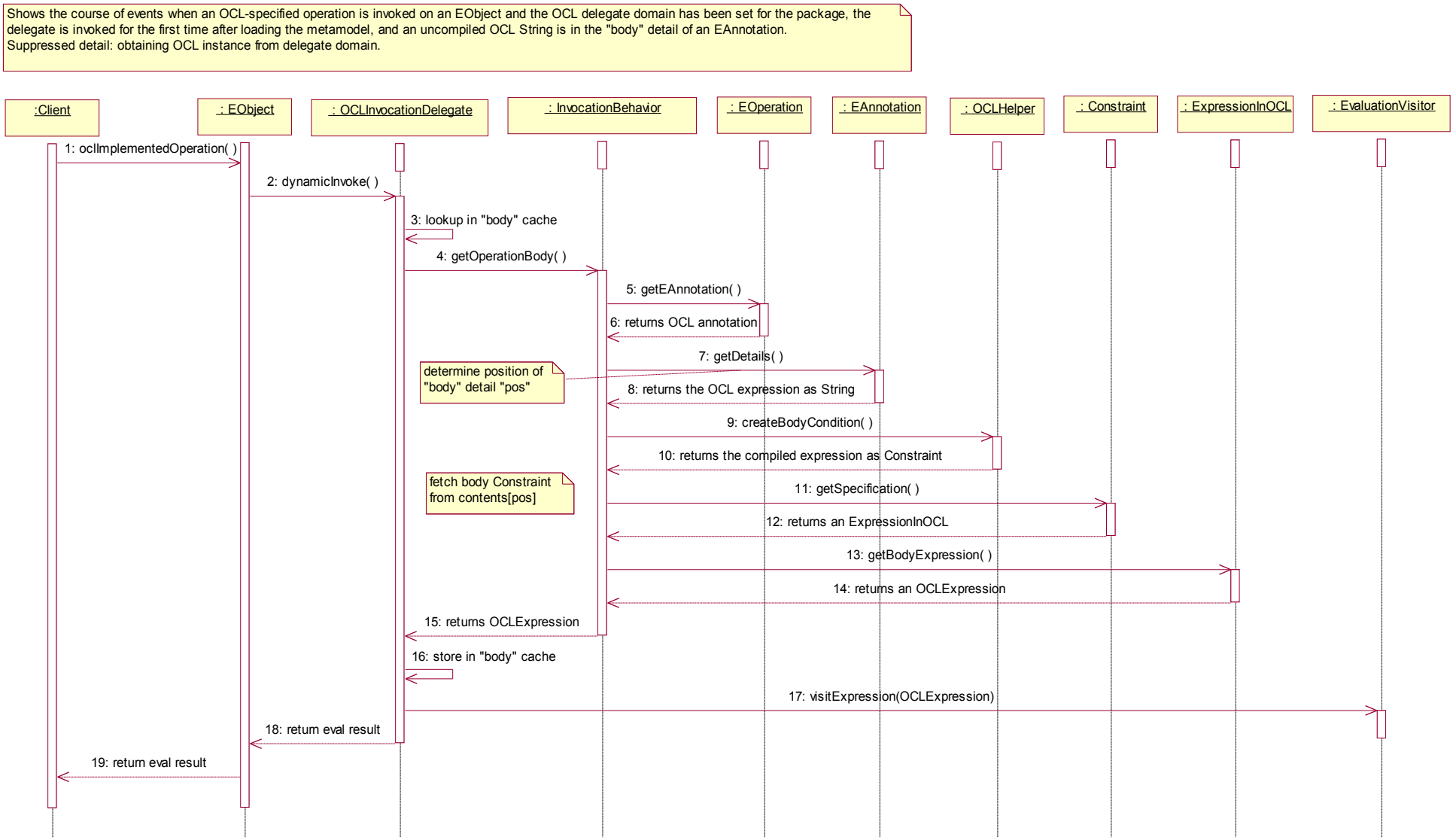
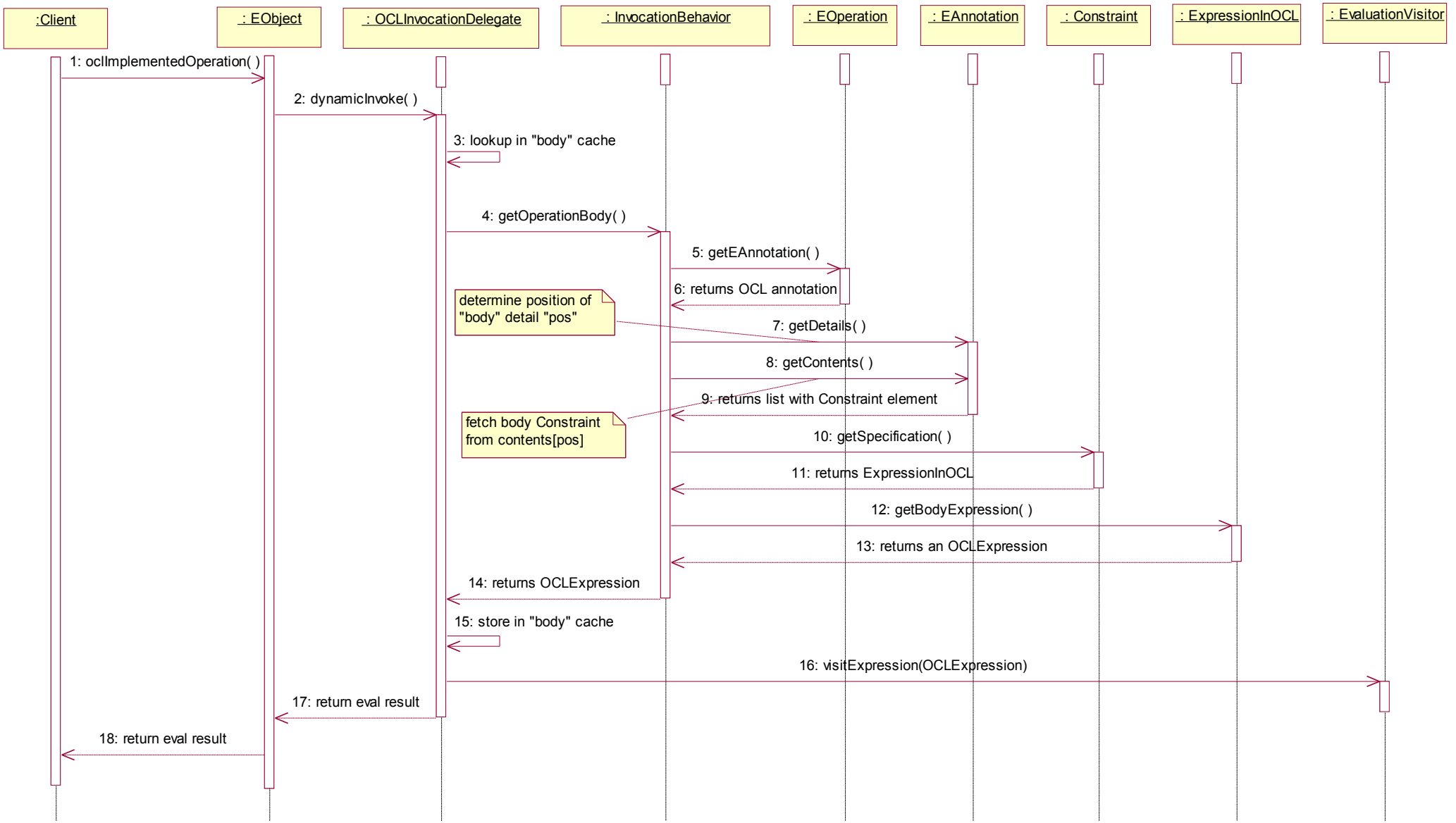
# Operation Invocation (String Annotation)



