Feedback on FND

Status

This feedback is based on the version of FIBO BE & FND & FND that I got from David Newman on December 3, 2015. Some this may need to be updated.

Request for Clarification

IndependentParty vs. AutonomousAgent

What if any relationship is there between IndependentParty and AutonomousAgent? It makes sense that every IndependentParty is an AutonomousAgent, but that subclass relationship is not there.

Occurrence, OccurrenceKind & TransactionEvent

- *Occurrence* and *OccurrenceKind* are not clearly distinct. Why is a Transaction event not just a subclass of Occurrence? It fits the definition.

Payment and Payment Event

- What is the difference - the English definition does not adequately distiguish the meanings of these two.

Recommended Changes

Definition not match axioms

- oc:Occurrence definition says there is both a date and a location, but there is no restriction for a date. Should be added.
  "*An Occurrence is a happening of an OccurrenceKind. Each Occurrence has a DateTimeStamp, which identifies when the Occurrence happened, and a Location*"

Making things simpler

Contract and ContractTermsSet

- The LOAN SMEs want the ability to directly link a contract to a single term, not requiring an intermediate ContractTermSet. However it is represented, an individual term usually/always amounts to a specification of an obligation that applies to one or both parties.
- I suggest ContractTermSet be subclass of fnd-arr-arr:Collection linked to individual terms using rel:hasMember. EK suggested as much in a comment.
- Is it ok for a Contract to have no terms? If not, then add hasTerms min 1 restriction.
- there is a restriction: [rel:hasPart only ContractTermsSet]. It is highly peculiar that a set of terms and conditions, if it had a part, that part could only be another SET of terms and conditions. More natural to say a TermSet hasMember [an individual] Term.
- Is a term of a contract text, or is it the obligation itself? This is analogous to the distinction between a creative work like “Moby Dick” vs. the specific rendition as text. The commitment is analogous to the work, the contract term is text.

Restrictions that should be min 0

Not all Occurrences have a location – oc:Occurrence has the restriction: fibo-fnd-plc-loc:isLocatedAt exactly 1 fibo-fnd-plc-loc:Location where Location can be either vitural of physical. However, not all Occurrences can be said to occur at some location. For example, consider a requst from a lender to a credit agency for a credit report. The request might be send via email, or it might be direct to another sytem via an API. In neither case is there any virtual location that can usefully be said to be where the request happened.

Naming Conventions for MediatingThings, ThingsInRole & RelativeThings & connecting properties

There is a property called isProducedBy which is supposed to only be used with RelativeThings. For the record, the definition is currently: “*The target or range of this property should be read as always being some kind of 'relative thing', that is a thing defined in some context. Generally this will be a 'party in role'. This property is not intended to be used to relate a thing to some independent thing which it is provided by, only to something in the role of being that which provides it.*”

That means if one wants to say MichaelUschold provides a Document, you have to create a fictitious instance that means MichaelUscholdAsProvider. To avoid that, I need another property for isProvidedBy(provides) that connects directly to MichaelUschold. What should that be, and how can it be made obvious that it connects to the other version of the property?

There is another problem. isProvidedBy and isUsedBy use the exact same grammatical form for the name. Yet, one is for exclusive use with Relative things, and the other is not. It would be nice if there was a hint about this. Given that use with relative things should be the exception rather than the rule, we could have a convention to highlight that the propery is for use with only Relative Things. The same solution can apply to subclasses of ThingInRole.

Common names like Asset and Bank cannot be used in the common sense way because they are subclasses of a the class: ThingInRole. There are two things:

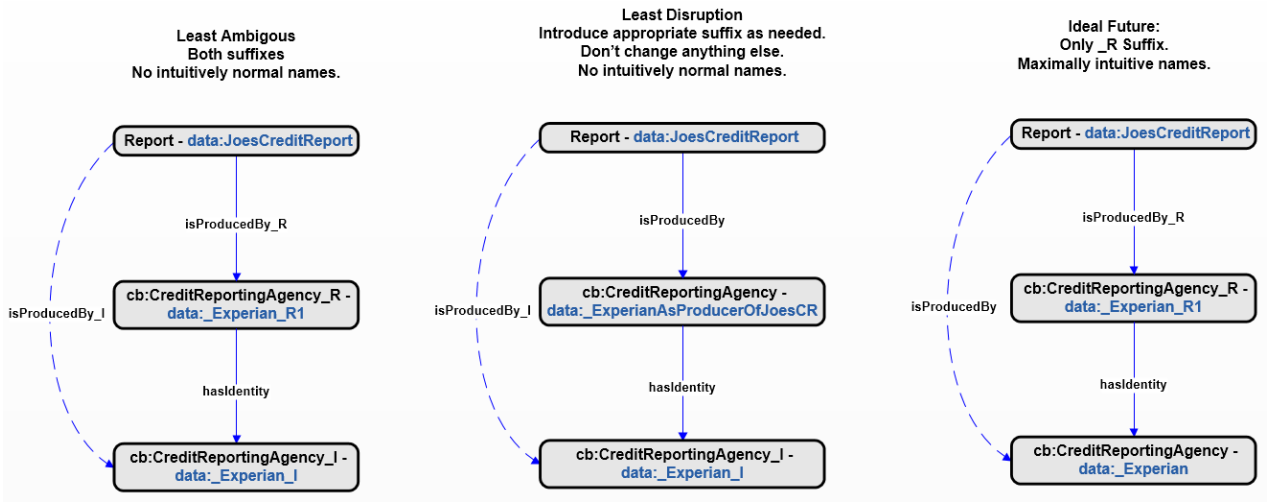
1. An independent party that is a corporation that provides banking services. We use the term “Bank” for this in everyday parlance. It is a real thing.
2. A party in the role of a bank: an independent corporation providing banking services. There is no everyday parlance for this, it is a fiction created to solve complex modeling scenarios about roles that occasionaly arise . FIBO calls this a Bank.

