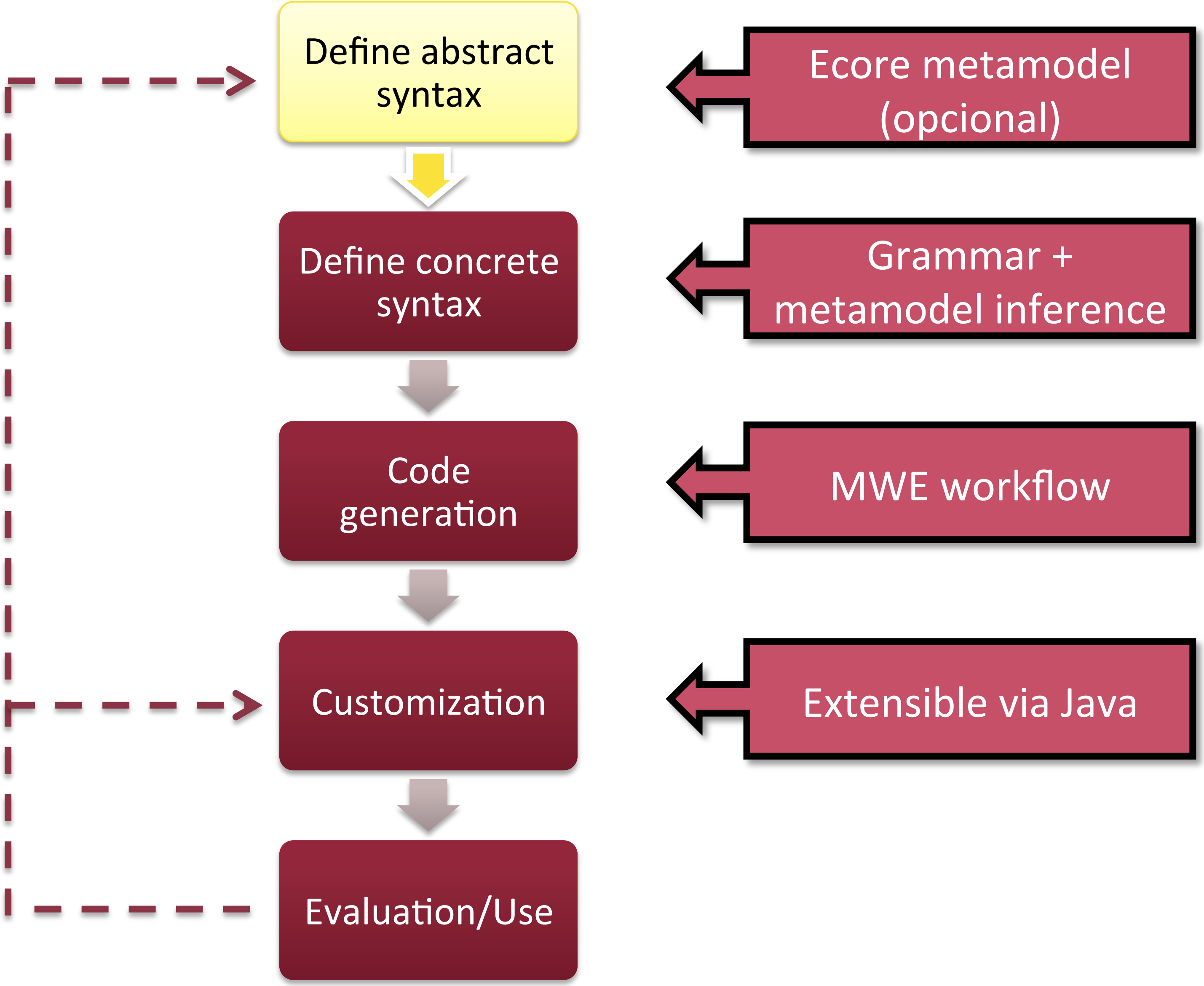
Textual DSL Creation III.