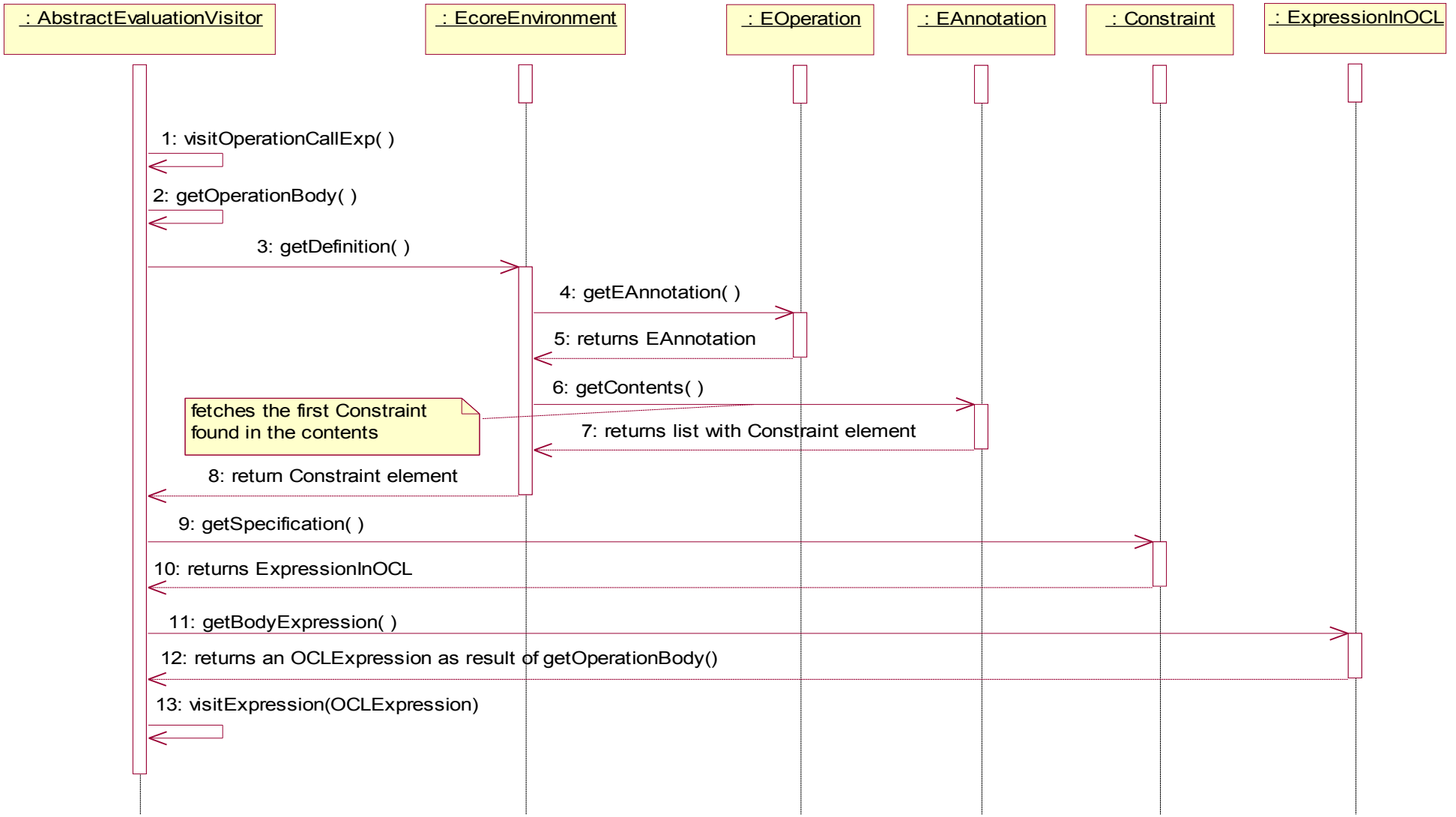
# Operation Invocation (AST in contents)

