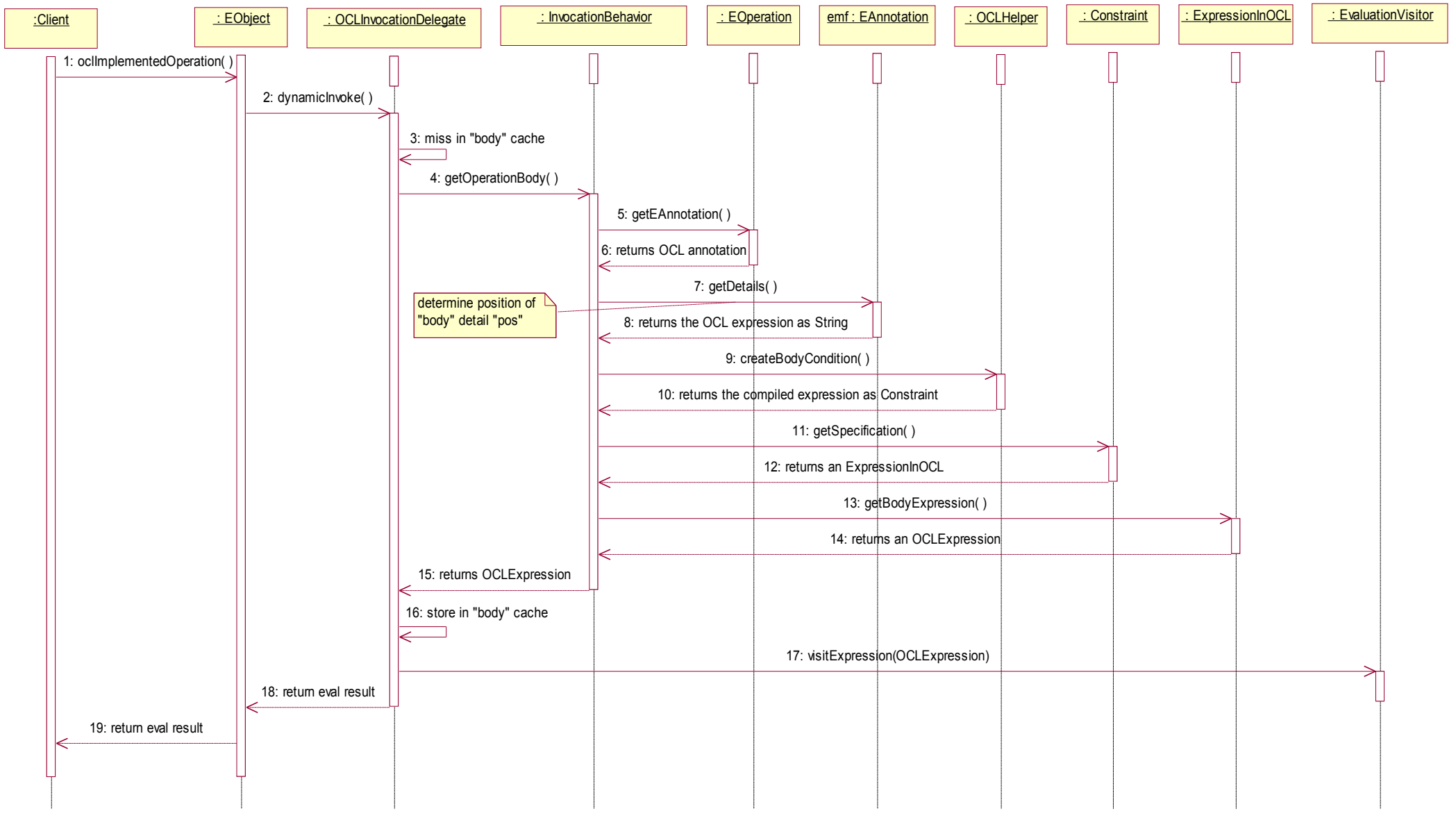
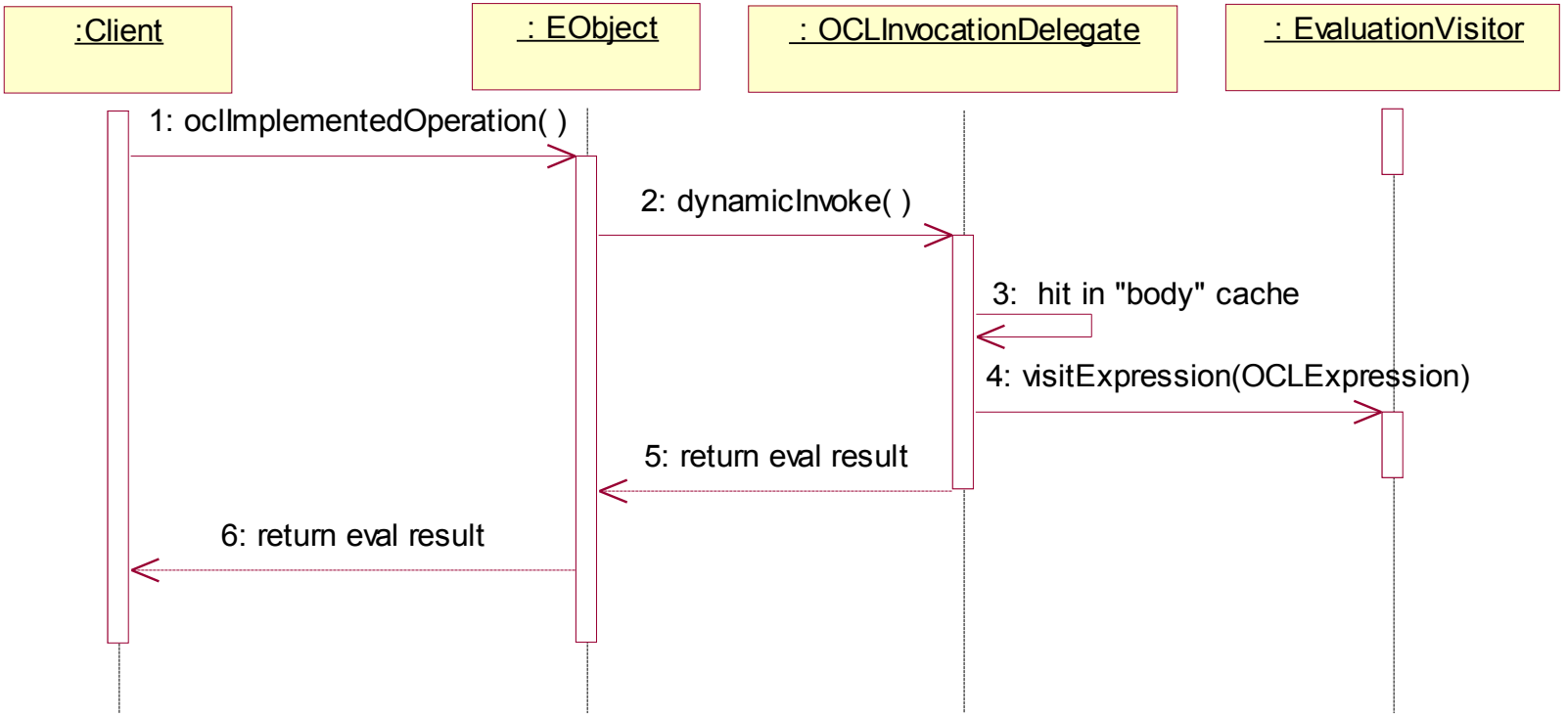
Operation Invocation (String Annotation)