Xtext – Advanced capabilities

Previously

- Xtext workflow
- Grammar specification
- AST inference
- Basic customization (in lab)

Usage process



Define abstract syntax

Define concrete syntax

Code generation

Customization

Evaluation/Use

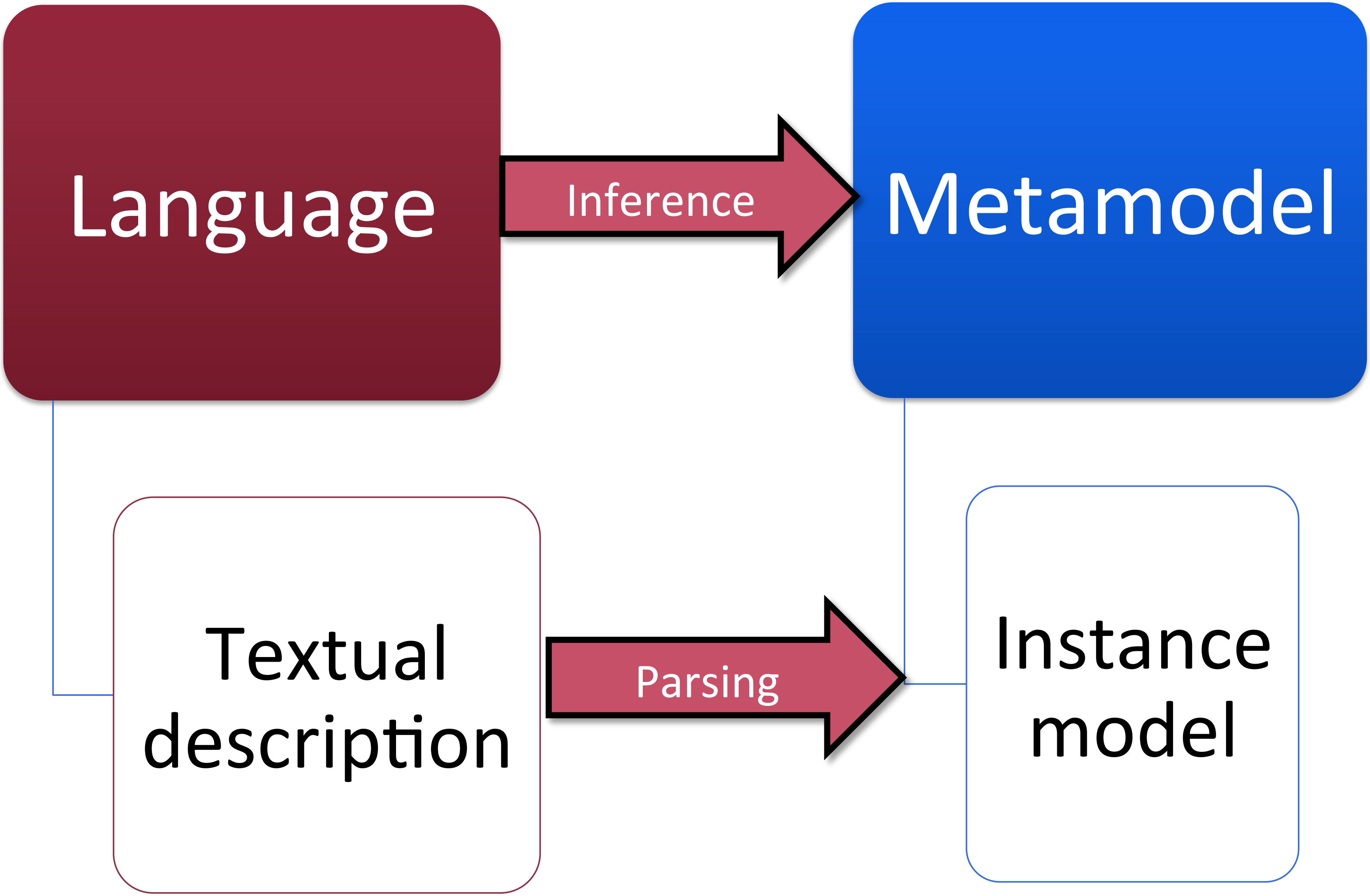
Ecore metamodel (opcional)

