Overview

- Associations
- Debt (SysML1 & SST)
  - Paid (SST February)
  - Unresolved (still paying interest)
- Proposals
- Summary
Assoc “Ends” (textual syntax)

- KerML “end” features are actually participant features.
  - Values are the things being linked, exactly one each.
  - Better term? Needs to be short, not an abbreviation.
Assoc “Ends” (abstract syntax)

- Same term in the metamodel.
“End” multiplicity is about **cross features** ..., ... which **might not exist**.

- **Useful** for one-way and non-navigable associations.
- **Looks like participant** multiplicity (if you know “end”).
“End” multiplicity redundant and possibly conflicting
- between “end” and cross features.
- In textual and abstract syntax.

Same for ordering and uniqueness
“End” changes feature multiplicity semantics...
- ...to be about cross features.
- Number of participant feature values not restricted.
KerML Associations
(instances of metamodel)

assoc Selection {
  end cart: ShoppingCart[0..*];
  end selectedProduct: Product[0..*];
  feature info: SelectionInfo [1..*];
}
Every cart and product must satisfy cross multiplicity.
– via Selection links.
Assoc Semantics (cross multiplicity)

- **Informal** text currently.
- Applies only to *instances of association* (links), rather than instances of the associated classifiers.

[7.4.5] If an association end has a multiplicity specified other than 1..1, then this is interpreted as follows: For each association end, the multiplicity, ordering and uniqueness constraints specified for that end apply to each set of instance of the association that have the same (single) values for each of the other ends. For a binary association, this corresponds to the multiplicity resulting from "navigating" across the association given a value at one end of the association to the other end of the association.

[8.4.4.5.1] If an associationEnd has a declared multiplicity other than 1..1, then this shall be interpreted as follows: For an Association with $N$ associationEnds, consider the $i$-th associationEnd $e_i$. The multiplicity, ordering and uniqueness constraints specified for $e_i$ apply to each set of instances of the Association that have the same (singleton) values for each of the $N-1$ associationEnds other than $e_i$. 
Assoc Semantics (cross multiplicity)

assoc Selection {
   end linkedCart: ShoppingCart[0..1];
   end linkedProduct: Product[1..*];
}

Model (M1)

Things Being Modeled (M0)

Misses this violation

Catches this violation

Semantics only applies to association instances

Cross features (if any)

Participant features

KERML-40
Nothing in the textual or abstract syntax for them.

- Cross-subsetting chain pattern is modeler option.
- API access can’t depend on it, eg, for automated analysis.
Association Debt Summary

1. Misleading textual syntax
   – Term: “end” for participant rather than cross features   KERML-34
   – Cross feature multiplicity looks like participant’s’.   KERML-26

2. Two meanings for feature multiplicity
   – Requires special casing in modeling/analysis tools.   KERML-37
   – Redundant cross feature multiplicity   KERML-38

3. Incomplete semantics
   – Number of participant values not restricted (should be [1]).
   – Cross multiplicity semantics is informal and incomplete.

4. Cross features not identified   KERML-41
   – Subsetting pattern is non-standard (tool builders and modelers cannot depend on it).
Overview

- Associations
- Debt (SysML1 & SST)
  - Paid (SST February)
  - Unresolved (still paying interest)
- Proposals
- Summary
Keyword and cross multiplicity position change.
- Leaves (textual) room for participant multiplicity in library.
Assoc Participant Abstract Syntax (Proposal)

- Same term in the metamodel.
No change in feature multiplicity semantics …
- … to be about cross features.
- Number of participant feature values restricted to 1.
Assoc Participant
(proposed instances of
metamodel)

Selection : Association

assoc Selection { 
  end cart: ShoppingCart[1];
  end selectedProduct: Product[1];
  feature info: SelectionInfo [1..*]; 
}
Assoc Cross Feature Semantics (How?)

