The abstract syntax of the EventArch Language

- <Software> ::= (<EventType> | <PrimitiveAEM> | <CompositeAEM>)*

- <EventType> ::= <TypeName> (‘**extends’** <TypeName>)?
  (<AttributeType> <AttributeIdentifier>)*

- <PrimitiveAEM> ::= <ModuleName> <PrimitiveInterface>+ <PrimitiveReactor>

- <PrimitiveReactor> ::= <ApplicationName><StateMachine>
- <PrimitiveInterface> ::= <InterfaceName> (‘**local to’** <ModuleName>)?
  <RequiredInterface> <ProvidedInterface>

- <RequiredInterface> ::= (<SelectorExpression> | <OnExpression>)*
- <SelectorExpression> ::= <TypeName> <SelectorName> <EventQuery>
- <EventQuery> ::= … <!-- Boolean expressions over even attributes -->
- <OnExpression> ::= ‘**on**’ <SelectorName> ‘**invoke**’ <Class> <Method> <Arguments>
- <Class> ::= … <!-- a string literal denoting a class name -->
- <Method> ::= … <!-- a string literal denoting a method name -->
- <Arguments> ::= … <!-- a string literal or <SelectorName> -->

- <ProvidedInterface> ::= (<EventDef> | <WaitExpression>)*
- <EventDef> ::= <TypeName> <EventName> (<PointcutExpression>
  (‘**&&**’ | ‘||’) <PointcutExpression>)* <Initialization>)?
- <PointcutExpression> ::= (‘**before**’ | ‘**after**’) (‘**execution**’ | ‘**call**’) <MethodPattern>
- <Initialization> ::= … <!--initializing the attributes of the event before publishing -->
- <WaitExpression> ::= ‘**wait when**’ <EventName> ‘**until**’ <SelectorName>+
  <Condition> <ControlCommand>
- <MethodPattern> ::= … <!-- a language-independent specification of methods signature -->
- <Condition> ::= … <!-- conditional statements over the attributes of <SelectorName> -->
- <ControlCommand> ::= ‘**retry’** | ‘**proceed’** | ‘**suspend’**

- <CompositeAEM> ::= <ModuleName> <CompositeInterface>+ <CompositeReactor>
- <CompositeInterface> ::= <ModuleName> <!-- refers to a primitive AEM; the primitive module can only be part of one composite AEM -->
- <CompositeReactor> ::= <ModuleName> <!-- refers to a primitive AEM; the primitive module can only be part of one composite AEM -->

- <StateMachine> ::= <InitialState> <State>*
- <InitialState> ::= **‘initial’** <State>
- <State> ::= **'state'** <StateName> (**'entry'** (<Invoke> | <PublishEvent>)+)?
- (**'during'** (<StateMachineOnExpression>)+)?
- (**'exit'** (<Invoke> | <PublishEvent>)+)?
- <StateMachineOnExpression> ::= **'on'** (<SelectorName> (<Invoke> | <PublishEvent> | <Transition>)
- <Invoke> ::=‘**invoke**’ <Class> <Method> <Arguments>
- <PublishEvent> ::= **'send'** <EventName> **'='** **'new'** <TypeName> (<Initialization>)?
- <Transition> ::= **'->‘** <StateName>