

(株)富士通研究所 & (株)RICOH
OKAR ワーキンググループ
2005年2月1日

Ontology for Knowledge Activity Resources(OKAR) ガイド

ドラフト 2005-02-01

dc:title=" Ontology for Knowledge Activity Resources(OKAR) ガイド ドラフト 2005-02-01"

dc:date="2005-02-01"

dc:creator="OKAR ワーキンググループ"

dc:right="Copyright ©2004,2005 FUJITSU LABORATORIES LTD. & RICOH COMPANY, LTD."

目次

1. はじめに.....	1
1. 1. 本書の位置付け	1
1. 2. OKAR の必要性	1
1. 3. OKAR の利用例	2
2. 設計方針	4
3. 仕様概要	5
3. 1. 名前空間	5
3. 2. 基本クラスの定義(概要)	5
3. 2. 1. 基本クラス階層	5
3. 2. 2. Agent クラス	6
3. 2. 3. Role クラス	6
3. 2. 3. Event クラス	8
3. 2. 4. Artifact クラス	8
3. 2. 5. その他(補助)のクラス	9
3. 3. 基本プロパティの定義(概要)	9
3. 3. 1. インポートしたプロパティと拡張定義	9
3. 3. 2. OKAR 独自のプロパティ	11
3. 4. インスタンスの記述例	12
3. 4. 1. 一般的な記述例	12
3. 4. 2. 組織階層の記述例	13
3. 4. 3. 人事異動の記述例	14
4. 拡張に関する概要	16
4. 1. 拡張方針、拡張の仕方	16
4. 2. 拡張例	16
4. 2. 1. クラスの拡張例	16
4. 2. 2. プロパティの拡張例	17
5. 今後の予定	19
6. 連絡先	19
7. 変更履歴	19
付録:OKAR の OWL 定義(N3 形式)	i

1. はじめに

株式会社富士通研究所(以下、富士通研究所)と、株式会社リコー(以下、リコー)は、オフィスにおける知的業務活動情報を記述するためのオントロジを開発する目的で、共同研究を開始した。本概要書は、共同研究のメンバによるワーキンググループが開発した“Ontology for Knowledge Activity Resources (以下、OKAR)”の仕様概要について述べる。

OKAR は、富士通研究所とリコーの共同研究による成果物であるが、第3者でも商用利用を含めて、無償で利用可能である。現在、ワーキンググループでは、OKAR に関する賛同者を広く募っており、本概要書に関するフィードバックを募集している。

1. 1. 本書の位置付け

2005年2月1日公開時点での本概要書の位置付けを以下に示す。

- ・ 本概要書は、ワーキンググループメンバやそれ以外でも関心のある関係者によるレビューのためのワーキングドラフトである。
- ・ 本概要書は、いつでも他のドキュメントによって更新、置換、破棄され得る。
- ・ 本概要書に関するコメントや本概要書が発行された以降の最新情報に関しては、6章のメールアドレスにご連絡いただきたい。

1. 2. OKAR の必要性

現代の企業において、企業の持つ知識の重要性が増大している。これらの知識は、共有された文書として企業内のデータベースに蓄積されることが望ましい。しかし、一般には、「業務に必要な情報の50-75%は人から得る」「企業内の情報の80%は個人PC内に存在し、従業員の退社と共に失われる」といった困難さが指摘されている (Gartner Research, Knowledge Worker Investment Paradox, 2002)。このような属人的な知識を管理するには、知識を記述した共有文書のみを情報源として管理するばかりでなく、従業員やグループが業務でどういった情報を作成したり入手したかの情報も管理し、自動的に更新していく仕組みが必要となっている。

人と情報(知識)の関係は、業務活動の中で様々な形態があり得る。最も簡単な関係として、文書と著者の関係がある。また、あるミーティングを介して、ミーティングで使われた資料とミーティングに参加した人の関係もある。さらに、誰と誰と一緒に仕事をしているかといった人と人との関係もあり、これらは全て業務活動における知識となり得る。

インターネットやブロードバンドの普及に伴い、ネットワーク上には、多種多様な機器やシステムが接続されるようになり、それらを介して、様々な情報が流れるようになった。オフィスにおいても、パソコンやプリンタの情報機器や、数多くの社内システムがネットワークに接続され、業務の効率化が図られている。こうした情報機器やシステムを組み合わせ、人が文書を提示したり、コミュニケーションしたりする行動の中にも、人と人との関係や人と情報の関係が存在する。これらの情報も

業務活動における知識となり得る。

上記のような業務活動における様々な知識を活用するためには、特定の情報機器やシステムに依存せず、また組織にも依存しない共通のフォーマット(以下、オントロジ)が必要となる。企業における様々な情報機器やシステムが、このオントロジに基づいて、企業内の業務活動情報を出し、それを蓄積することができるようになれば、様々な情報源から属人的な情報を企業内知識として自動的に管理していくことが可能となる。

そこで、今回、富士通研究所とリコーは、知的業務活動の様々な場面で現れる情報を記述するためのオントロジを共同で開発した。このオントロジを「OKAR: Ontology for Knowledge Activity Resources(仮称)」と呼んでいる。

OKARでは、業務活動における「人」や「モノ」に注目し、それらに関する基本情報と、互いの関係を記述することができる。具体的には、企業を構成する人や組織、業務で利用するシステムや機器、業務で生産されるドキュメント、業務で行なわれるミーティングなどのイベントに関する情報と、それらの間の関係を記述することができる。

OKARは、セマンティックWebにおけるオントロジ記述言語OWL(Web Ontology Language)で定義されている。今後、セマンティックWebの発展と共に、OWLに基づいたメタデータが世界的にも増えてくることが予想され、それらと容易に情報交換が可能である。

1. 3. OKARの利用例

OKARは、様々な用途への利用が考えられる。例えば、個人やプロジェクトに付随するスキル情報の管理、あるプロジェクトに関連して発生したイベント情報の管理などがある。

富士通研究所とリコーは、OKARの利用シーンとして、それぞれデモを作成し、ISWC2004(International Semantic Web Conference) Exhibition¹で展示した。

富士通研究所は、「タスクコンピューティング²」技術をベースにして、各種の機器が連携するミーティング情報管理システムを開発した。このシステムでは、RFID³タグによるミーティング出席者の管理を行なうと共に、会議室の様々な機器を連携したプレゼンテーションを容易に実現できる。同時に、「誰が誰にどういう資料をプレゼンした」「誰がどういう会議に出た」という活動情報をOKAR形式で自動的に記録する。さらに、記録した活動情報を自動分析し、「誰がどの仕事に詳しいか(スキル情報)」「誰と誰と一緒に仕事をしているか(人脈情報)」といった情報をベースとした、高度なKnowWho⁴検索を実現している。

¹ ISWC2004: <http://iswc2004.semanticweb.org/>

² タスクコンピューティング: 富士通研究所、米国富士通研究所、メリーランド大が提案をしているセマンティック Web による情報機器を意味的に連携するための枠組み。 <http://taskcomputing.org/>

³ RFID: Radio Frequency Identification. 小さな無線チップを人やモノにつけることにより、識別を行う仕組み。

⁴ KnowWho: 「誰がその知識を有しているのか」など、人を中心とした様々な業務情報を示す言葉。知識を持った人をいかに効率的に探し出すかは、企業の競争力強化にとって不可欠であり、企業内の情報共有、ナレッジマネジメントの一分野として注目されつつある。

