Using Code Generation to Build a Platform for Developing & Testing Dialogue Games

Tommy Yuan, Suresh Manandhar, & Simon Wells
Universities of {York | Edinburgh Napier}
CMNA 14 @ JURIX 2014, Krakow

Introduction

- ProtOCL -
  - a prototype tool & workflow for describing & implementing dialogue games
  - (1) Describe game using industry standard tools
  - (2) Implement using code generation
  - (*n*) Build on generated code using API
- Execution Platform -
  - API, Code, & Tools for using ProtOCL games
  - Exemplar of the process of integrating ProtOCL with a wider system

Motivation

- Not always a big intersection between academic & industrial/ commercial tools
- But, increasing intersection of academia & business
  - projects (particularly larger EU), spin-outs
- Legitimate to investigate applied issues
- NB. Also an increasing focus on argumentation in relation to HCI & UX

Specification Methods

- Natural Language
- Formal/logical Notation
- Domain Specific Language (DSL)
- Diagrammatic
  - + various hybrids

Move Types

Assertions: The content of an assertion is a statement P, Q, etc. or the truth-functional compounds of statements: "Not P", "If P then Q", "P and Q".
Questions: The question of the statement P is "Is it the case that P?"
Challenges: The challenge of the statement P is "Why P?"
Withdrawals: The withdrawal of the statement P is "no commitment P".
Resolution demands: The resolution demand of the statement P is "resolve whether P".

Dialogue Rules

R$_{FORM}$**:** Participants may make one of the permitted types of move in turn.
R$_{REPSTAT}$**:** Mutual commitment can only be asserted when a question or challenge is responded.
R$_{QUEST}$**:** The question P can be answered only by P, "Not P" or "no commitment P".
R$_{CHALL}$**:** "Why P?" has to be responded to by either a withdrawal of P, a statement that challenger accept, or a resolution demands of the previous commitments of the challenger which immediately imply P.
R$_{RESOLVE}$**:** A resolution demand can be made only in situations that the other party of the dialogue has committed in an immediate inconsistent conjunction of statements, or he withdraws or challenges an immediate consequent of previous commitments.
R$_{RESOLUTION}$**:** A resolution demand has to be responded by either the withdrawal of the offending conjuncts or confirmation of the disputed consequent.
R$_{LEGALCHALL}$**:** "Why P?" cannot be used unless P has been explicitly stated by the dialogue partner.

Commitment Rules

Initial commitment, CR$_0$: The initial commitment of each participant is null.
Withdrawals, CR$_W$: After the withdrawal of P, the statement P is not included in the move makers store.
Statements, CR$_S$: After a statement P, unless the preceding event was a challenge, P is included in the move makers store.
Defence, CR$_{YS}$: After a statement P, if the preceding event was Why Q?, P and If P then Q are included in the move makers store.
Challenges, CR$_Y$: A challenge of P results in P being removed from the store of the move maker if it is there.