Shows the course of events when an OCL-specified operation is invoked on an EObject and the OCL delegate domain has been set for the package, the delegate is invoked for the first time after loading the metamodel, and an uncompiled OCL String is in the "body" detail of an EAnnotation.
Suppressed detail: obtaining OCL instance from delegate domain.



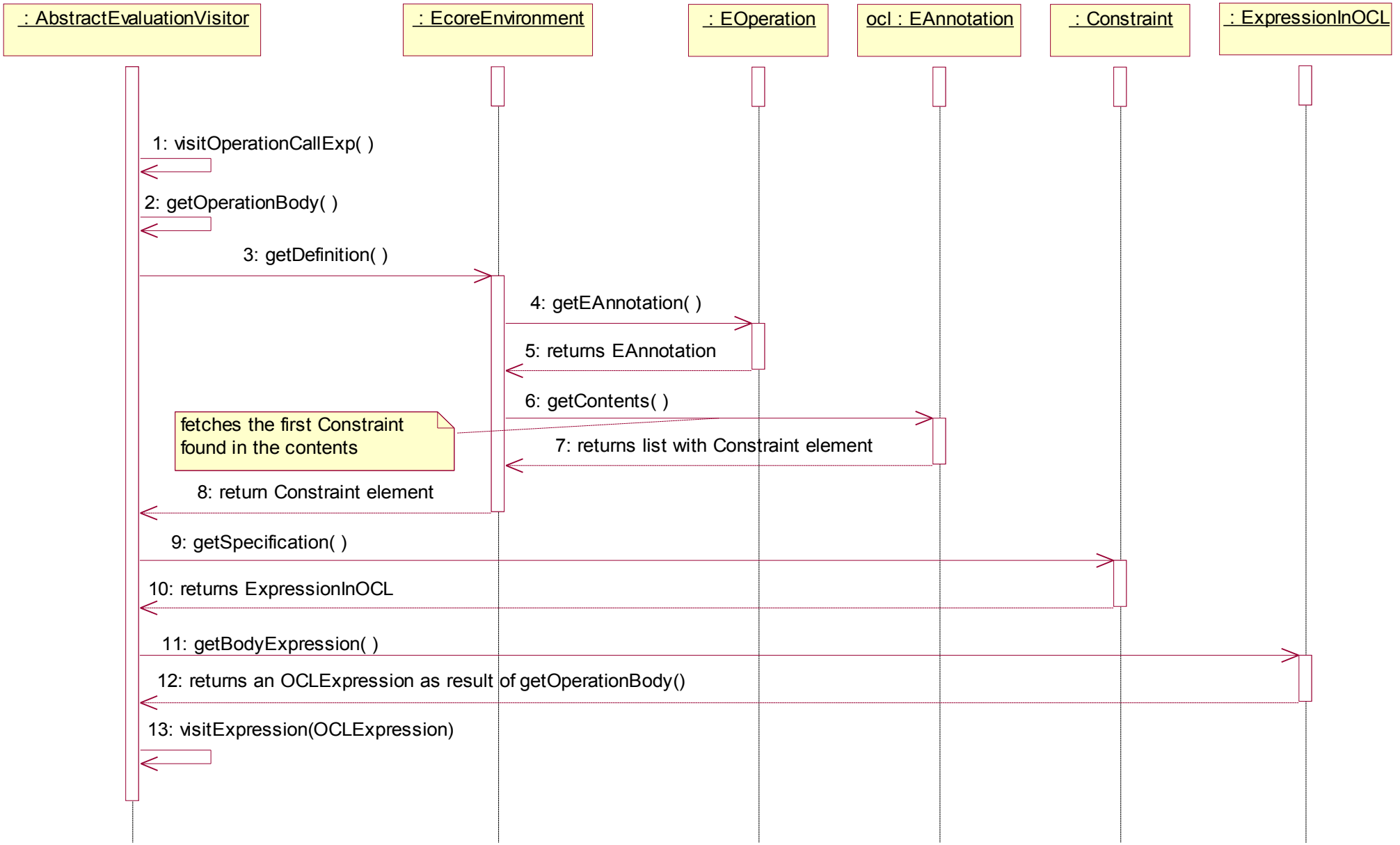
Operation Invocation (Cache Hit)

Shows the course of events when an OCL-specified operation is invoked on an EObject and the OCL delegate domain has been set for the package, and the delegate has been invoked before where it cached the compiled OCL expression in its "body" field.



OperationCallExp with AST in Contents

Shows the course of events when an OCL-specified operation is invoked by means of evaluating an OperationCallExp in the EvaluationVisitor. We assume a compiled Constraint object is embedded in the contents of the OCL EAnnotation.
This efficient scenario can be exploited by pre-compiling OCL expressions and storing them persistently in the OCL EAnnotation's contents.

