phải tất cả tác động thuộc về bản chất của tất cả dịch vụ).

10.12. ServiceContract – Các ví dụ

10.12.1. Các thỏa thuận mức độ dịch vụ

Một thỏa thuận mức độ dịch vụ (service-level agreement – SLA) về một dịch vụ được tổ chức A và B cùng chấp thuận. Rất quan trọng khi nhận ra rằng SLA phụ thuộc vào ngữ cảnh của các bên tham gia chấp nhận nó, liên quan đến ít nhất một “nhà sử dụng” và một “nhà cung cấp” hợp pháp. Điều này được thể hiện trong bản thể luận như sau:

- *A* và *B* là các thể hiện của **HumanActor**;

- *Service* là một thể hiện của **Service**;

- *ServiceContract* là một thể hiện của **ServiceContract**;

- *ServiceContract* là hợp đồng cho *Service*;

- *ServiceContract* bao gồm thành phần *A* ;

- *ServiceContract* bao gồm thành phần *B* ;

- Đặc tính kiểu dữ liệu **legalAspect** thuộc *ServiceContract* mô tả SLA.

10.12.2. Nguồn gốc dịch vụ

Tổ chức A và B cùng chấp thuận việc tổ chức B cung cấp một vài dịch vụ cho A, tuy nhiên, tổ chức B muốn thể hiện nguồn gốc cung cấp thật sự những dịch vụ đó cho một bên thứ ba là C. Điều này có thể được thể hiện trong bản thể luận như sau:

- *A, B* và *C* là các thể hiện của **HumanActor** ;

- *Service* là một thể hiện của **Service** ;

- *C* cung cấp *Service* ;

- *ServiceContract* là một thể hiện của **ServiceContract** ;

- *ServiceContract* là hợp đồng cho Service ;

- *ServiceContract* bao gồm thành phần *A* ;

- *ServiceContract* bao gồm thành phần *B* ;

- Đặc tính kiểu dữ liệu **legalAspect** thuộc ServiceContract mô tả quy định pháp lý của B khi cung cấp Service cho A.

10.12.3. Ví dụ về rửa xe

Tham khảo phu luc A về các khía cạnh **Service** và **ServiceContract** hoàn chỉnh của ví dụ rửa xe.

10.13. Lớp ServiceInterface (Giao diện dịch vụ)

```
<owl:Class rdf:about="#ServiceInterface">
   <owl:disjointWith>
    <owl:Class rdf: about="#Service"/>
   </owl:disjointWith>
   <owl:disjointWith>
    <owl:Class rdf: about="#ServiceContract"/>
   </owl:disjointWith>
   <owl:disjointWith>
    <owl:Class rdf: about="#Effect"/>
   </owl:disjointWith>
   <owl:disjointWith>
    <owl:Class rdf: about="#HumanActor"/>
   </owl:disjointWith>
   <owl:disjointWith>
    <owl:Class rdf: about="#Task"/>
   </owl:disjointWith>
</owl:Class>
```

Một đặc điểm rất quan trọng của các dịch vụ đó là chúng có giao diện đơn giản và được định nghĩa rõ ràng. Điều này hỗ trợ việc tương tác với chúng trở nên dễ dàng và cho phép các phần tử khác sử dụng chúng theo cách có cấu trúc. Một *service interface* định nghĩa cách mà các phần tử khác có thể tương tác và trao đổi thông tin với một dịch vụ. Khái niệm này được thể hiện bởi lớp **ServiceInterface**, minh họa trong Hình 10.

Hình 10 – Lớp ServiceInterface

Khái niệm giao diện nhìn chung được hiểu rõ bởi những người triển khai thực tế, với quan niệm rằng các giao diện xác định các tham số cho các thông tin đi vào và đi ra khỏi chúng khi được triệu gọi. Sự khác nhau giữa miền này với miền kia là ở bản chất cụ thể của cách thức mà giao diện được triệu gọi và cách thức thông tin được truyền qua lại. Các giao diện dịch vụ thường, nhưng không nhất thiết, dựa trên thông điệp (hỗ trợ kết nối lỏng). Ngoài ra, các giao diện luôn độc lập với việc triển khai dịch vụ (hỗ trợ kết nối lỏng và thành phần trung gian).

Theo quan điểm thiết kế, các giao diện có thể có nhiều các hoạt động chi tiết hoặc có thể gồm nhiều giao diện khác, tuy nhiên, phần tiêu chuẩn ISO/IEC 18384 này vẫn giữ ở mức khái niệm và không bao gồm những quan điểm thiết kế như vậy trong bản thể luận.

ServiceInterface được định nghĩa tách rời với lớp **Service**, **ServiceContract** và **Effect**. Các thể hiện của các lớp này được xem như không định nghĩa (bởi chúng) cách thức mà các phần tử khác tương tác và trao đổi thông tin với một dịch vụ. Lưu ý rằng tồn tại sự phối hợp tự nhiên giữa **ServiceInterface** và đặc tính kiểu dữ liệu **interactionAspect** thuộc **ServiceContract**, vì sau này định nghĩa một đa tương tác bất kỳ và/hoặc các ràng buộc tuần tự về cách thức sử dụng một dịch vụ thông qua tương tác với các giao diện dịch vụ của nó.

10.14. Đặc tính kiểu dữ liệu Constraints

```
<owl:DatatypeProperty rdf:about="#constraints">
   <rdfs:domain rdf:resource="#ServiceInterface"/>
</owl:DatatypeProperty>

<owl:Class rdf:about="#ServiceInterface">
   <rdfs:subClassOf>
    <owl:Restriction>
     <owl:onProperty>
      <owl:DatatypeProperty rdf: about="#constraints"/>
     </owl:onProperty>
     <owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
     >1</owl:minCardinality>
    </owl:Restriction>
   </rdfs:subClassOf>
   <rdfs:subClassOf>
    <owl:Restriction>
     <owl:onProperty>
      <owl:DatatypeProperty rdf:about="#constraints"/>
     </owl:onProperty>
```

```
     <owl:maxCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
     >1</owl:maxCardinality>
    </owl:Restriction>
   </rdfs:subClassOf>
</owl:Class>
```

Đặc tính kiểu dữ liệu **Constraints** (ràng buộc) trên **ServiceInterface** chứa quan niệm rằng có thể có những ràng buộc về sự tương tác, chẳng hạn như chỉ cho phép tham số được nhận một số giá trị nhất định. Phụ thuộc vào bản chất của dịch vụ và giao diện dịch vụ đã đề cập, những ràng buộc này có thể được định nghĩa chính thức hoặc không chính thức (trường hợp không chính thức liên quan đến một số lượng tối thiểu các kiểu dịch vụ thực tế).

10.15. Đặc tính hasInterface và IsInterfaceOf

```
<owl:ObjectProperty rdf:about="#hasInterface">
   <rdfs:domain rdf:resource="#Service"/>
   <rdfs:range rdf:resource="#ServiceInterface"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf: about="#isInterfaceOf">
   <owl:inverseOf>
    <owl:ObjectProperty rdf:about="#hasInterface"/>
   </owl:inverseOf>
</owl:ObjectProperty>

<owl:Class rdf:about="#Service">
   <rdfs:subClassOf>
    <owl:Restriction>
     <owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
     >1</owl:minCardinality>
     <owl:onProperty>
      <owl:ObjectProperty rdf: about="#hasInterface"/>
     </owl:onProperty>
    </owl:Restriction>
   </rdfs:subClassOf>
</owl:Class>
```

Đặc tính **hasInterface** (có giao diện) và **isInterfaceOf** (là giao diện của) thể hiện quan niệm trừu tượng về một dịch vụ có giao diện dịch vụ cụ thể.

Theo định hướng, một dịch vụ bất kỳ có ít nhất một giao diện dịch vụ; tất cả những điều khác sẽ không đúng với định nghĩa về một dịch vụ là đại diện cho một tập các hoạt động có kết quả đầu ra cụ thể và là một "hộp đen" với người sử dụng. Theo một hướng khác, có thể có nhiều giao diện dịch vụ không phải là giao diện của bất kỳ dịch vụ được định nghĩa trước. Ngoài ra, một giao diện dịch vụ cũng có thể là giao diện của nhiều dịch vụ. Điều này không có nghĩa rằng các dịch vụ này là giống nhau, hoặc thậm chí

chúng có cùng tác động, nó chỉ mang ý nghĩa rằng có thể tương tác với tất cả các dịch vụ này theo cách thức đã xác định trước bằng giao diện dịch vụ đã đề cập.

10.16. Lớp InformationType

```
<owl:Class rdf: about="#InformationType">
   <owl:disjointWith>
    <owl:Class rdf: about="#Effect"/>
   </owl:disjointWith>
   <owl:disjointWith>
    <owl:Class rdf: about="#ServiceContract"/>
   </owl:disjointWith>
</owl:Class>
```

Một giao diện dịch vụ có thể cho phép một phần tử khác cung cấp thông tin hoặc nhận thông tin từ một dịch vụ (khi nó sử dụng dịch vụ đó), đặc biệt các loại thông tin cung cấp hoặc được nhận. Khái niệm về *information type* (loại thông tin) được thể hiện bởi lớp **InformationType** OWL, minh họa trong Hình 11.

Hình 11 – Lớp InformationType

Trong một tương tác cụ thể bất kỳ thông qua một giao diện dịch vụ, loại thông tin trên giao diện đó được tạo ra bởi các mục thông tin, tuy nhiên, đối với bản thân giao diện dịch vụ, đây là các loại thông tin rất quan trọng.

Nên nhớ rằng đặc tính kiểu dữ liệu **constraints** trên **ServiceInterface**, nếu cần thiết, có thể được sử dụng để thể hiện các ràng buộc cho các giá trị được phép sử dụng đối với các loại thông tin nhất định.

10.17. Đặc tính hasInput và isInputAt

```
<owl:ObjectProperty rdf: about="#hasInput">
   <rdfs:domain rdf:resource="#ServiceInterface"/>
   <rdfs:range rdf:resource="#InformationType"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf: about="#isInputAt">
   <owl:inverseOf>
    <owl:ObjectProperty rdf: about="#hasInput"/>
   </owl:inverseOf>
</owl:ObjectProperty>
```

Đặc tính **hasInput** (có đầu vào) và **isInputAt** (là đầu vào tại) thể hiện quan niệm trừu tượng về một loại thông tin cụ thể được đưa ra khi tương tác với một dịch vụ thông qua một giao diện dịch vụ cụ thể.

Ghi nhớ rằng tồn tại mối quan hệ nhiều-nhiều giữa các giao diện dịch vụ với các loại thông tin đầu vào. Một loại thông tin được đưa ra có thể không phải hoặc là đầu vào tại rất nhiều giao diện dịch vụ. Tương tự như vậy, một giao diện dịch vụ nhất định có thể có nhiều loại thông tin như là đầu vào hoặc không có loại thông tin nào. Điều quan trọng phải nhận ra rằng một số dịch vụ có thể chỉ có đầu vào (kích hoạt một hành động không đồng bộ mà không có phản hồi xác định) và các dịch vụ khác có thể chỉ có đầu ra (các phần tử thực hiện các dịch vụ này thực thi một cách độc lập nhưng vẫn có thể đưa ra kết quả đầu ra được sử dụng bởi các phần tử khác).

10.18. Đặc tính hasOutput và isOutputAt

```
<owl:ObjectProperty rdf: about="#hasOutput">
   <rdfs:domain rdf:resource="#ServiceInterface"/>
   <rdfs:range rdf:resource="#InformationType"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf: about="#isOutputAt">
   <owl:inverseOf>
    <owl:ObjectProperty rdf: about="#hasOutput"/>
   </owl:inverseOf>
</owl:ObjectProperty>
```

Đặc tính **hasOutput** (có kết quả đầu ra) và **isOutputAt** (là kết quả đầu ra tại) thể hiện quan niệm trừu tượng về một loại thông tin cụ thể được nhận khi tương tác với một dịch vụ thông qua một giao diện dịch vụ.

Ghi nhớ rằng tồn tại quan hệ nhiều-nhiều giữa các giao diện dịch vụ với các loại thông tin đầu ra. Một loại thông tin được đưa ra có thể không phải hoặc là đầu ra tại rất nhiều giao diện dịch vụ. Tương tự, một giao diện dịch vụ được đưa ra có thể có nhiều loại thông tin như kết quả đầu ra hoặc không có loại thông tin nào. Điều quan trọng phải nhận ra rằng một số dịch vụ có thể chỉ có đầu vào (kích hoạt một hành động không đồng bộ mà không có phản hồi xác định) và các dịch vụ khác có thể chỉ có đầu ra (các phần tử thực hiện các dịch vụ này thực thi một cách độc lập nhưng có thể cung cấp kết quả đầu ra được sử dụng bởi các phần tử khác).

10.19. Các ví dụ

10.19.1. Trình tự tương tác

Một hợp đồng dịch vụ thể hiện rằng các giao diện của dịch vụ đó được sử dụng theo một trình tự nhất định.

- *Service* là một thể hiện của **Service**

- *ServiceContract* là một thể hiện của **ServiceContract**.

- *ServiceContract* là hợp đồng cho Service.

- *X* là một thể hiện của *ServiceInterface*.

- *X* là một giao diện của *Service*.

- *Y* là một thể hiện của *ServiceInterface*.

- *Y* là một giao diện của *Service*.

- Đặc tính kiểu dữ liệu **interactionAspect** về *ServiceContract* mô tả *X* được sử dụng trước khi *Y* có thể được sử dụng.

10.19.2. Ví dụ về rửa xe

Tham khảo Phụ lục A về khía cạnh **ServiceInterface** của ví dụ rửa xe hoàn chỉnh.

11. Hợp thành và các lớp thành phần

11.1. Tổng quan

Quan niệm về Hợp thành (Composition)) là khái niệm cốt lõi của Kiến trúc hướng dịch vụ SOA. Các dịch vụ có thể bao gồm các dịch vụ khác. Các quy trình bao gồm các tác nhân con người, tác vụ và có thể gồm cả các dịch vụ. Các chuyên gia triển khai SOA có kinh nghiệm một cách trực giác thường áp dụng hợp thành như một thành phần không thể tách rời về kiến trúc, thiết kế và thực hiện các hệ thống SOA; trong thực tế, bất kỳ một môi trường SOA tốt nào cũng có tính hợp thành theo cách thức các dịch vụ và các quy trình hỗ trợ các khả năng nghiệp vụ. Điểm khác biệt thể hiện giữa các chuyên gia triển khai thực tế là bản chất chính xác của hợp thành và mô hình hợp thành được áp dụng.

Khoản này mô tả các lớp sau đây của bản thể luận :

Composition (lớp thành phần của **System**)

ServiceComposition (lớp thành phần của **Composition**)

Process (lớp thành phần của **Composition**)

Ngoài ra, nó còn định nghĩa đặc tính kiểu dữ liệu sau đây:

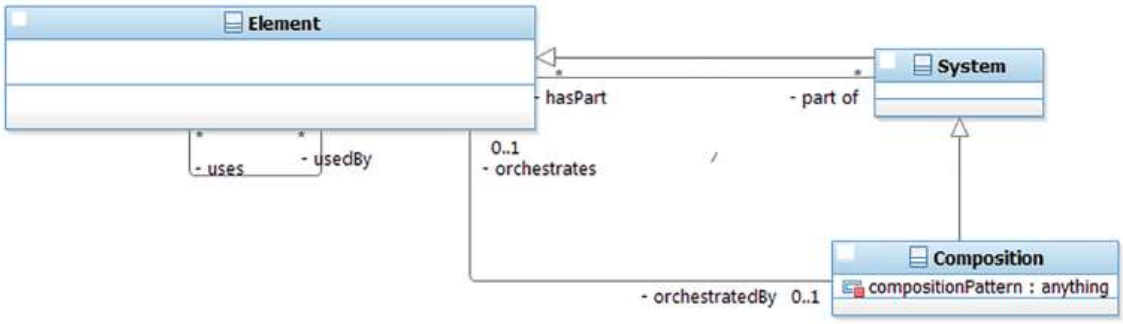
compositionPattern

11.2. Lớp Composition

```
<owl:Class rdf:about="#Composition">
```

```
    <rdfs:subClassOf>
     <owl:Class rdf: about="#System"/>
    </rdfs:subClassOf>
    <owl:disjointWith>
     <owl:Class rdf: about="#Task"/>
    </owl:disjointWith>
</owl:Class>
```

Một *composition* (hợp thành) là kết quả của việc tập hợp một nhóm các vật cho một mục đích cụ thể. Lưu ý, cần phân biệt theo mục đích về hành động tạo ra hợp thành với hợp thành kết quả như là một vật, và trong trường hợp này, sử dụng khái niệm *composition* thứ hai. Khái niệm *composition* được thể hiện bởi lớp **Composition** OWL, minh họa trong Hình 12.