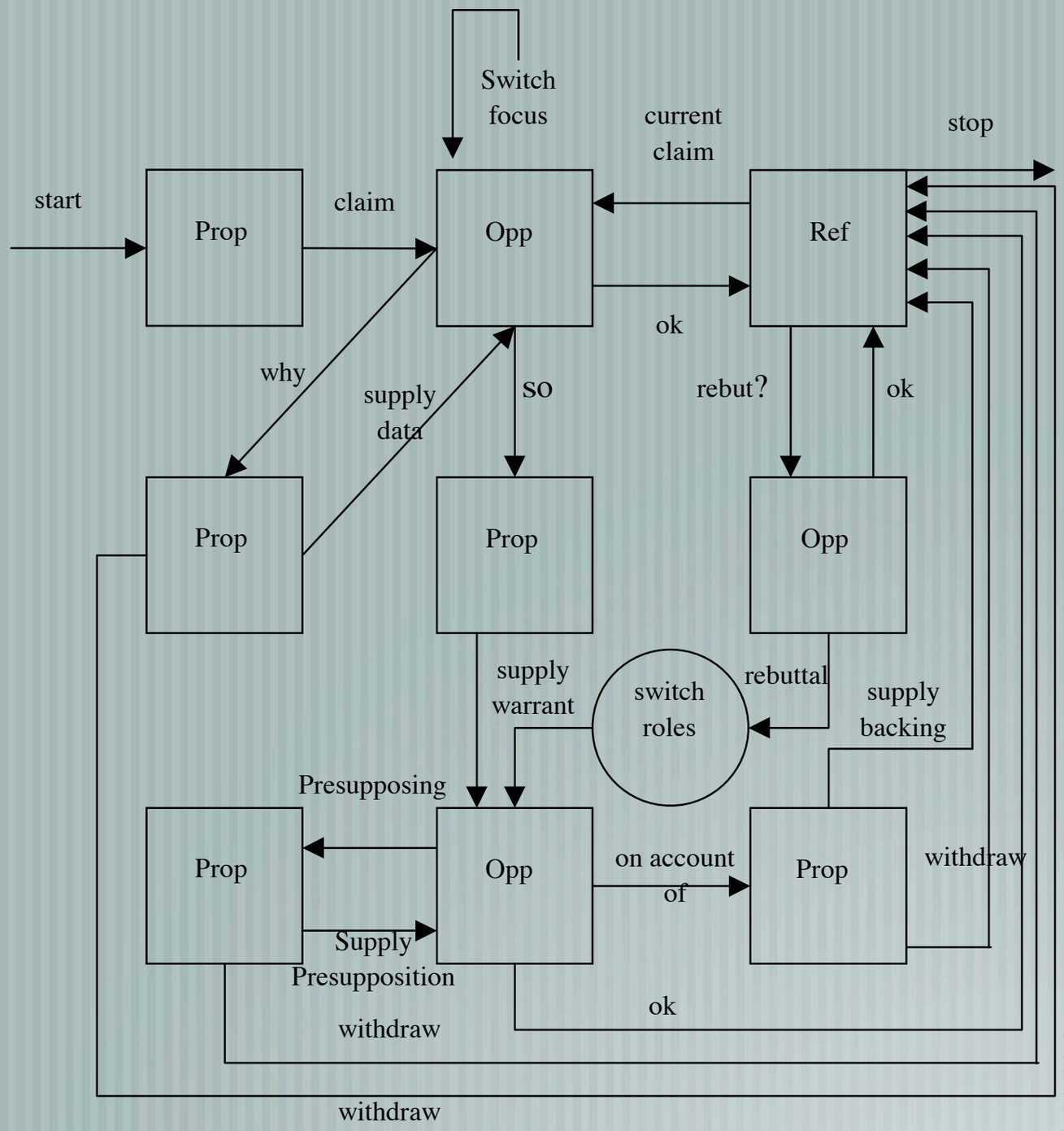
Termination Rules

1. The game will be ended when a participant accepts another participants view.

Pre-Conditions - *Commitment Store Contents*	
$\mathrm{C} \in \mathrm{CS}_n$	Commitment C is currently in commitment store CS
$\mathrm{C} \notin \mathrm{CS}_n$	Commitment C is not currently in commitment store CS
Post-Conditions - *Alterations to Commitment Stores*	
$\mathrm{CS}_{n+1} = \mathrm{CS}_n \cup \{\mathrm{C}\}$	Commitment C is added to commitment store CS
$\mathrm{CS}_{n+1} = \mathrm{CS}_n \setminus \{\mathrm{C}\}$	Commitment C is removed from commitment store CS