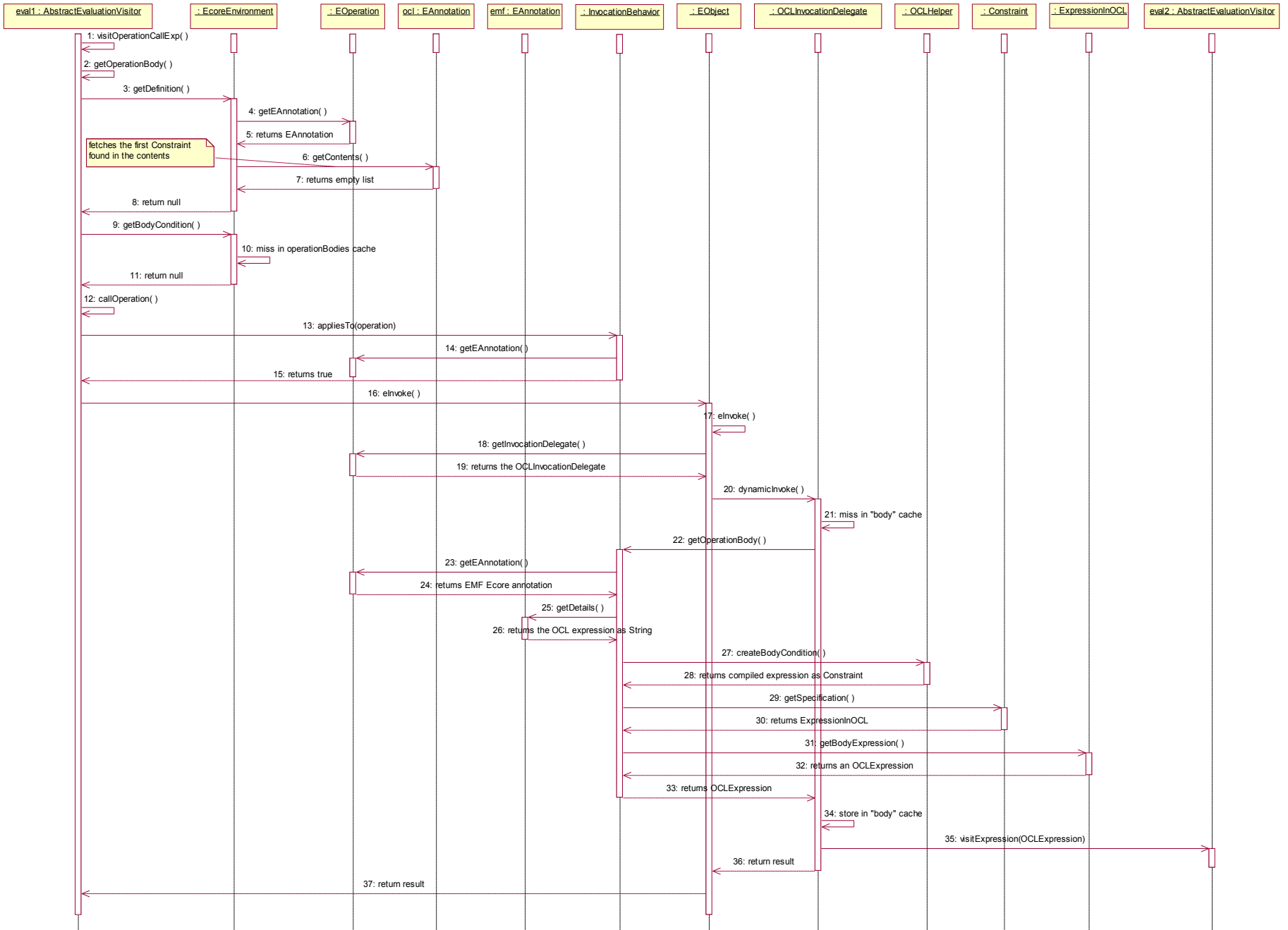


OperationCallExp w/ dynamically compiled AST

Shows the course of events when an OCL-specified operation is invoked by means of evaluating an OperationCallExp in the EvaluationVisitor. We assume the OCL expression has been compiled from sources after loading the metamodel and using the AbstractOCLAnalyzer which uses AbstractEnvironment.setBodyCondition(...) to cache the Constraint in its operationBodies map.