Hình 12 – Lớp Composition

Về bản chất (cũng có thể), một tập các vật đơn giản có tổ chức, lớp **Composition** là một lớp thành phần của lớp **System**. Trong khi một hợp thành luôn luôn là một hệ thống, thì không nhất thiết một hệ thống phải là một hợp thành trong đó phải là một kết quả nào đó, ở đây, cần lưu ý sự khác biệt giữa một hệ thống tạo ra một kết quả và một hệ thống chính là một kết quả. Sự khác biệt có thể dễ thấy hơn giữa một hệ thống và một hợp thành chính là sự khác biệt thứ hai có liên quan đến một mô hình (pattern) hợp thành cụ thể cung cấp sự hợp thành (là một tổng thể) là kết quả khi mô hình hợp thành đó được áp dụng cho các phần tử sử dụng trong hợp thành. Hàm ý của điều này nghĩa là không có một thành viên riêng lẻ nào của một hợp thành có tính đại diện (là một phần tử) cho hợp thành đó như một tổng thể; nói một cách khác, bản thân hợp thành đó không phải là một trong số các vật được hợp thành. Mặt khác, *composition* trong thực tế là một khái niệm có tính đệ quy (là tất cả các lớp thành phần của **System**), là một hệ thống, một hợp thành cũng là một phần tử mang ý nghĩa rằng nó có thể được sử dụng bởi một hợp thành mức cao hơn.

Trong ngữ cảnh bản thể luận về SOA, chỉ có những hợp thành về chức năng thuộc miền SOA được xem xét một cách chi tiết. Lưu ý rằng một trường hợp được mô tả đầy đủ về **Composition** là do bản chất mối quan hệ *uses* với ít nhất một thể hiện của **Element**. (Không cần thiết phải có nhiều thể hiện hơn khi mô hình hợp thành áp dụng có thể, ví dụ, chỉ đơn giản là một chuyển đổi.) Một lần nữa (đối với **System**), điều quan trọng phải nhận ra rằng một hợp thành có thể sử dụng các phần tử bên ngoài nó.

Vì **Composition** là một lớp thành phần của **Element**, nên tất cả các hợp thành đều có ranh giới và không trong suốt đối với người bên ngoài (quan điểm hộp đen). Mô hình hợp thành lần lượt là điểm nhìn bên trong (quan điểm hộp trắng) về hợp thành. Ví dụ, đối với quan niệm về một hợp thành dịch vụ, điều này tương ứng với sự khác biệt giữa cách nhìn hợp thành dịch vụ như một phần tử cung cấp dịch vụ (mức cao) hay cách nhìn hợp thành dịch vụ như một cấu trúc gồm các dịch vụ (mức thấp).

11.3. Đặc tính kiểu dữ liệu compositionPattern

11.3.1. Tổng quan

```
<owl:DatatypeProperty rdf:about="#compositionPattern">
   <rdfs:domain rdf:resource="#Composition"/>
</owl:DatatypeProperty>

<owl:Class rdf:about="#Composition">
<rdfs:subClassOf>
   <owl:Restriction>
    <owl:maxCardinality
rdf:datatype="http://www.w3.org/2001/XMLSchema#int">1</owl:maxCardinality>
    <owl:onProperty>
     <owl:DatatypeProperty rdf: about="#compositionPattern"/>
    </owl:onProperty>
   </owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
    <owl:Restriction>
     <owl:onProperty>
      <owl:DatatypeProperty rdf: about="#compositionPattern"/>
     </owl:onProperty>
     <owl:minCardinality
rdf:datatype="http://www.w3.org/2001/XMLSchema#int">1</owl:minCardinality>
    </owl:Restriction>
   </rdfs:subClassOf>
</owl:Class>
```

Như thảo luận trong phần 11.2, một hợp thành bất kỳ có liên quan với một mô hình hợp thành cụ thể, mô hình đó mô tả cách thức trong đó một tập các phần tử được hợp thành với nhau thành một kết quả. Khái niệm về một mô hình hợp thành được thể hiện bởi đặc tính kiểu dữ liệu **compositionPattern** (mô hình hợp thành). Nên nhớ rằng mặc dù một vài loại mô hình hợp thành nhận được sự quan tâm đặc biệt trong SOA (tham khảo phần 11.3.2), nhưng đặc tính kiểu dữ liệu **compositionPattern** có thể có giá trị bất kỳ miễn là giá trị đó mô tả cách thức tập hợp các phần tử được sử dụng bởi hợp thành mà nó liên quan.

11.3.2. Mô hình hợp thành điều phối

Một loại mô hình hợp thành nhận được sự quan tâm đặc biệt trong SOA là *Orchestration* (Mô hình hợp thành điều phối). Trong mô hình điều phối (hợp thành mà mô hình là một thành phần điều phối), có một

phần tử cụ thể được sử dụng bởi hợp thành giám sát và điều khiển các phần tử khác. Lưu ý rằng phần tử điều khiển một điều phối theo định nghĩa khác biệt với chính thành phần điều phối (thể hiện của **Composition**).

Có thể xem một luồng công việc được điều phối và có tính khả thi như một ví dụ về một thành phần điều phối. Bản thân cấu trúc luồng công việc là một trong các phần tử được sử dụng trong hợp thành, tuy nhiên, điểm khác biệt so với hợp thành là bản thân hợp thành là một kết quả của việc áp dụng (thực thi) một luồng công việc theo các quy trình, tác nhân con người, dịch vụ,… được điều phối bởi cấu trúc luồng công việc.

Một ví dụ không liên quan đến CNTT như một người quản lý của một nhóm sửa chữa đường bộ. Nếu người quản lý lựa chọn sử dụng kiểm soát trực tiếp nhiệm vụ của từng người trong nhóm thì hợp thành kết quả sẽ là một thành phần điều phối (với người quản lý là giám đốc và là nhà cung cấp mô hình hợp thành). Lưu ý rằng trong các trường hợp khác, với một mô hình hợp thành khác, một nhóm sửa chữa đường cũng có thể hoạt động theo cách Cộng tác (collaboration) hay cách Thiết kế theo trình tự định sẵn (choreography) (tham khảo 11.3.3 và 11.3.4 về định nghĩa cộng tác và theo trình tự định sẵn).

Ví dụ cuối đã cho thấy việc sử dụng một mô hình hợp thành điều phối không đảm bảo rằng “không có điều gì có thể sai”. Thực tế, nó phụ thuộc vào khả năng xử lý các ngoại lệ của người chỉ huy việc điều phối.

11.3.3. Mô hình hợp thành thiết kế theo trình tự định sẵn

Một loại khác của mô hình hợp thành được đặc biệt quan tâm trong SOA là *Choreography* (mô hình hợp thành thiết kế theo trình tự định sẵn). Trong một thiết kế theo trình tự định sẵn (một hợp thành mà mô hình hợp thành là một thiết kế theo trình tự định sẵn), các phần tử được sử dụng bởi tương tác hợp thành theo kiểu không được điều khiển nhưng với từng thành viên chủ động hiểu rõ và theo dõi mô hình hành vi đã xác định trước cho toàn bộ hợp thành.

Có thể thấy mô hình quy trình như là một ví dụ về thiết kế theo trình tự định sẵn. Mô hình quy trình không điều khiển các phần tử trong nó nhưng lại thực hiện cung cấp một mô hình hành vi đã được xác định trước mà mỗi phần tử sẽ tuân theo khi “thực thi”.

11.3.4. Mô hình hợp thành cộng tác.

Một loại mô hình hợp thành thứ ba cũng nhận được quan tâm đặc biệt trong SOA là *Collaboration* (Mô hình hợp thành cộng tác). Trong cộng tác (một hợp thành mà mô hình hợp thành là cộng tác), các phần tử được sử dụng bởi tương tác hợp thành theo kiểu không được điều khiển, mỗi một phần tử thực hiện theo kế hoạch và mục đích riêng mà không tuân theo một mô hình hành vi đã được xác định trước. Mỗi phần tử đơn giản chỉ biết việc mà nó phải làm và thực hiện độc lập, khởi tạo tương tác với các thành viên khác của hợp thành theo ý thích của mình. Điều này có nghĩa là sẽ không có một “luồng” chung được xác định trước về sự hợp tác mặc dù có thể thấy một “luồng tương tác” theo thời gian chạy.

Một ví dụ điển hình về cộng tác là một cuộc họp. Không có một kịch bản nào về cách thức mà cuộc họp sẽ diễn ra mà chỉ có thể mô tả chuỗi tương tác đã xảy ra sau khi cuộc họp kết thúc.

11.4. Đặc tính orchestrates và orchestratedBy

```
<owl:ObjectProperty rdf:about="#orchestratedBy">
   <rdfs:domain rdf:resource="#Composition"/>
   <rdfs:range rdf:resource="#Element"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="#orchestrates">
   <owl:inverseOf>
    <owl:ObjectProperty rdf: about="#orchestratedBy"/>
   </owl:inverseOf>
</owl:ObjectProperty>

<owl:Class rdf:about="#Composition">
   <rdfs:subClassOf>
    <owl:Restriction>
     <owl:maxCardinality
rdf:datatype="http://www.w3.org/2001/XMLSchema#int">1</owl:maxCardinality>
     <owl:onProperty>
     <owl:ObjectProperty rdf: about="#orchestratedBy"/>
    </owl:onProperty>
   </owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
   <owl:Restriction>
    <owl:minCardinality
rdf:datatype="http://www.w3.org/2001/XMLSchema#int">0</owl:minCardinality>
    <owl:onProperty>
     <owl:ObjectProperty rdf: about="#orchestratedBy"/>
    </owl:onProperty>
   </owl:Restriction>
</rdfs:subClassOf>
</owl:Class>

<owl:Class rdf:about="#Element">
   <rdfs:subClassOf>
    <owl:Restriction>
     <owl:minCardinality
rdf:datatype="http://www.w3.org/2001/XMLSchema#int">0</owl:minCardinality>
     <owl:onProperty>
      <owl:ObjectProperty rdf: about="#orchestrates"/>
     </owl:onProperty>
    </owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
    <owl:Restriction>
```

```
     <owl:maxCardinality
rdf:datatype="http://www.w3.org/2001/XMLSchema#int">1</owl:maxCardinality>
     <owl:onProperty>
      <owl:ObjectProperty rdf:about="#orchestrates"/>
     </owl:onProperty>
    </owl:Restriction>
   </rdfs:subClassOf>
</owl:Class>
```

Một điều phối có một phần tử cụ thể thực hiện giám sát và điều khiển các phần tử khác sử dụng bởi hợp thành. Kiểu quan hệ như này có đủ tính quan trọng mà quan niệm trừu tượng thể hiện trong đặc tính **orchestrates** (điều phối) và **orchestratedBy** (được điều phối bởi).

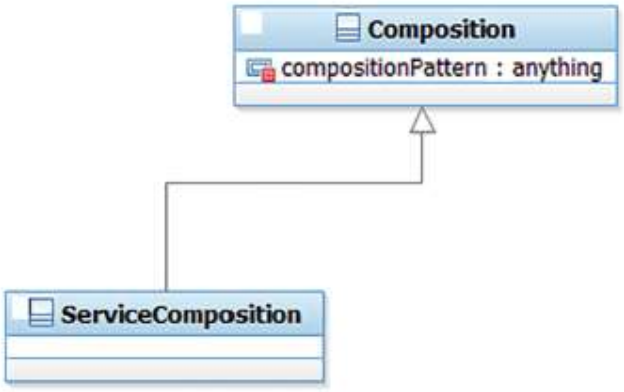
Theo định hướng, một hợp thành có thể có nhiều nhất một phần tử điều phối nó, và số lượng chỉ có thể là một nếu trong thực tế mô hình hợp thành là một điều phối. Theo một hướng khác, một phần tử có thể điều phối nhiều nhất một hợp thành mà có một điều phối như là mô hình hợp thành của nó.

Lưu ý rằng trong các ứng dụng thực tế của bản thể luận, mặc dù **Service** là một lớp thành phần của **Element** nhưng một dịch vụ (là biểu diễn logic thuần túy) không phải để điều phối một hợp thành.

11.5. Lớp ServiceComposition

```
<owl:Class rdf: about="#ServiceComposition">
   <rdfs:subClassOf>
    <owl:Class rdf: about="#Composition"/>
   </rdfs:subClassOf>
   <owl:disjointWith>
    <owl:Class rdf: about="#ServiceContract"/>
   </owl:disjointWith>
   <owl:disjointWith>
    <owl:Class rdf:about="#ServiceInterface"/>
   </owl:disjointWith>
</owl:Class>
```

Một khái niệm cốt lõi của SOA là quan niệm về *service composition* (hợp thành dịch vụ), kết quả của việc kết hợp một tập các dịch vụ để thực hiện một dịch vụ mới ở mức cao. Khái niệm về *service composition* được thể hiện bởi lớp **ServiceComposition** OWL, minh họa trong Hình 13.



Hình 13 – Lớp ServiceComposition

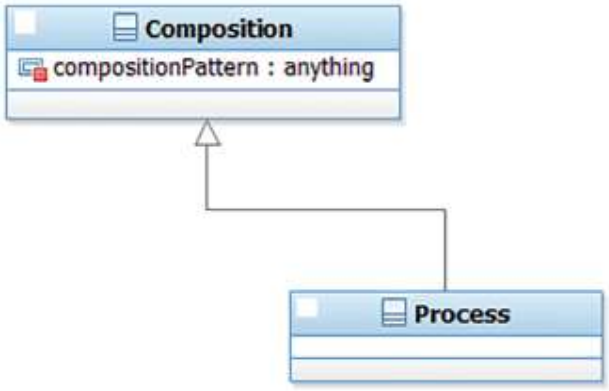
Do một *service composition* là kết quả của việc kết hợp một tập các dịch vụ với nhau nên **ServiceComposition** về bản chất là một lớp thành phần của **Composition**.

Một hợp thành dịch vụ có thể và thông thường sẽ bổ sung thêm tính logic (hoặc thậm chí là "mã") thông qua mô hình hợp thành (composition pattern). Lưu ý rằng một hợp thành dịch vụ bản thân nó không phải là dịch vụ mới mức cao hơn (do các lớp System và Service tách rời nhau); mà nó thực hiện (như một phần tử) dịch vụ mức cao hơn.

11.6. Lớp Process

```
<owl:Class rdf: about="#Process">
   <rdfs:subClassOf>
    <owl:Class rdf: about="#Composition"/>
   </rdfs:subClassOf>
   <owl:disjointWith>
    <owl:Class rdf: about="#ServiceContract"/>
   </owl:disjointWith>
   <owl:disjointWith>
    <owl:Class rdf: about="#ServiceInterface"/>
   </owl:disjointWith>
</owl:Class>
```

Một khái niệm cốt lõi khác của SOA là quan niệm về *process* (quy trình). Một *process* là một hợp thành mà các phần tử của nó được tạo thành một chuỗi hay luồng các hoạt động và tương tác với mục tiêu thực hiện công việc nhất định. Định nghĩa này nhất quán với định nghĩa về một quy trình trong Ký hiệu Mô hình hóa quy trình nghiệp vụ (BPMN) 2.0. (tham khảo Tài liệu [4]). Khái niệm *process* được thể hiện bởi lớp **Process** OWL, minh họa trong Hình 14.



Hình 14 – Lớp Process

Các phần tử trong hợp thành quy trình có thể là các tác nhân con người, các tác vụ, các dịch vụ, các quy trình khác,… Một quy trình luôn luôn thêm tính logic thông qua mô hình hợp thành, kết quả là nhiều các phần hơn. Theo mô hình cộng tác của chúng, các quy trình có thể như sau:

- **Orchestrated** (được điều phối): Khi một quy trình được điều phối trong một Hệ thống quản lý quy trình nghiệp vụ, thì sản phẩm CNTT thực tế sẽ là một điều phối; nghĩa là có một mô hình cộng tác điều phối. Kiểu quy trình này thường được gọi là "Điều phối quy trình" (Process Orchestration).