Move Specifications (*utilising pre- & post-conditions*)	
Statement(S_x)	Pre: $\emptyset$
	Post: $\mathrm{CP}_{n+1} = \mathrm{CP}_n \cup \{\mathrm{S}_x\} \wedge \mathrm{CO}_{n+1} = \mathrm{CO}_n \cup \{\mathrm{S}_x\}$
Withdrawal(S_x)	Pre: $\emptyset$
	Post: $\mathrm{CP}_{n+1} = \mathrm{CP}_n \setminus \{\mathrm{S}_x\}$

```
Simple{
    {turns,magnitude:single,ordering:strict}
    {players,min:2,max:2}
    {player,id:Player1}
    {player,id:Player2}
    {store,id:CStore,owner:Player1}
    {store,id:CStore,owner:Player2}
    {Assert,{p},"I assert that",
        {store(add, {p}, CStore, Speaker),store(add, {p}, CStore, Listener)}
    }
}
```



claim (C)

Description:	P asserts that C
Preconditions:	P has control of the dialogue
Postconditions:	O has control of the dialogue
	C is pushed onto the claim stack
	P is committed to C
Completion Conditions:	C is popped from the claim stack

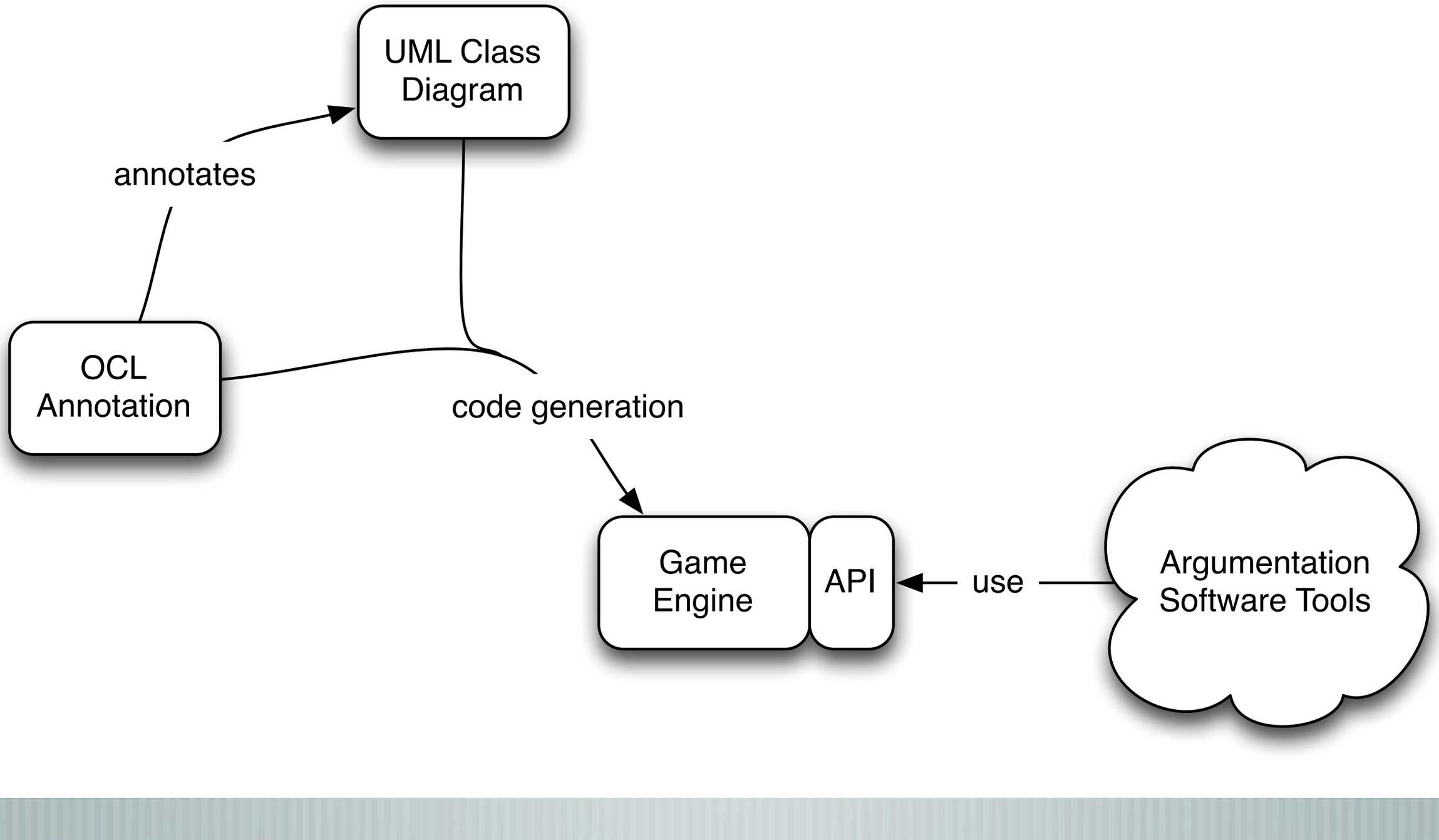
why (C)