Grammar + metamodel inference

MWE workflow

Extensible via Java

Languages and models



Language

Inference

Metamodel

Textual description

Parsing

Instance model

Customization

Customization

- Re-defineable services
  - **Scoping: to help link resolution**
  - **Formatting: rule-based pretty printing**
  - **Validation: well-formedness constraint validation**
  - Content Assist: e.g. filtering
  - Labeling: based on JFace Label Provider API
  - Outline: Outline view customization
  - Quick Fix: suggesting automatic fixes

Scoping

- Problem:
  - Variables/objects have scope defined
  - Scoping is language-dependant
  - Xtext default
    - Only container and sibling object available

Scoping

- Problem:
  - Variables/objects have scope defined
  - Scoping is language-dependant
  - Xtext default
    - Only container and sibling object available

```
community BME {
  community FTSRG
}
person UjhelyiZoltan {
  member …
}
```

Scoping

- Problem:
  - Variables/objects have scope defined
  - Scoping is language-dependant
  - Xtext default
    - Only container and sibling object available

```
community BME {
   community FTSRG
}
person UjhelyiZoltan {
   member …
}
```

Which communities are visible here?

Scoping

- Problem:
  - Variables/objects have scope defined
  - Scoping is language-dependant
  - Xtext default
    - Only container and sibling object available

```
community BME {
  community FTSRG
}
person UjhelyiZoltan {
  member …
}
```

Which communities are visible here?

Only BME

Scoping service

- Problem
  - List all possible targets for a selected reference
  - Ignore name
- Implementation
  - Global scoping
    - For references crossing files
    - Usually based on explicit import declarations
  - Local scoping
    - For listing targets from the current file
    - **Declarative, rule-based approach**

Declarative scoping

- AbstractDeclarativeScopeProvider class
  - Two kind of generic solutions
    - `IScope scope_<RefDeclaringEClass>_<Reference>`
      `(ContextType> ctx, EReference ref)`
    - `IScope scope_<TypeToReturn>`
      `(<ContextType> ctx, EReference ref)`
  - Similar to visitor pattern
    - But without all the required methods in base class
      - Parent classes does not list all possible scope_* methods
      - Selected via (EMF) reflective API

Declarative scoping

- Methods used to
  - Enumerate all available items
  - Parameters:
    - Source of reference (context, or declaring item)
    - Type of reference (EClass)

Declarative scoping: Example

```
public IScope scope_Person_membership(final Person context,
EReference reference) {
  //Naive scoping
  SocialNetwork network = (SocialNetwork) context.eContainer();
  Iterable<Community> communities =
       Iterables.filter(network.getEntities(),Community.class);
  List<Community> communityList = new ArrayList<Community>();
  for (Community community : communities) {
   addChildren(communityList, community);
  }
  return Scopes.scopeFor(communityList);
}
```

Google Collections

- Google Guava
  - http://code.google.com/p/guava-libraries/
- Functional-style collection operations
  - Closures
  - Lazy evaluation
- Preconditions

Validation

- Well-formedness validation
  - After parsing/resolution finishes
  - Approach
    - Any EMF-based approach might work (e.g. OCL)
    - Xtext provides a rule-based validator

Java-based validator

- Validation rule: a method
  - Annotated with @Check
    - Parameter selects how often it should be validated
  - Single parameter: a type from the AST
- Base class defines warning/error methods
  - EMF metamodel literals are used to locate error
  - Error code can be added
  - Additional information might be added
    - Untyped object array
    - Usable by quick fixes

Validator example

```
@Check
public void noNameCollision(Community entity) {
  noNameCollision(entity, entity.eContainer().eContents(),
SocialNetworkPackage.Literals.SOCIAL_ENTITY__NAME);
}

private void noNameCollision(EObject eObject, List<EObject>
siblings, EStructuralFeature nameFeature) {
  String name = (String) eObject.eGet(nameFeature);
  for (EObject sibling : siblings) {
       if(name.equals(sibling.eGet(nameFeature)) && eObject !=
sibling) {
       error("Duplicate name", nameFeature.getFeatureID());
       }
  }
}
```