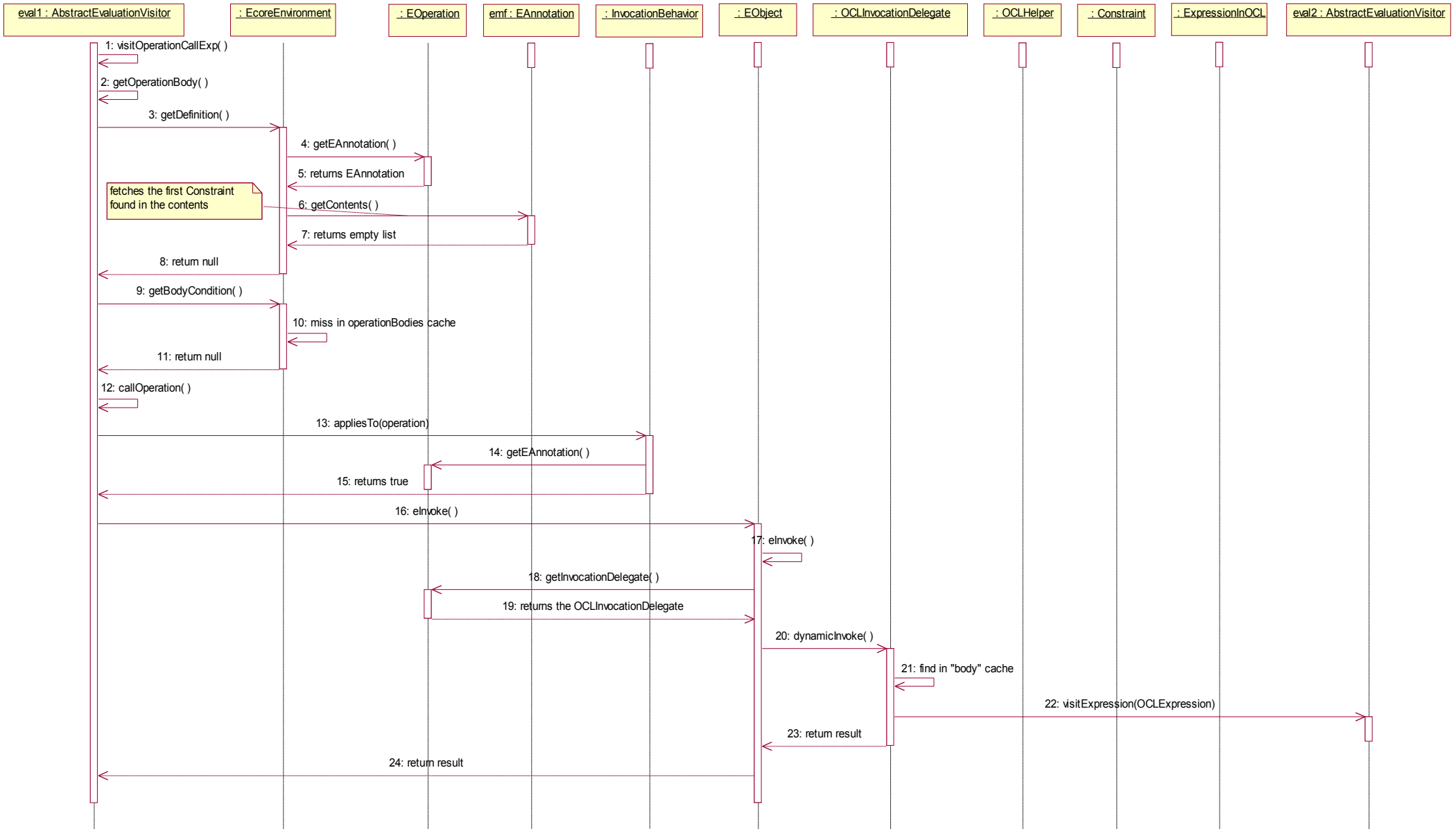
OperationCallExp w/ String Annotation, Cache Miss

Shows the course of events when an OCL-specified operation is invoked by means of evaluating an OperationCallExp in the EvaluationVisitor. We assume the OCL expression has been attached as a String in the "body" details of an EMF Ecore EAnnotation on the EOperation and hasn't been compiled yet. Note the unsuccessful checks for annotation contents and operationBodies cache hits before a Java reflection-based call through the delegate infrastructure is placed.