Description:	O seeks data supporting C
Preconditions:	O has control of the dialogue
	C is top of claim stack
Postconditions:	P has control of the dialogue
Completion Conditions:	C is not top of claim stack

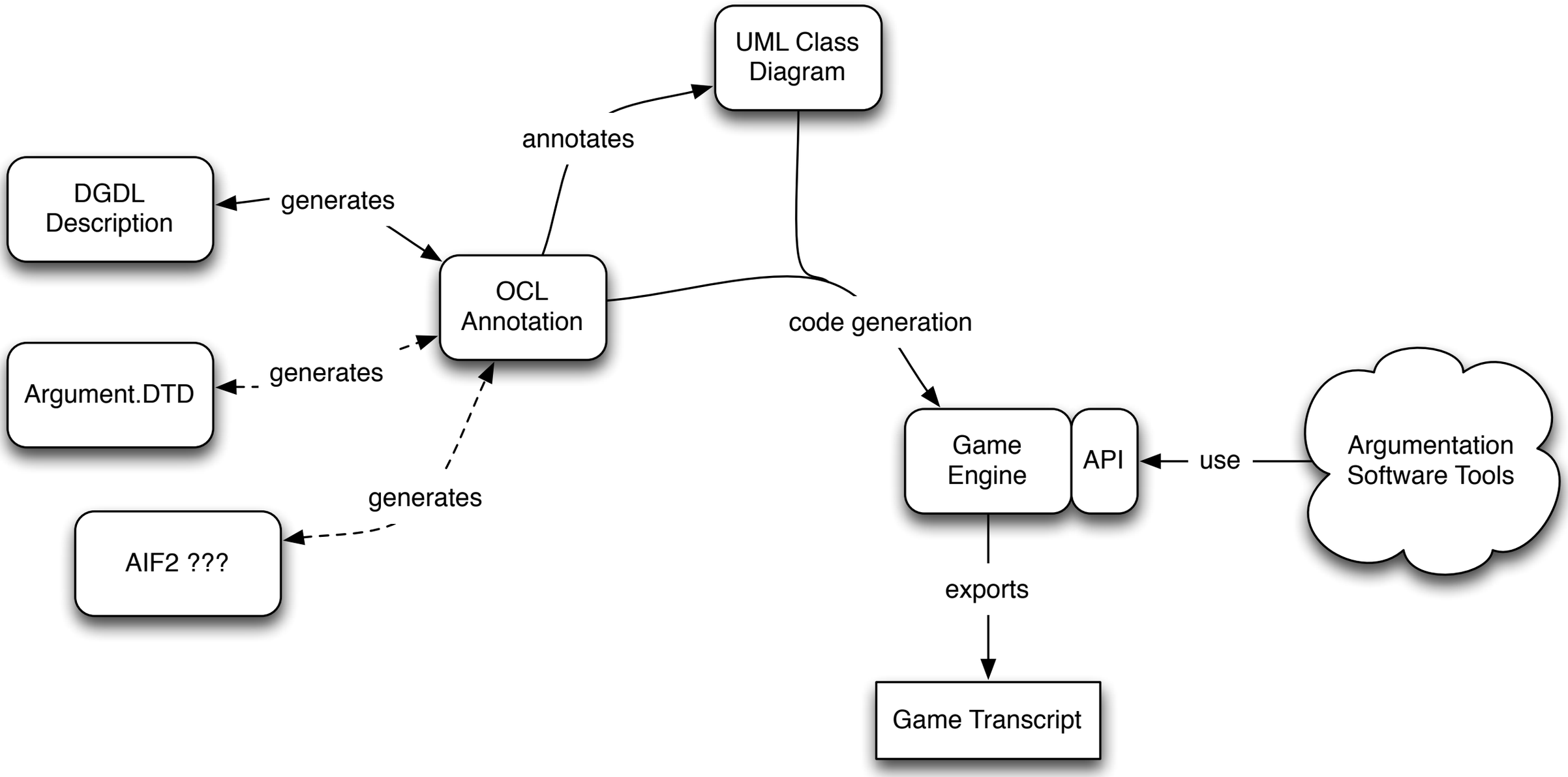
ProtOCL

1. Describe a generic dialogue game UML object model
2. Describe specific rules for updating that model in OCL
- Use standard UML tools to produce the OCL description
- Compile against object model
- Auto-generates a dialogue game framework with Java API