一方、リコーは、情報機器やアプリケーションを一元化する「ドキュメントハイウェイ⁵」と連携する知識リソースの検索システムのプロトタイプを開発した。このシステムでは、ドキュメント管理システム内のプロジェクト情報や、FOAF⁶プロジェクトにより記述された主に人間関係の情報を基に、人、組織、プロジェクトといった多様なリソースが保有する知識を抽出し、特定の知識を持った人、組織、プロジェクトといった様々なリソースを検索することができる。また、企業や組織をまたがる情報を扱う際には不可欠なアクセス制御のためにXACML⁷を導入している。

⁵ ドキュメントハイウェイ：リコーが提唱する、ネットワーク上のデジタル複合機、プリンタなどの情報機器やアプリケーションを連携させ、オフィスにおけるスムーズな情報の活用・管理を可能にするプラットフォーム。 <http://www.ricoh.co.jp/src/en/highway/>

⁶ FOAF: <http://www.foaf-project.org/>

⁷ XACML: <http://www.oasis-open.org/committees/xacml/>

2. 設計方針

OKAR は、「オフィスにおける一般的な知的業務活動」を記述することを目的としている。そのため、業務活動における「人」と「モノ」が OKAR の主な記述対象である。具体的には、「活動を行なう主体(Agent)」「活動の対象物や活動によって生産された成果物(Artifact)」「主体によって行なわれる活動および行為そのもの(Event)」が記述対象となるリソースであり、OKAR では、それら 3 者間の関係を記述できるようにしている。

また、4 つ目のリソースとして、「Agent の多面性」を記述するために、Agent の「役割(Role)」を記述対象としている。例えば、「人」は複数の「組織」に所属することがあり、所属する「組織」によって、「人」は異なる「役割」を持ち得る。また、このような役割は時間によって変化する。OKAR では、役割を 1 つのリソースとして記述することで、Agent の多面性や役割の時間的推移をより正確に記述できるようにしている。

OKAR の初期バージョンでは、オフィスに出現する様々なリソースを網羅的に定義するのではなく、その核となるものだけを「基本クラス」として定義することを目的とした。基本クラスの設計に当たり、ワーキンググループでは、富士通研究所とリコーの 2 社を例として、2 社において共通に出現するリソースや、2 社間で情報交換する際に必要と思われるリソースを選別した。また、「実際の知的業務活動の事例」として、今回のワーキンググループの活動自身が記述できることを確認した。基本クラスの詳細に関しては 3 章で解説する。

ユーザは必要に応じて、OKAR の基本クラスを拡張定義することにより、より詳細な記述を行なうことができる。拡張定義を行なう場合の詳細については 4 章で解説する。

各リソースの属性や関係の記述(プロパティ)には、積極的に他の標準で規定されている語彙(以下、ボキャブラリ)を利用し、不足分だけを OKAR で定義するようにした。これは、既存アプリケーションとの相互運用性(interoperability)を考慮した結果である。

3. 仕様概要

本概要書はドラフトの位置付けであり、本章で解説する OKAR の仕様は、予告なく変更される可能性がある。本概要書が発行された以降の変更に関しては、6 章のメールアドレスにご連絡いただきたい。

3. 1. 名前空間

OKAR の名前空間を以下のように定める。

```
http://www.labs.fujitsu.com/jp/techinfo/okar/0.9#
```

OKAR の利用者は、RDF 記述の最初にて、上記の名前空間を宣言する必要がある。また、OKAR では、vCard, iCalendar, Dublin Core のボキャブラリを利用している。よって、OKAR によって記述される RDF/XML のヘッダは、一般的には以下ようになる。

```
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:okar="http://www.labs.fujitsu.com/jp/techinfo/okar/0.9#"
  xmlns:vcard="http://www.w3.org/2001/vcard-rdf/3.0#"
  xmlns:dc="http://purl.org/dc/elements/1.1/"
  xmlns:ical="http://www.w3.org/2002/12/cal/ical#"
>
...
```

3. 2. 基本クラスの定義(概要)

OKAR では、4つのコアクラスと、コアクラスから派生する(owl:subClassOf 関係にある)7つのサブクラスを定義している。これら11個のクラスを OKAR 基本クラスと呼ぶ。

3. 2. 1. 基本クラス階層

基本クラスは、Agent, Role, Event, Artifact の4つのコアクラスと、Agent のサブクラスである Person/Organization/Equipment/Software、Event のサブクラスである Action/GroupEvent、Artifact のサブクラスである Document から構成される。また、補助クラスとして、Location, PersonName の2つのクラスを定義している。

図1に基本クラス(および補助クラス)のクラス階層図を示す。楕円が各クラスを表しており、点線矢印は、owl:subClassOf を表している。

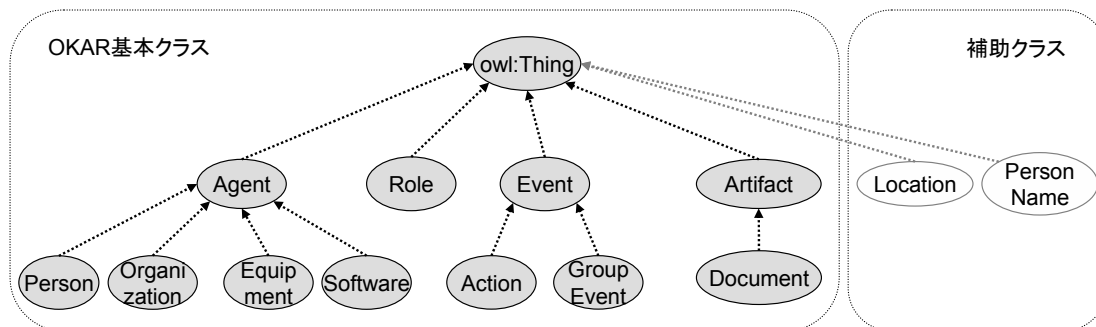


図1: 基本クラス階層

補助クラスは、他のオントロジの owl:imports(以下インポート)を想定している。例えば、PersonName は、vCard オントロジがインポート可能になった段階で、vCard のボキャブラリによって置き換えられる。

3. 2. 2. Agent クラス

Agent クラスは、「活動(Event)の主体または知識所有者となる Identity を持った物(生物,人工物を含む)」を記述するクラスである。主に、人(Person)や人の集合体である組織(Organization)が含まれ、それがサブクラスとして定義されている。また、活動(Event)の主体として、Equipment や Software やなどの無機物を含んでいる点が特徴である。

Agent クラスは、主に vCard ボキャブラリによって、付加情報を記述できる。一般的な Agent インスタンスの記述は、以下のようなになる(以下は、Person クラスの例である)。

```

<okar:Person rdf:about="#person:TaroYamada">
  <vcard:FN>Taro Yamada</vcard:FN>
  <vcard:N rdf:parseType="Resource">
    <vcard:Family>Yamada</vcard:Family>
    <vcard:Given>Taro</vcard:Given>
  </vcard:N>
</okar:Person>
  
```

Person クラスは人、Organization クラスは会社組織(一時的なグループを含む)、Equipment クラスは機器などの物理的なボディを持つ無機物、Software クラスはシステムやアプリケーションなどの物理的なボディを持たない無機物、を記述するとき使用する。

3. 2. 3. Role クラス

OKAR の最も特徴的なクラスが、Role クラスである。Role クラスは、Agent クラスと他の基本クラス(Role 以外)とを結びつける機能を持ち、両者の関係を記述する際に必ず利用される。

