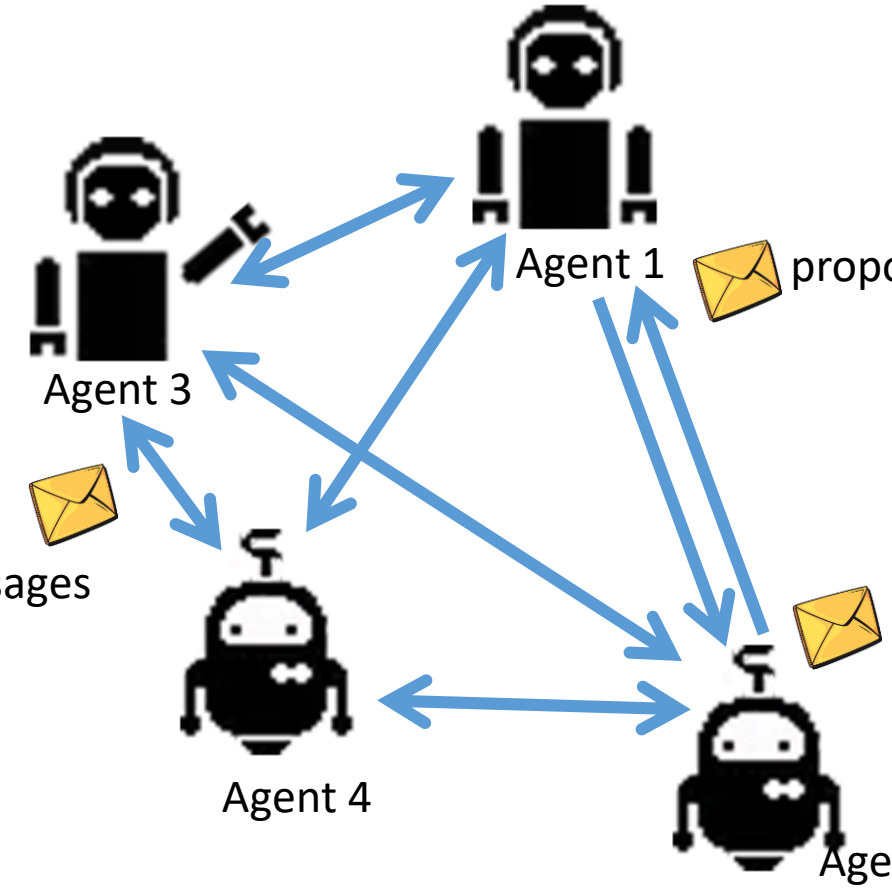
Agent Communication

الدكتور مصطفى رشيد

Definitions

- Protocol is modular, potentially reusable specifications of interactions between two or more components

Agents are not ordinary components. They are autonomous and heterogeneous



Interactions ≡ messages

Sends acceptance to A1
Sends counter proposal to A1
Sends disagreement to A1
Sends rejection to A1

Autonomy Vs. Protocol

- Autonomy means that the agents are free to interact as they please

Protocol : you must ….

Autonomy: would you like to……..

AUTONOMY is interpreted as the ability of an agent to perform high-level reasoning (intelligent agents) or as the degree to which an agent can operate without the supervision of its principal (autonomic agents).

Traditional Software Engineering Approaches (TSEA)

