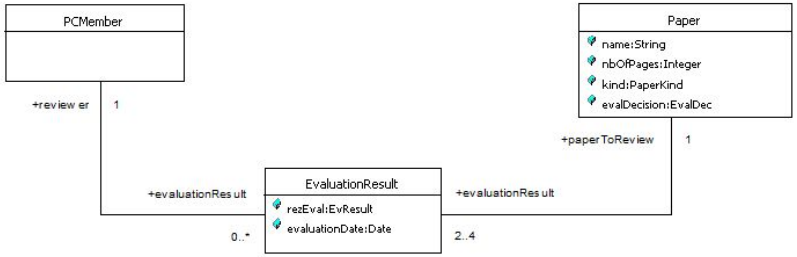
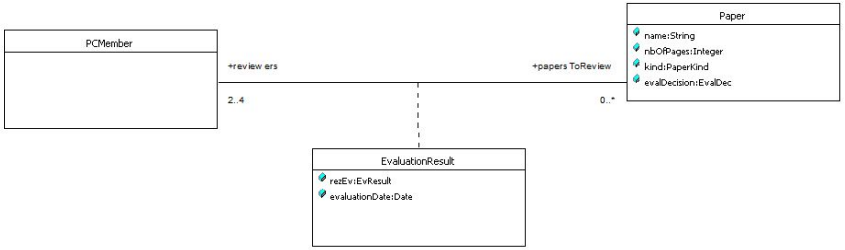
Object Design – Specifying Interfaces

The Conference Management model

Dan CHIOREAN

UML – Equivalent architectures





OCL – Accessing Overridden properties

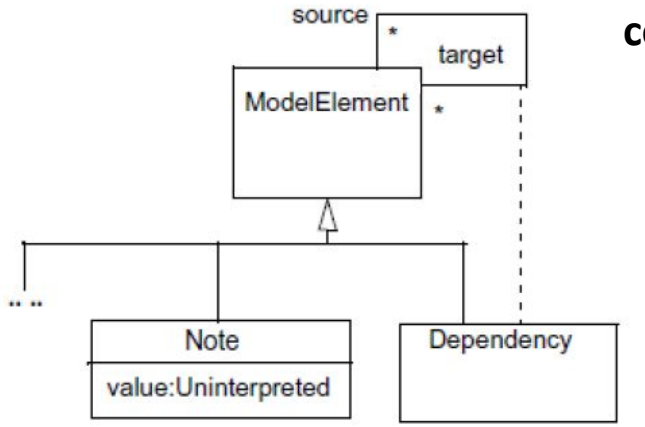


Figure 7.4 - Accessing Overridden Properties Example

context Dependency

inv: self.source <> **self** /*ambiguous specification*/

inv: **self**.oclAsType(Dependency).source->isEmpty()

inv: **self**.oclAsType(ModelElement).source->isEmpty()

OCL – few operations used in today specifications

/* All these are literal expressions – context independent */

Set{1,5,7,9, 8} **-** Set{2, 4, 6, 8, 5} = Set{1, 7, 9}

Set{1,5,7,9, 8}->**symmetricDifference**(Set{2, 4, 6, 8, 5}) = Set{1, 7, 9, 2, 4, 6}

Set{1,5,7,9, 8}->any(true) = 1

Set{1,5,7,9, 8}->any(i | i > 8) = 9

Set{1,5,7,9, 8}->any(i | i >= 7) = 7

Set{1,5,7,9, 8}->any(i | i >= 8) = 9

The purpose of using assertions

- Support in writing correct software, including the means to formally define correctness

– The writing of explicit contracts comes as a prerequisite of their enforcement in software

- Support for a better software documentation

– Essential when it comes to reusable assets, see the case of Ariane!