- **Choreographed** (được thiết kế theo trình tự định sẵn): Một mô hình quy trình thể hiện một mẫu hành vi đã được xác định trước thường được gọi là "Thiết kế theo trình tự định sẵn" (Process Choreography).

- **Collaborative** (Cộng tác): Không có hình mẫu về hành vi được xác định trước; quy trình thể hiện những hành vi (đã được thực hiện) quan sát được.

11.7. Ví dụ về hợp thành dịch vụ và quy trình

11.7.1. Ví dụ về hợp thành dịch vụ đơn giản

Sử dụng một ví dụ về hợp thành dịch vụ, các dịch vụ A và B là thể hiện của **Service** và hợp thành của A và B là một thể hiện của **ServiceComposition** (sử dụng cả A và B):

- *A* và *B* là thể hiện của **Service**,

- *X* là một thể hiện của **ServiceComposition**, và

- *X* sử dụng cả *A* và *B* (tạo ra chúng theo mô hình hợp thành dịch vụ).

Lưu ý rằng có rất nhiều cách mà theo đó mô hình hợp thành dịch vụ có thể tạo ra A và B, tất cả các cách này đều liên quan đến một tình huống này hoặc một tình huống khác. Ví dụ, các giao diện của X có hoặc không bao gồm một vài tập thành phần các giao diện của A và B. Ngoài ra, các giao diện của A và B có thể hoặc không thể triệu gọi (trực tiếp) mà không thông qua X, đó là một vấn đề của hợp đồng dịch vụ

và/hoặc các chính sách truy cập áp dụng cho cả A và B. Cuối cùng, X có thể sử dụng các phần tử khác không phải là dịch vụ (ví dụ là mã hợp thành, các bộ kết nối,…).

11.7.2. Ví dụ về quy trình

Sử dụng một ví dụ về quy trình, tác vụ T1 và T2 là các thể hiện của **Task**, vai trò R1 và R2 là các thể hiện của **Element**, hợp thành của T1, T2, R1 và R2 là một thể hiện của **Process** (sử dụng T1, T2, R1 và R2) :

- *T1* và *T2* là các thể hiện của **Task**,

- *R1* và *R2* là các thể hiện của **Element**,

- *Y* là một thể hiện của **Process**, và

- *Y* sử dụng cả *T1, T2, R1* và *R2* (tạo ra chúng theo mô hình hợp thành quy trình).

11.7.3. Ví dụ về quy trình và hợp thành dịch vụ

Để làm rõ ví dụ về quy trình trong phần 11.7.2, nếu T1 được làm khi sử dụng dịch vụ S thì :

- *S* là một thể hiện của **Service**, và

- *T1* sử dụng *S*.

Lưu ý rằng phụ thuộc vào việc lựa chọn phương thức thiết kế cụ thể (và mô hình hợp thành kết quả), Y có hoặc không sử dụng S một cách trực tiếp. Điều này phụ thuộc vào việc Y mang tính ràng buộc giữa T1 và S hay ràng buộc được đóng gói trong T1.

11.7.4. Ví dụ về rửa xe

Tham khảo phụ lục A về khía cạnh **Process** của ví dụ rửa xe ô tô.

12. **Chính sách**

12.1. Tổng quan

Các chính sách, các tác nhân con người đưa ra, và các sự vật mà chúng áp dụng đều là những khía cạnh quan trọng của một hệ thống bất kỳ, chắc chắn cả với các hệ thống SOA với rất nhiều các phần tử tương tác khác nhau. Các chính sách có thể áp dụng cho phần tử bất kỳ trong một hệ thống. Khái niệm về *Policy* được thể hiện bởi lớp **Policy** và mối quan hệ với các lớp **HumanActor** và **Thing**.

Khoản này mô tả các lớp sau đây của bản thể luận :

Policy

Ngoài ra, nó còn định nghĩa các đặc tính sau đây :

appliesTo và **isSubjectTo**

setsPolicy và **isSetby**

12.2. Lớp Policy

```
<owl:Class rdf:about="#Policy">
   <owl:disjointWith>
    <owl:Class rdf: about="#InformationType"/>
   </owl:disjointWith>
   <owl:disjointWith>
    <owl:Class rdf: about="#ServiceInterface"/>
   </owl:disjointWith>
   <owl:disjointWith>
    <owl:Class rdf: about="#Element"/>
   </owl:disjointWith>
   <owl:disjointWith>
    <owl:Class rdf: about="#Effect"/>
   </owl:disjointWith>
   <owl:disjointWith>
    <owl:Class rdf: about="#Event"/>
   </owl:disjointWith>
   <owl:disjointWith>
    <owl:Class rdf: about="#ServiceContract"/>
   </owl:disjointWith>
</owl:Class>
```

Một *policy* là một tuyên bố về định hướng mà một tác nhân con người có ý định làm theo hoặc tác nhân con người khác nên làm theo. Việc biết các chính sách áp dụng cho một sự vật giúp dễ dàng và minh bạch hơn khi tương tác với sự vật đó. Khái niệm *policy* được thể hiện bởi lớp **Policy** OWL, minh họa trong Hình 15.

Hình 15 – Lớp Policy

Policy như là một khái niệm chung và có liên quan đến phạm vi bên ngoài miền SOA. Với mục đích của bản thể luận về SOA này, không cần thiết hay không liên quan đến việc giới hạn về bản chất chung của chính lớp **Policy**. Mối quan hệ giữa **Policy** và **HumanActor** dĩ nhiên bị ràng buộc bởi những giới hạn cụ thể liên quan đến SOA mà được áp dụng trong định nghĩa về **HumanActor**.

Theo quan điểm thiết kế, các chính sách có thể gồm nhiều thành phần chi tiết hơn hoặc có thể được thể hiện và có hiệu lực thông qua các quy tắc cụ thể. Phần này của tiêu chuẩn ISO/IEC 18384 vẫn chỉ ở mức khái niệm và không bao gồm các khía cạnh về thiết kế như vậy trong bản thể luận.

Policy khác biệt với tất cả các khái niệm khác trong bản thể luận này; do đó, lớp **Policy** được định nghĩa tách rời với tất cả các lớp khác. Cụ thể, **Policy** tách rời với **ServiceContract**. Trong khi các chính sách có thể áp dụng với các hợp đồng dịch vụ, chẳng hạn như các chính sách về người có thể thay đổi hợp đồng dịch vụ đã đưa ra hoặc ngược lại được tham chiếu đến bởi hợp đồng dịch vụ như một phần của các điều khoản, điều kiện hợp đồng và các quy định tương tác mà các thành phần tham gia phải tuân theo, các hợp đồng dịch vụ bản thân nó không phải là chính sách bởi vì chúng không mô tả tiến trình của hành động hướng đến.

12.3. Đặc tính appliesTo và isSubjectTo

```
<owl:ObjectProperty rdf: about="#appliesTo">
   <rdfs:domain rdf:resource="#Policy"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf: about="#isSubjectTo">
   <owl:inverseOf>
    <owl:ObjectProperty rdf: about="#appliesTo"/>
   </owl:inverseOf>
</owl:ObjectProperty>
```

Các chính sách có thể áp dụng cho các sự vật chứ không phải chỉ áp dụng cho mỗi phần tử; trong thực tế, các chính sách có thể áp dụng cho bất kỳ sự vật nào và có thể gồm các chính sách khác. Ví dụ, một chính sách về bảo mật có thể chỉ rõ những tác nhân con người nào có quyền thay đổi một chính sách khác. Đặc tính **appliesTo** và **isSubjectTo** thể hiện quan điểm trừu tượng mà một chính sách có thể áp dụng cho bất kỳ thể hiện nào của **Thing**. Lưu ý đặc biệt rằng **Element** là một lớp thành phần của **Thing**, do vậy, bằng sự suy luận, các chính sách có thể áp dụng cho bất kỳ thể hiện nào của **Element**.

Theo định hướng, một chính sách có thể không áp dụng (khi một chính sách đã được xây dựng nhưng chưa được áp dụng cho rõ ràng cho bất kỳ một sự vật nào) hoặc được áp dụng cho một hoặc nhiều thể hiện của **Thing**. Lưu ý rằng việc một chính sách áp dụng cho rất nhiều sự vật không có nghĩa rằng các sự vật này là giống nhau, nó chỉ có ý nghĩa rằng chúng được quy định bởi cùng một mục đích giống nhau. Theo một hướng khác, một thể hiện của **Thing** có thể không phụ thuộc hoặc bị phụ thuộc vào một hoặc nhiều chính sách. Lưu ý rằng khi nhiều chính sách áp dụng cho cùng một thể hiện của **Thing**, điều này thường do nhiều chính sách thuộc các miền khác nhau (chẳng hạn như bảo mật và điều hành).

Bản thể luận về SOA không cố gắng làm rõ các miền chính sách khác nhau; những thông tin chi tiết tập trung vào chính sách như vậy được coi là phù hợp với bản thể luận về chính sách hơn. Cần thiết phải chỉ ra rằng một bản thể luận về chính sách cụ thể cũng có thể giới hạn một số loại sự vật mà chính sách có thể áp dụng (nếu muốn).

12.4. Đặc tính setsPolicy và isSetBy

```
<owl:ObjectProperty rdf:about="#setsPolicy">
<rdfs:domain rdf:resource="#HumanActor"/>
<rdfs:range rdf:resource="#Policy"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf: about="#isSetBy">
<owl:inverseOf>
<owl:ObjectProperty rdf: about="#setsPolicy"/>
</owl:inverseOf>
</owl:ObjectProperty>
```

Đặc tính **setsPolicy** (thiết lập chính sách) và **isSetBy** (được thiết lập bởi) thể hiện quan niệm trừu tượng mà một chính sách có thể được thiết lập bởi một hoặc nhiều tác nhân con người.

Theo định hướng, một chính sách có thể được thiết lập không bởi một tác nhân con người nào (trong trường hợp không xác định được tác nhân con người thiết lập chính sách) hoặc bởi một hoặc nhiều tác nhân con người. Đặc biệt lưu ý rằng một số chính sách được thiết lập bởi nhiều tác nhân con người, có nghĩa là tất cả những tác nhân con người này cần thảo luận và thống nhất về chính sách trước khi nó có hiệu lực. Một ví dụ trong thực tế là hai bố mẹ cùng nhau đưa ra các quy định về các hành vi có thể chấp nhận được của con trẻ. Theo một hướng khác, một tác nhân con người có thể thiết lập (hoặc tham gia một phần) rất nhiều chính sách khác nhau.

Bản thể luận về SOA có mục đích tách biệt việc thiết lập chính sách với việc áp dụng chính sách cho một hoặc nhiều thể hiện của **Thing**. Trong nhiều trường hợp, hai hành động này có thể bị ràng buộc và không thể tách rời, nhưng trong nhiều trường hợp khác, chúng lại không như vậy. Một ví dụ về trường hợp này là một chính sách về sự tuân thủ tổng thể được xây dựng ở cấp độ đoàn thể nhưng được áp dụng bởi các nhân viên trong từng dòng nghiệp vụ.

Ngoài ra, một trường hợp cụ thể cần quan tâm trong bản thể luận này là khi nhà cung cấp một dịch vụ có một chính sách về dịch vụ nhưng một chính sách về dịch vụ không nhất thiết phải do một nhà cung cấp sở hữu. Ví dụ, các quy định về thực phẩm và vệ sinh an toàn thực phẩm của chính phủ (chính sách được ban hành dạng luật) có hiệu lực với các dịch vụ của nhà hàng và không liên quan đến mong muốn hay quy định nào của chủ nhà hàng.

12.5. Các ví dụ

12.5.1. Ví dụ về rửa xe

Tham khảo Phần A.5 về khía cạnh Policy của ví dụ rửa xe

13. Sự kiện

13.1. Tổng quan

Sự kiện (event) và các phần tử sinh ra hoặc phản hồi chúng đều là những khía cạnh quan trọng của một hệ thống phát sinh sự kiện bất kỳ. Các hệ thống SOA trong thực tế thường phát sinh sự kiện, do đó, *event* được định nghĩa là một khái niệm trong bản thể luận về SOA.

Khoản này mô tả các lớp sau đây của bản thể luận:

Event

Ngoài ra, nó đình nghĩa các đặc tính sau đây:

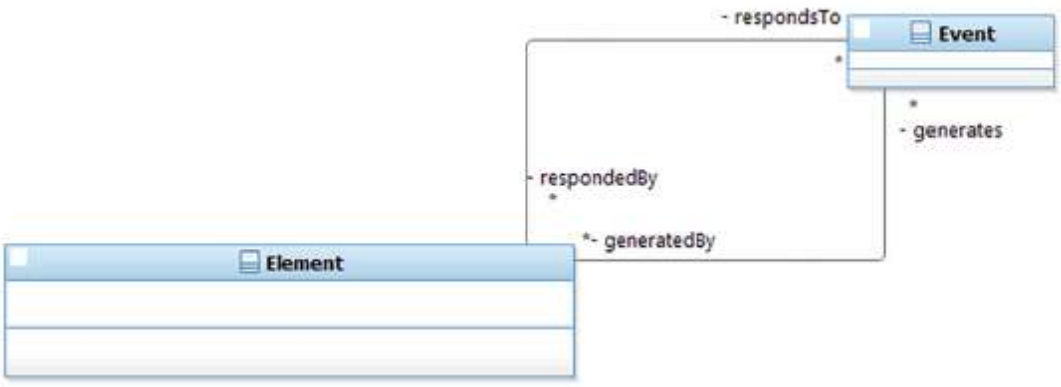
generates (sinh ra) và **generatedBy** (sinh ra bởi)

respondsTo (phản hồi đến) và **respondedToBy** (được phản hồi bởi)

13.2. Lớp Event

```
<owl:Class rdf:about="#Event">
   <owl:disjointWith>
    <owl:Class rdf: about="#Policy"/>
   </owl:disjointWith>
   <owl:disjointWith>
    <owl:Class rdf: about="#ServiceContract"/>
   </owl:disjointWith>
   <owl:disjointWith>
    <owl:Class rdf: about="#ServiceInterface"/>
   </owl:disjointWith>
</owl:Class>
```

Một *event* (sự kiện) là một sự vật xảy ra mà một phần tử có thể lựa chọn phản hồi đến nó. Các sự kiện có thể được phản hồi bởi một phần tử bất kỳ. Tương tự, các sự kiện có thể được sinh ra bởi một phần tử bất kỳ. Việc biết các sự kiện được sinh ra và được phản hồi bởi một phần tử sẽ giúp dễ dàng và minh bạch hơn trong việc tương tác với phần tử đó. Lưu ý rằng một số sự kiện vẫn có thể xảy ra cho dù được sinh ra và được phản hồi bởi một phần tử hoặc không bởi một phần tử nào cả. Khái niệm về *event* được thể hiện trong lớp **Event** OWL, minh họa trong Hình 16.



Hình 16 – Lớp Event

Event như là một khái niệm chung và có liên quan đến miền SOA cũng như nhiều miền khác. Với mục đích của bản thể luận này, *Event* được sử dụng với ý nghĩa chung.

Theo quan điểm thiết kế, các sự kiện có thể bao gồm nhiều thành phần hoặc cũng có thể được biểu diễn và hoạt động thông qua cú pháp và ngữ nghĩa cụ thể. Phần này của tiêu chuẩn ISO/IEC 18384 chỉ mô tả ở mức khái niệm và không bao gồm các thông tin về khía cạnh thiết kế trong bản thể luận.

13.3. Đặc tính generates và generatedBy

```
<owl:ObjectProperty rdf: about="#generates">
   <rdfs:domain rdf:resource="#Element"/>
   <rdfs:range rdf:resource="#Event"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf: about="#generatedBy">
   <owl:inverseOf>
    <owl:ObjectProperty rdf: about="#generates"/>
   </owl:inverseOf>
</owl:ObjectProperty>
```

Các sự kiện có thể, nhưng không nhất thiết, được tạo ra bởi các phần tử. Đặc tính **generates** (sinh ra) và **generatedBy** (được sinh ra bởi) thể hiện quan niệm trừu tượng mà một phần tử sinh ra một sự kiện.

Lưu ý rằng cùng một sự kiện có thể được sinh ra bởi nhiều phần tử khác nhau. Tương tự, một phần tử có thể sinh ra rất nhiều sự kiện.