This is backwards, the thing with no common name should have a name that gives some hint that it is the fictitious thing that means PartyInTheRoleOfABank, and the common everyday thing that people call a bank should be called Bank in FIBO. ThingsInRole: should be hidden from view, and only used on occasion, not baked into FIBO infrastructure.

Proposed Solution:

IDEALLY: have the non-ThingInRole name be the usual name that you expect, e.g. isProducedBy, and if you want one that is only to be used for ThingsInRole then append _R to the name. e.g. isProducedBy_R. This way you can see that the two properties are linked, one is defined as a property chain using the other. Subclasses of ThingInRole, like Asset and Bank should be renamed to Asset_R and Bank_R and Bank and Asset should be used to mean what every one will think they mean. In this world, there may be a handful of classes and properties that have an _R suffix, and most will not.

ALTERNATIVE: A less disruptive (and hopefully temporary) option that can be implemented immediately is to use the suffixes as needed when creating a version that is not there. For example, I created a property called isProducedBy_I for what I needed above. Now it is clearly linked to the other version. All the ThingInRole classes like Bank and Asset can have their IndependentParty versions called Bank_I & Asset_I. I needed to have a new Class called CreditRepository, but I sure don't want the _R version. For now, given the current naming convention, I will create a class called CreditRepository_I and not bother creating the _R version since I cannot imagine a time when it would be needed.



This example points out something else. The definition tells us to only use the isProducedBy property when the property is acting in the producer role. In this case they are producing the credit report in their role as a credit reporting agency, not more norrowly as a producer. Experian might produce something else but not in their role as a CreditReportingAgency, this would require yet another role instance for Experian. Yet another reason to avoid mediating things whenver possible.

ReFactoring Suggestions

- **Suggest move the hasName data property to the relations ontology.** Many things that are not agents have names . Then there may be no need for relations ontology to be importing agents ontology. It seems a bit backwards.
- **Suggest move identifies(isIdentifiedBy) properties to the relations ontology.** Many things that are not agents have identifiers. Then the agents ontology can import the relations ontology, like most other ontologies do.
- **Suggest change name of rel:hasUniqueIdentifier to rel:hasUniqueString** to avoid confusion with the class, Identifier.
- **Suggest move fibo-fnd-qt-qtu:isDerivedFrom to relations ontology** so it can be used for other things, like deriving information products from accounts.

Properties with too narrow domain and/or range

rel:refersTo property has too narrow domain (Reference)

The property refersTo in the relations ontology has an overly narrow domain: Reference. Every Reference refersTo something, but a person or a document or a report can refer to something, and it is not true that individual persons, documents or reporets are examples of " a concept that refers to (or stands in for) another concept".

For loans, I have a class called PublicRecord for bankruptcies, tax liens, wage garnishments etc. These are legal documents that often refer to specific accounts. I want to use the refersTo property but this will generate an untrue inference. Given the dearth of disjoint axioms, this will probabaly not be caught by inference, but we don't want incorrect inferences, just the same. I doubt we need two refersTo properties, one specific to Referencee and one more general.

There are many other examples of problems.