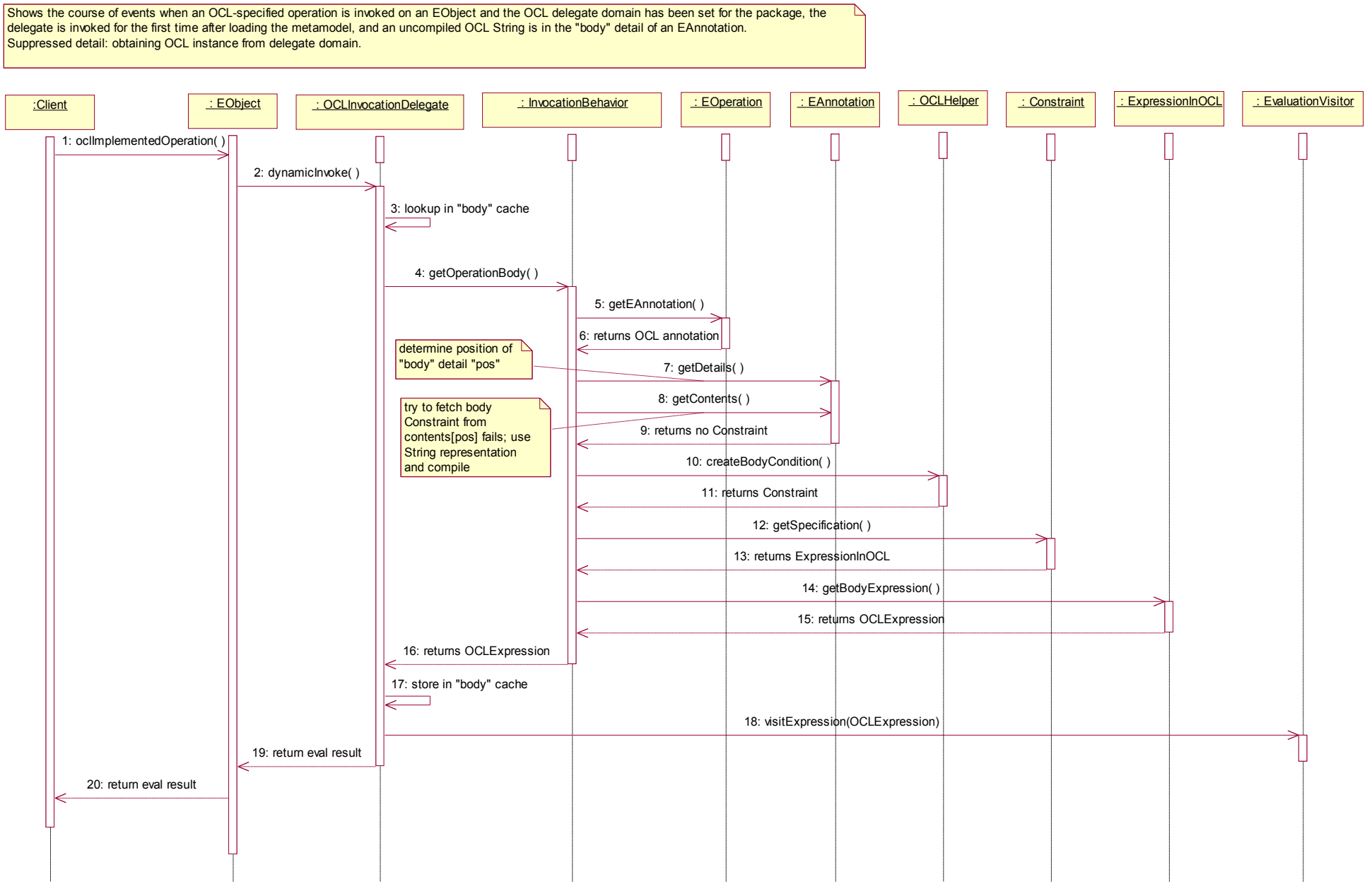


OperationCallExp w/ String Annotation, Cache Hit

Shows the course of events when an OCL-specified operation is invoked by means of evaluating an OperationCallExp in the EvaluationVisitor. We assume the OCL expression has been attached as a String in the "body" details of an EMF Ecore EAnnotation on the EOperation and has previously been compiled by the delegate, caching the compilation result in the delegate's "body" field.
Note the unsuccessful checks for annotation contents and operationBodies cache hits before a Java reflection-based call through the delegate infrastructure is placed.