一般に、Agent は複数の役割(Role)を持つと考えられる。例えば、Person は同時に複数の組

織に存在しえるし、**Software** は複数の機能を実装している場合がある。各役割が全く同一ならば問題ないが、通常、各役割はそれぞれ他リソースとの関係上、異なる性質を持ちえる。**Role** クラスは、このような異なる性質を持った複数の役割を持つ **Agent** を記述するのに使用する。

図2は、1つの **Person**(“Taro Yamada”) が、2つの **Organization**(“A-division” と “B-division”) に所属している例を示している。図2は、“A-division”における“Taro Yamada”の役職が“Director”、“B-division”における役職が“Senior Researcher”であった場合を示している。

Person と **Organization** の関係を単純に1つのプロパティ(例えば、vcard:ORG など)で表現しようとすると、役職の違いが表現できない問題が発生する。この問題を解決するため、Role を用いて、それぞれの組織に属している **Role**(**Organization** から okar:member でリンクされているクラス)の vcard:TITLE にそれぞれの役職を記述できる。つまり、この場合、1つの **Role** インスタンスが1つのビジネスカードに対応することになる。

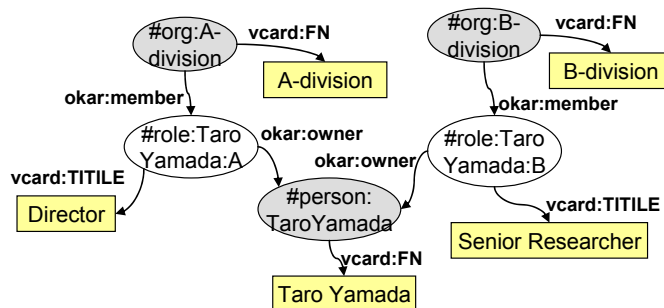


図2: 2つの組織で異なる役職を持つAgentの記述例

```
<okar:Role rdf:about="#role:TaroYamada:A">
  <okar:owner rdf:resource="#person:TaroYamada"/>
  <vcard:TITLE>Director</vcard:TITLE>
</okar:Person>
```

Role クラスは、**Agent** クラスと同様に vCard ボキャブラリによって、付加情報を記述できる。そのため、**Role** クラスの記述内容と、その基になる **Agent** クラス(okar:owner でリンクされた **Agent**)の記述内容に衝突が発生しえる。たとえば、“Taro Yamada”の **Agent** インスタンスは vcard:NICKNAME として “Taro” を持つが、C-division での **Role** インスタンスは vcard:NICKNAME として “Yamachan” を持つかもしれない。

OKAR 自体は何らそれらに対する制限を規定するものではない。記述内容の衝突は、アプリケーションでの解決が必要となるが、通常、「**Agent** の記述内容を **Role** の記述内容で上書きする」などのルールの適用が推奨される。

また、**Role** インスタンスをどのタイミングで更新するか、どのような記述のときに別の **Role** インスタンスとして記述するか、などの **Role** インスタンスの記述方針は、アプリケーションに依存する。例えば、“Taro Yamada”が昇格し、**Role** インスタンスの1つの属性値である vcard:TITLE が変更されたときに別の **Role** インスタンスとして記述してもよい。この場合、各 **Role** インスタンスに有

効期限を記述し、**Role** インスタンスを時間順に並べることにより、その **Agent** の役割の変化 (**Person** ならば履歴書相当)が記述できる。

3. 2. 3. Event クラス

Event クラスは、「誰かがある時にある場所で起こす何か」を記述するクラスである。サブクラスとして、**Agent**の単体行動である**Action**クラスと、複数の**Agent**の協調行動である**GroupEvent**クラスの2つのサブクラスがある。

Event クラスは、主に iCalendar ボキャブラリによって、付加情報を記述できる。一般的な**Event**インスタンスの記述は、以下のようになるだろう(以下は、**GroupEvent**クラスの例である)。

```
<okar:GroupEvent rdf:about="#meeting:NextProductMeeting">
  <ical:summary>Next Product Meeting</ical:summary>
  <ical:dtstart rdf:parseType="Resource">
    <ical:dateTime>2004-10-12T15:00:00</ical:dateTime>
  </ical:dtstart>
  <ical:dtend rdf:parseType="Resource">
    <ical:dateTime>2004-10-12T17:00:00</ical:dateTime>
  </ical:dtend>
  <okar:hasLocation rdf:resource="#loc:Room-B"/>
  <ical:attendee rdf:resource="#role:TaroYamada:A"/>
  <ical:attendee rdf:resource="#role:YumikoMoriyama"/>
  <ical:attendee rdf:resource="#role:KenjiKitamura"/>
</okar:GroupEvent>
```

上記は、「Next Product Meeting」という打合せに、3人の出席者がいることが記述された例である。**Event** インスタンスの記述では、「いつ(when)」「どこで(where)」「誰が(who)」「何を(what)」が記述される。後述の「3. 3. 基本プロパティの定義(概要)」にて解説するが、who を示す okar:actor や ical:attendee の取る値が **Role** クラスである点が特徴である。上記の例では、「Taro Yamada」は、「role:TaroYamada:A」(A-division の所属)の **Role** で出席していることを意味している。

3. 2. 4. Artifact クラス

Artifact クラスは、「**Agent** によって生産された成果物や何らかの行為の対象物となる人工物」を記述するクラスである。サブクラスとして、「何かしらのシンボル(TEXT やその他のフォーマット)を用いて、**Agent**の思考や命令、その他が記述されたもの」を記述する**Document**クラスを持つ。

Artifact クラスは、主に Dublin Core ボキャブラリによって、付加情報を記述できる。一般的な**Artifact**インスタンスの記述は、以下のようになるだろう(以下は、**Document**クラスの例である)。

```

<okar:Document rdf:about="#doc:NextProductTarget">
  <dc:title>Target for Next Product</dc:title>
  <dc:description>
    次期製品のターゲットをまとめた。ターゲットは大きく4つあり、…
  </dc:description>
  <dc:creator rdf:resource="#role:TaroYamada:A"/>
  <dc:date>2004-10-11</dc:date>
</okar:Document>

```

上記は、作成者が”Taro Yamada”であり、「Target for Next Product」というタイトルを持つドキュメントを記述された例である。後述の「3. 3. 基本プロパティの定義(概要)」にて解説するが、dc:creator の取る値が **Role** クラスである点が特徴である。上記の例では、”Taro Yamada”は、”role:TaroYamada:A” (A-division の所属)の **Role** でドキュメントを作成していることを意味している。

3. 2. 5. その他(補助)のクラス

OKAR では、基本クラス以外の補助クラスとして、**PersonName** と **Location** を定義している。**PersonName** は、「人の構造化された名前」を記述するクラスであり、姓(Family Name)や名(Given Name)を区別して記述するクラスである。**Location** クラスは、**Event** クラスにおけるイベントの発生場所や、vcard:ADR などによって表される住所を記述するクラスである。

これらを基本クラスとせずに補助クラスとしている理由は、他のオントロジのインポートを想定していることによる。例えば、**PersonName** は、vCard オントロジがインポート可能になった段階で、補助クラスから削除する予定である。**Location** クラスは、現時点でのインポート候補は存在しないが、良いものがあれば、それに置き換える予定である。

3. 3. 基本プロパティの定義(概要)