13.4. Đặc tính respondsTo và respondedToBy

```
<owl:ObjectProperty rdf: about="#respondsTo">
   <rdfs:domain rdf:resource="#Element"/>
   <rdfs:range rdf:resource="#Event"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf: about="#respondedToBy">
   <owl:inverseOf>
```

```
    <owl:ObjectProperty rdf: about="#respondsTo"/>
   </owl:inverseOf>
</owl:ObjectProperty>
```

Các sự kiện có thể, nhưng không nhất thiết, được phản hồi bởi các phần tử. Đặc tính **respondsTo** (phản hồi đến) và **respondedToBy** (được phản hồi bởi) thể hiện quan niệm trừu tượng về việc một phần tử phản hồi một sự kiện.

Cùng một sự kiện có thể được phản hồi bởi rất nhiều phần tử. Tương tự, một phần tử có thể phản hồi rất nhiều sự kiện khác nhau.

Phụ lục A

(Thông tin bổ sung)

Ví dụ về rửa xe hoàn chỉnh

A.1. Giới thiệu chung

Phụ lục này trình bày ví dụ rửa xe hoàn chỉnh đã được sử dụng trong các phần bên trên của bản thể luận.

A.2. Khía cạnh về tổ chức

Joe – người chủ lựa chọn tổ chức doanh nghiệp thành 2 đơn vị thành phần: *Administration* (Quản lý) và *CarWash* (Rửa xe):

- CarWashBusiness là một thể hiện của cả **HumanActor** và **System**,

- *Administration* là một thể hiện của **HumanActor** (đơn vị tổ chức),

- *CarWash* là một thể hiện của **HumanActor** (đơn vị tổ chức),

- *CarWashBusiness* sử dụng (gồm các đơn vị tổ chức) *Administration* và *CarWash*,

- *AdministrativeSystem* là một thể hiện của **System**,

- *Administration* đại diện cho *AdministrativeSystem*,

- *CarWashSystem* là một thể hiện của **System**, và

- *CarWash* đại diện cho *CarWashSystem*.

Và sử dụng vai trò được xác định rõ ràng trong mỗi tổ chức:

- *Owner* (Người chủ) (vai trò) là một thể hiện của **Element** và được sử dụng bởi *AdministrativeSystem*,

- *Joe* là một thể hiện của **HumanActor** và được đại diện bởi (có vai trò) *Owner*,

- *Secretary* (Thư ký) (vai trò) là một thể hiện của **Element** và được sử dụng bởi *AdministrativeSystem*,

- *Mary* là một thể hiện của **HumanActor** và được đại diện bởi (có vai trò) *Secretary*,

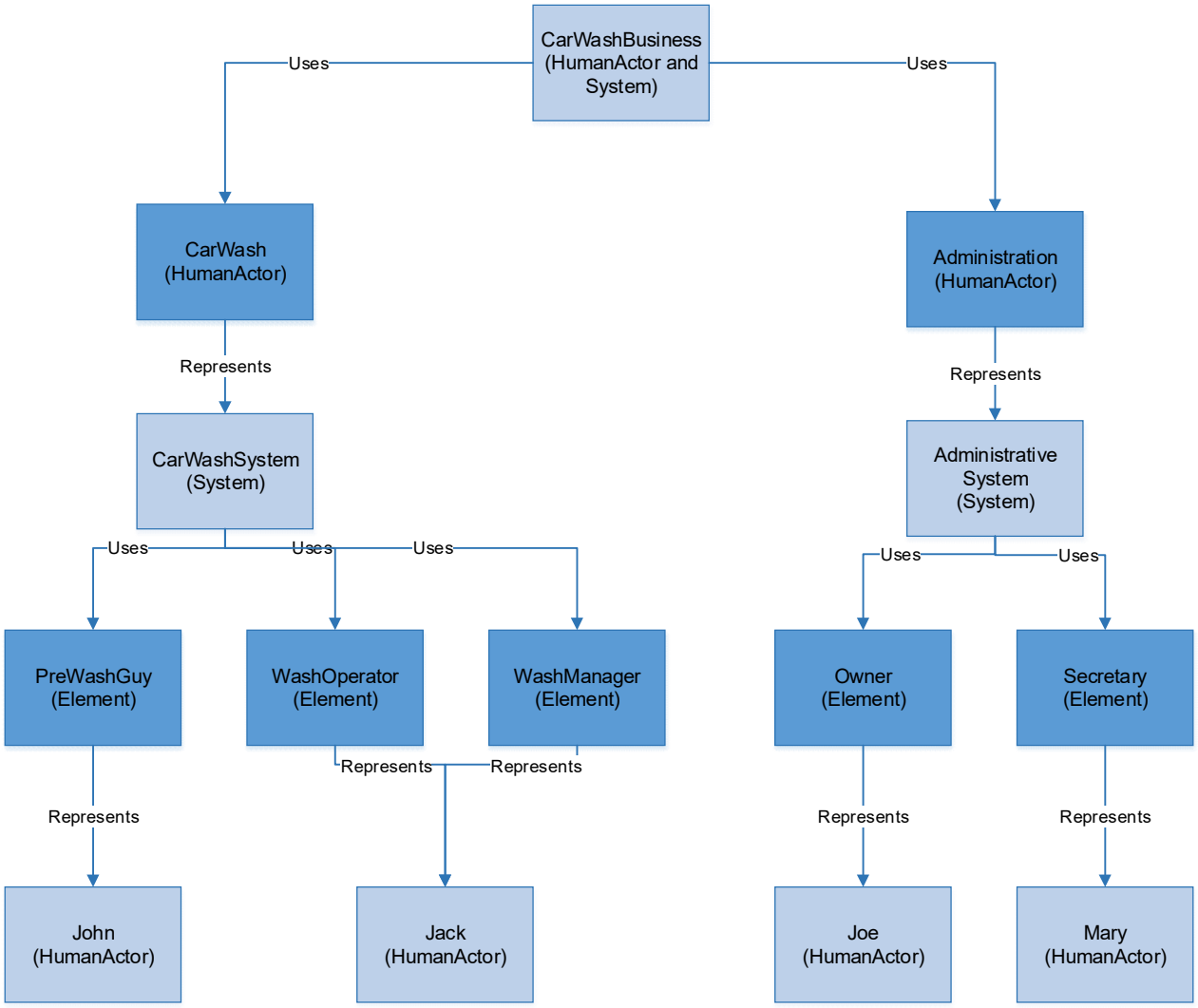
- *PreWashGuy* (vai trò) là một thể hiện của **Element** và được sử dụng bởi *CarWashSystem*,

- *John* là một thể hiện của **HumanActor** và được đại diện bởi (có vai trò) *PreWashGuy*,

- *WashManager* (vai trò) là một thể hiện của **Element** và được sử dụng bởi *CarWashSystem*,

- *WashOperator* (vai trò) là một thể hiện của **Element** và được sử dụng bởi *CarWashSystem*, và

- *Jack* là một thể hiện của **HumanActor** và được đại diện bởi (có vai trò) bởi cả *WashManager* và *WashOperator*.



Hình A.1 – Ví dụ rửa xe – Khía cạnh về tổ chức

A.3. Các dịch vụ rửa xe

Joe cung cấp 2 dịch vụ khác nhau cho khách hàng: Rửa xe cơ bản (basic wash) và rửa xe chất lượng cao (gold wash):

- *GoldWash* là một thể hiện của **Service**,

- *BasicWash* là một thể hiện của **Service**,

- *CarWash* thực hiện cả hai *BasicWash* và *GoldWash*, và

- *WashManager* đại diện cho cả *BasicWash* và *GoldWash* (nghĩa là nó là điểm tương tác mà khách hàng có thể đặt hàng dịch vụ cũng như trả tiền phí).

Đối với việc thanh toán, dịch vụ *BasicWash* của Joe làm sạch xe của khách hàng Judy:

- *Judy* là một thể hiện của **HumanActor** (khách hàng),

- *BasicWashContract* là một thể hiện của **ServiceContract**,

- **BasicWash** có hợp đồng *BasicWashContract*,

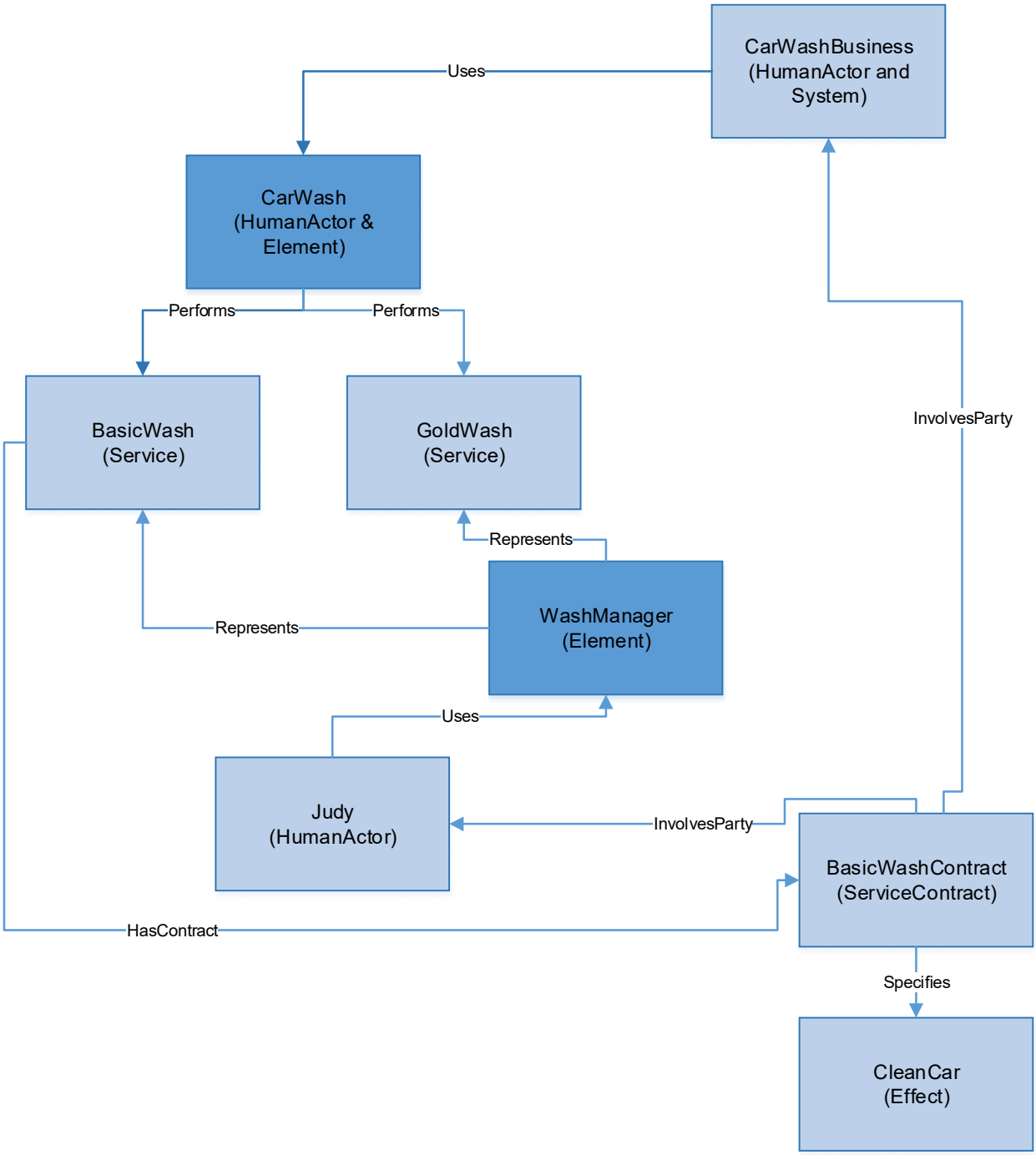
- *CleanCar* là một thể hiện của **Effect**,

- *BasicWashContract* chỉ rõ CleanCar như tác động của nó,

- *BasicWashContract* bao gồm các bên tham gia *CarWashBusiness* và *Judy* và xác định rõ Judy (là khách hàng hợp pháp) thanh toán *CarWashBusiness* (với tư cách là nhà cung cấp hợp pháp) 10 đô la cho một lần sử dụng *BasicWash* với tác động là một *CleanCar*. Lưu ý rằng *BasicWash* thực sự được thực hiện bởi *CarWash* chứ không phải bởi nhà cung cấp hợp pháp *CarWashBusiness*, trong ví dụ cụ thể này *CarWash* là thành viên của *CarWashBusiness*, nhưng không phải lúc nào cũng như vậy, *CarWash* có thể là nhà cung cấp thứ ba và

- *Judy* sử dụng *WashManager* (để triệu gọi dịch vụ *BasicWash*).

Lưu ý rằng, trong ví dụ này, Judy không tương tác trực tiếp với dịch vụ rửa xe *BasicWash*, thay vào đó, Judy tương tác với *WashManager* đại diện cho dịch vụ. Điều này là do Joe quyết định, khi rửa xe, khách hàng của mình không tương tác trực tiếp với máy rửa.



Hình A.2 – Ví dụ rửa xe – Các dịch vụ rửa xe

A.4. Các giao diện cho các dịch vụ rửa xe

Cách thức để tương tác với các dịch vụ rửa xe rất đơn giản đối với khách hàng; Họ chỉ cần trả tiền cho người quản lý rửa xe và yêu cầu một trong hai dịch vụ rửa xe được cung cấp. Do trong thực tế, Joe đã quyết định đưa thêm người quản lý giữa khách hàng và máy rửa xe, nên khách hàng không bao giờ phải

tương tác với các dịch vụ rửa xe này. Một dịch vụ proxy được cung cấp bởi người quản lý rửa xe đã được định nghĩa một cách chính thức, nhưng trong ví dụ thực tế này, mức độ chính thức đã bị bỏ qua.

Người quản lý rửa xe lần lượt tương tác với các dịch vụ rửa xe qua giao diện của chúng, được định nghĩa như sau:

- *WashingMachineInterface* là một thể hiện của **ServiceInterface**;

- *TypeOfWash* là một thể hiện của **InformationType**;

- *WashingMachineInterface* có đầu vào *TypeOfWash*;

- *BasicWash* có giao diện *WashMachineInterface*;

- *GoldWash* có giao diện *WashMachineInterface*.

Lưu ý rằng, thực tế thì cả hai dịch vụ rửa xe đều có cùng một giao diện dịch vụ. Mặc dù, Joe đã chọn lựa cung cấp rửa xe cơ bản và rửa bằng chất lượng cao là hai dịch vụ khác nhau, nhưng cả hai đều được làm bởi cùng một máy rửa (một việc đơn giản phải lựa chọn loại rửa xe khi khởi tạo máy rửa).