- How constrain all instances collectively?
  - Of association and its associated classifiers
- Usually done in math semantics
  - Trying to use math only in Core.
Cross Feature Modeled Semantics (Proposal)

Model (M1)

assoc Selection {
    linkEnd [0..1] linkedCart: ShoppingCart;
    linkEnd [1..*] linkedProduct: Product;
}

linkEnd linkedCart: ShoppingCart
    subsets linkedProduct. (linkedCart::inCart);

feature inCart: ShoppingCart [0..1] featured by Product;

linkEnd linkedProduct: Product
    subsets linkedCart. (linkedProduct::selectedProduct);

feature selectedProduct: Product [1..*] featured by ShoppingCart

- Cross features in participant namespace.
Cross Feature Redundancy (Proposal)

- Move cross features to associated classifiers
  - As UML/SysML tools currently do with “association-owned ends”.

```plaintext
Model (M1)

classifier ShoppingCart {  
  feature selectedProduct: Product [1..*];
}

classifier Product {  
  feature inCart: ShoppingCart [0..1];
}

assoc Selection {  
  linkEnd [0..1] linkedCart: ShoppingCart  
  subsets linkedProduct.inCart;
  ⊕ feature inCart: ShoppingCart [0..1] featured by Product;
  linkEnd [1..*] linkedProduct: Product  
  subsets linkedCart.selectedProduct;  
  ⊕ feature selectedProduct: Product [1..*] featured by ShoppingCart;
}
```

KERML-36
Cross Feature Abstract Syntax (Proposal)

- Participant (link end) features identify cross features
  - As UML/SysML associations currently do with assoc ends.
Participant, Identify Cross Subsets (Proposal)  

Metamodel (M2)

For binary associations:
• source must be link end feature that identifies a cross feature
• target must chain through other link end feature then cross feature

Model (M1)

assoc Selection {
  linkEnd [0..*] linkedCart: ShoppingCart across inCart;
  linkEnd [0..1] linkedProduct: Product across selectedProduct;
}
“N”-aries ( > 2 participants)

assoc Selection {
  linkEnd linkedCart: ShoppingCart [1] subsets participant;
  linkEnd linkedProduct: Product [1] subsets participant;
  linkEnd linkedAccount: Account [1] subsets participant;
  feature redefines participant: Anything [3] }  

assoc Link {
  feature participant: Anything [2..*]; }  

- Specialize Link & participant directly.
“N”-aries (> 2 participants) “Navigation”

- Association end (cross) multiplicity, ordering, and uniqueness apply to “navigation” from n-1 things.
  - Generalized from n=2.
Domains are sequences of “n-1” data types.

```plaintext
datatype CartProductPair specializes OrderedPair { 
    redefines element1: Cart; 
    redefines element2: Product; }
}

datatype CartAccountPair specializes OrderedPair { 
    redefines element1: Cart; 
    redefines element2: Account; }
}

datatype ProductAccountPair specializes OrderedPair { 
    redefines element1: Product; 
    redefines element2: Account; }
}

datatype OrderedPair specializes OrderedCollection { 
    feature redefines elements [2]; 
    feature element1 [1] = elements#(1); feature element2 [1] = elements#(2); }
```
N-ary Assoc Modeled Semantics (Proposal)

Model (M1)

assoc Selection {
  linkEnd [0..1] linkedCart: ShoppingCart;
  linkEnd [1..*] linkedProduct: Product;
  linkEnd [1] linkedAccount: Account;
}

Parses to

assoc Selection {
  linkEnd linkedCart: ShoppingCart;
  – feature inCart: ShoppingCart [0..1] featured by ProductAccntPair;
  linkEnd linkedProduct: Product;
  – feature selectedProduct: Product [1..*] featured by CartAccntPair;
  linkEnd linkedAccount: Account;

Same for ordering and uniqueness

N-ary Assoc Modeled Semantics (Proposal)

```
assoc Selection {
    linkEnd [0..1] linkedCart: ShoppingCart;
    linkEnd [1..*] linkedProduct: Product;
    linkEnd [1] linkedBuyer: Account;
}
```

Same for ordering and uniqueness

```
linkEnd linkedCart: ShoppingCart
    subsets (linkedProduct, linkedAccount). (linkedCart::inCart);
    ⊕ feature inCart: ShoppingCart [0..1] featured by ProductAccntPair;

linkEnd linkedProduct: Product
    subsets (linkedCart, linkedAccount). (linkedProduct::selectedProduct);
    ⊕ feature selectedProduct: Product [1..*] featured by CartAccntPair;

linkEnd linkedAccount: Account
    subsets (linkedCart, linkedProduct). (linkedBuyer::productBuyer);
```
For n-ary associations:
• source must be link end feature that identifies a cross feature
• target must chain through sequence of other “n-1” link end values, then cross feature

assoc Selection {
  linkEnd [0..1] linkedCart: ShoppingCart across inCart;
  linkEnd [1..*] linkedProduct: Product across selectedProduct;
  linkEnd [1] linkedAccount: Account across chargedAccount;
}