Formatter

- Automatic code gormatting:
  - Inserts white space characters
- When to execute
  - Started by user
  - After AST editing during serialization
- Formatting rules
  - Selecting lexer tokens
  - Adds selected white spaces
  - Defined in Java

Selection primitives

- after(token)
- before(token)
- around(token)
- between(token1, token2)
- bounds(token1, token2)
- range(token1, token2)

White space insertion

- setIndentationIncrement
- setIndentationDecrement
- setLinewrap
- setSpace
- setNoSpace

Formatting example

```
protected void configureFormatting(FormattingConfig c) {
  SocialNetworkGrammarAccess access =
(SocialNetworkGrammarAccess) getGrammarAccess();

  SocialNetworkElements sne = access.getSocialNetworkAccess();
  c.setLinewrap(1, 1, 1).after(sne.getLeftCurlyBracketKeyword_2());
  c.setLinewrap(1, 1, 1).before
          (sne.getRightCurlyBracketKeyword_5());
  c.setLinewrap(1, 1, 1).after(sne.getEntitiesAssignment_3());
  c.setLinewrap(2, 2, 2).before
          (sne.getAcquaintancesAcquaintanceParserRuleCall_4_0());
  c.setIndentationIncrement().after
          (sne.getLeftCurlyBracketKeyword_2());
  c.setIndentationDecrement().before
          (sne.getRightCurlyBracketKeyword_5());
```

Service Registration in Xtext

Xtext Service Registration

Xtext Service Registration

- Based on dependency injection
  - Google Guice: http://code.google.com/p/google-guice/
- Uses Generation gap pattern extensively
  - Generic and Generated implementations of services
  - Custom implementations can be injected

Xtext Service Registration

- Based on dependency injection
  - Google Guice: http://code.google.com/p/google-guice/
- Uses Generation gap pattern extensively
  - Generic and Generated implementations of services
  - Custom implementations can be injected

Dependency Injection

Dependency Injection

- Depedency injection design pattern
  - A class should not instantiate its dependencies
  - Instead settable via setters or constructor parameters
- DI frameworks available
  - Annotation-based injection
  - Management of injectable services/components

Dependency Injection

- Depedency injection design pattern
  - A class should not instantiate its dependencies
  - Instead settable via setters or constructor parameters
- DI frameworks available
  - Annotation-based injection
  - Management of injectable services/components

