Appendices

Structural Enhancement Rules (SER) Language EBNF Grammar

A. Base syntax

```
String Char = Printable - ["]
StringLiteral = '"'String Char*'"'
IdLetter = Letter
IdAlphaNumeric = Alphanumeric
Identifier = IdLetterIdAlphaNumeric*
IndirectCharLiteral = QuoteCharSign1Quote
''Case Sensitive'' = 'True'
''Start Symbol'' = <CompilationUnit>
Comment Line = '--'
<Literal>
  ::= StringLiteral
<PatternVarType>
  ::= 'Pattern'
<IteratorVarType>
  ::= 'Iterator'
<Type>
  ::= 'Class' | 'Interface'
<ClassType>
  ::= 'OrgClass' | 'EnhClass'
<Modifiers>
        ::= <Modifier>
         |  <Modifiers> <Modifier>
<Modifier>
        ::= 'public' | 'protected'
         |  'private' |  'static'
         |  'abstract' |  'final'
         |  'native' |  'synchronized'
         |  'transient' |  'volatile'
<ClassAccess>
  ::= <ClassType> '.' <ClassAttribute>
<ClassAttribute>
  ::= 'Name' | 'Type'
<ModuleAccess>
  ::= Identifier '.' <ClassAccess>
<AssignOptr>
  ::= '='
   |  '+='
<NamingConvention>
  ::= <Literal>
<Prefix>
  ::= <Literal>
<Name>
  ::= <SimpleName>
   |  <QualifiedName>
<SimpleName>
  ::= Identifier
<QualifiedName>
  ::= <Name> '.' Identifier
<CompilationUnit>
  ::= <ProgramDclr>
   |  <ProgramDclr> | <ModuleDclr>
<ProgramDclr>
  ::= 'Program' <Name>
<ModuleDclr>
  ::= 'Module' <Name>
<ProgramBody>
  ::= 'Begin' 'End'
   |  'Begin' <ModuleBody> 'End'
<ModuleBody>
  ::= 'Begin' 'End'
```

B. Iterator syntax

```
<IterVar>
  ::= Identifier
<IteratorVarId>
  ::= 'Var' <IteratorVarType> <IterVar>
<IteratorVarDclr>
  ::= <IteratorVarId> <AssignOptr> <MethodInvocation>
<MethodInvocation>
  ::= ClassType '.' <Accessors>
   |  ClassType '.' <IteratationHandler>
   |  ClassType '.' <EnhancementHandler>
```

C. Pattern syntax

```
<PatternVar>
  ::= Identifier
<PatternVarId>
  ::='Var' <PatternVarType> <PatternVar>
<PatternVarDclr>
  ::= <PatternVarId> <PatternDclrBody>
<PatternDclrBody>
  ::= 'Begin' <PatternDclr> 'End'
<PatternDclr>
  ::= <PatternModifierDclr>
   |  <PatternTypeDclr>
   |  <PatternNameDclr>
   |  <PatternSignatureDclr>
<PatternModifierDclr>
  ::= <ModiferAccess> <AssignOptr> <Modifier>
<PatternTypeDclr>
  ::= <TypeAccess> <AssignOptr> <ModuleAccess>
<PatternNameDclr>
  ::= <NameAccess> <AssignOptr> <NamingConvention>
<PatternSignatureDclr>
  ::= <SignatureAccess> <AssignOptr> <ModuleAccess>
<PatternAccess> ::= <ModiferAccess>
                 |  <TypeAccess>
                 |  <NameAccess>
                 |  <SignatureAccess>
<ModiferAccess>   ::= 'modifiers'
<TypeAccess>      ::= 'type'
<NameAccess>      ::= 'name'
<SignatureAccess> ::= 'signature'
```

D. Access and Replication syntax

```
<Accessors>
  ::= <Constructors>
   |  <Methods>
   |  <Fields>
<IteratationHandler>
```

```
  ::= <FieldGetReplacer> | <FieldSetReplacer>
   |  <CreateNewIterator>
<EnhancementHandler>
  ::= <AddInterface>   | <RemoveInterface>
   |  <AddSuperclass>  | <RemoveSuperclass>
   |  <AddConstructor> | <RemoveConstructor>
   |  <AddMethod>      | <RemoveMethod>
   |  <AddField>       | <RemoveField>
<Constructors>
  ::= 'Constructors' '(' ')'
   |  'Constructors' '(' <PatternVar> ')'
<Methods>
  ::= 'Methods' '(' ')'
   |  'Methods' '(' <PatternVar> ')'
<Fields>
  ::= 'Fields' '(' ')'
   | 'Fields' '(' <PatternVar> ')'
<FieldGetReplacer>
  ::= 'FieldGetReplacer' '(' <Prefix> ',' <IterVar> ')'
<FieldSetReplacer>
  ::= 'FieldSetReplacer' '(' <Prefix> ',' <IterVar> ')'
<CreateNewIterator>
  ::= 'CreateNewIterator' '(' <IterVar> ')'
```

E. Add and removal syntax

```
<AddInterface>
  ::= 'AddInterface' '(' <IterVar> ')'
<AddSuperclass>
  ::= 'AddSuperclass' '(' <IterVar> ')'
<AddConstructor>
  ::= 'AddConstructor' '(' <PatternVar> ')'
<AddMethod>
  ::= 'AddMethod' '(' <IterVar> ')'
   |  'AddMethod' '(' <PatternVar> ')'
<AddField>
  ::= 'AddField' '(' <IterVar> ')'
  ::= 'AddField' '(' <PatternVar> ')'
<RemoveInterface>
  ::= 'RemoveInterface' '(' <IterVar> ')'
<RemoveSuperclass>
  ::= 'RemoveSuperclass' '(' <IterVar> ')'
<RemoveConstructor>
  ::= 'RemoveConstructor' '(' <PatternVar> ')'
<RemoveMethod>
  ::= 'RemoveMethod' '(' <IterVar> ')'
   |  'RemoveMethod' '(' <PatternVar> ')'
<RemoveField>
  ::= 'RemoveField' '(' <IterVar> ')'
  ::= 'RemoveField' '(' <PatternVar> ')'
```