A.5. Các quy trình rửa xe

Một phần quan trọng của hệ thống rửa xe là quy trình rửa xe được thể hiện như sau:

- *AutomatedCarWashProcess* là một thể hiện của cả **Process** và **Orchestration**;

- *Wash* là một thể hiện của **Task** và được sử dụng bởi *AutomatedCarWashProcess*;

- *Dry* (Làm khô) là một thể hiện của **Task** và được sử dụng bởi *AutomatedCarWashProcess*;

- *AutomatedCarWash* là một thể hiện của **Element** (máy rửa tự động) và đại diện cho *AutomatedCarWashProcess* (đóng gói quy trình) cũng như điều khiển *AutomatedCarWashProcess*;

- *CarWashProcess* là một thể hiện của **Process** và được sử dụng bởi (một phần của) *CarWashSystem* (không cần phải tạo một khối xây dựng rõ ràng);

- *AutomatedCarWash* được sử dụng bởi *CarWashProcess* (hoạt động tự động trong quy trình);

- *WashWindows* là một thể hiện của **Task** và được làm bởi *John*;

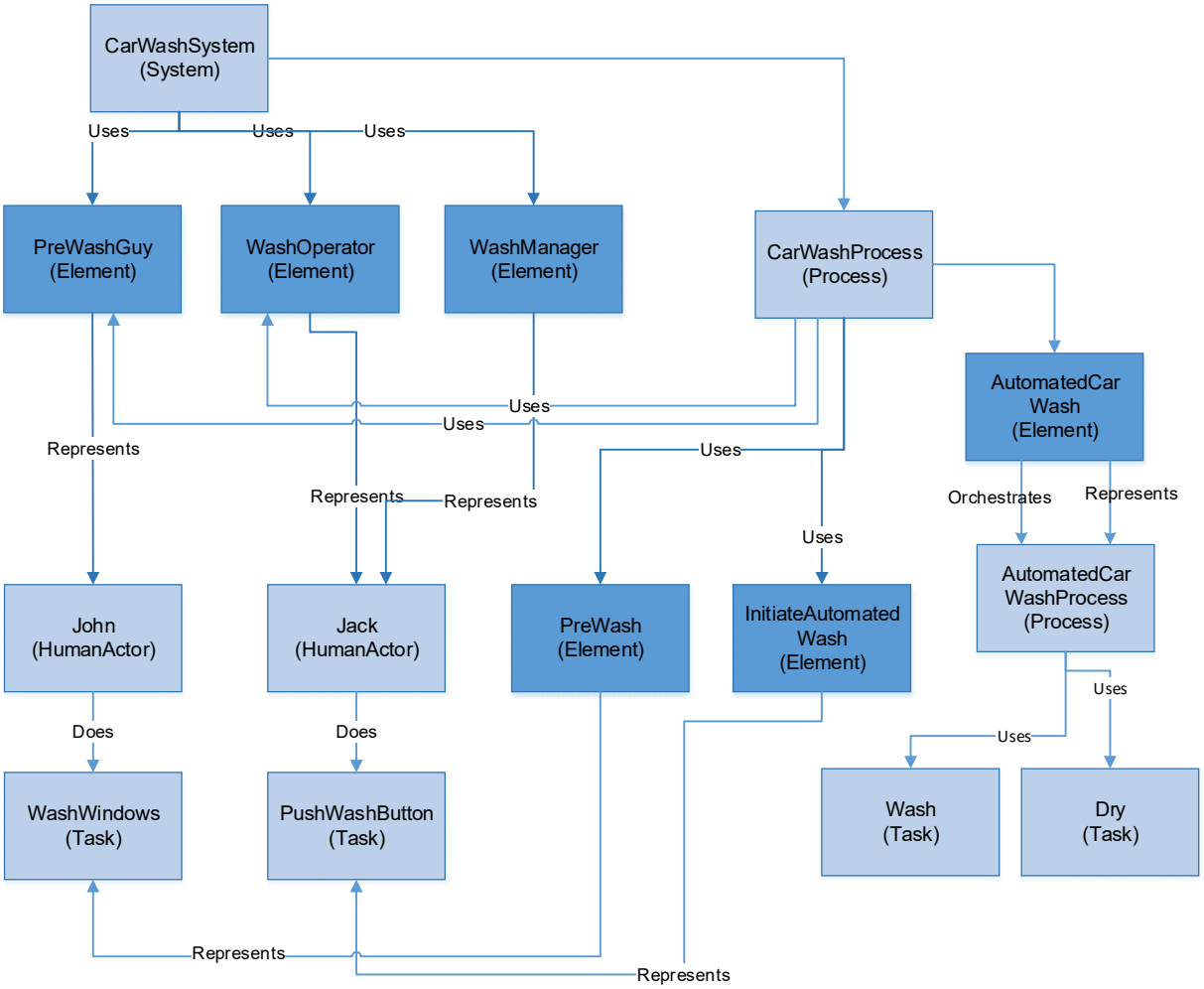
- *PreWash* là một thể hiện của **Element**, đại diện cho *WashWindows*, và được sử dụng bởi *CarWashProcess* (hoạt động logic trong quy trình);

- *PrewashGuy* là một thành viên của *CarWashProcess* (vai trò trong quy trình);

- *PushWashButton* là một thể hiện của **Task** và được làm bởi *Jack*;

- *InitiateAutomatedWash* là một thể hiện của **Element**, đại diện cho *PushWashButton*, và được sử dụng bởi *CarWashProcess* (hoạt động logic trong quy trình);

- *WashOperator* là một thành viên của *CarWashProcess* (vai trò trong quy trình).



Hình A.3 – Ví dụ về rửa xe – Quy trình rửa xe

A.5.1. Các chính sách về rửa xe

Joe đặt ra chính sách tiền thanh toán tiền trước đối với các dịch vụ rửa xe như sau:

- *PaymentUpFront* là một thể hiện của **Policy**;

- *PaymentUpFront* được thiết lập bởi *Joe*;

- *PaymentUpFront* áp dụng cho cả *GoldWash* và *BasicWash*.

Lưu ý cách thức chính sách *PaymentUpFront* mở rộng hợp đồng dịch vụ *BasicWashContract*. Mặc dù *BasicWashContract* chỉ quy định rằng *Judy* phải trả 10 đô la cho một lần sử dụng dịch vụ *BasicWash*, nhưng chính sách *PaymentUpFront* cho biết cụ thể rằng cần phải thực hiện thanh toán trước đó. Một trong những ưu điểm của việc tách biệt chính sách ra khỏi hợp đồng dịch vụ là chính sách thanh toán có thể được thay đổi một cách độc lập với hợp đồng dịch vụ. Ví dụ, sau này, Joe có thể thay đổi rằng khách hàng không cần phải trả tiền trước và có thể thực hiện thay đổi này với chính sách mà không ảnh hưởng đến những sự vật khác liên quan đến *CarWashBusiness*.

Phụ lục B

(Thông tin bổ sung)

Ví dụ về mua hàng Internet

Jill đang mua một chiếc Ti vi mới trên Internet thông qua một trang bán hàng trực tuyến:

- Jill là một thể hiện của **Actor** (người).

- *PurchaseTV* là một thể hiện của **Task**.

- *Jill* làm *PuschaseTV*.

- *BuyTVOnline* là một thể hiện của **Service**.

- *PurchaseTV* sử dụng *BuyTVOnline*.

OnlineTVSales là công ty bán Ti vi:

- *OnlineTVSales* là một thể hiện của **Actor** (tổ chức).

- *BuyTVOnlineContract* là một thể hiện của **ServiceContract** (và mô tả cách thức tương tác với *BuyTVOnline* cũng như hợp đồng pháp lý giữa người mua Ti vi và *OnlineTVSales*).

- *BuyTVOnline* có hợp đồng *BuyTVOnlineContract*.

- *OnlineTVSales* là bên tham gia *BuyTVOnlineContract*.

- *Jill* là bên tham gia *BuyTVOnlineContract*.

Các trang web trực tuyến được triển khai bằng cách sử dụng phần mềm trang web:

- *OnlineSalesComponent* là một thể hiện của **Element**.

- *OnlineSalesComponent* thực hiện *OnlineTVSales*.

- *SelectWhatToBuyComponent* là một thể hiện của **Element**.

- *SelectWhatToBuyService* là một thể hiện của **Service**.

- *SelectWhatToBuyComponent* thực hiện *SelectWhatToBuyService*.

- *PayComponent* là một thể hiện của **Element**.

- *PayService* là một thể hiện của **Service**.

- *PayComponent* thực hiện *PayService*.

- *OnlineSalesComponent* cũng là một thể hiện của **ServiceComposition**.

- *OnlineSalesComponent* sử dụng *SelectWhatToBuyService* và *PayService*.

Để hoàn tất giao dịch mua hàng, Jill cần thanh toán tiền và sau đó Ti vi sẽ được chuyển đến:

- *PayForTV* là một thể hiện của **Task**.

- *Jill* làm *PayForTV*.

- *PayForTV* sử dụng *BuyTVOnline*.

- *DeliverTV* là một thể hiện của **Task**.

- *OnlineTVSales* làm *DeliverTV*.

- *OnlineTVSalesProcess* là một thể hiện của **Process**.

- *OnlineTVSalesProcess* sử dụng *Jill*, *OnlineTVSales*, *PurchaseTV*, *PayForTV* và *DeliverTV*.

Phụ lục C

(Chuẩn hóa)

Định nghĩa OWL về bản thể luận

Bản thể luận về OWL có thể tham khảo trực tuyến tại: http://www.w3.org/2007/OWL/wiki/OWL_Working_Group