Dependency Injection: Example

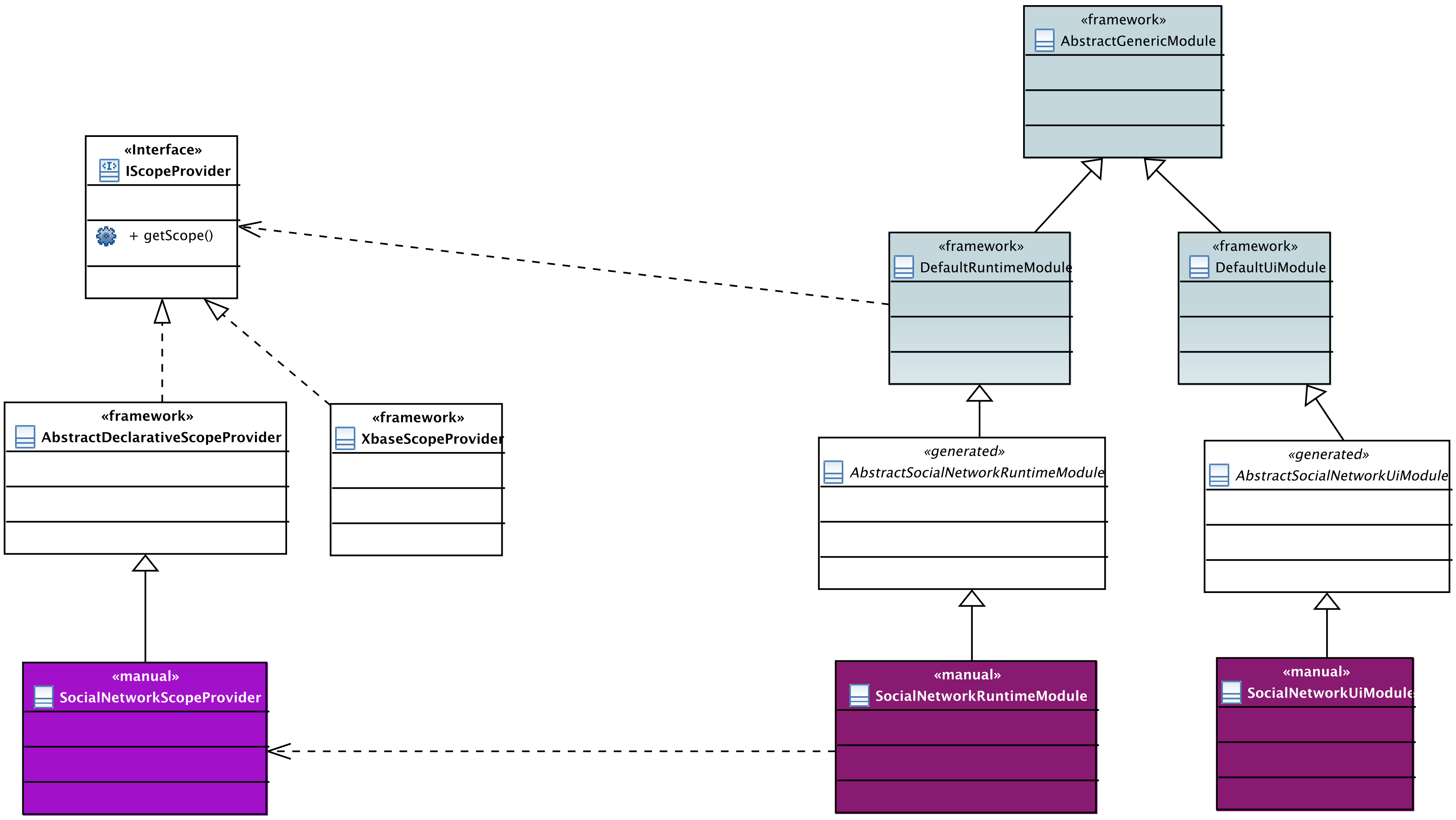
```
public interface IGenerator {

  /**
   * @param input - the input for which to generate resources
   * @param fsa - file system access to be used to generate files
   */
  public void doGenerate(Resource input, IFileSystemAccess fsa);

}
```

File System Access:
abstraction of file operations

Dependency Injection in Xtext – 1.

- Injectable services are configured in **modules**
  - RuntimeModule
    - Core services
    - Works in any Java application
  - UIModule
    - Editor, user interface
    - Eclipse-specific services
  - TestModule
    - Test implementations
    - Might include sources

Registering Services



Dependency Injection in Xtext – 2.

- Abstract module parents generated
- Overridable descendant available
  - Custom implementation registration
  - Custom service registration