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.
Suppressed detail: obtaining OCL instance from delegate domain.



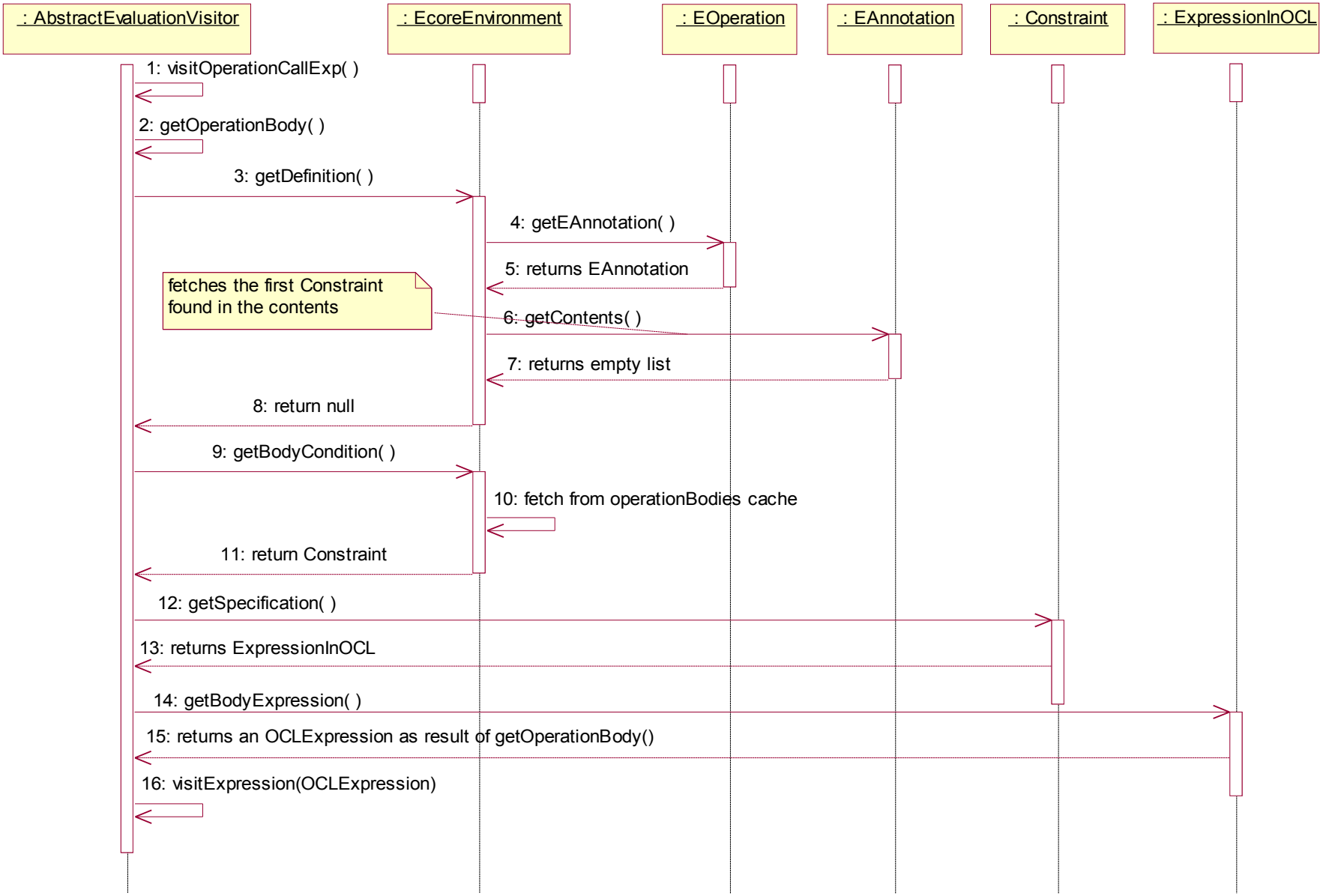
# OperationCallExp with AST in contents

Shows the course of events when an OCL-specified operation is invoked by means of evaluating an OperationCallExp in the EvaluationVisitor. We assume 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:
Suppressed detail: obtaining OCL instance from delegate domain.
Note: if the annotation contents were used to cache compiled OCLExpressions as Constraint elements, the getDefinition() and getBodyCondition() call would use the same information in the same place, compatibly.