Bản thể luận được mô phỏng như dưới đây.

```
<?xml version="1.0"?>

<rdf:RDF
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
   xmlns:owl="http://www.w3.org/2002/07/owl#"
   xmlns="http://www.semanticweb.org/ontologies/2010/01/core-soa.owl#"
   xml:base="http://www.semanticweb.org/ontologies/2010/01/core-soa.owl"
>

 <!-- bản thể luận -->
 <owl:Bản thể luận rdf:about=""/>
 <!-- classes -->

 <owl:Class rdf:about="#Event">
   <owl:disjointWith>
    <owl:Class rdf:about="#Policy"/>
   </owl:disjointWith>
   <owl:disjointWith>
    <owl:Class rdf:about="#ServiceContract"/>
   </owl:disjointWith>
   <owl:disjointWith>
    <owl:Class rdf:about="#ServiceInterface"/>
   </owl:disjointWith>
</owl:Class>

<owl:Class rdf:about="#InformationType">
   <owl:disjointWith>
    <owl:Class rdf:about="#Policy"/>
   </owl:disjointWith>
   <owl:disjointWith>
    <owl:Class rdf:about="#ServiceContract"/>
   </owl:disjointWith>
   <owl:disjointWith>
    <owl:Class rdf:about="#Effect"/>
   </owl:disjointWith>
</owl:Class>
```

```
<owl:Class rdf:about="#ServiceComposition">
   <rdfs:subClassOf>
    <owl:Class rdf:about="#Composition"/>
   </rdfs:subClassOf>
   <owl:disjointWith>
    <owl:Class rdf:about="#ServiceContract"/>
   </owl:disjointWith>
   <owl:disjointWith>
    <owl:Class rdf:about="#ServiceInterface"/>
   </owl:disjointWith>
</owl:Class>

<owl:Class rdf:about="#Effect">
   <owl:disjointWith>
    <owl:Class rdf:about="#Policy"/>
   </owl:disjointWith>
   <owl:disjointWith>
    <owl:Class rdf:about="#ServiceInterface"/>
   </owl:disjointWith>
   <owl:disjointWith>
    <owl:Class rdf:about="#InformationType"/>
   </owl:disjointWith>
   <rdfs:subClassOf>
    <owl:Restriction>
     <owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
     >1</owl:minCardinality>
     <owl:onProperty>
      <owl:ObjectProperty rdf:about="#isSpecifiedBy"/>
     </owl:onProperty>
    </owl:Restriction>
   </rdfs:subClassOf>
</owl:Class>

<owl:Class rdf:about="#Task">
   <owl:disjointWith>
    <owl:Class rdf:about="#Policy"/>
   </owl:disjointWith>
   <owl:disjointWith>
    <owl:Class rdf:about="#System"/>
   </owl:disjointWith>
   <owl:disjointWith>
    <owl:Class rdf:about="#HumanActor"/>
   </owl:disjointWith>
   <owl:disjointWith>
    <owl:Class rdf:about="#Service"/>
   </owl:disjointWith>
   <owl:disjointWith>
    <owl:Class rdf:about="#ServiceContract"/>
   </owl:disjointWith>
   <owl:disjointWith>
    <owl:Class rdf:about="#ServiceInterface"/>
```

```
   </owl:disjointWith>
   <owl:disjointWith>
    <owl:Class rdf:about="#Composition"/>
   </owl:disjointWith>
   <rdfs:subClassOf>
    <owl:Class rdf:about="#Element"/>
   </rdfs:subClassOf>
   <rdfs:subClassOf>
    <owl:Restriction>
     <owl:onProperty>
      <owl:ObjectProperty rdf:about="#doneBy"/>
     </owl:onProperty>
     <owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
     >0</owl:minCardinality>
    </owl:Restriction>
   </rdfs:subClassOf>
   <rdfs:subClassOf>
    <owl:Restriction>
     <owl:maxCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
     >1</owl:maxCardinality>
     <owl:onProperty>
      <owl:ObjectProperty rdf:about="#doneBy"/>
     </owl:onProperty>
    </owl:Restriction>
   </rdfs:subClassOf>
</owl:Class>

<owl:Class rdf:about="#System">
   <owl:disjointWith>
    <owl:Class rdf:about="#Task"/>
   </owl:disjointWith>
   <owl:disjointWith>
    <owl:Class rdf:about="#Service"/>
   </owl:disjointWith>
   <rdfs:subClassOf>
    <owl:Class rdf:about="#Element"/>
   </rdfs:subClassOf>
</owl:Class>

<owl:Class rdf:about="#Service">
   <owl:disjointWith>
    <owl:Class rdf:about="#System"/>
   </owl:disjointWith>
   <owl:disjointWith>
    <owl:Class rdf:about="#Task"/>
   </owl:disjointWith>
   <owl:disjointWith>
    <owl:Class rdf:about="#HumanActor"/>
   </owl:disjointWith>
   <owl:disjointWith>
    <owl:Class rdf:about="#ServiceInterface"/>
```

```
   </owl:disjointWith>
   <rdfs:subClassOf>
    <owl:Class rdf:about="#Element"/>
   </rdfs:subClassOf>
   <rdfs:subClassOf>
    <owl:Restriction>
     <owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
     >1</owl:minCardinality>
     <owl:onProperty>
      <owl:ObjectProperty rdf:about="#hasInterface"/>
     </owl:onProperty>
    </owl:Restriction>
   </rdfs:subClassOf>
</owl:Class>

<owl:Class rdf:about="#Policy">
   <owl:disjointWith>
    <owl:Class rdf:about="#InformationType"/>
   </owl:disjointWith>
   <owl:disjointWith>
    <owl:Class rdf:about="#ServiceInterface"/>
   </owl:disjointWith>
   <owl:disjointWith>
    <owl:Class rdf:about="#Element"/>
   </owl:disjointWith>
   <owl:disjointWith>
    <owl:Class rdf:about="#Effect"/>
   </owl:disjointWith>
   <owl:disjointWith>
    <owl:Class rdf:about="#Event"/>
   </owl:disjointWith>
   <owl:disjointWith>
    <owl:Class rdf:about="#ServiceContract"/>
   </owl:disjointWith>
</owl:Class>

<owl:Class rdf:about="#HumanActor">
   <rdfs:subClassOf>
    <owl:Class rdf:about="#Element"/>
   </rdfs:subClassOf>
   <owl:disjointWith>
    <owl:Class rdf:about="#Task"/>
   </owl:disjointWith>
   <owl:disjointWith>
    <owl:Class rdf:about="#Service"/>
   </owl:disjointWith>
   <owl:disjointWith>
    <owl:Class rdf:about="#ServiceContract"/>
   </owl:disjointWith>
   <owl:disjointWith>
```

```
    <owl:Class rdf:about="#ServiceInterface"/>
   </owl:disjointWith>
</owl:Class>

<owl:Class rdf:about="#Composition">
   <owl:disjointWith>
    <owl:Class rdf:about="#Task"/>
   </owl:disjointWith>
   <rdfs:subClassOf>
    <owl:Class rdf:about="#System"/>
   </rdfs:subClassOf>
   <rdfs:subClassOf>
    <owl:Restriction>
     <owl:maxCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
     >1</owl:maxCardinality>
     <owl:onProperty>
      <owl:DatatypeProperty rdf:about="#compositionPattern"/>
     </owl:onProperty>
    </owl:Restriction>
   </rdfs:subClassOf>
   <rdfs:subClassOf>
    <owl:Restriction>
     <owl:onProperty>
      <owl:DatatypeProperty rdf:about="#compositionPattern"/>
     </owl:onProperty>
     <owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
     >1</owl:minCardinality>
    </owl:Restriction>
   </rdfs:subClassOf>
   <rdfs:subClassOf>
    <owl:Restriction>
     <owl:maxCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
     >1</owl:maxCardinality>
     <owl:onProperty>
      <owl:ObjectProperty rdf:about="#orchestratedBy"/>
     </owl:onProperty>
    </owl:Restriction>
   </rdfs:subClassOf>
   <rdfs:subClassOf>
    <owl:Restriction>
     <owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
     >0</owl:minCardinality>
     <owl:onProperty>
      <owl:ObjectProperty rdf:about="#orchestratedBy"/>
     </owl:onProperty>
    </owl:Restriction>
   </rdfs:subClassOf>
</owl:Class>

<owl:Class rdf:about="#ServiceInterface">
   <owl:disjointWith>
```

```
    <owl:Class rdf:about="#Service"/>
   </owl:disjointWith>
   <owl:disjointWith>
    <owl:Class rdf:about="#ServiceContract"/>
   </owl:disjointWith>
   <owl:disjointWith>
    <owl:Class rdf:about="#Effect"/>
   </owl:disjointWith>
   <owl:disjointWith>
    <owl:Class rdf:about="#Policy"/>
   </owl:disjointWith>
   <owl:disjointWith>
    <owl:Class rdf:about="#HumanActor"/>
   </owl:disjointWith>
   <owl:disjointWith>
    <owl:Class rdf:about="#Task"/>
   </owl:disjointWith>
   <owl:disjointWith>
    <owl:Class rdf:about="#ServiceComposition"/>
   </owl:disjointWith>
   <owl:disjointWith>
    <owl:Class rdf:about="#Process"/>
   </owl:disjointWith>
   <owl:disjointWith>
    <owl:Class rdf:about="#Event"/>
   </owl:disjointWith>
   <rdfs:subClassOf>
    <owl:Restriction>
     <owl:onProperty>
      <owl:DatatypeProperty rdf:about="#constraints"/>
     </owl:onProperty>
     <owl:maxCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
     >1</owl:maxCardinality>
    </owl:Restriction>
   </rdfs:subClassOf>
   <rdfs:subClassOf>
    <owl:Restriction>
     <owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
     >1</owl:minCardinality>
     <owl:onProperty>
      <owl:DatatypeProperty rdf:about="#constraints"/>
     </owl:onProperty>
    </owl:Restriction>
   </rdfs:subClassOf>
</owl:Class>

<owl:Class rdf:about="#Element">
   <owl:disjointWith>
    <owl:Class rdf:about="#Policy"/>
   </owl:disjointWith>
```

```
   <rdfs:subClassOf>
    <owl:Restriction>
     <owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
     >0</owl:minCardinality>
     <owl:onProperty>
      <owl:ObjectProperty rdf:about="#orchestrates"/>
     </owl:onProperty>
    </owl:Restriction>
   </rdfs:subClassOf>
   <rdfs:subClassOf>
    <owl:Restriction>
     <owl:maxCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
     >1</owl:maxCardinality>
     <owl:onProperty>
      <owl:ObjectProperty rdf:about="#orchestrates"/>
     </owl:onProperty>
    </owl:Restriction>
   </rdfs:subClassOf>
</owl:Class>

<owl:Class rdf:about="#ServiceContract">
   <owl:disjointWith>
    <owl:Class rdf:about="#ServiceInterface"/>
   </owl:disjointWith>
   <owl:disjointWith>
    <owl:Class rdf:about="#Policy"/>
   </owl:disjointWith>
   <owl:disjointWith>
    <owl:Class rdf:about="#HumanActor"/>
   </owl:disjointWith>
   <owl:disjointWith>
    <owl:Class rdf:about="#Task"/>
   </owl:disjointWith>
   <owl:disjointWith>
    <owl:Class rdf:about="#ServiceComposition"/>
   </owl:disjointWith>
   <owl:disjointWith>
    <owl:Class rdf:about="#Process"/>
   </owl:disjointWith>
   <owl:disjointWith>
    <owl:Class rdf:about="#Event"/>
   </owl:disjointWith>
   <owl:disjointWith>
    <owl:Class rdf:about="#InformationType"/>
   </owl:disjointWith>
   <rdfs:subClassOf>
    <owl:Restriction>
     <owl:onProperty>
      <owl:DatatypeProperty rdf:about="#legalAspect"/>
     </owl:onProperty>
```

```
      <owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1</owl:minCardinality>
     </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
     <owl:Restriction>
      <owl:maxCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1</owl:maxCardinality>
      <owl:onProperty>
       <owl:DatatypeProperty rdf:about="#legalAspect"/>
      </owl:onProperty>
     </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
     <owl:Restriction>
      <owl:onProperty>
       <owl:DatatypeProperty rdf:about="#interactionAspect"/>
      </owl:onProperty>
      <owl:maxCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1</owl:maxCardinality>
     </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
     <owl:Restriction>
      <owl:onProperty>
       <owl:DatatypeProperty rdf:about="#interactionAspect"/>
      </owl:onProperty>
      <owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1</owl:minCardinality>
     </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
     <owl:Restriction>
      <owl:onProperty>
       <owl:ObjectProperty rdf:about="#isContractFor"/>
      </owl:onProperty>
      <owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1</owl:minCardinality>
     </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
     <owl:Restriction>
      <owl:onProperty>
       <owl:ObjectProperty rdf:about="#specifies"/>
      </owl:onProperty>
      <owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1</owl:minCardinality>
     </owl:Restriction>
    </rdfs:subClassOf>
</owl:Class>
```

```
<owl:Class rdf:about="#Process">
   <owl:disjointWith>
    <owl:Class rdf:about="#ServiceContract"/>
   </owl:disjointWith>
   <owl:disjointWith>
    <owl:Class rdf:about="#ServiceInterface"/>
   </owl:disjointWith>
   <rdfs:subClassOf>
    <owl:Class rdf:about="#Composition"/>
   </rdfs:subClassOf>
</owl:Class>

<!-- object properties -->

<owl:ObjectProperty rdf:about="#isPartyTo">
   <rdfs:domain rdf:resource="#HumanActor"/>
   <rdfs:range rdf:resource="#ServiceContract"/>
    </owl:ObjectProperty>
   <owl:ObjectProperty rdf:about="#involvesParty">
   <owl:inverseOf>
    <owl:ObjectProperty rdf:about="#isPartyTo"/>
   </owl:inverseOf>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="#orchestratedBy">
   <rdfs:domain rdf:resource="#Composition"/>
   <rdfs:range rdf:resource="#Element"/>
 </owl:ObjectProperty>

<owl:ObjectProperty rdf:about="#orchestrates">
   <owl:inverseOf>
    <owl:ObjectProperty rdf:about="#orchestratedBy"/>
   </owl:inverseOf>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="#isContractFor">
   <rdfs:domain rdf:resource="#ServiceContract"/>
   <rdfs:range rdf:resource="#Service"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="#hasContract">
   <owl:inverseOf>
    <owl:ObjectProperty rdf:about="#isContractFor"/>
   </owl:inverseOf>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="#setsPolicy">
   <rdfs:domain rdf:resource="#HumanActor"/>
   <rdfs:range rdf:resource="#Policy"/>
</owl:ObjectProperty>
```

```
<owl:ObjectProperty rdf:about="#isSetBy">
   <owl:inverseOf>
    <owl:ObjectProperty rdf:about="#setsPolicy"/>
   </owl:inverseOf>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="#generates">
   <rdfs:domain rdf:resource="#Element"/>
   <rdfs:range rdf:resource="#Event"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="#generatedBy">
   <owl:inverseOf>
    <owl:ObjectProperty rdf:about="#generates"/>
   </owl:inverseOf>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="#represents">
   <rdfs:domain rdf:resource="#Element"/>
   <rdfs:range rdf:resource="#Element"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="#representedBy">
   <owl:inverseOf>
    <owl:ObjectProperty rdf:about="#represents"/>
   </owl:inverseOf>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="#hasInput">
   <rdfs:domain rdf:resource="#ServiceInterface"/>
   <rdfs:range rdf:resource="#InformationType"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="#isInputAt">
   <owl:inverseOf>
    <owl:ObjectProperty rdf:about="#hasInput"/>
   </owl:inverseOf>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="#doneBy">
   <rdfs:domain rdf:resource="#Task"/>
   <rdfs:range rdf:resource="#HumanActor"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="#does">
   <owl:inverseOf>
    <owl:ObjectProperty rdf:about="#doneBy"/>
   </owl:inverseOf>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="#specifies">
   <rdfs:domain rdf:resource="#ServiceContract"/>
```

```
   <rdfs:range rdf:resource="#Effect"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="#isSpecifiedBy">
   <owl:inverseOf>
    <owl:ObjectProperty rdf:about="#specifies"/>
   </owl:inverseOf>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="#appliesTo">
   <rdfs:domain rdf:resource="#Policy"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="#isSubjectTo">
   <owl:inverseOf>
    <owl:ObjectProperty rdf:about="#appliesTo"/>
   </owl:inverseOf>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="#hasInterface">
   <rdfs:domain rdf:resource="#Service"/>
   <rdfs:range rdf:resource="#ServiceInterface"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="#isInterfaceOf">
   <owl:inverseOf>
    <owl:ObjectProperty rdf:about="#hasInterface"/>
   </owl:inverseOf>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="#respondsTo">
   <rdfs:domain rdf:resource="#Element"/>
   <rdfs:range rdf:resource="#Event"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="#respondedToBy">
   <owl:inverseOf>
    <owl:ObjectProperty rdf:about="#respondsTo"/>
   </owl:inverseOf>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="#performs">
   <rdfs:domain rdf:resource="#Element"/>
   <rdfs:range rdf:resource="#Service"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="#performedBy">
   <owl:inverseOf>
    <owl:ObjectProperty rdf:about="#performs"/>
   </owl:inverseOf>
</owl:ObjectProperty>
```

```
<owl:ObjectProperty rdf:about="#uses">
   <rdfs:domain rdf:resource="#Element"/>
   <rdfs:range rdf:resource="#Element"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="#usedBy">
   <owl:inverseOf>
    <owl:ObjectProperty rdf:about="#uses"/>
   </owl:inverseOf>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="#hasOutput">
   <rdfs:domain rdf:resource="#ServiceInterface"/>
   <rdfs:range rdf:resource="#InformationType"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="#isOutputAt">
   <owl:inverseOf>
    <owl:ObjectProperty rdf:about="#hasOutput"/>
   </owl:inverseOf>
</owl:ObjectProperty>

<!-- datatype properties -->
<owl:DatatypeProperty rdf:about="#legalAspect">
   <rdfs:domain rdf:resource="#ServiceContract"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:about="#constraints">
   <rdfs:domain rdf:resource="#ServiceInterface"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:about="#compositionPattern">
   <rdfs:domain rdf:resource="#Composition"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:about="#interactionAspect">
   <rdfs:domain rdf:resource="#ServiceContract"/>
</owl:DatatypeProperty>

</rdf:RDF>
```

Phụ lục D

(Thông tin bổ sung)

Ma trận quan hệ của lớp

Phụ lục này gồm một ma trận quan hệ của lớp thể hiện các mối quan hệ bản chất giữa các lớp trong các định nghĩa OWL của bản thể luận về SOA. Ma trận được xác định từ các định nghĩa OWL của bản thể luận. Mỗi hàng X và mỗi cột Y tương ứng với một lớp OWL. Một mối quan hệ sẽ xuất hiện trong ô (X, Y), nếu và chỉ khi nào, lớp X là một phần thuộc miền và lớp Y là một phần của phạm vi của đặc tính OWL tương ứng. Lưu ý rằng điều này có nghĩa là đặc tính datatype (không có giới hạn) sẽ không được bao gồm trong ma trận quan hệ của lớp.

Như đã nêu trong phần nội dung chính của tài liệu này, có bốn quan hệ trong bảng (ngoài ra còn có các biến thể và các mở rộng của lớp thành phần) được phép sử dụng theo các định nghĩa OWL, nhưng không được kỳ vọng sẽ xảy ra khi áp dụng thực tế của bản thể luận. Cụ thể, các dịch vụ không được kỳ vọng để thực hiện các dịch vụ, các dịch vụ không được kỳ vọng sử dụng các phần tử (một cách trực tiếp), các dịch vụ không được kỳ vọng thể hiện các phần tử, và các dịch vụ không được kỳ vọng điều phối các hợp thành, tất cả do lớp **Service** được định nghĩa như một biểu diễn logic của một tập các hoạt động; tham khảo 10.3, 10.5.1, 10.5.2 và 11.3.2 để biết thêm thông tin chi tiết.

Thing						
Policy	isSubjectTo	isSubjectTo	isSubjectTo	setsPolicy, isSubjectTo	isSubjectTo	isSubjectTo
Event	Generates, respondsTo	Generates, respondsTo	Generates, respondsTo	Generates, respondsTo	Generates, respondsTo	Generates, respondsTo
InfomationType						
Service interface			hasInterface			
Effect						
Service contract			hasContract	isPartyTo		
Service composition	Uses, usedBy represents, representedBy, orchestrates	Uses, usedBy represents, representedBy, orchestrates	Uses, usedBy represents, representedBy, performedBy, orchestrates	Uses, usedBy represents, representedBy, orchestrates	Uses, usedBy represents, representedBy, orchestrates	Uses, usedBy represents, representedBy, orchestrates, orchestratedBy
Process	Uses, usedBy represents, representedBy, orchestrates	Uses, usedBy represents, representedBy, orchestrates	Uses, usedBy represents, representedBy, performedBy, orchestrates	Uses, usedBy represents, representedBy, orchestrates	Uses, usedBy represents, representedBy, orchestrates	Uses, usedBy represents, representedBy, orchestrates, orchestratedBy

Composition	Uses, usedBy represents, representedBy, orchestrates	Uses, usedBy represents, representedBy, orchestrates	Uses, usedBy represents, representedBy, performedBy, orchestrates	Uses, usedBy represents, representedBy, orchestrates	Uses, usedBy represents, representedBy, orchestrates	Uses, usedBy represents, representedBy, orchestrates, orchestratedBy
Task	Uses, usedBy represents, representedBy	Uses, usedBy represents, representedBy	Uses, usedBy represents, representedBy, performedBy	Uses, usedBy represents, representedBy, does	Uses, usedBy represents, representedBy	Uses, usedBy represents, representedBy, orchestratedBy
Human Actor	Uses, usedBy represents, representedBy	Uses, usedBy represents, representedBy	Uses, usedBy represents, representedBy, performedBy	Uses, usedBy represents, representedBy	Uses, usedBy represents, representedBy, doneBy	Uses, usedBy represents, representedBy, orchestratedBy
Service	Uses, usedBy represents, representedBy, performs	Uses, usedBy represents, representedBy, performs	Uses, usedBy represents, representedBy, performs, performedBy	Uses, usedBy represents, representedBy, performs	Uses, usedBy represents, representedBy, performs	Uses, usedBy represents, representedBy, performs, orchestratedBy
System	Uses, usedBy represents, representedBy	Uses, usedBy represents, representedBy	Uses, usedBy	Uses, usedBy represents, representedBy	Uses, usedBy represents, representedBy	Uses, usedBy represents, representedBy, performs, orchestratedBy

			represents, representedBy, performedBy			
Element	Uses, usedBy represents, representedBy	Uses, usedBy represents, representedBy	Uses, usedBy represents, representedBy, performedBy	Uses, usedBy represents, representedBy	Uses, usedBy represents, representedBy	Uses, usedBy represents, representedBy, performs, orchestratedBy
	Element	System	Service	Human Actor	Task	Composition

Thing								appliesTo	
Policy	isSubjectTo	isSubjectTo	isSubjectTo	isSubjectTo	isSubjectTo	isSubjectTo	isSubjectTo	appliesTo, isSubjectTo	isSubjectTo
Event	Generates, respondsTo	Generates, respondsTo	Generates, respondsTo	Generates, respondsTo				appliesTo	
InfomationType					hasInput, hasOuput			appliesTo	
Service interface						isInputAt, isOutputAt		appliesTo	
Effect			Specifies					appliesTo	
Service contract				isSpecifiedBy				appliesTo	
Service composition	Uses, usedBy represents, representedBy, orchestrates, orchestratedBy	Uses, usedBy represents, representedBy, orchestrates, orchestratedBy					generatedBy, respondedToBy	appliesTo	

Process	Uses, usedBy represents, representedBy, orchestrates, orchestratedBy	Uses, usedBy represents, representedBy, orchestrates, orchestratedBy					generatedBy, respondedToBy	appliesTo	
Composition	Uses, usedBy represents, representedBy, orchestrates, orchestratedBy	Uses, usedBy represents, representedBy, orchestrates, orchestratedBy					generatedBy, respondedToBy	appliesTo	
Task	Uses, usedBy represents, representedBy, orchestratedBy	Uses, usedBy represents, representedBy, orchestratedBy					generatedBy, respondedToBy	appliesTo	
Human Actor	Uses, usedBy represents, representedBy, orchestratedBy	Uses, usedBy represents, representedBy, orchestratedBy	involvesParty				generatedBy, respondedToBy	isSetBy, appliesTo	
Service	Uses, usedBy represents, representedBy,	Uses, usedBy represents, representedBy,	isContractFor		isInterfaceOf		generatedBy, respondedToBy	appliesTo	

	performs, orchestratedBy	performs, orchestratedBy							
System	Uses, usedBy represents, representedBy, orchestratedBy	Uses, usedBy represents, representedBy, orchestratedBy					generatedBy, respondedToBy	appliesTo	
Element	Uses, usedBy represents, representedBy, orchestratedBy	Uses, usedBy represents, representedBy, orchestratedBy					generatedBy, respondedToBy	appliesTo	
	Process	Service composition	Service contract	Effect	Service interface	InfomationType	Event	Policy	Thing

Phụ lục E

(Thông tin bổ sung)

Ánh xạ các thuật ngữ giữa các phần của SOA RA

Trong phần phụ lục này có một bảng ánh xạ các định nghĩa trong các phần của Tiêu chuẩn ISO/IEC 18384. Cụ thể, các định nghĩa trong tiêu chuẩn ISO/IEC 18384-1 với các thuật ngữ của bản thể luận trong tiêu chuẩn ISO/IEC 18384-3 để làm nổi bật các thông tin về mỗi thuật ngữ và khái niệm có thể được tìm thấy cũng như minh họa khi các định nghĩa là phù hợp hoặc sai lệch. Nếu xảy ra sai lệch giữa các định nghĩa, thì phân tích định nghĩa và sự khác biệt sẽ được in nghiêng. Cột thứ ba chứa các khoản trong tiêu chuẩn ISO/IEC 18384-2, trong đó, các khái niệm được xác định, thảo luận hoặc được tham chiếu. Cột cuối cùng chỉ ra nguồn tham khảo của các thuật ngữ đã được định nghĩa. Nếu không tồn tại khái niệm trong tiêu chuẩn ISO/IEC 18384-3 hoặc tiêu chuẩn ISO/IEC 18384-4 thì điền 'n / a' vào cột.