「2. 設計方針」でも記述したが、OKAR では他のオントロジのインポートを積極的に行なっている。これは、既存アプリケーションとの相互運用性(interoperability)を重視していることによる。OKAR では、他のオントロジで定義されていないプロパティだけを OKAR 独自のプロパティとして定義し、インポートしたプロパティの使用を推奨している。インポートしたプロパティの定義域(rdf:domain)や値域(rdf:range)は、OKAR の基本クラスに合致するよう拡張定義している。以下では、それぞれのプロパティに関する概要を解説する。

3. 3. 1. インポートしたプロパティと拡張定義

OKAR では、vCard オントロジ⁸、iCalendar オントロジ⁹、Dublin Core オントロジ¹⁰のインポートを予定している。現時点では、OWL に対応している iCalendar のみを実際にインポートし、定義

⁸ RDF vCard: <http://www.w3.org/TR/2001/NOTE-vcard-rdf-20010222/>

⁹ RDF iCalendar: <http://www.w3.org/2002/12/cal/>

¹⁰ Dublin Core: <http://dublincore.org/documents/dces/>

域(rdf:domain)や値域(rdf:range)にオリジナルと違いがあるプロパティを拡張定義している。

また、vCard, Dublin Core は、現時点で OWL に対応していないため、実際にはインポートせず、各オントロジにおける代表的なプロパティのみ定義している。

OKAR では、vCard オントロジのプロパティを **Agent** クラスと **Role** クラス、iCalendar オントロジのプロパティを **Event** クラス、Dublin Core オントロジのプロパティを **Artifact** クラスに用いるように定義している。例えば、vCard オントロジのプロパティである vcard:FN (Formatted Name)の定義域(rdf:domain)は、**Agent** クラスと **Role** クラスの owl:unionOf クラスとして定義している。Dublin Core オントロジのプロパティも同様に、それぞれの定義域(rdf:domain)を **Artifact** クラスとして定義している。iCalendar オントロジのプロパティは、OKAR の **Event** クラスを ical:Vevent クラスの owl:subClassOf と定義しているため、iCalendar オントロジの ical:Vevent クラスに使用できるプロパティが、OKAR オントロジの **Event** クラスのプロパティとして記述可能である。

それぞれのプロパティの値域(rdf:range)において、OKAR の各種クラスを値域に取るよう拡張定義しているものがある。以下、代表的な拡張定義を列挙する。

[vCard の拡張定義(rdf:range)]

- vcard:PHOTO, vcard:LOGO ⇒ okar:Document クラス
- vcard:ADR ⇒ okar:Location クラス
- vcard:BDAY ⇒ xsd:dateTime(XML スキーマの dateTime 型)
- vcard:ORG ⇒ okar:Organization クラス

[iCalendar の拡張定義(rdf:range)]

- ical:attendee, ical:organizer ⇒ okar:Role クラス
- ical:attach ⇒ okar:Artifact クラス
- ical:relatedTo ⇒ xsd:anyURI(XML スキーマの URI 型)¹¹

[Dublin Core の拡張定義(rdf:range)]

- dc:creator, dc:publisher, dc:contributor ⇒ okar:Role クラス
- dc:date ⇒ xsd:dateTime(XML スキーマの dateTime 型)
- dc:source, dc:relation ⇒ okar:Artifact クラス

ただし、それぞれのプロパティの意味の解釈は、それぞれのオントロジの意味の解釈に準じるものとする。

¹¹ "http://www.w3.org/2002/12/cal/ical#"では range が xsd:string で定義されているため拡張定義。本来ならば、okar:Event を rdf:range に定義すべきだが、ical:relatedTo が DatatypeProperty で宣言されているため、URI に限定した拡張定義を行なっている。

3.3.2. OKAR 独自のプロパティ

OKAR では、他のオントロジで定義されていないプロパティを独自で定義している。例えば、Agent や Role の発生日(誕生日)は、vCard の vcard:BDAY で記述できるが、消滅日(死亡日)に該当するプロパティが定義されていないため、okar:DDAY を独自定義している。

また、組織構成に関するプロパティをいくつか定義している(okar:member や okar:groupMember など)。

表1に、OKAR 独自のプロパティ定義をまとめる。現在、20個のプロパティが定義されている。

表1: OKAR 独自のプロパティ定義

プロパティ名	意味	rdf:domain	rdf:range	備考
DDAY	消滅日(死亡日)	Agent, Role	xsd:dateTime	
hasRole	Agent が持つ Role	Agent	Role	owner の owl:inverseOf
owner	Role のオーナー	Role	Agent	hasRole の同上
member	組織の構成員	Organization	Role(Person)	Person の Role のみ
leader	組織のリーダー	Organization	Role(Person)	member のサブプロパティ
subLeader	組織のサブリーダー	Organization	Role(Person)	同上
regulerMember	正規の組織構成員	Organization	Role(Person)	同上
temporaryMember	非正規の組織構成員	Organization	Role(Person)	同上
groupMember	下位組織	Organization	Role(Organization)	Organization の Role のみ
regularGroupMember	正規の下位組織	Organization	Role(Organization)	groupMember のサブプロパティ、transitiveProperty
temporaryGroupMember	非正規の下位組織	Organization	Role(Organization)	groupMember のサブプロパティ
relatedRole	Role 間の関係	Role	Role	
purpose	Role が持つ目的	Role	xsd:string	
hasLocation	Event の発生場所	Event	Location	脚注 ¹² 参照
actor	Event の主行為者	Event	Role	
target	Event の対象物	Event	owl:Thing	
knows	知識を持っている	Agent	owl:Thing	
mate	知り合い、仕事仲間	Person	Role(Person)	knows のサブプロパティ
roleWeight	Agent が Role に掛ける重み	Role	xsd:float	値はアプリ依存
roleRank	組織における階級	Role	xsd:float	値はアプリ依存

¹²<http://www.w3.org/2002/12/cal/ical#>の ical:location は、DatatypeProperty で定義されているため、イベントの発生場所を表すプロパティを別途定義している。

表 1 に示されるように、主に、Organization クラスと Person クラス(が持つ Role クラス)の関係、および Organization クラスと Organization クラス(が持つ Role クラス)の関係を記述するためのプロパティが定義されている。これは、企業における組織階層や組織構成を詳細に記述することを目的としたものである。

3. 4. インスタンスの記述例

本節では、OKAR インスタンスの記述例のいくつかを紹介する。主に OKAR 独自プロパティの記述例を示し、vCard, iCalendar, Dublin Core ボキャブラリの記述例は、それぞれの仕様書に譲る。

3. 4. 1. 一般的な記述例

一般的な OKAR インスタンスの記述例(RDF モデル)を図3に示す。これは、「3. 2. 基本クラスの仕様(概要)」で示した例を基にしたものであり、あるミーティングに3人の出席者がおり、1つのドキュメントがミーティング資料として使用された例を示している。3人の出席者のうち、1人(“Taro Yamada”)は2つの所属を持ち、残り2人は1つの所属(“Taro Yamada”の1つの所属と同じ)を持っている。RDF/XML 形式の記述例は、前述した「3. 2. 基本クラスの仕様(概要)」の例を参照のこと。