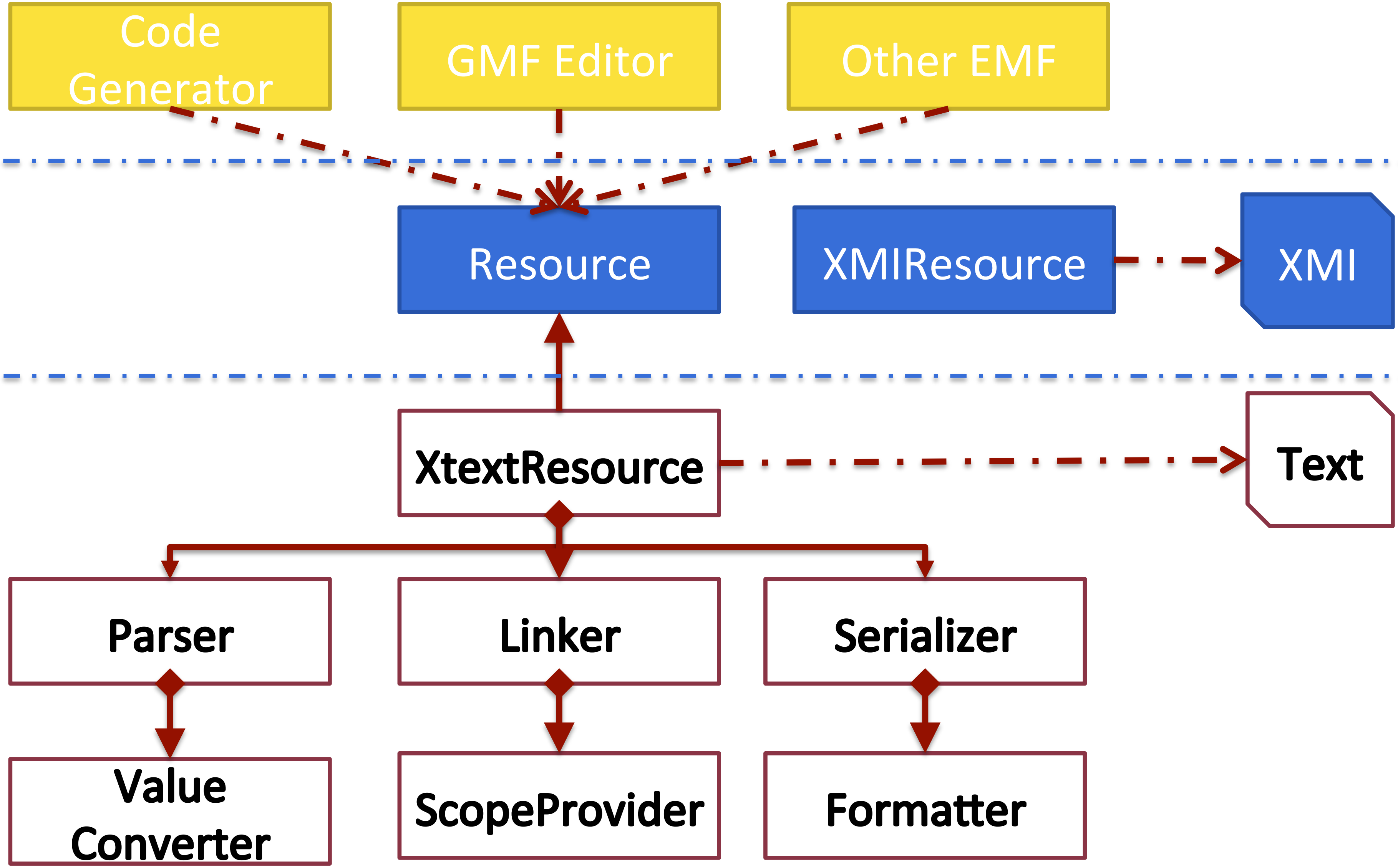
Xbase

Xbase

- Expression language
  - Embeddable into custom DSL (base language)
- Code generator via inferred JVM model
  - Generated code always in Java
- Extra features
  - Debugger
  - IDE links