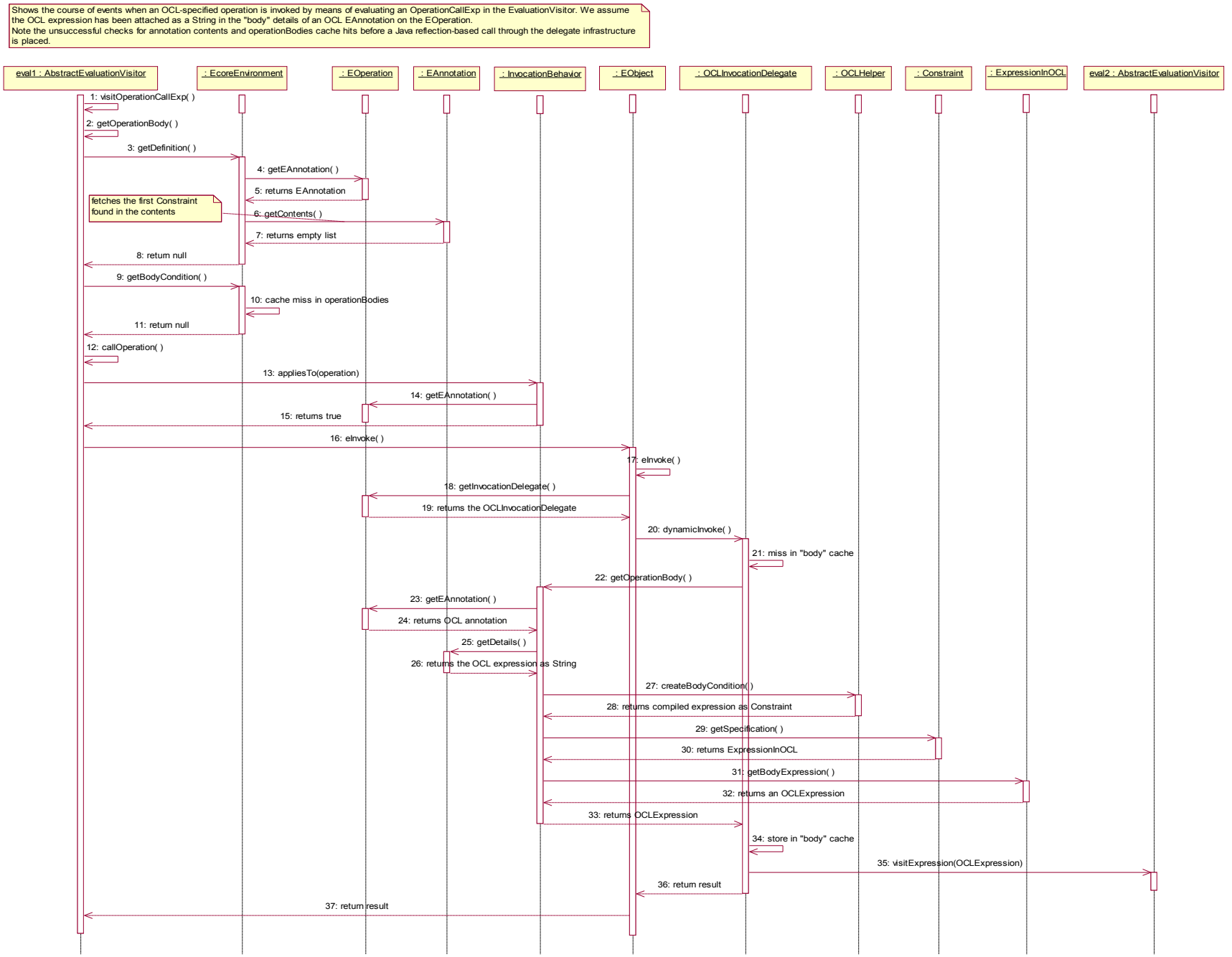


# OperationCallExp w/ dynamically compiled AST

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.