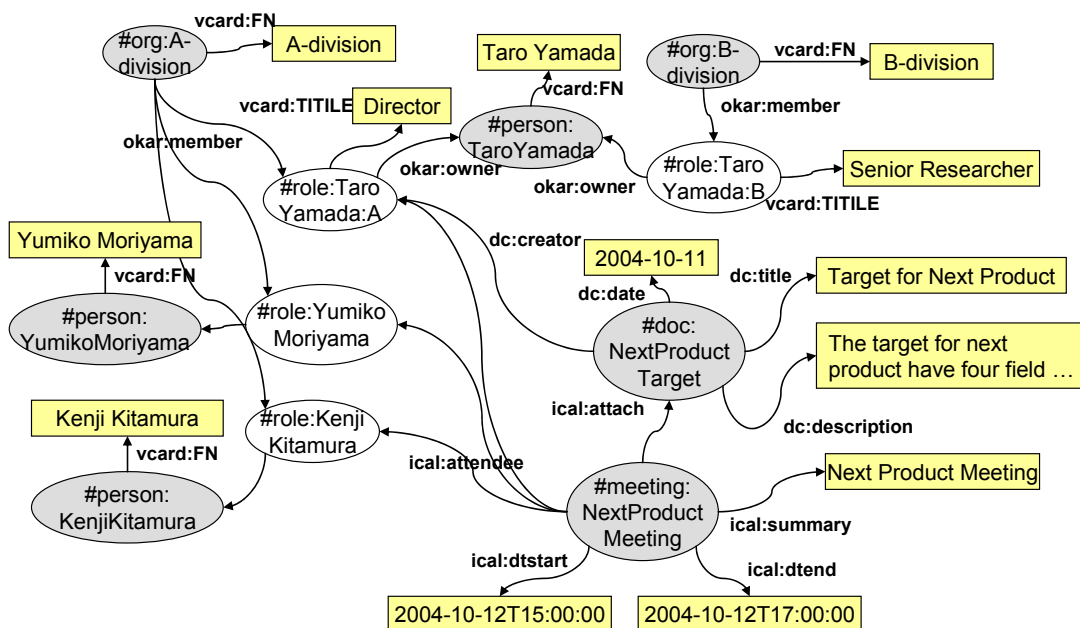


図3: 一般的なOKARインスタンスの記述例(RDFモデル)

3. 4. 2. 組織階層の記述例

組織階層は、owl:groupMember によって記述する。図4に組織階層の記述例(RDF モデル)を示す。これは、“AAA-Company”の下位組織には、“A-Division”、“B-Division”、“C-Division”の3つの部があり、“A-Division”の下位組織には、“X-Section”、“Y-Section”、“Z-Section”の3つの課があることを記述した例である。owl:groupMember の rdf:domain が **Organization** クラスであり、rdf:range が (**Organization** が okar:hasRole として持っている) **Role** クラスである点に注意されたい。

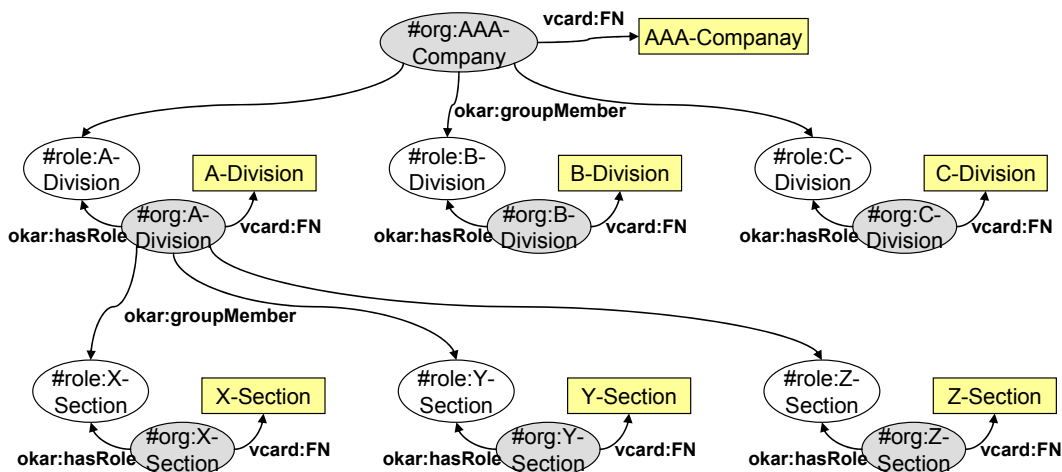


図4：組織階層のインスタンス記述例(RDFモデル)

RDF/XML 形式のインスタンス記述例は、以下のようになる。

```
<okar:Organization rdf:about="#org:AAA-Company">
  <vcard:FN>AAA Company</vcard:FN>
  <okar:groupMember rdf:resource="role:A-Division"/>
  <okar:groupMember rdf:resource="role:B-Division"/>
  <okar:groupMember rdf:resource="role:C-Division"/>
</okar:Organization>
```

```
<okar:Organization rdf:about="#org:A-Division">
  <vcard:FN>A Division</vcard:FN>
  <okar:hasRole>
    <okar:Role rdf:about="#role:A-Division"/>
  </okar:hasRole>
  <okar:groupMember rdf:resource="role:X-Section"/>
  <okar:groupMember rdf:resource="role:Y-Section"/>
  <okar:groupMember rdf:resource="role:Z-Section"/>
</okar:Organization>
```

```
<okar:Organization rdf:about="#org:X-Section">
  <vcard:FN>X Section</vcard:FN>
  <okar:hasRole>
    <okar:Role rdf:about="#role:X-Section"/>
  </okar:hasRole>
</okar:Organization>
```

3. 4. 3. 人事異動の記述例

組織の構成員は、owl:member によって記述する。owl:member の rdf:domain が Organization であり、rdf:range が (Person クラスが okar:hasRole として持っている) Role クラスである点に注意されたい。

通常の所属情報ならば、図3(一般的な OKAR インスタンスの記述例)のような単純な形となるが、人事異動を記述する場合には、複数の Role インスタンスを記述し、それぞれの Role インスタンスの有効期限を記述すればよい。図5に人事異動の記述例 (RDF モデル) を示す。これは、“A-Division” の “Director” であった “Taro Yamada” が、2004/10/25 に “C-Division” の “Senior Director” に人事異動したことを記述した例である。

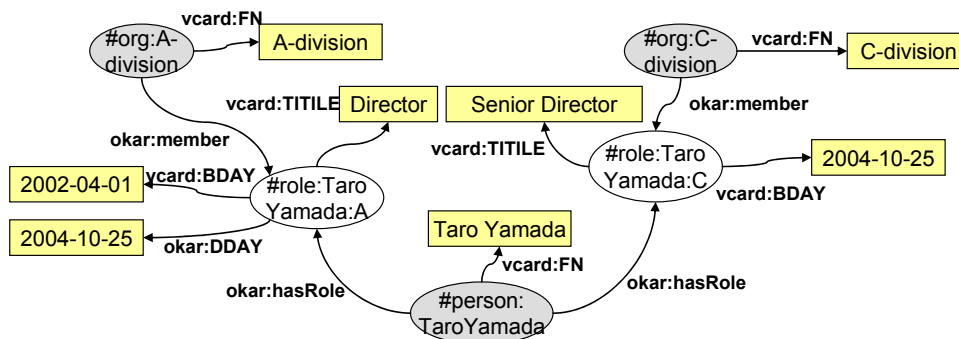


図5: 人事異動のインスタンス記述例 (RDFモデル)