Integration with EMF technologies

Integration with EMF technologies



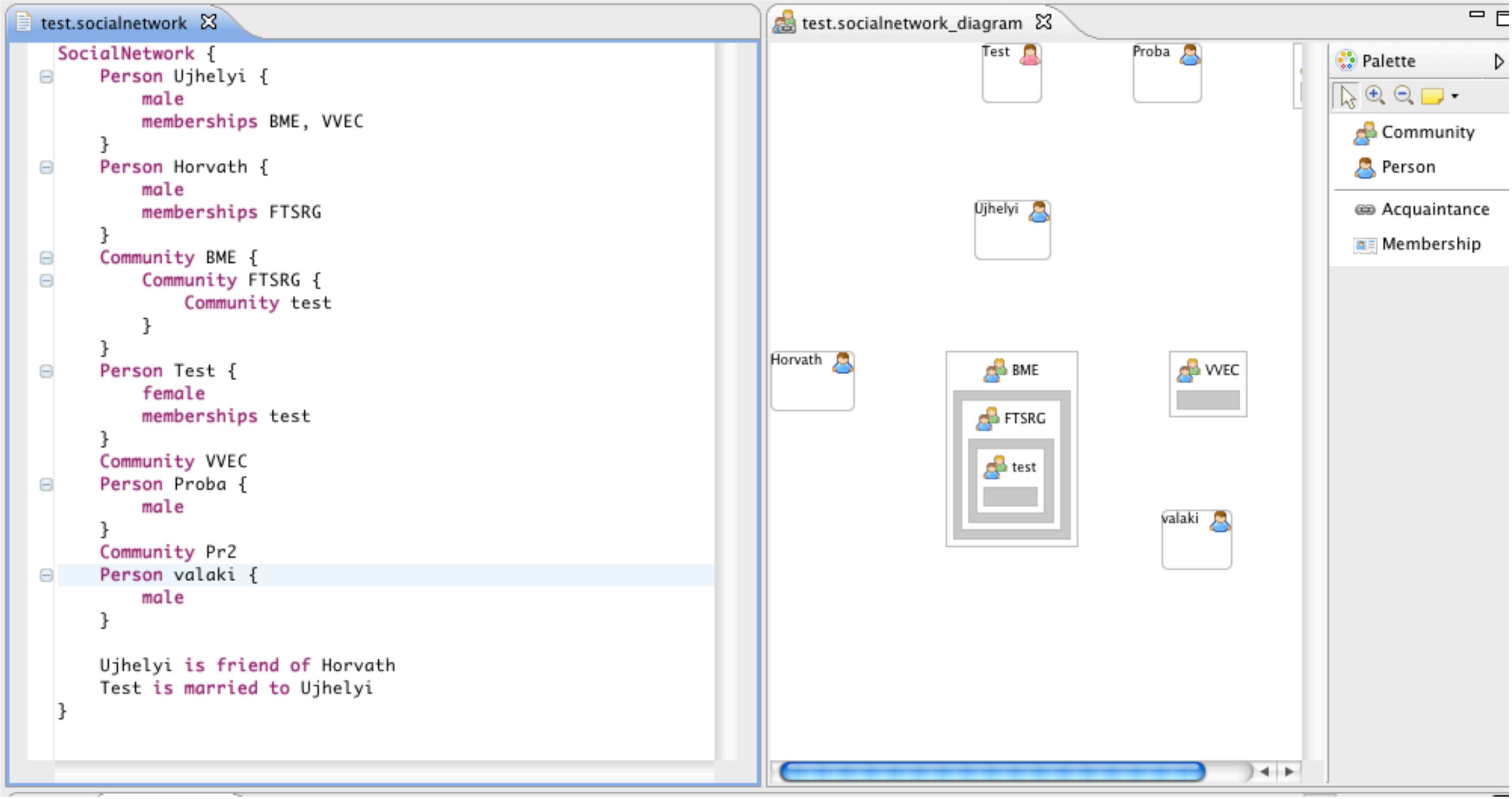
Integration

- Xtext
  - Custom EMF resource implementation
  - Registered via EMF extension point
    - org.eclipse.emf.ecore.extension_parser
  - Can be handled transparently
    - Except when using Xbase ☹
    - In case of Xbase additional registration required

GMF editor

- Domain model can be Xtext resource
  - Automatic synchronization on save
- GMF editor must maintain serializability!
  - GMF does not know grammar
  - Luckily, not much is required here

GMF and Xtext editor



test.socialnetwork

```
SocialNetwork {
    Person Ujhelyi {
        male
        memberships BME, VVEC
    }
    Person Horvath {
        male
        memberships FTSRG
    }
    Community BME {
        Community FTSRG {
            Community test
        }
    }
    Person Test {
        female
        memberships test
    }
    Community VVEC
    Person Proba {
        male
    }
    Community Pr2
    Person valaki {
        male
    }

    Ujhelyi is friend of Horvath
    Test is married to Ujhelyi
}
```

Summary - Xtext

- Easy to extend implementation
  - Sane defaults
  - Easy to use
- Problems
  - Xbase is highly experimental
  - Can be quite complex
  - Somewhat large dependecy list