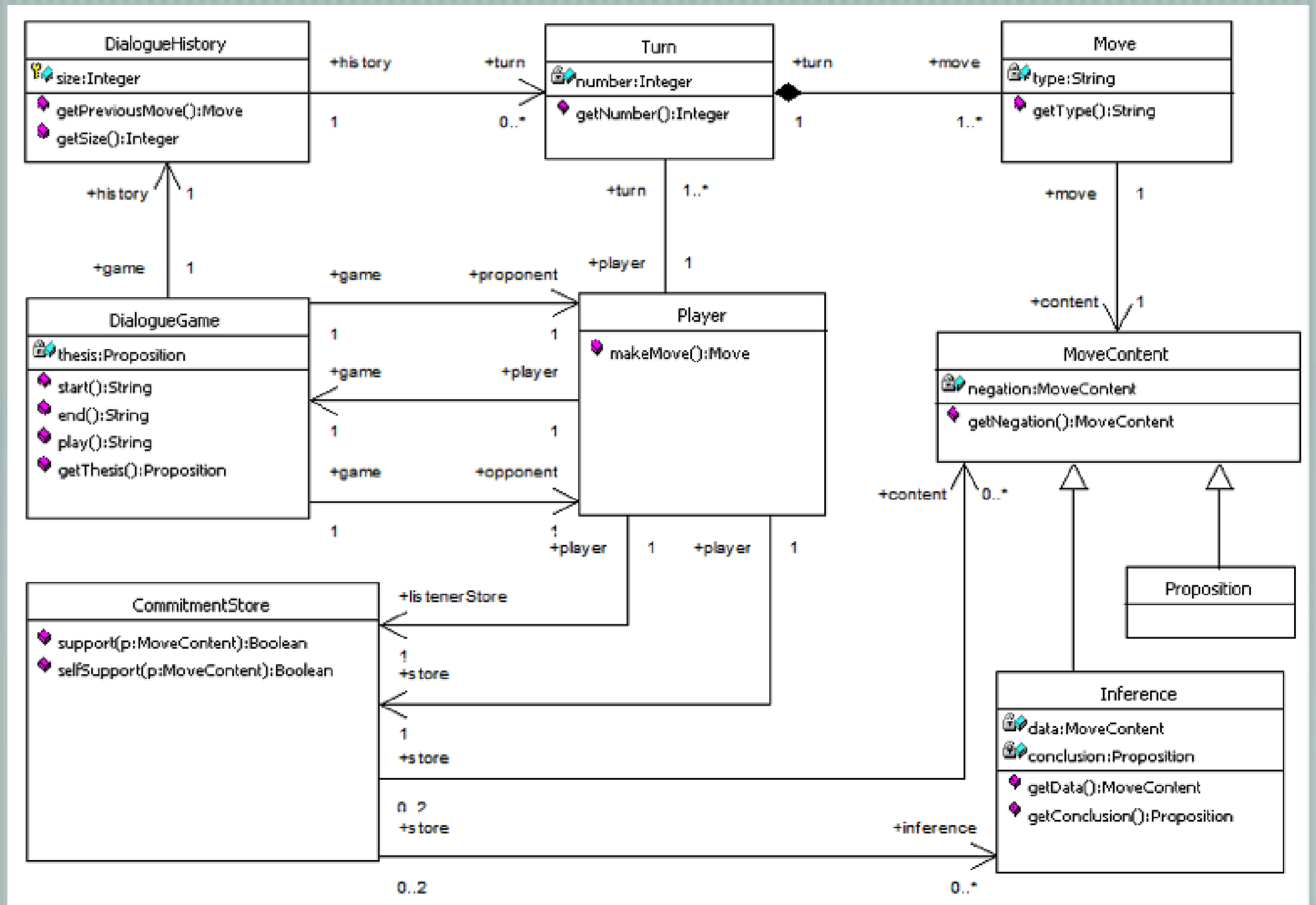
Overview: ProtOCL Lite



Overview: ProtOCL



Object Model



OCL Fragments

Move Types

Assertions: The content of an assertion is a statement P, Q, etc. or the truth-functional compounds of statements: “Not P”, “If P then Q”, “P and Q”.

Questions: The question of the statement P is “Is it the case that P?”
Challenges: The challenge of the statement P is “Why P?”
Withdrawals: The withdrawal of the statement P is “no commitment P”.
Resolution demands: The resolution demand of the statement P is “resolve whether P”.

```
--Player makes a legal move
context Player::makeMove():Move
    --Permitted move types:
    post: Set{'Assertion', 'Question', 'Challenge', 'Resolve', 'Withdrawal'}
        ->includes(result.getType())
```

Execution Platform

- Rules+Agents+Knowledge = Platform
- ProtOCL generated rules
- Java Agents (extend abstract agent classes from the platform) - should be an agent framework (e.g. JADE)
- XML Knowledge-Bases (KBManager Graphical Tool)

Dialogue Game Platform

Game Settings Help

Dialogue History

01: agent2>Is it the case that CP is acceptable?
02: agent1>Yes, I think CP is acceptable.
03: agent2>I think CP is not acceptable.
04: agent1>Is it the case that mistakes rarely happen during judicial process?
05: agent2>Yes, I think mistakes rarely happen during judicial process.
06: agent1>I think 'it is wrong to take a human life' implies 'CP is not acceptable'.
07: agent2>Is it the case that 'execution of murderers is fair for the people being murdered' implies 'murderers should receive capit
08: agent1>Yes, I think 'execution of murderers is fair for the people being murdered' implies 'murderers should receive capital puni
09: agent2>Is it the case that scientific techniques will increase the success of justice?
10: agent1>No, I think It is not the case scientific techniques will increase the success of justice.
11: agent2>I think murderers should receive capital punishment.
12: agent1>I don't think murderers should receive capital punishment.

Move Type Choice

Move Content | Student Position | Computer Position

CP is acceptable
* mistakes rarely happen during judicial process
'it is wrong to take a human life' implies 'CP is not acceptable'
'execution of murderers is fair for the people being murdered' implies 'murderers shou
It is not the case scientific techniques will increase the success of justice
'the recent survey shows that 60% British people support CP' implies 'most people war

Welcome to the game, please choose 'New game' item from the top menu bar!

Benefits

- Flexible, Expressive, & Comprehensive:
  - Dialogue Game API
  - Object Model
- Common/Popular Rules
- Increased testability of game rules
- Reduced likelihood of implementation errors (code gen)

Conclusions/Discussion

- Approaches to specification - many too distant from user(dev) experience
- Identified existing, well supported tools within industry/ commercial software dev
- Developed preliminary workflow for bringing together those software tools with concepts from argumentation domain.