- Consider subproperty rel:characterizes. This causes many odd things to be inferred to be subclass of Reference. These include Funds, SecuritiesTransaction, RegistrationScheme, AccountSpecificServiceAgreement, SystemOfUnits, AccountingTransactionEvent and Catalog. A Catalog contains things that refer to products, but it does not itself refer to anything, it IS something.
- Consider subproperty rel:appliesTo. Many thing that are not references can apply to other things. A rule or law applies to specific circumstances. This infers every Rule and Law into the class Reference. It infers a SystemOfUnits to be a subclass of Reference. This has to be wrong. Other inferred subclasses of Reference: Funds, SecuritiesTransaction, RegistrationScheme, AccountSpecificServiceAgreement, AccountingTransactionEvent
- Every Classifier is inferred to be a Reference. So the classifier for mortgage purpose has instances say, home improvement, home purchase and refinance. That means these instances are "concept(s) that refer or stand in for anothere concept. What concept does a Classifier stand in for, other than the trivial sense in which every thing in owl is stands in for an concept in the world being modelled.
- A PublicRecord of say a bankruptcy, is inferred to be a Reference if I want to say that PublicRecord refers to an account, (which is does!).

Proposed solution: just remove the domain for refersTo, and from all other properties that have indeterminately broad potential usage.

doc:hasExpirationDate has too narrow domain (Document)

It is currently not possible for an agreement to have an expiration date, because only Documents are allowed to have expiration dates

Proposed solution: just remove the domain for doc:hasExpirationDate

doc:hasDateOfIssuance has too narrow domain (Document)

It is currently not possible for anything to be issued unless it is a Document, thus the current ontology requires every financial instrument (which is currently a WrittenContract) to also be a Document. Is that desirable?

Proposed solution: just remove the domain for doc:hasDate

fd:hasCount has too narrow domain (RegularSchedule)

Data property: hasCount only applies to a RegularSchedule, but many things can be counted, e.g. the number of entries in a report, or number of members of a collection.

Proposed solution: just remove the domain for fd:hasCount

rel:uses has too narrow domain (AutonomousAgent)

The domain for uses is AutonomousAgent. Unlike the other examples, this one immediately seems reasonable and defensible from an intuitive and ontological perspective. Yet from a practical perspective, the domain is still too narrow. Outside of finance per se you can say an Argument uses a line of reasoning or that ontology uses a design pattern. This is convenient and useful (as it were), even though strictly, the agent creating the argument or the ontology is doing the direct using. In finance, there is a CreditScoring Occurrence whereby an algorithm is used to compute the credit score. Strictly the agent doing the work was using the model. But there is no reason to introduce an Agent here, that information is never tracked. I want to use the rel:uses property to link the credit scoring occurrence directly to the model. In English this is fine, it makes sense to say algorithm X was used by this Occurrence of coming up with a credit score. Lots of things might use something else, and it is hard to say what they all might be, so probably best to remove the domain. An alternate strategy is to try soemthing like Agent OR Occurrence (and anything else we can thyink of) and then if we run inference one day and find a problem we can keep extending the domain as needed. Not clear what the pros/cons are.

Proposed solution: just remove the domain for rel:uses

rel:isMemberOf has too narrow domain

isMemberOf should be general, now there is need to create a isGenericMemberOf.
UPDATE: this may have been fixed now.

Suggested Changes to class and property hierarchies

Changes to class hierarchy

Changes to property hierarchy

Check for date properties that should be a subproperty of fd:hasDate. For example:

- ctr:hasEffectiveDate,
- doc:hasDateOfIssuance
- doc:hasExpirationDate (maybe via fd:hasEndDate)

Make corp:hasDateOfIncorporation a subproperty of (a new property called hasCreationDate which is a subproperty of) fd:hasStartDate

Recommended Additions

Chunks of ontology

1. for representing communication, which includes request, reponse, from/to and content of the communication. MU has some clear ideas abou this to be shared at a future date.
2. ordering products is a special kind of request., link with FND products and services.

Classes

MarketCategory:

Definition: Specifies the market domain in which a product is offered. This is a superclass of LoanMarketCategory which is: specifies the market domain in which the loan product is offered.

RiskAssessment

A subclass of oc:Occurrence that means the event of determining the risk of something. For loans, we have a subclass called CreditRiskAssessment, which entails among other things, getting a credit report.

InformationProduct

For things like credit reports. A Product that is informational in nature, rather than physical.

PostalAddress

There needs to be a way to spell out details of an address, city, street etc, ideally beyond just a string. Cities and zip codes are real things.

Properties

identifies:

should be functional (or isIdentifiedBy inverse functional).

conformsTo:

To be in alignment or agreement or obeying some kind of specification, e.g. terms in a master agreement. Also policies, conditions, guidelines, a recipe.

hasCreationDate (make it a subproperty of hasStartDate)

For reports, contracts, and many other things.

asOfDate a subproperty of hasDate

Used for credit reports. Not necessarily the same as creation date, which could be later.
ALSO: ctr:effectiveDate should probably be a subproperty of asOfDate, since it means effective "as of" a given date.

onBehalfOf

Relates a party to the party for whom the are doing something. E.g a point of contact is acting as such on behalf of a company. It might be this is quite general for FND, or one might want it in BE?

datatype property for version indicator

hasVersion string valued.