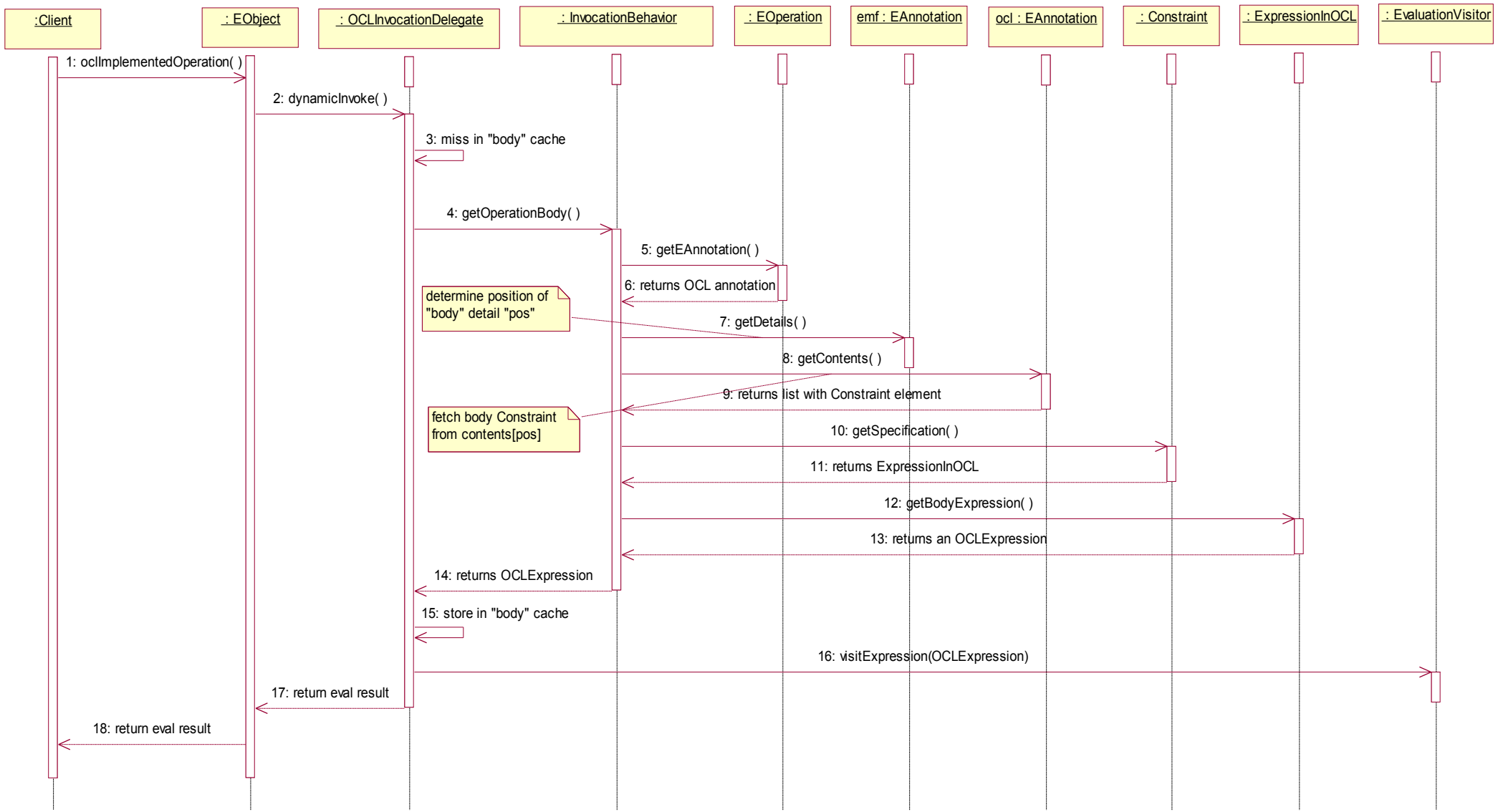


Proposed OperationCallExp w/ String Annotation

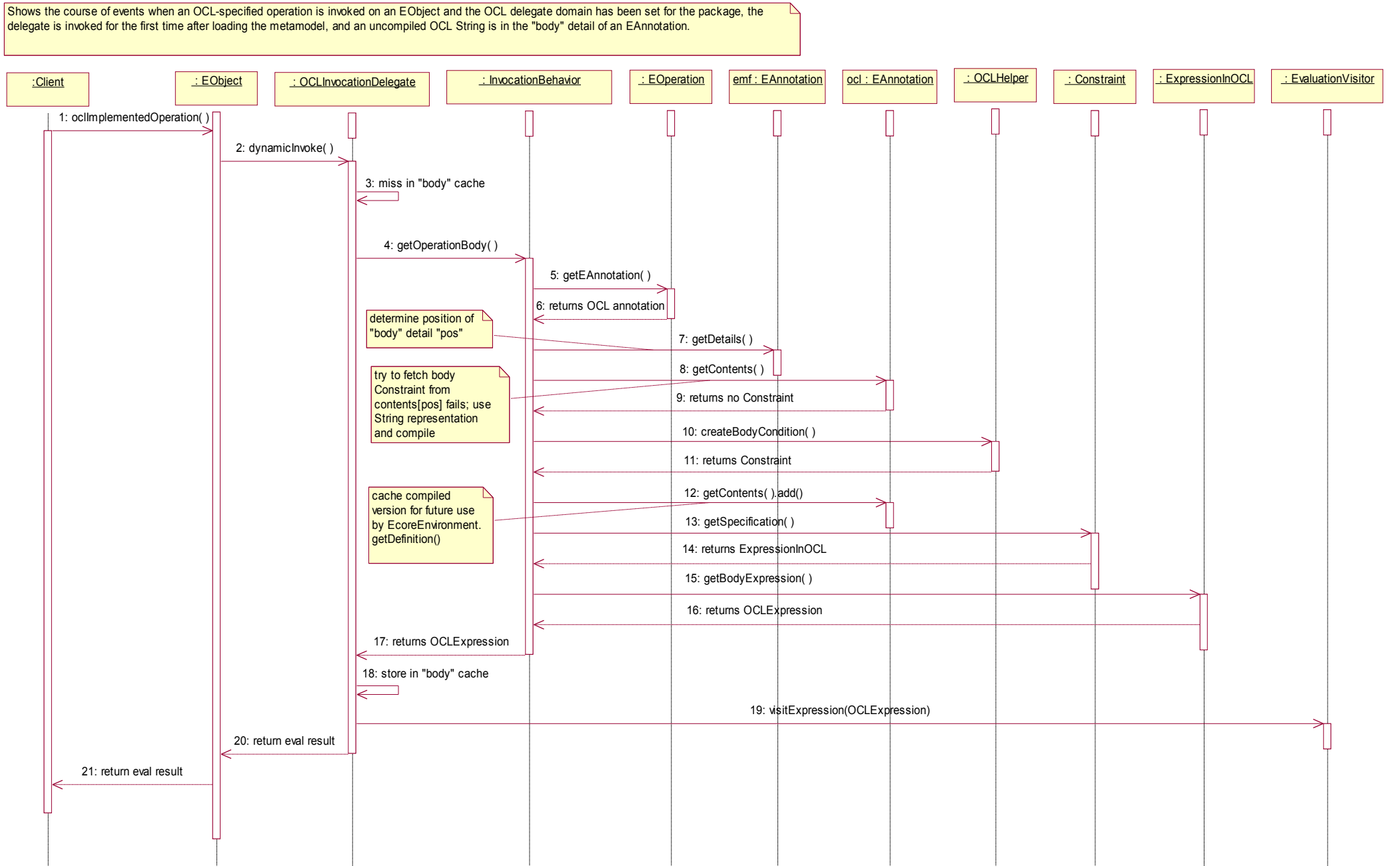


Proposed Invocation w/ AST Annotation

Shows the course of events when an OCL-specified operation is invoked on an EObject and the OCL delegate domain has been set for the package, the delegate is invoked for the first time after loading the metamodel, and a compiled Constraint object is embedded in the contents of an EAnnotation which has a "body" detail at the same position as the Constraint has in the "contents" list.
Benefit: saves compilation effort upon first invocation of operation through delegate



Proposed Invocation w/ String annotation



Summary

- Proposal
  - Consistent storage of compiled AST in Eannotation.getContents() for annotation with source `http://www.eclipse.org/ocl/1.1.0/OCL`
  - Transient (as cache)
  - Persistent (pre-compiled)
- Benefits
  - Accelerated OperationCallExp evaluation
  - Strategic enabling of persistent pre-compilation