- Support for testing, debugging and quality assurance

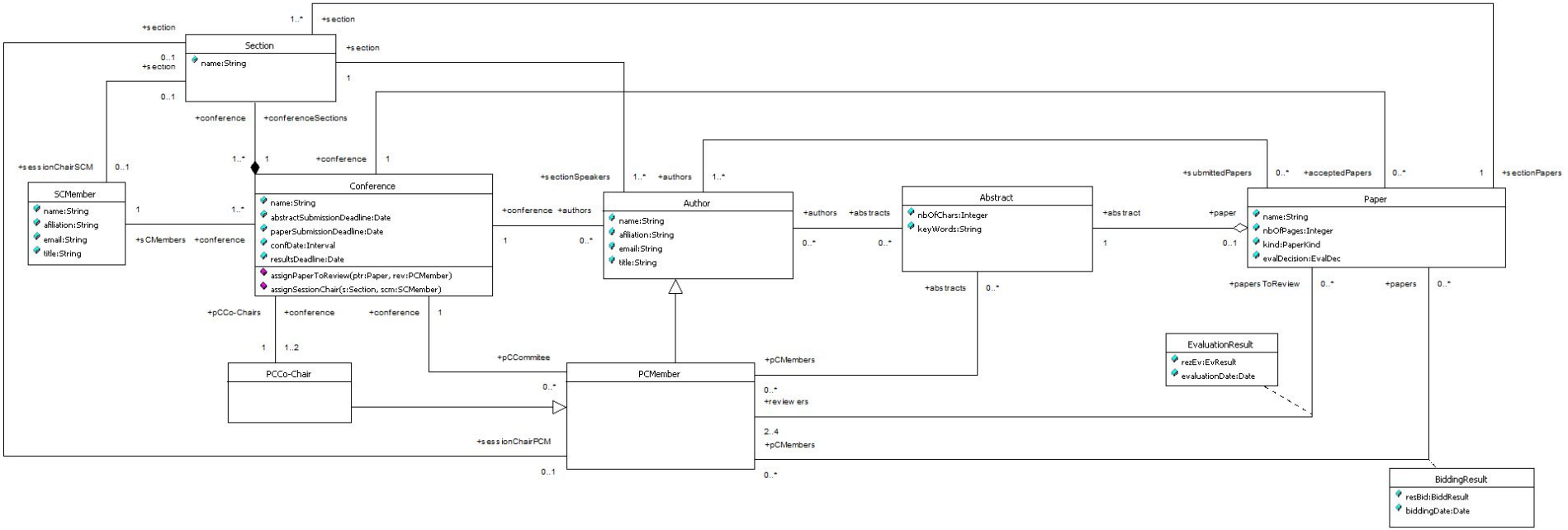
– Levels of runtime assertion monitoring:

- 1.preconditions only
- 2.preconditions and postconditions
- 3.all assertions

– While testing, enforce level 3, in production, there is a tradeoff between trust in the code, efficiency level desired and critical nature of the application