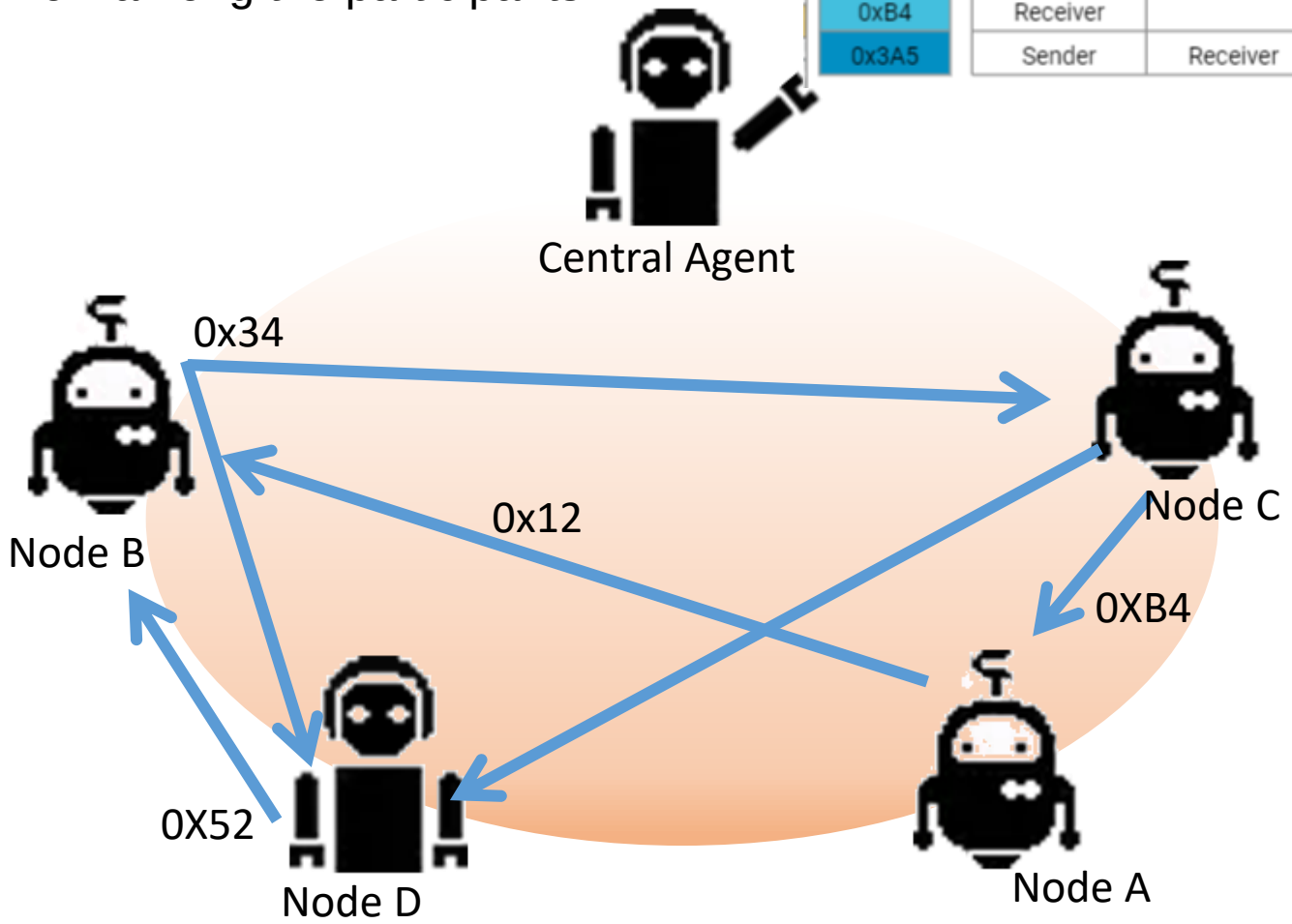
Traditional software engineering approaches for specifying protocols are operational in nature

- Choreographies
- Sequence Diagrams.
- State Machines

Choreographies

A choreography is a specification of the message flow among the participants.