RDF/XML 形式のインスタンス記述例は、以下のようなになる。

```
<okar:Organization rdf:about="#org:A-Division">
  <vcard:FN>A Division</vcard:FN>
  <okar:hasRole>
    <okar:Role rdf:about="#role:A-Division"/>
  </okar:hasRole>
  <okar:member rdf:resource="role:TaroYamada:A"/>
  <okar:member rdf:resource="role:YumikoMoriyama"/>
  <okar:member rdf:resource="role:KenjiKitamura"/>
  ...
</okar:Organization>

<okar:Organization rdf:about="#org:C-Division">
  <vcard:FN>C Division</vcard:FN>
  <okar:hasRole>
    <okar:Role rdf:about="#role:C-Division"/>
  </okar:hasRole>
  <okar:member rdf:resource="role:TaroYamada:C"/>
  ...
</okar:Organization>
```

```
<okar:Person rdf:about="#Person:TaroYamada">
  <vcard:FN>Taro Yamada</vcard:FN>
  <okar:hasRole>
    <okar:Role rdf:about="#role:TaroYamada:A">
      <vcard:TITLE>Director</vcard:TITLE>
      <vcard:BDAY>2002-04-01</vcard:BDAY>
      <okar:DDAY>2004-10-25</okar:DDAY>
    </okar:Role>
  </okar:hasRole>
  <okar:hasRole>
    <okar:Role rdf:about="#role:TaroYamada:C">
      <vcard:TITLE>Senior Director</vcard:TITLE>
      <vcard:BDAY>2004-10-25</vcard:BDAY>
    </okar:Role>
  </okar:hasRole>
</okar:Person>
```

組織変更の場合も、組織間の関係の変更のみであれば、人事異動の記述と同様に複数の **Role** インスタンスを用いることにより記述できる。組織の分割が生じる変更の場合には、新しい **Organization** インスタンスを用いて記述すればよい。

4. 拡張に関する概要

4. 1. 拡張方針、拡張の仕方

OKAR のボキャブラリによる記述よりも、さらに詳細な記述を行ないたい場合は、ユーザは OKAR の定義を個別に拡張定義してよい。拡張定義は、OKAR の基本クラスの派生クラスか、OKAR のプロパティの派生プロパティとして定義することが望ましい。これは、異なるユーザ間で拡張定義された複数の記述に対しても、OKAR の基本クラスやプロパティによって、統一的な意味の解釈を与えるためである。もし、拡張定義において、OKAR 以外のオントロジをインポートし、混合した記述を行ないたい場合、可能な限り OKAR オントロジとのマッピングを定義することが推奨される。OKAR オントロジにマッピングすることができないか、あるいはマッピングが定義されていないボキャブラリは、OKAR を解釈するプロセッサにおいて無視されることを想定しなければならない。

尚、拡張定義を行なう場合は、OKAR オントロジをインポートし、必要な拡張定義を行なえばよい。

4. 2. 拡張例

本節では、いくつかの OKAR の拡張定義例を示す。尚、本節のサンプル定義における"&okar;"や"&local;"は文字エンティティであり、それぞれ OKAR の名前空間や拡張定義したオントロジの名前空間を意味している。具体的な記述例は以下のようになる。

```
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [
<!ENTITY okar "http://www.labs.fujitsu.com/jp/techinfo/okar/0.9#">
<!ENTITY local "http://somewhere/LocalOKAR#">
]>
<rdf:RDF xmlns:okar="&okar;" xmlns:local="&local;" ...>

  <owl:Ontology rdf:about="http://somewhere/LocalOKAR">
    <!-- OKAR オントロジの import -->
    <owl:imports
      rdf:resource="http://www.labs.fujitsu.com/jp/techinfo/okar/0.9" />
  </owl:Ontology>
  ...

```

4. 2. 1. クラスの拡張例

「2. 設計方針」に示したように、OKAR の初期バージョンでは、「オフィスにおける業務活動の中で核となるもの」のみを基本クラスとして定義している。よって、それ以外のクラスを記述したい場合には、基本クラスのサブクラスとして拡張定義すればよい。

例えば、複数の Agent が協調して行なうイベントは **GroupEvent** クラスしか定義されておらず、アプリケーションによっては、「討議目的のミーティング」と「親睦目的のパーティ」を区別して記述

したい場合もあるだろう。このような場合、**GroupEvent** クラスのサブクラスとして、ミーティング (**local:Meeting**)やパーティ(**local:Party**)を拡張定義すればよい。以下に、拡張定義の例を示す。

```
<owl:Class rdf:about="&local;Meeting">
  <rdfs:comment>討議目的のミーティングを表わすクラス</rdfs:comment>
  <owl:subClassOf rdf:resource="&okar;groupEvent"/>
</owl:Class>

<owl:Class rdf:about="&local;Party">
  <rdfs:comment>親睦目的のパーティを表わすクラス</rdfs:comment>
  <owl:subClassOf rdf:resource="&okar;groupEvent"/>
</owl:Class>
```

その他、現在、**Artifact** クラスのサブクラスには、**Document** クラスしか定義されていないが、**Artifact** クラスのサブクラスには様々なものが想定される。代表的なものとしては、製品があるだろう。この場合も、上記のミーティングの拡張例と同様に、**Artifact** クラスのサブクラスとして、製品クラス(**local:Product**)を拡張定義すればよい。以下に、拡張定義の例を示す。

```
<owl:Class rdf:about="&local;Product">
  <rdfs:comment>製品を表わすクラス</rdfs:comment>
  <owl:subClassOf rdf:resource="&okar;Artifact"/>
</owl:Class>
```

4. 2. 2. プロパティの拡張例

「3. 4. 3. 人事異動の記述例」で示したように、人事異動の記録を全て記述しようとする、ある組織が存在している間に短期間でも在籍した全ての構成員が **okar:member** として記述されることになる。

例えば、「現在の組織構成員」だけを求めようとする、現在の OKAR のボキャブラリだけで記述されたメタデータでは、

1. **okar:member** の値域となる **Role** インスタンスを参照
2. 各 **Role** インスタンスの **okar:DDAY** の値をチェック
3. 現在でも有効な **Role** インスタンスか確認

といった処理が必要になる。このような場合、**okar:member** のサブプロパティとして、「現在、有効な組織構成員」を表わすプロパティ(**local:currentMember**)を拡張定義すればよい。以下に、拡張定義の例を示す。

```
<owl:ObjectProperty rdf:about="&local;currentMember">
  <rdfs:comment>現在、有効な組織構成員を表わす拡張プロパティ</rdfs:comment>
  <owl:subPropertyOf rdf:resource="&okar;member"/>
</owl:ObjectProperty>
```

Person インスタンスに関しても `okar:member` の例と同様のことが言え、「現在の有効な役割」を素早く参照したい場合には、`okar:hasRole` のサブプロパティとして、「現在の有効な役割」を表わすプロパティ(`local:currentRole`)を拡張定義すればよい。また、「現在の主な役割」を表わすプロパティ(`local:mainRole`)を拡張定義してもよい。以下に、拡張定義の例を示す。

```
<owl:ObjectProperty rdf:about="&local;currentRole">
  <rdfs:comment>現在の有効な役割を表わす拡張プロパティ</rdfs:comment>
  <owl:subPropertyOf rdf:resource="&okar;hasRole"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="&local;mainRole">
  <rdfs:comment>現在の主な役割を表わす拡張プロパティ</rdfs:comment>
  <owl:subPropertyOf rdf:resource="&okar;hasRole"/>
</owl:ObjectProperty>
```

さらに別のアプリケーションでは、ある役割について、誰がその役割を命令したのかを明らかにしたいことも考えられる。このような場合には `okar:relatedRole` のサブプロパティとして「任命した関係」を表わすプロパティ(`local:appoint`)を拡張定義すればよい。以下に拡張定義の例を示す。

```
<owl:ObjectProperty rdf:about="&local;appoint">
  <rdfs:comment>任命の関係を表わす拡張プロパティ</rdfs:comment>
  <owl:subPropertyOf rdf:resource="&okar;relatedRole"/>
</owl:ObjectProperty>
```