- Support for the development of fault tolerant systems

Conference Management – Specifying Interfaces

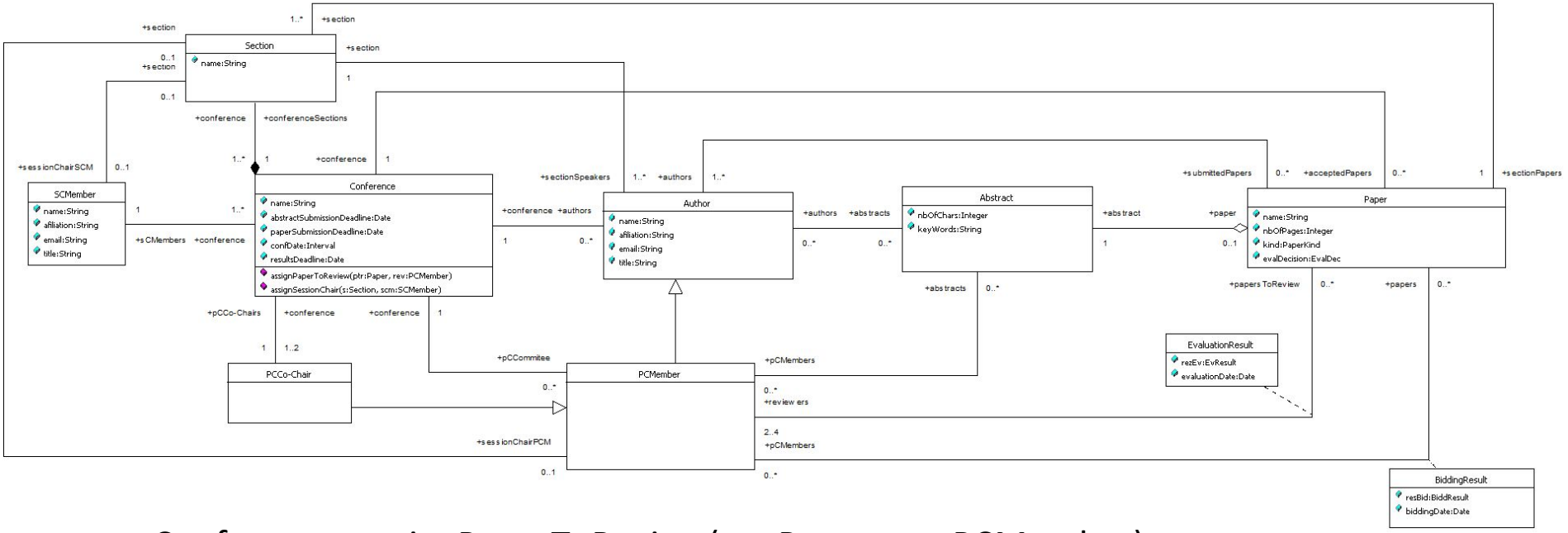


context PCMember

inv approprPapToReview:

self.papersToReview->select(p:Paper | Set{BiddResult::conflict, BiddResult::refuseToEv}->includes(p.biddingResult->**any**(br| br.pCMembers->includes(**self**)).resBid))->isEmpty **and** **self**.papersToReview.authors->excludes(**self**.oclAsType(Author))