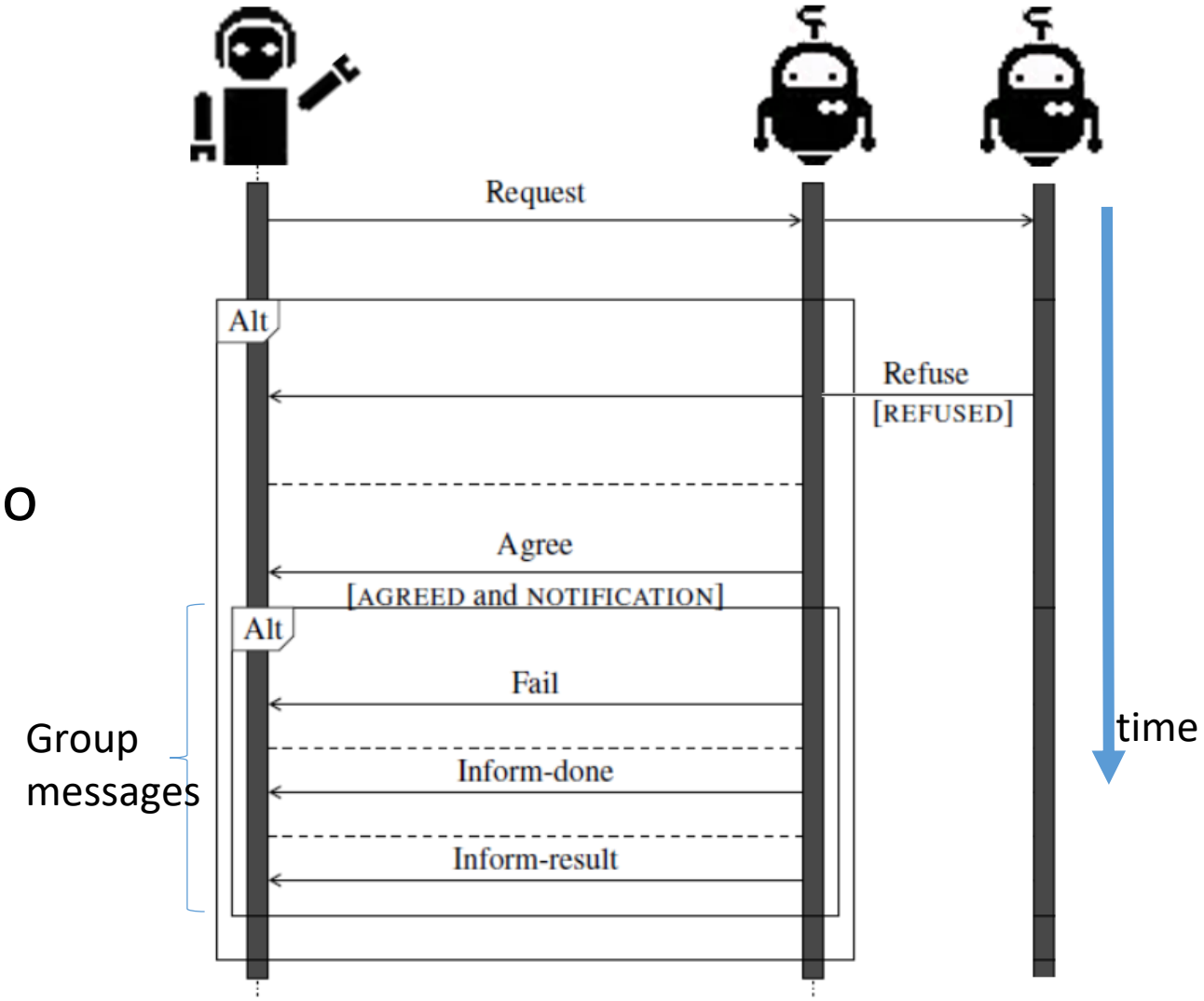
Data Frame	CAN Node A	CAN Node B	CAN Node C	CAN Node D
0x12	Sender	Receiver		
0x34		Sender	Receiver	Receiver
0x52		Receiver		Sender
0x67	Receiver	Receiver	Sender	Receiver
0xB4	Receiver		Sender	
0x3A5	Sender	Receiver	Receiver	Receiver