ISO/IEC 18384-1	ISO/IEC 18384-3	ISO/IEC 18384-2	Tài liệu tham chiếu khác
Khoản và định nghĩa *Phân tích in nghiêng khi có sự khác biệt*	**Khoản và định nghĩa/thảo luận** *Phân tích in nghiêng khi có sự khác biệt*	**Số thứ tự khoản khi khái niệm được thảo luận** n/a cho biết định nghĩa không xuất hiện trong ISO/IEC 18384-2	
3.1 actor (tác nhân) (ISO/IEC 16500-8:1999, 3.1) Người hoặc thành phần (component) hệ thống có tương tác với hệ thống đó và cung cấp tác động triệu gọi hành động	n/a	Được thảo luận trong 4.5, Khoản 9, Khoản 11, Khoản 12	(ISO/IEC 16500-8:1999, 3.1) BPMN
3.2. architecture (kiến trúc) (ISO/IEC 16500-8:1999, 3.2) Khái niệm hay đặc tính cơ bản của một hệ thống trong môi trường thể hiện các phần tử, các mối quan hệ và theo các nguyên tắc về thiết kế và tiến hóa của nó	n/a	Được thảo luận trong Khoản 4	(ISO/IEC/IEEE 42010:2011, 3.1)

3.3. choreography (Thiết kế theo trình tự định sẵn) Kiểu hợp thành (3.5) trong đó các thành phần của nó (3.8) tương tác theo một cách thức không bị điều khiển với từng thành phần độc lập hiểu rõ và tuân theo một mô hình hành vi đã được định nghĩa trước cho toàn bộ hợp thành. *Đặc trưng quan sát thấy được bổ sung vào định nghĩa của Bản thể luận*	11.3.3 Trong một Điều phối theo trình tự định sẵn với kết quả mong đợi (một hợp thành mà mô hình hợp thành là một thiết kế theo trình tự định sẵn), các phần tử được sử dụng bởi hợp thành tương tác theo cách thức không bị điều khiển, nhưng với mỗi thành viên độc lập hiểu rõ và tuân theo một mô hình hành vi đã được định nghĩa trước cho toàn bộ hợp thành đó.	Được thảo luận trong Khoản 8	
3.4. collaboration (cộng tác) Kiểu hợp thành (3.5) trong đó các phần tử (3.8) tương tác theo cách thức không bị điều khiển, mỗi phần tử theo một kế hoạch và mục đích riêng của chúng mà không có một mô hình hành vi được xác định trước	11.3.4 Trong cộng tác (một hợp thành mà mô hình hợp thành của chúng là collaboration), các phần tử được sử dụng bởi hợp thành tương tác theo cách không bị điều khiển, mỗi phần tử theo một kế hoạch và mục đích riêng của chúng mà không có một mô hình hành vi được định nghĩa trước	Được thảo luận trong Khoản 4, Khoản 8	

3.5. composition (hợp thành) Kết quả của việt kết hợp một tập các phần tử theo một mục đích cụ thể	11.2 Hợp thành là kết quả của việc kết hợp một tập các sự vật theo một mục đích cụ thể	Được thảo luận trong Khoản 4, Khoản 5, Khoản 8, Khoản 9, Khoản 10, Khoản 11, Khoản 12, Khoản 15 Cũng được sử dụng theo nghĩa tiếng Anh	
3.6. endpoint (điểm cuối) Vị trí mà tại đó nhận được thông tin để triệu gọi và cấu hình tương tác	n/a	Được thảo luận trong Khoản 4, Khoản 10	
3.7. effect (tác động)	10.10 Tương tác với một vật thực hiện một dịch vụ sẽ có những effect (tác động). Những tác động này bao gồm đầu ra (outcome) của tương tác đó, và cách thức cung một dịch vụ (thông qua phần tử thực hiện nó) mang lại giá trị đến người dùng.	Được thảo luận trong Khoản 4, Khoản 6, Khoản 11	
3.8 element (phẩn tử) Đơn vị không thể phân chia nhỏ hơn ở mức trừu tượng và có ranh giới được xác định rõ ràng	8.2 Một *element* là một thực thể (entity) không trong suốt và không thể phân chia nhỏ hơn ở mức trừu tượng. Mỗi	Được thảo luận trong Khoản 4 và trong toàn bộ tài liệu Ngoài ra, nó cũng được sử dụng với nghĩa tiếng Anh	

	phần tử có ranh giới được xác định rõ ràng		
3.9. entity Phần tử độc lập (3.8) trong một hệ thống với một định danh có thể hoạt động như một nhà cung cấp dịch vụ (3.49) hoặc một người sử dụng dịch vụ	n/a chỉ được sử dụng trong định nghĩa phần tử và được viện dẫn theo Tài liệu tham khảo 4	Được thảo luận trong Khoản 4, Khoản 5	SoaML
3.10 event (sự kiện) Sự vật xảy ra mà một phần tử có thể lựa chọn phản hồi	13.2 Một *event* là một sự vật xảy ra mà một phần tử có thể lựa chọn phản hồi. Sự kiện có thể được phản hồi bởi một phần tử bất kỳ. Tương tự, các sự kiện có thể được sinh ra (tạo ra) bởi một phần tử bất kỳ	Được thảo luận trong Khoản 4, Khoản 8, Khoản 10, Khoản 11, Khoản 12	
3.11. ngữ cảnh thực hiện Tập các phần tử kỹ thuật và nghiệp vụ (3.8) mà nhà cung cấp dịch vụ (3.49) và người dùng (3.29) cần theo nhu cầu và khả năng của họ	n/a	n/a	SOA RM

3.12. human actor (tác nhân con người) Tác nhân (3.1) dưới điều khiển của một thực thể con người hoặc một thực thể tổ chức (3.9)	9.2. Hunman Actor Một *human actor* là một người hoặc một tổ chức. tách rời với lớp *Service* và *Task*	Được thảo luận trong Khoản 4, Khoản 9	
3.13. human task (tác vụ của con người) Tác vụ là việc được làm bởi tác nhân con người (3.12)	*Tác vụ không được định nghĩa.* 9.4. task (tác vụ) Một *task* là một hành động hoàn thành một kết quả xác định. Các tác vụ được làm bởi người hoặc tổ chức, cụ thể là bởi các thể hiện của **HumanActor**	Được thảo luận trong Khoản 8	
3.14. Interface (giao diện) Một ranh giới (boundary) được chia sẻ giữa hai đơn vị (units) chức năng, được xác định bởi rất nhiều đặc tính liên quan đến các chức năng, các kết nối vật lý, trao đổi tín hiệu và các đặc tính khác liên quan [NGUỒN: ISO/IEC 2382-1:1993, 01.01.38]	10.13 *thảo luận giao diện: (nhất quán)* Khái niệm về một giao diện nhìn chung dễ hiểu cho người triển khai thực tế, bao gồm quan niệm rằng các giao diện xác định các tham số cho phép thông tin đi vào và đi ra khỏi chúng khi được triệu gọi. Điểm khác nhau giữa miền này với miền kia là bản chất cụ thể về cách thức một giao diện được triệu dọi	Được thảo luận trong Khoản 4, Khoản 7, Khoản 9, Khoản 14	[NGUỒN: ISO/IEC 2382-1:1993, 01.01.38]

	và cách thức thông tin được truyền qua truyền lại. Theo quan điểm của người thiết kế, giao diện có thể có nhiều hoạt động hơn hoặc có thể chứa nhiều giao diện khác.		
3.15 loose coupling (kết nối lỏng) Nguyên tắc mà ở đó sự phụ thuộc giữa các dịch vụ được tối giản hóa	n/a 10.13. *thảo luận kết nối lỏng không chủ định nhưng nhất quán* Các giao diện dịch vụ thường (nhưng không nhất thiết) dựa trên thông điệp (hỗ trợ kết nối lỏng). Hơn nữa, các giao diện dịch vụ luôn luôn được xác định độc lập với dịch vụ triển khai chúng (hỗ trợ kết nối lỏng và thành phần trung gian)	Được thảo luận trong Khoản 4, Khoản 6, Khoản 10	
3.16 orchestration (điều phối) Kiểu hợp thành (3.5) trong đó, một phần tử cụ thể (3.8) được sử dụng bởi hợp thành để giám sát và điều hướng các phần tử khác	11.3 Trong một điều phối (một hợp thành mà mô hình của nó là một điều phối), có một phần tử cụ thể được sử dụng bởi hợp thành giám sát và điều hướng các phần tử khác. Lưu ý rằng phần tử	Được thảo luận trong Khoản 4, Khoản 8, Khoản 9, Khoản 10	

	điều hướng một điều phối bởi định nghĩa khác với bản thân một điều phối (thể hiện của **Composition**)		
3.17. policy (chính sách) Tuyên bố về việc một thực thể (3.9) có thể tuân theo hoặc một thực thể khác nên tuân theo	12 Một chính sách là một tuyên bố về hướng dẫn rằng một tác nhân con người có thể tuân theo hoặc một tác nhân con người khác nên tuân theo. *Các chính sách có thể áp dụng cho phần tử bất kỳ trong một hệ thống.* *Bản thể luận này thể hiện phạm vi hẹp hơn so với Phần 1, chỉ cho phép tác nhân con người định nghĩa/tuân theo chính sác*h.	Được thảo luận trong Khoản 4, Khoản 11, Khoản 13, Khoản 14	
3.18 process (quy trình) Kiểu hợp thành (3.5) trong đó, các phần tử (3.8) được tạo thành một chuỗi hoặc một luồng các hoạt động và tương tác với mục tiêu thực hiện một công việc cụ thể	11.6 Một quy trình là một hợp thành trong đó, các phần tử được tạo thành một chuỗi hoặc một luồng các hoạt động và tương tác với mục tiêu thực hiện một công việc cụ thể	Được thảo luận trong Khoản 4, Khoản 8, Khoản 11, Khoản 13, Khoản 14 Ngoài ra, nó cũng được sử dụng theo nghĩa tiếng Anh	

3.19 real world effect (tác động của thế giới thực) Thay đổi liên quan đến và được trải nghiệm bởi các bên liên quan cụ thể (Tham khảo Tài liệu [6])	n/a *tương đương với 'effect'* effect (tác động): tương tác với vật thực hiện một dịch vụ có *effects*. Những tác động này bao gồm đầu ra (outcome) của tương tác đó và là cách thức mà một dịch vụ (thông qua phần tử thực hiện nó) mang lại giá trị cho người dùng.	Được thảo luận trong Khoản 4, Khoản 6	SOA RM
3.20. service (dịch vụ) Một biểu diễn logic của một tập các hoạt động có đầu ra cụ thể, có tính khép kín (self-contained), có thể bao gồm các dịch vụ khác và là một "hộp đen" đối với người sử dụng dịch vụ (tham khảo ISO/IEC 18384-3:--,7.2)	10.2 Một dịch vụ là một biểu diễn logic của một tập các hoạt động có đầu ra cụ thể, có tính khép kín (self-contained), có thể bao gồm các dịch vụ khác và là một "hộp đen" đối với người sử dụng dịch vụ.	Được thảo luận trong Khoản 4, và trong các nội dung còn lại của tài liệu	
3.21. service broker (trung gian dịch vụ) Phần tử có khả năng giao tiếp với các dịch vụ (3.20) cả mức nghiệp vụ hay	n/a	n/a	

mức triển khai, nghĩa là với thành phần trung chuyển			
3.22. service bus (trục dịch vụ) Mô hình thiết kế và chạy thật phục vụ việc tương tác của các dịch vụ (3.20), ví dụ như truyền thông, truy cập, sử dụng, chuyển đổi, trung chuyển và định tuyến thông điệp	Được thảo luận trong Khoản 7, Khoản 11, Khoản 13	n/a	
3.23. service candidate (ứng viên dịch vụ) Các dịch vụ (3.20) được định danh trong suốt vòng đời SOA (2.1.58), đáp ứng các yêu cầu dịch vụ rộng rãi và theo đó, một hoặc nhiều dịch vụ được lựa chọn để tiếp tục phát triển như là một phần của một giải pháp SOA tổng thể (3.56)	n/a	Được thảo luận trong Khoản 8	
3.24. service registry/repository service catalogue (đăng ký dịch vụ/danh mục dịch vụ) Tập các mô tả logic về dịch vụ (3.31) và các sản phẩm (artifacts) liên quan	n/a	Được thảo luận trong Khoản 4, Khoản 9, Khoản 12, Khoản 13	

nhằm hỗ trợ việc xuất bản, đăng ký, tìm kiếm, quản lý và thu hồi các sản phẩm đó.			
3.25. service choreography (Thiết kế dịch vụ theo trình tự định sẵn) Điều phối (3.3) trong đó các phần tử (3.8) là các dịch vụ (3.20) (tham khảo ISO/IEC 18384-3:-, Khoản 8)	*Khoản 11 đã thảo luận – nội dung nhất quán* *Không có định nghĩa cụ thể nhưng là mở rộng của hợp thành dịch vụ - sau đó, là "kết quả của việc kết hợp một tập các dịch vụ", do vậy , nhất quán với việc điều phối của các dịch vụ*	Được thảo luận trong Khoản 4, Khoản 8, Khoản 15 Thảo luận điều phối của các dịch vụ	
3.26. service collaboration (cộng tác dịch vụ) Cộng tác (3.5) trong đó các phần tử (3.8) là các dịch vụ (3.20) (tham khảo ISO/IEC 18384-3:-, Khoản 8)	*Khoản 11 đã thảo luận – nội dung nhất quán* *Không có định nghĩa cụ thể nhưng là mở rộng của hợp thành dịch vụ - sau đó, là "kết quả của việc kết hợp một tập các dịch vụ", do vậy , nhất quán với việc cộng tác của các dịch vụ*	Được thảo luận trong Khoản 4, Khoản 8, Khoản 15 Thảo luận cộng tác các dịch vụ Cộng tác theo ngữ cảnh tiếng Anh	
3.27. service component (cấu phần dịch vụ)	n/a	Được thảo luận trong Khoản 4, Khoản 6, Khoản 7, Khoản 8, Khoản 15	