Conference Management – Specifying Interfaces

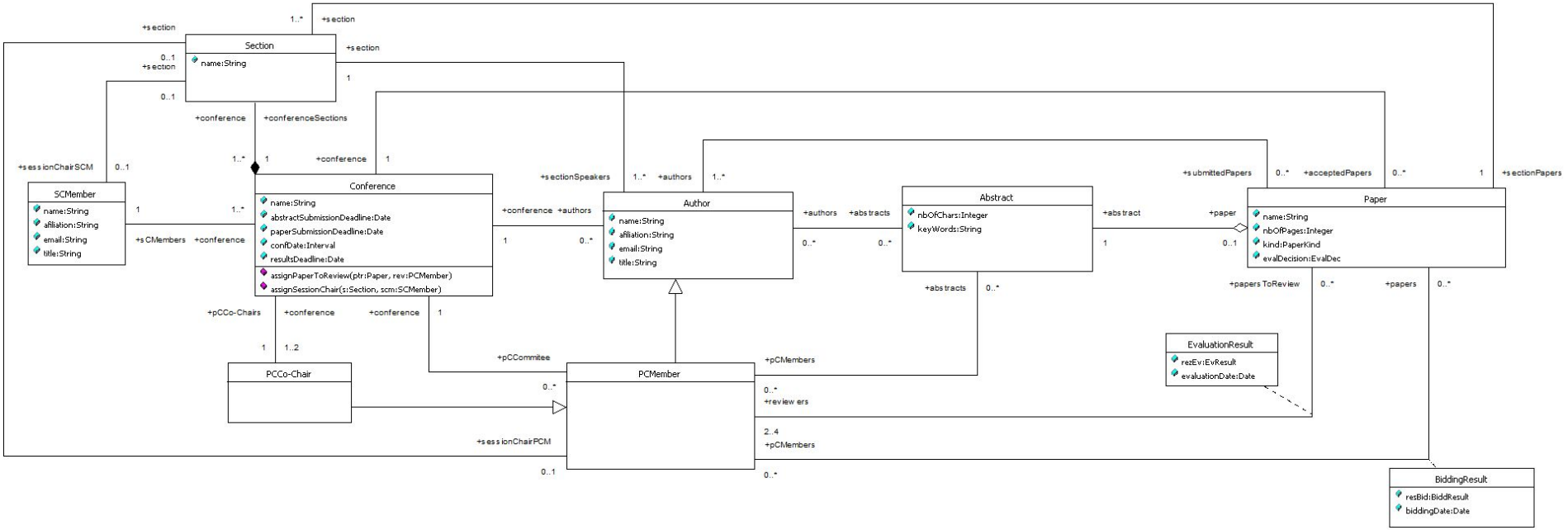


context Conference::assignPaperToReview(ptr:Paper, rev:PCMember)

pre: ptr.reviewers->size < 4 **and** ptr.reviewers->excludes(rev) **and**
self.submittedPapers->includes(ptr) **and self**.pCCommitee->includes(rev) **and**
Set{BiddResult::conflict, BiddResult::refuseToEv}->excludes(ptr.biddingResult->
select(br | br.pCMembers = rev)->**any**(true).resBid)

post: ptr.reviewers->includes(rev) **and** ptr.reviewers->size = ptr.reviewers@pre->size + 1

Conference Management – Specifying Interfaces



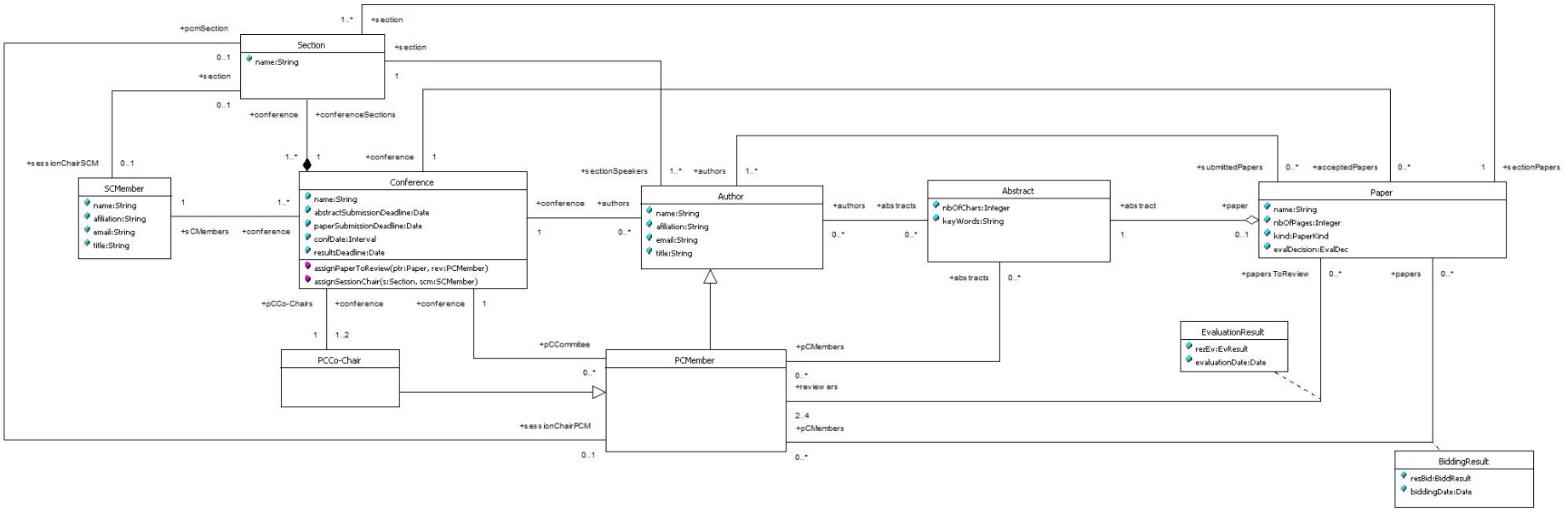
context PCMember

inv sessionChair:

not self.oclAsType(PCMember).section.isDefined **implies**

self.oclAsType(PCMember).section.sectionSpeakers->excludes(**self**.oclAsType(Author))

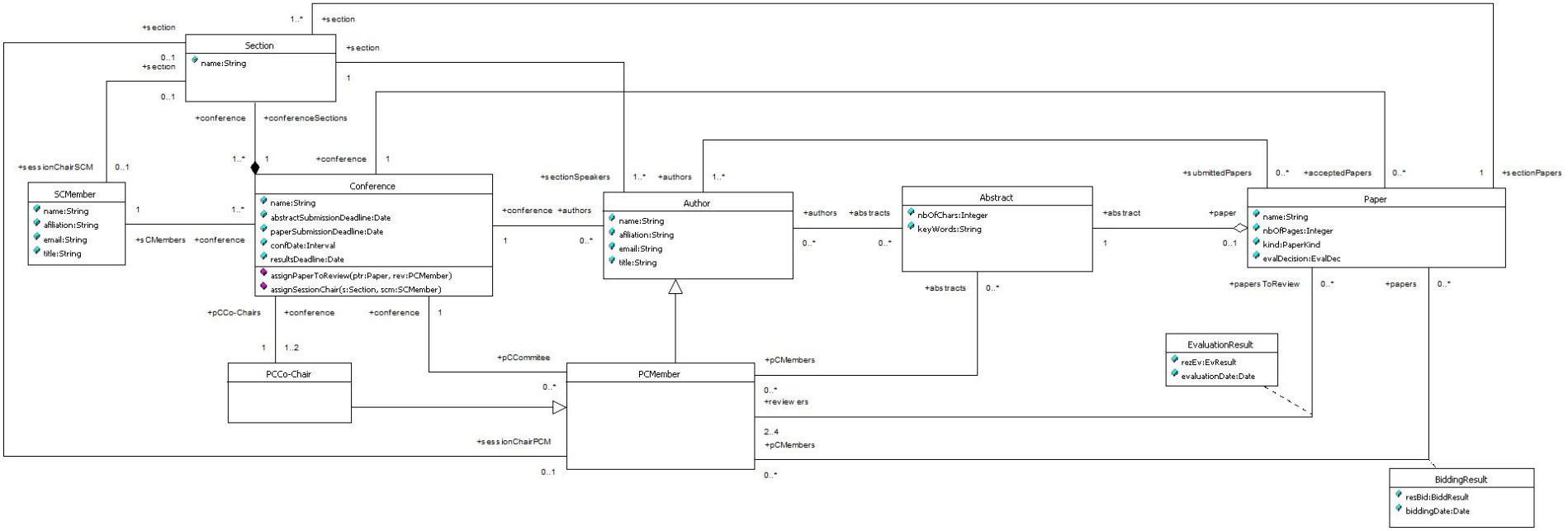
Conference Management – Specifying Interfaces



The previous OCL specification, using uppercast for the section property visible in the PCMember context was made for an uncompilable model because in the namespace of PCMember there is a conflict name. The correct solution is this.

context PCMember
inv sessionChair:
not self.pcmSection.isDefined **implies**
self.pcmSection.sectionSpeakers->excludes(**self**.oclAsType(Author))