Sequence Diagrams

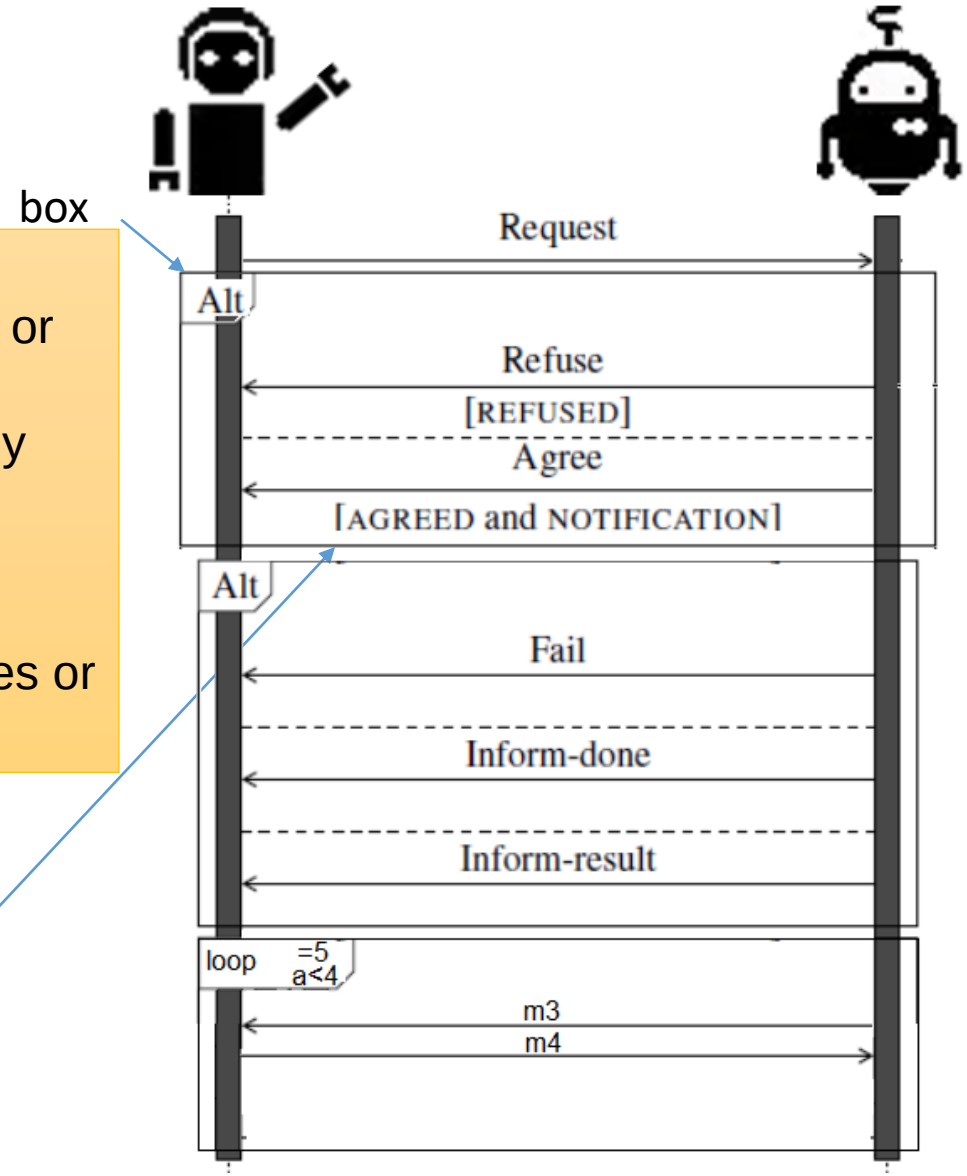
MSCs (Message Sequence Communications) support :-

- Primitives for grouping messages into blocks.
- Alternatives,
- Parallel blocks, or iterative blocks.



Agent Unified Modeling Language (AUML)