5. 今後の予定

OKAR ワーキンググループでは、OKAR の今後のよりよい発展のために、本概要書に関するコメントを広く募っている。OKAR に興味のある方は、是非、次章のメールアドレスにご連絡いただきたい。

また、以下の URL にて、OKAR に関する公開している。ホームページでは、OKAR の最新仕様、最新情報を公開するだけでなく、FOAF (friend of a friend)などの既存オントロジとのマッピングツールや OKAR エディタなど、いくつかのツールの公開を企画している。

富士通研究所: <http://www.labs.fujitsu.com/jp/techinfo/okar/>

リコー: http://www.rioh.co.jp/src/lab/lab_us_uc3.html

6. 連絡先

(株)富士通研究所

IT メディア研究所 知能システム研究部

E-mail: okar-question@ml.labs.fujitsu.com

(株)リコー

ソフトウェア研究開発本部 ユビキタスソリューション研究所 UC 研究センター

E-mail: okar-question@src.rioh.co.jp

7. 変更履歴

- Draft 2004-11-10 から Draft 2005-02-01 の変更点
 - 3.1 節「名前空間」: OKAR URI の変更(URN⇒URL)
 - 5 節「今後の予定」: OKAR ホームページ情報の追加

付録: OKAR の OWL 定義 (N3 形式)

以下に OKAR の OWL 定義を示す。紙面節約のため、以下では N3 形式を用いているが、RDF/XML 形式の OWL 定義は、Web 上で公開している。

```

@prefix okar: <http://www.labs.fujitsu.com/jp/techinfo/okar/0.9#> .
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix dcterms: <http://purl.org/dc/terms/> .
@prefix ical: <http://www.w3.org/2002/12/cal/ical#> .
@prefix vcard: <http://www.w3.org/2001/vcard-rdf/3.0#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

### Ontology Header
< http://www.labs.fujitsu.com/jp/techinfo/okar/0.9>
  rdf:type owl:Ontology ;
  rdfs:label "Ontology for Knowledge Activity Resources (OKAR) vocabulary" ;
  owl:imports <http://www.w3.org/2002/12/cal/ical> .

### Class

okar:Agent rdf:type owl:Class ;
  rdfs:subClassOf [
    rdf:type owl:Restriction ;
    owl:onProperty vcard:FN ;
    owl:minCardinality "1"^^xsd:nonNegativeInteger
  ] ;
  owl:disjointWith okar:Role, okar:Event, okar:Artifact, okar:Location,
    okar:PersonName .

okar:Person rdf:type owl:Class ;
  rdfs:subClassOf okar:Agent ;
  owl:disjointWith okar:Organization, okar:Equipment, okar:Software .

okar:Organization rdf:type owl:Class ;
  rdfs:subClassOf okar:Agent ;
  owl:disjointWith okar:Person, okar:Equipment, okar:Software .

okar:Equipment rdf:type owl:Class ;
  rdfs:subClassOf okar:Agent ;
  owl:disjointWith okar:Person, okar:Organization, okar:Software .

okar:Software rdf:type owl:Class ;
  rdfs:subClassOf okar:Agent ;
  owl:disjointWith okar:Person, okar:Organization, okar:Equipment .

okar:Role rdf:type owl:Class ;
  rdfs:subClassOf [
    rdf:type owl:Restriction ;
    owl:onProperty okar:owner ;
    owl:cardinality "1"^^xsd:nonNegativeInteger
  ] ;
  owl:disjointWith okar:Agent, okar:Event, okar:Artifact, okar:Location,
    okar:PersonName .

```

```
okar:Event rdf:type owl:Class ;
  rdfs:subClassOf ical:Vevent ;
  owl:disjointWith okar:Agent, okar:Role, okar:Artifact, okar:Location,
    okar:PersonName .

okar:Action rdf:type owl:Class ;
  rdfs:subClassOf [
    rdf:type owl:Restriction ;
    owl:onProperty okar:actor ;
    owl:maxCardinality "1"^^xsd:nonNegativeInteger
  ] ;
  rdfs:subClassOf okar:Event ;
  owl:disjointWith okar:GroupEvent .

okar:GroupEvent rdf:type owl:Class ;
  rdfs:subClassOf okar:Event ;
  owl:disjointWith okar:Action .

okar:Artifact rdf:type owl:Class ;
  owl:disjointWith okar:Agent, okar:Role, okar:Event, okar:Location,
    okar:PersonName .

okar:Document rdf:type owl:Class ;
  rdfs:subClassOf okar:Artifact .

okar:Location rdf:type owl:Class ;
  owl:disjointWith okar:Agent, okar:Role, okar:Event, okar:Artifact,
    okar:PersonName .

okar:PersonName rdf:type owl:Class ;
  owl:disjointWith okar:Agent, okar:Role, okar:Event, okar:Artifact,
    okar:Location .

### Property

# vCard Properties
vcard:FN rdf:type owl:DatatypeProperty ;
  rdfs:domain [ owl:unionOf (okar:Agent okar:Role) ] ;
  rdfs:range xsd:string .

vcard:N rdf:type owl:ObjectProperty ;
  rdfs:domain okar:Person ;
  rdfs:range okar:PersonName .

vcard:Family rdf:type owl:DatatypeProperty ;
  rdfs:domain okar:PersonName ;
  rdfs:range xsd:string .

vcard:Given rdf:type owl:DatatypeProperty ;
  rdfs:domain okar:PersonName ;
  rdfs:range xsd:string .

vcard:Other rdf:type owl:DatatypeProperty ;
  rdfs:domain okar:PersonName ;
  rdfs:range xsd:string .

vcard:Prefix rdf:type owl:DatatypeProperty ;
  rdfs:domain okar:PersonName ;
  rdfs:range xsd:string .

vcard:Suffix rdf:type owl:DatatypeProperty ;
  rdfs:domain okar:PersonName ;
  rdfs:range xsd:string .
```

```

vcard:NICKNAME rdf:type owl:DatatypeProperty ;
  rdfs:domain [ owl:unionOf (okar:Agent okar:Role) ] ;
  rdfs:range xsd:string .

vcard:PHOTO rdf:type owl:ObjectProperty ;
  rdfs:domain [ owl:unionOf (okar:Agent okar:Role) ] ;
  rdfs:range okar:Document .

vcard:LOGO rdf:type owl:ObjectProperty ;
  rdfs:domain [ owl:unionOf (okar:Agent okar:Role) ] ;
  rdfs:range okar:Document .

vcard:ADR rdf:type owl:ObjectProperty ;
  rdfs:domain [ owl:unionOf (okar:Agent okar:Role) ] ;
  rdfs:range okar:Location .

vcard:TEL rdf:type owl:ObjectProperty ;
  rdfs:domain [ owl:unionOf (okar:Agent okar:Role) ] .

vcard:EMAIL rdf:type owl:ObjectProperty ;
  rdfs:domain [ owl:unionOf (okar:Agent okar:Role) ] .

vcard:BDAY rdf:type owl:DatatypeProperty, owl:FunctionalProperty ;
  rdfs:domain [ owl:unionOf (okar:Agent okar:Role) ] ;
  rdfs:range xsd:dateTime .

vcard:ORG rdf:type owl:ObjectProperty, owl:FunctionalProperty ;
  owl:inverseOf okar:owner ;
  rdfs:domain [
    owl:intersectionOf (okar:Role [
      rdf:type owl:Restriction ;
      owl:onProperty okar:owner ;
      owl:allValuesFrom okar:Person
    ] ) ] ;
  rdfs:range okar:Organization .

vcard:TITLE rdf:type owl:DatatypeProperty, owl:FunctionalProperty ;
  rdfs:domain okar:Role ;
  rdfs:range xsd:string .

vcard:ROLE rdf:type owl:DatatypeProperty, owl:FunctionalProperty ;
  rdfs:domain okar:Role ;
  rdfs:range xsd:string .

vcard:NOTE rdf:type owl:DatatypeProperty ;
  rdfs:domain [ owl:unionOf (okar:Agent okar:Role) ] ;
  rdfs:range xsd:string .

vcard:CLASS rdf:type owl:DatatypeProperty, owl:FunctionalProperty ;
  rdfs:domain [ owl:unionOf (okar:Agent okar:Role) ] ;
  rdfs:range xsd:string .

vcard:KEY rdf:type owl:DatatypeProperty ;
  rdfs:domain [ owl:unionOf (okar:Agent okar:Role) ] ;
  rdfs:range xsd:string .

# OKAR Properties
okar:DDAY rdf:type owl:DatatypeProperty, owl:FunctionalProperty ;
  rdfs:domain [ owl:unionOf (okar:Agent okar:Role) ] ;
  rdfs:range xsd:dateTime .

okar:roleWeight rdf:type owl:DatatypeProperty, owl:FunctionalProperty ;
  rdfs:domain okar:Role ;
  rdfs:range xsd:float .