Phần tử (3.8) triển khai các dịch vụ (3.20)			
3.28. service composition (hợp thành dịch vụ) Hợp thành (3.5) cung cấp (ngữ cảnh vận hành) các dịch vụ mức cao hơn (3.20) chỉ bao gồm các dịch vụ khác	11.5 Một khái niệm cốt lõi của SOA là quan niệm về *service composition*, kết quả của việc kết hợp một tập các dịch vụ để thực hiện một dịch vụ mới mức cao hơn	Được thảo luận trong Khoản 4, Khoản 5, Khoản 8	
3.29. service consumer (người dùng dịch vụ) Thực thể (3.9) sử dụng dịch vụ (3.20)	10.4 Phần tử sử dụng (tiêu thụ) một dịch vụ	Được thảo luận trong Khoản 4, Khoản 8, Khoản 9, Khoản 10	
3.30. service contract (hợp đồng dịch vụ) Các điều khoản, điều kiện và quy tắc về tương tác mà người dùng dịch vụ (3.29) và nhà cung cấp dịch vụ (3.49) cùng thỏa thuận (trực tiếp hoặc gián tiếp) *Phần 1 giới hạn những người tham gia*	10.6 Một *service contract* xác định các điều khoản, điều kiện và quy tắc về tương tác mà những người tham gia cùng thỏa thuận (trực tiếp hoặc gián tiếp)	Được thảo luận trong Khoản 4, Khoản 11, Khoản 13, Khoản 14	

3.31. service description (mô tả dịch vụ) Thông tin cần thiết để sử dụng hoặc xem xét sử dụng một dịch vụ (3.20)	n/a *được sử dụng theo viện dẫn trong tài liệu số 10.1* "một cơ chế cho phép truy cập một hoặc nhiều khả năng, trong đó, việc truy cập được cung cấp bằng cách sử dụng một giao diện quy định và được thực hiện phù hợp với những ràng buộc và chính sách như đã đặc tả bởi mô tả dịch vụ	Được thảo luận trong Khoản 4, Khoản 6, Khoản 7, Khoản 14	
3.32. service deployment (triển khai dịch vụ) Hoạt động tạo ra việc triển khai các dịch vụ (3.20) để cho phép chạy trong môi trường phần cứng, phần mềm cụ thể và có khả năng sử dụng bởi người dùng dịch vụ (3.29)	n/a	Được thảo luận trong Khoản 4, Khoản 6, Khoản 14	
3.33. service development (phát triển dịch vụ) Hoạt động xác định các yêu cầu và ràng buộc, thiết kế các dịch vụ như một thành phần của giải pháp SOA	n/a	Được thảo luận trong Khoản 6, Khoản 14	

(3.56) để giải quyết các yêu cầu theo ràng buộc ở trên			
3.34. service implementtation (thực hiện dịch vụ) Hoạt động thực hiện phát triển kỹ thuật và thực hiện vật lý của dịch vụ (3.20), là thành phần của một vòng đời dịch vụ (3.40), và kết quả trong việc tạo ra một cấu phần dịch vụ (3.27)	n/a	Được thảo luận trong Khoản 4, Khoản 5, Khoản 14, Khoản 15	
3.35. service discovery (khai phá dịch vụ) Hoạt động trong đó một người dùng dịch vụ có thể tìm kiếm các dịch vụ đáp ứng yêu cầu chức năng/phi chức năng cụ thể của họ	n/a	Được thảo luận trong Khoản 7, Khoản 8, Khoản 10, Khoản 13	
3.36. service governance (quản trị dịch vụ) Chiến lược và cơ chế kiểm soát áp dụng trong vòng đời dịch vụ (3.40) và danh mục dịch vụ, bao gồm quy định trách nhiệm, hướng dẫn giám sát việc tuân thủ các chính sách bằng việc	n/a	Được thảo luận trong Khoản 13	

cung cấp các quy trình và cách đo phù hợp như là thành phần của việc quản trị giải pháp SOA (3.57)			
3.37. service interaction (tương tác dịch vụ) Hoạt động tạo ra việc sử dụng một khả năng được cung cấp, thông thường là đi qua một ranh giới để đạt được một tác động mong muốn cụ thể trong thế giới thực (3.18) (tham khảo Tài liệu [6])	10.8. *thảo luận khía cạnh tương tác – không mâu thuẫn với tương tác dịch vụ, nhưng không định nghĩa cụ thể* Khía cạnh tương tác Bất kỳ ai muốn sử dụng dịch vụ đều phải tuân theo khía cạnh tương tác (như định nghĩa trong đặc tính kiểu dữ liệu interaction-Aspect) của một hợp đồng dịch vụ bất kỳ áp dụng cho tương tác đó. Theo cách đó, khía cạnh tương tác của một hợp đồng dịch vụ là độc lập với ngữ cảnh; chúng thể hiện cách thức cơ bản được định nghĩa trước trong đó một dịch vụ có thể được sử dụng	Được thảo luận trong Khoản 4, Khoản 7, Khoản 10	SOA RM
3.38. service interface (giao diện dịch vụ)	10.13. Một giao diện dịch vụ xác định cách thức trong đó, các phần tử khác có thể	Được thảo luận trong Khoản 4, Khoản 9, Khoản 10, Khoản 14	

Giao diện mà các phần tử khác (3.8) có thể tương tác và trao đổi thông tin với dịch vụ trong đó, định dạng yêu cầu và đầu ra của yêu cầu nằm trong mô tả dịch vụ (tham khảo ISO/IEC 18384-3:-, 7.13)	tương tác và trao đổi thông tin với một dịch vụ		
3.39. service interoperability (tương tác dịch vụ) Khả năng của nhà cung cấp dịch vụ (3.49) và người dùng dịch vụ (3.29) có thể giao tiếp, triệu gọi dịch vụ (3.20) và trao đổi thông tin cả về ngữ nghĩa lẫn cú pháp dẫn đến các tác động như đã định nghĩa trong mô tả dịch vụ (3.31)	n/a	Được thảo luận trong Khoản 4, Khoản 6, Khoản 10, Khoản 14	
3.40. Service Level Agreement (thỏa thuận mức độ dịch vụ) Kiểu hợp đồng dịch vụ (3.30) xác định các điều kiện tương tác có thể định lượng được giữa một nhà cung cấp dịch vụ (3.49) với một người dùng dịch vụ (3.29)	n/a	Được thảo luận trong Khoản 4, Khoản 11, Khoản 13, Khoản 14	

3.41. service lifecylce (vòng đời dịch vụ) Tập các giai đoạn cho việc thực hiện một dịch vụ (3.20) từ lúc nhận thức, xác định cho đến khởi tạo và nghỉ hưu (kết thúc)	n/a	Được thảo luận trong Khoản 4, Khoản 11, Khoản 13, Khoản 14	
3.42. service management (quản lý dịch vụ) Giám sát, kiểm soát, duy trì, tối ưu hóa và vận hành dịch vụ (3.20)	n/a	Được thảo luận trong Khoản 4, Khoản 11, Khoản 13	
3.43. service modelling (mô hình hóa dịch vụ) Tập các hoạt động để phát triển một chuỗi các ứng viên dịch vụ (3.23) cho các chức năng và hành động về một giải pháp SOA (3.57) sử dụng các quy trình phân tích hướng dịch vụ (3.46)	n/a	Được thảo luận trong Khoản 4, Khoản 11, Khoản 12, Khoản 13	
3.44. service monitoring (giám sát dịch vụ) Theo dõi tình trạng và các điều kiện vận hành liên quan đến việc thực thi,	n/a	Được thảo luận trong Khoản 4, Khoản 11, Khoản 14	

thực hiện dịch vụ (3.20) và các tác động thực tế (3.18)			
3.45. service orchestration (điều phối dịch vụ) Điều phối (3.15) trong đó có các phần tử (3.8) được điều phối là các dịch vụ (3.20)	*Khoản 11 đã thảo luận – nội dung nhất quán* *Không có định nghĩa cụ thể nhưng là mở rộng của hợp thành dịch vụ - sau đó, là "kết quả của việc kết hợp một tập các dịch vụ", do vậy , nhất quán với việc điều phối của các dịch vụ*	Được thảo luận trong Khoản 4, Khoản 8, Khoản 10	
3.46. service orientation (hướng dịch vụ) Cách tiếp cận để thiết kế hệ thống theo điều khoản dịch vụ (3.20) và phát triển dựa trên dịch vụ	*n/a*	Được thảo luận trong Khoản 4	
3.47. service oriented analysis (phân tích hướng dịch vụ) Các bước thu thập thông tin mở đầu được hoàn thành để hỗ trợ một quy trình thành phần trong quá trình mô	n/a	n/a	

hình hóa dịch vụ dẫn đến việc tạo ra một tập các dịch vụ (3.20)			
3.48. service oriented architecture (kiến trúc hướng dịch vụ) Phong cách kiến trúc hỗ trợ hướng dịch vụ (3.45) và là mô hình kiểu mẫu để xây dựng các giải pháp nghiệp vụ	Giới thiệu Kiến trúc hướng dịch vụ (SOA) là một phong cách kiến trúc, trong đó, các hệ thống nghiệp vụ và CNTT được thiết kế theo điều khoản dịch vụ sẵn sàng tại một giao diện và các đầu ra của các dịch vụ này. Một dịch vụ là một biểu diễn logic của một tập các hoạt động có đầu ra xác định, có tính khép kín, nó có thể bao gồm các dịch vụ khác nhưng người dùng dịch vụ không cần biết cấu trúc nội bộ bên trong	Được thảo luận trong phần giới thiệu, Khoản 4 và trong toàn bộ tài liệu này	
3.49. service policy (chính sách dịch vụ) Chính sách được áp dụng cho một dịch vụ (3.20)	*Chính sách dịch vụ được thảo luận nhưng không được định nghĩa rõ ràng - được sử dụng nhất quán áp dụng cho một dịch vụ và tách biệt với một hợp đồng* 12.2. Chính sách Một *policy* là một tuyên bố về hướng dẫn mà một tác nhân con người có thể	Được thảo luận trong Khoản 4, Khoản 7, Khoản 11, Khoản 13	

	tuân theo hoặc có thể hướng một tác nhân con người khác nên tuân theo. Nhà cung cấp dịch vụ có một chính sách đối với dịch vụ, một chính sách cho một dịch vụ không nhất thiết thuộc về một nhà cung cấp dịch vụ Một ưu điểm của việc tách biệt chính sách với hợp đồng dịch vụ đó là chính sách thanh toán có thể thay đổi mà không phụ thuộc vào hợp đồng dịch vụ		
3.50. service provider (nhà cung cấp dịch vụ) Thực thể cung cấp các dịch vụ (3.20) *Phần 1 giới hạn nhà cung cấp là một thực thể*	10.4 Một phần tử thực hiện (cung cấp) một dịch vụ	Được thảo luận trong Khoản 4, Khoản 7, Khoản 10, Khoản 13, Khoản 14	
3.51. service publishing/service registration (xuất bản dịch vụ/đăng ký dịch vụ) Danh mục mô tả dịch vụ trong một vị trị có thể truy cập được, ví dụ như một đăng ký/kho dịch vụ (3.74) hỗ trợ các hoạt động tìm kiếm và thu hồi mô tả,	n/a	Được thảo luận trong Khoản 4, Khoản 6, Khoản 8, Khoản 14, Khoản 15 Có một số thảo luận về xuất bản các sự kiện	

giúp cho thông tin dịch vụ luôn sẵn có và dễ nhìn thấy đối với người dùng dịch vụ (3.29)			
3.52. SOA implementation (thực hiện SOA) Phương thức và ký thuật được sử dụng để phát triển các giải pháp SOA (3.47)	n/a	n/a	
3.53. SOA maturity (mức độ trưởng thành của SOA) Đánh giá về khả năng của một tổ chức áp dụng SOA (3.47) và mức độ áp dụng hiện tại	n/a	Được thảo luận trong Khoản 4, Khoản 13	
3.54. SOA maturity model (mô hình trưởng thành SOA) Khung xác định các mục tiêu tổng thể và một phương thức để đánh giá mức độ trưởng thành về SOA của một tổ chức theo các mục tiêu đó	n/a	n/a	
3.55. SOA resource (tài nguyên SOA)	n/a	n/a	

Các phần tử (3.8) cung cấp các tài nguyên CNTT được sử dụng bởi các dịch vụ (3.20)			
3.56. SOA solution (giải pháp SOA) Các giải pháp, là một phần hoặc toàn bộ, được triển khai bằng việc áp dụng các nguyên tắc, khái niệm, phương thức và kỹ thuật SOA (3.47)	n/a (được sử dụng trong phần giới thiệu, nội dung nhất quán)	Được thảo luận trong Khoản 4, Khoản 5, Khoản 8, Khoản 9, Khoản 10, Khoản 11, Khoản 13, Khoản 14, Khoản 15	
3.57. SOA solution governance (quản trị giải pháp SOA) Cụ thể hóa việc quản trị CNTT đặc biệt là tập trung vào các chiến lược và cơ chế quản lý cho giải pháp SOA (3.56) cụ thể với người dùng cuối	n/a	Được thảo luận trong Khoản 4, Khoản 13	
3.58. SOA solution lifecycle (vòng đời của giải pháp SOA) Tập các hoạt động về kỹ nghệ phần mềm của các giải pháp SOA (3.56), bao gồm phân tích, thiết kế, thực hiện, triển khai, kiểm thử và quản lý	n/a	Được thảo luận trong khoản 4, Khoản 11	

3.60. task (tác vụ) Hành động nguyên tử (cơ bản, không thể chia nhỏ) hoàn thành một kết quả xác định (tham khảo 18384-3, 6.4)	9.4. Một tác vụ là một hành động nguyên tử (cơ bản và không thể chia nhỏ) hoàn thành một kết quả xác định	Được thảo luận trong Khoản 4, Khoản 8, Khoản 15 8.1.2: sẽ có những tác vụ có thể phân chia, điều này là không nhất quán với tính chất nguyên tử của tác vụ	BPMN 2.0
3.61. Web Services (các dịch vụ Web) Hệ thống phần mềm được thiết kế để hỗ trợ khả năng tương tác giữa các máy tính với nhau trên một mạng	n/a	Được thảo luận trong Khoản 4	

Tài liệu tham khảo

[1] ISO/IEC 19505-2, Information technology — Object Management Group Unified Modeling Language (OMG UML) — Part 2: Superstructure

[2] ISO/IEC/TR 24800-1:2007, Information technology — JPSearch — Part 1: System framework and components

[3] OASIS. Reference Model for SOA, Version 1.0, OASIS Standard, October 2006: Available from World Wide Web: http://docs.oasis-open.org/soa-rm/v1.0/soa-rm.pdf

[4] OMG. Business Process Management Notation (BPMN), see http://www.omg.org/spec/BPMN/2.0/

[5] ISO Technical Report TR9007, Concepts and Terminology for the Conceptual Schema and the Information Base

[6] OASIS. Reference Architecture for SOA Foundation, Version 1.0, OASIS Public Review Draft 1, April 2008: see docs.oasis-open.org/soa-rm/soa-ra/v1.0/soa-ra-pr-01.pdf

[7] OMG. Model Driven Architecture (MDA) Guide, Version 1.0.1, Object Management Group (OMG), June 2003: see www.omg.org/docs/omg/03-06-01.pdf

[8] OMG. Unified Modeling Language (OMG UML), Superstructure, Version 2.2, OMG Doc. No.: formal/2009-02-02, Object Management Group (OMG), February 2009: see www.omg.org/spec/UML/2.2/Superstructure

[9] OMG. SOA Modeling Language (OMG SoaML) Specification for the UML Profile and Metamodel for Services (UPMS), Revised Submission, OMG Doc. No.: ad/2008-11-01, Object Management Group (OMG), November 2008: see www.omg.org/cgi-bin/doc?ad/08-11-01

[10] OWL. Web Ontology Language, World Wide Web Consortium (W3C), February 2004: see www.w3.org/TR/owl-ref

[11] Beyond Concepts: Ontology as Reality Representation, by Barry Smith; available from http://ontology.buffalo.edu/bfo/BeyondConcepts.pdf.

[12] Std I.E.E.E. 1471-2000: IEEE Recommended Practice for Architectural Description of Software-intensive Systems (also published as ISO/IEC 42010: 2007); available from standards.ieee.org.

[13] ISO/IEC 42010: 2007, Systems and Software Engineering – Recommended Practice for Architectural Description of Software-intensive Systems; available from www.iso.org.

[14] What is an Ontology? Stanford University; available from www-ksl.stanford.edu/kst/what-is-an-ontology.html.

[15] OWL 2 Web Ontology Language (Second Edition), World Wide Web Consortium (W3C), December 2012: see http://www.w3.org/TR/owl-overview/

[16] OWL Web Ontology Language Reference, W3C Recommendation, 10 February 2004, World-Wide Web Consortium; available from www.w3.org/TR/owl-ref.