Conference Management – Specifying Interfaces



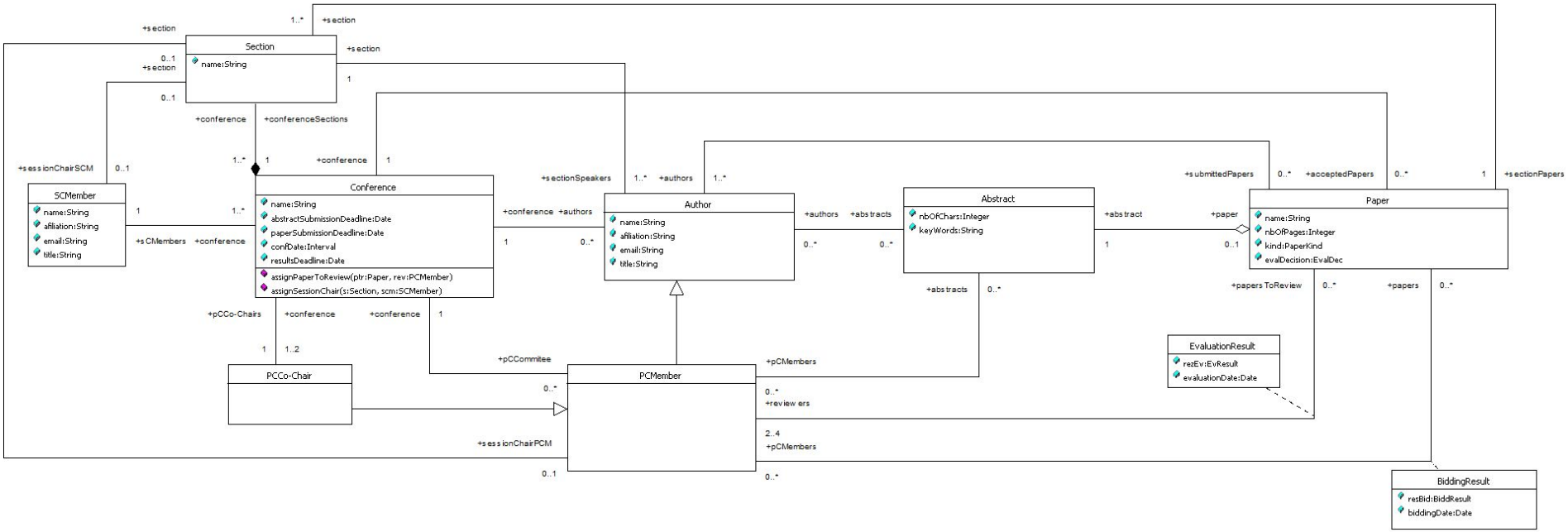
context Conference::assignSessionChair(s:Section, scm:SCMember)

pre:

s.sessionChairSCM.isUndefined **and** s.sessionChairPCM.**isUndefined and**
self.sCMembers->includes(scm) **and self**.conferenceSections->includes(s)

post: **not** s.sessionChairSCM.isUndefined /* isUndefined is oclIsUndefined() */