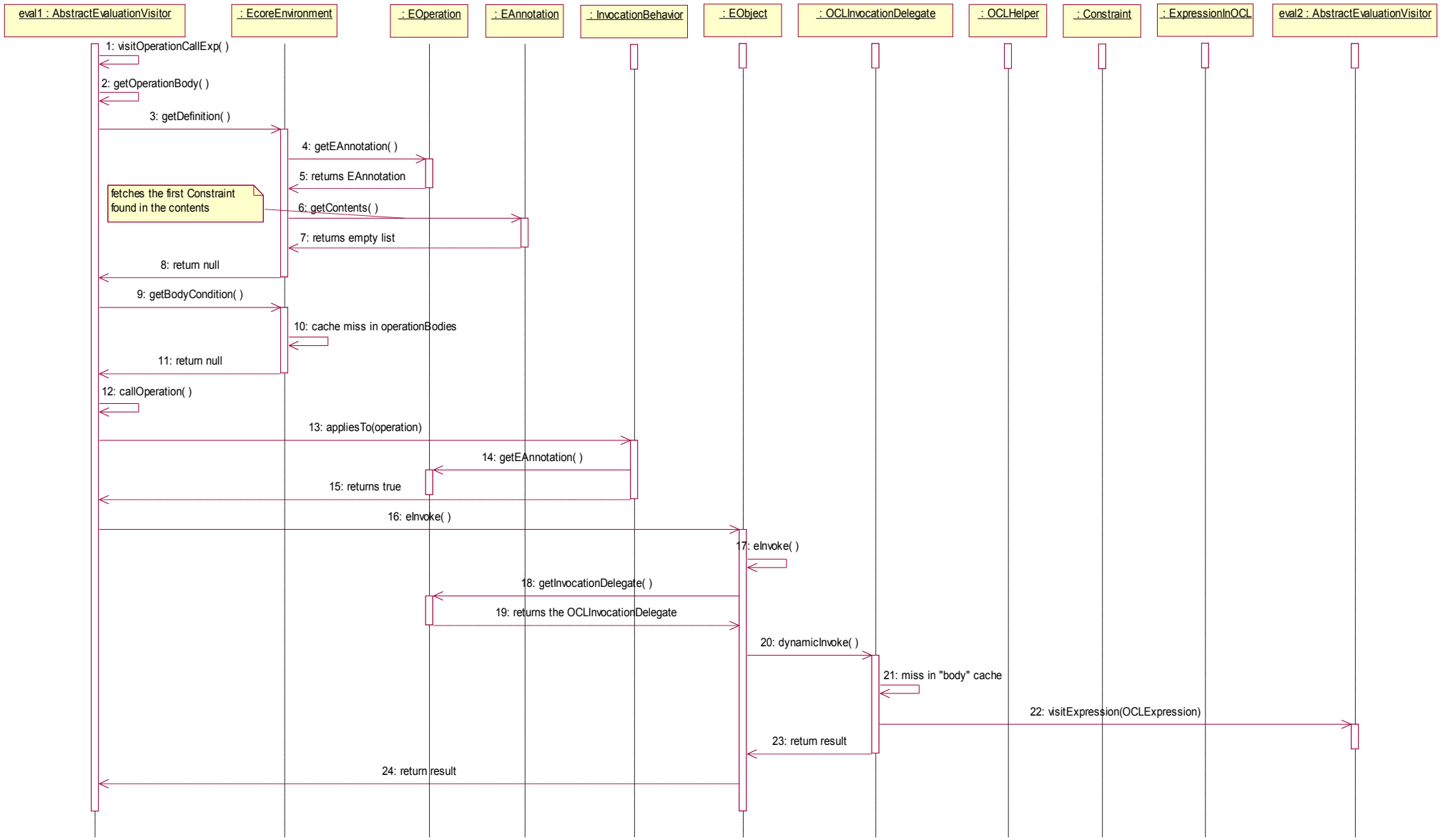


# OperationCallExp w/ String annotation, Cache Miss

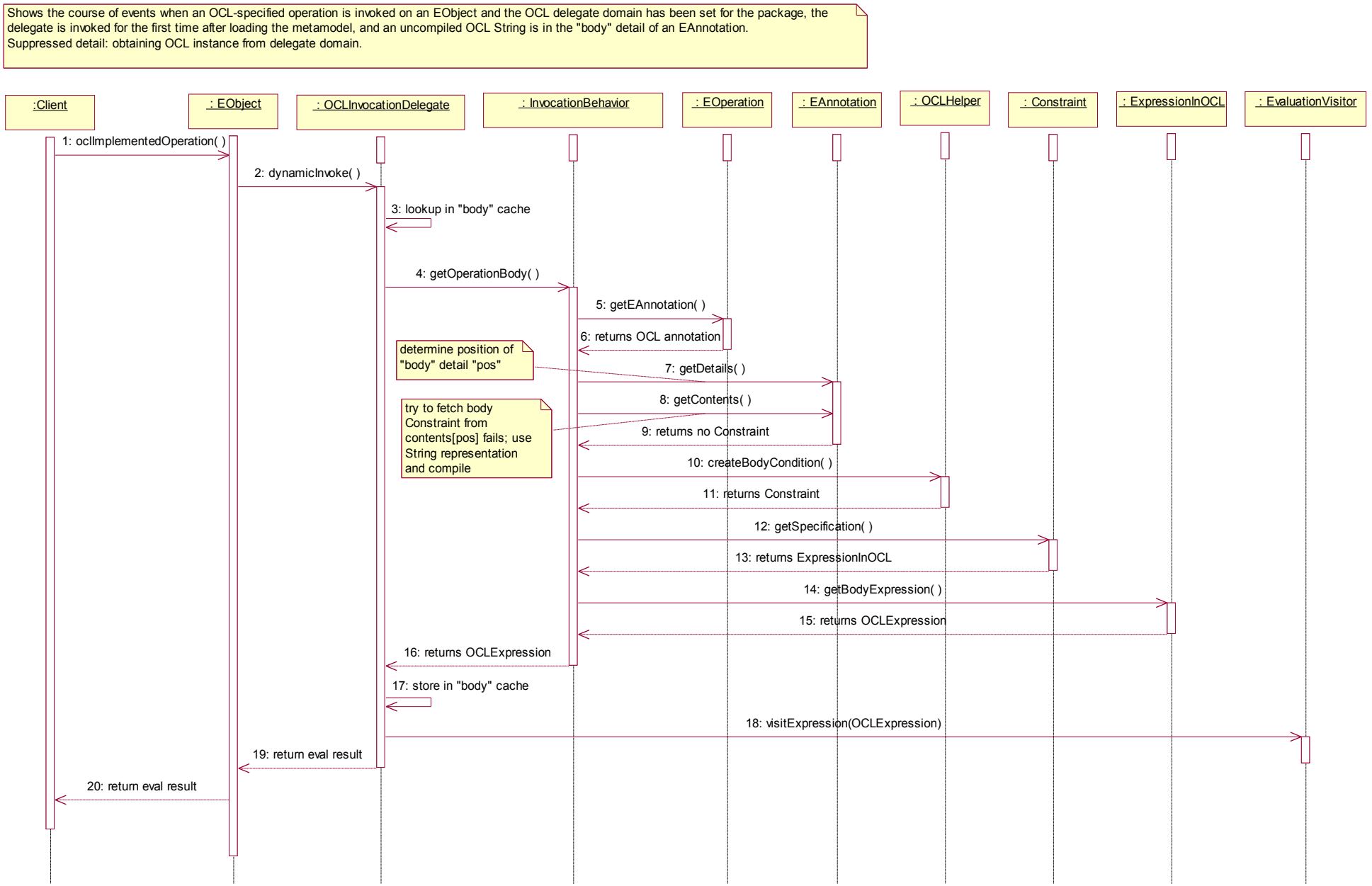


# OperationCallExp w/ String annotation, Cache Hit

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 OCL EAnnotation on the EOperation.
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



# Summary

- Proposal
  - Consistent storage of compiled AST in Eannotation.getContents()
  - Transient (as cache)
  - Persistent (pre-compiled)
- Benefits
  - Code simplification
  - Accelerated OperationCallExp evaluation
  - Strategic enabling of persistent pre-compilation