```

```
okar:roleRank rdf:type owl:DatatypeProperty, owl:FunctionalProperty ;
  rdfs:domain [
    owl:intersectionOf (okar:Role [
      rdf:type owl:Restriction ;
      owl:onProperty okar:owner ;
      owl:allValuesFrom okar:Person
    ] ) ] ;
  rdfs:range xsd:float .

okar:knows rdf:type owl:ObjectProperty ;
  rdfs:domain okar:Agent .

okar:mate rdf:type owl:ObjectProperty ;
  rdfs:subPropertyOf okar:knows ;
  rdfs:domain okar:Person ;
  rdfs:range [
    owl:intersectionOf (okar:Role [
      rdf:type owl:Restriction ;
      owl:onProperty okar:owner ;
      owl:allValuesFrom okar:Person
    ] )
  ] .

okar:hasRole rdf:type owl:ObjectProperty ;
  owl:inverseOf okar:owner ;
  rdfs:domain okar:Agent ;
  rdfs:range okar:Role .

okar:owner rdf:type owl:ObjectProperty ;
  owl:inverseOf okar:hasRole ;
  rdfs:domain okar:Role ;
  rdfs:range okar:Agent .

okar:member rdf:type owl:ObjectProperty ;
  rdfs:domain okar:Organization ;
  rdfs:range [
    owl:intersectionOf (okar:Role [
      rdf:type owl:Restriction ;
      owl:onProperty okar:owner ;
      owl:allValuesFrom okar:Person
    ] )
  ] .

okar:leader rdf:type owl:ObjectProperty ;
  rdfs:subPropertyOf okar:member .

okar:subLeader rdf:type owl:ObjectProperty ;
  rdfs:subPropertyOf okar:member .

okar:regularMember rdf:type owl:ObjectProperty ;
  rdfs:subPropertyOf okar:member .

okar:temporaryMember rdf:type owl:ObjectProperty ;
  rdfs:subPropertyOf okar:member .

okar:groupMember rdf:type owl:ObjectProperty ;
  rdfs:domain okar:Organization ;
  rdfs:range [
    owl:intersectionOf (okar:Role [
      rdf:type owl:Restriction ;
      owl:onProperty okar:owner ;
      owl:allValuesFrom okar:Organization
    ] )
  ] .
```

```

okar:regulerGroupMember rdf:type owl:ObjectProperty, owl:TransitiveProperty ;
  rdfs:subPropertyOf okar:groupMember .

okar:temporaryGroupMember rdf:type owl:ObjectProperty ;
  rdfs:subPropertyOf okar:groupMember .

okar:relatedRole rdf:type owl:ObjectProperty, owl:TransitiveProperty ;
  rdfs:domain okar:Role ;
  rdfs:range okar:Role .

okar:purpose rdf:type owl:DatatypeProperty ;
  rdfs:domain okar:Role ;
  rdfs:range xsd:string .

okar:hasLocation rdf:type owl:ObjectProperty ;
  rdfs:domain okar:Event ;
  rdfs:range okar:Location .

okar:actor rdf:type owl:ObjectProperty ;
  rdfs:domain okar:Event ;
  rdfs:range okar:Role .

okar:target rdf:type owl:ObjectProperty ;
  rdfs:domain okar:Event .

# iCalendar Properties
ical:relatedTo rdf:type owl:DatatypeProperty ;
  rdfs:domain okar:Event ;
  rdfs:range xsd:anyURI .

ical:attendee rdf:type owl:ObjectProperty ;
  rdfs:domain okar:GroupEvent ;
  rdfs:range okar:Role .

ical:organizer rdf:type owl:ObjectProperty ;
  rdfs:domain okar:GroupEvent ;
  rdfs:range okar:Role .

ical:attach rdf:type owl:ObjectProperty ;
  rdfs:domain okar:GroupEvent ;
  rdfs:range okar:Artifact .

# Dublin Core Properties
dc:title rdf:type owl:DatatypeProperty ;
  rdfs:domain okar:Artifact ;
  rdfs:range xsd:string .

dc:creator rdf:type owl:ObjectProperty ;
  rdfs:domain okar:Artifact ;
  rdfs:range okar:Role .

dc:subject rdf:type owl:ObjectProperty ;
  rdfs:domain okar:Artifact .

dc:description rdf:type owl:DatatypeProperty ;
  rdfs:domain okar:Artifact ;
  rdfs:range xsd:string .

dc:publisher rdf:type owl:ObjectProperty ;
  rdfs:domain okar:Artifact ;
  rdfs:range okar:Role .

dc:contributor rdf:type owl:ObjectProperty ;
  rdfs:domain okar:Artifact ;
  rdfs:range okar:Role .

```

```
dc:date rdf:type owl:DatatypeProperty ;
  rdfs:domain okar:Artifact ;
  rdfs:range xsd:dateTime .

dc:format rdf:type owl:DatatypeProperty ;
  rdfs:domain okar:Artifact ;
  rdfs:range xsd:string .

dc:source rdf:type owl:ObjectProperty ;
  rdfs:domain okar:Artifact ;
  rdfs:range okar:Artifact .

dc:relation rdf:type owl:ObjectProperty ;
  rdfs:domain okar:Artifact ;
  rdfs:range okar:Artifact .

dc:identifier rdf:type owl:DatatypeProperty ;
  rdfs:domain okar:Artifact ;
  rdfs:range xsd:anyURI .

dc:type rdf:type owl:ObjectProperty ;
  rdfs:domain okar:Artifact .

dc:language rdf:type owl:DatatypeProperty ;
  rdfs:domain okar:Artifact ;
  rdfs:range xsd:string .

dc:coverage rdf:type owl:DatatypeProperty ;
  rdfs:domain okar:Artifact ;
  rdfs:range xsd:string .

dc:rights rdf:type owl:DatatypeProperty ;
  rdfs:domain okar:Artifact ;
  rdfs:range xsd:string .
```