Conference Management – Specifying Interfaces



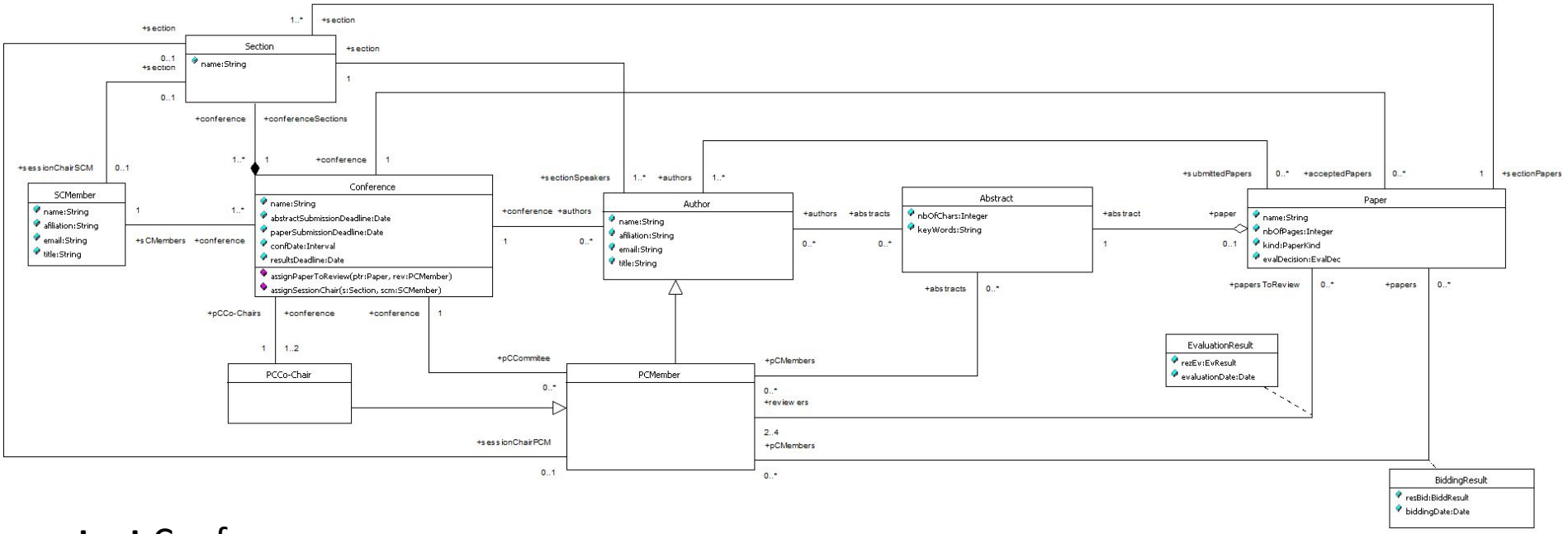
context Conference

def acceptRejectUndecided:

let allEvalResBorderline:Set(Paper)=**self**.authors.submittedPapers->asSet->select(p:Paper | p.evaluationResult.rezEv->forAll(rE | rE=EvResult::borderlinePaper))

let acceptedPapersC: Set(Paper) = **self**.authors.submittedPapers->asSet->select(p:Paper | Set{EvResult::strongAccept, EvResult::accept, EvResult::weakAccept, EvResult::borderlinePaper}->includesAll(p.evaluationResult.rezEv))-allEvalResBorderline

Conference Management – Specifying Interfaces



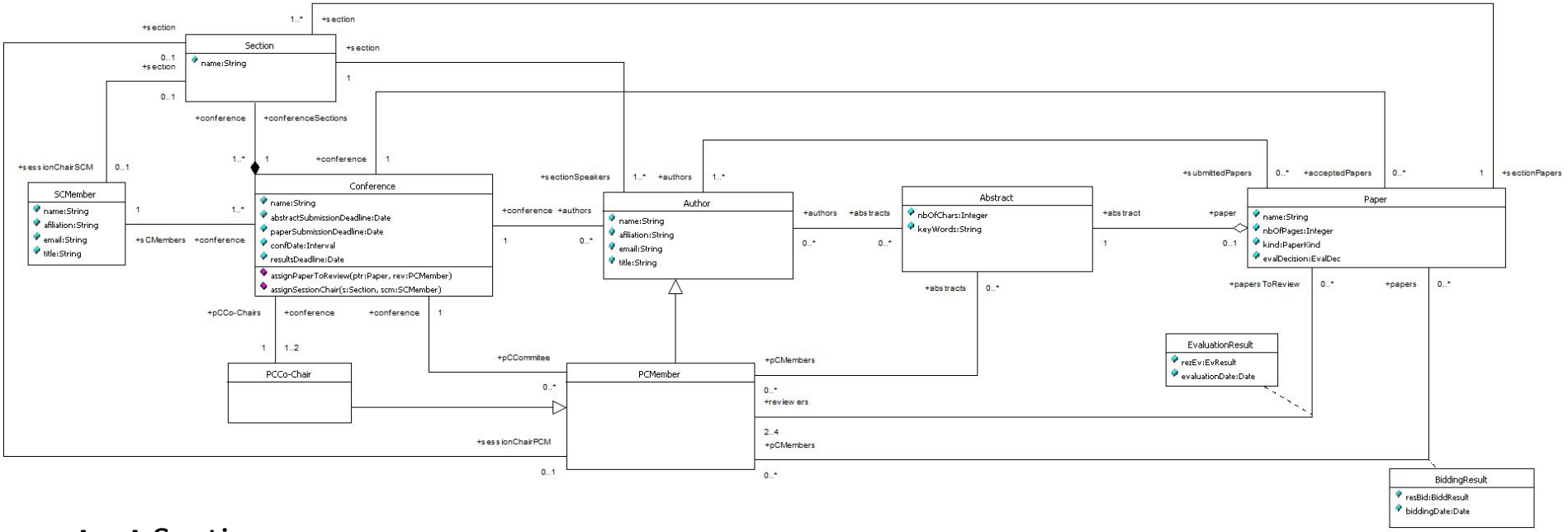
context Conference

def acceptRejectUndecided:

let rejectedPapersC: Set(Paper) = **self**.authors.submittedPapers->asSet->select(p:Paper |
Set{EvResult::strongReject, EvResult::reject, EvResult::weakReject, EvResult::borderlinePaper}->
includesAll(p.evaluationResult.rezEv))-allEvalResBorderline

let undecidedPapersC:Set(Paper) = **self**.authors.submittedPapers->asSet –acceptedPapersC
- rejectedPapersC

Conference Management – Specifying Interfaces

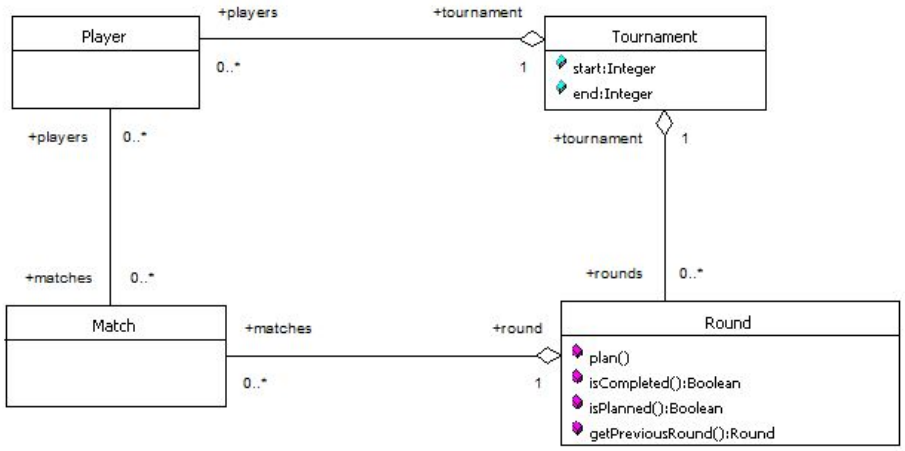


context Section

inv sessionChair**:**

self.sessionChairSCM->isEmpty **xor self**.sessionChairPCM->isEmpty

ARENA Case Study – Specifying Interfaces



context Player

inv:

/*A player cannot be assigned to more than one match per round */

self.matches->reject(m:Match | **self**.matches.round->count(m.round)=1)->isEmpty

context Round

inv:

/* choosing the context */

self.matches->forAll(m1:Match |

m1.players->forAll(p:Player | p.matches->forAll(m2:Match | m1<>m2 implies m1.round <> m2.round)))