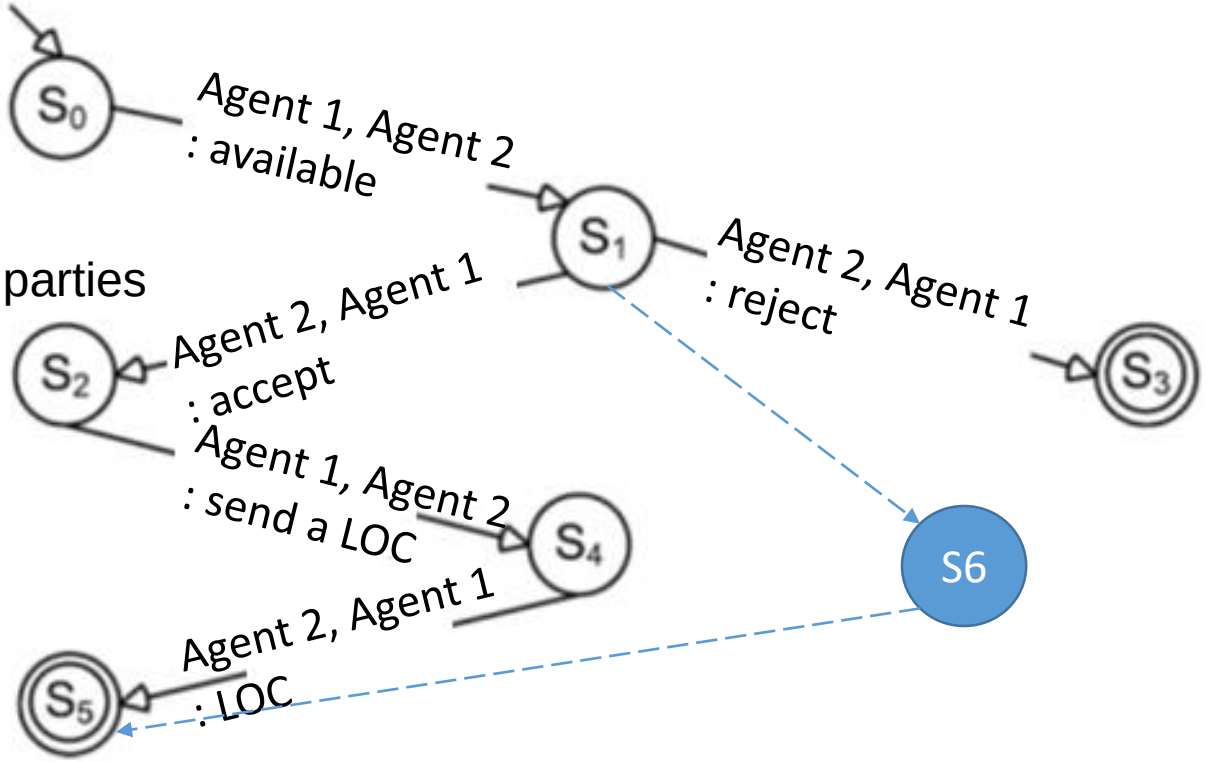
- Alternative: specifies the different available choices.
- Option: Can only have a single region. Specifies that this region may or may not occur.
- Parallel: Specifies that each of the regions takes place simultaneously and the sequence of messages is interleaved.
- Loop: Can only have a single region. Specifies that the region is repeated some number of times/Boolean condition
- Ref: This box type is a little different in that it doesn't contain subboxes or messages. Instead, it contains the name of another protocol.



A guard, denoted by text in square brackets, indicates a condition that must be true in order to send the message

State Machines

The transitions are labeled with messages and both parties



- State machines does not reflect the internal policies based upon which the customer accepts an offer.

Evaluation of TSEA

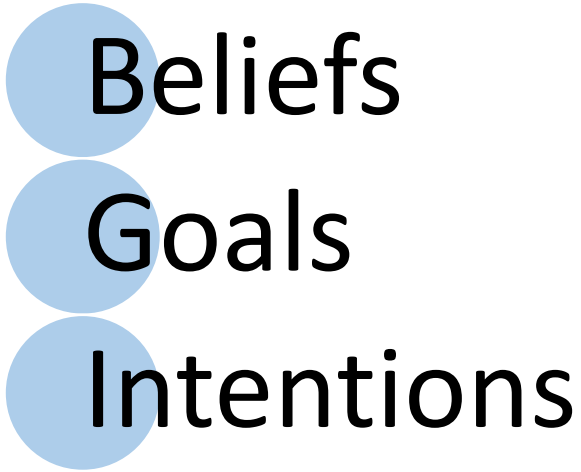
- Instead of specifying the meaning of a communication, they specify the flow of information among agents

Software Engineering		It is difficult to map the business requirements
Flexibility		Agents have little flexibility at runtime (there isn't space for deviation)
Compliance		Checking an agent's compliance with the protocol is easy

*Traditional Software Engineering Approaches

Traditional AI Approaches

- These approaches presume that the agents are constructed based on cognitive concepts



What these approaches had in common was that they were geared toward developing a tool that would assist a user in obtaining information from a database

Knowledge Query and Manipulation Language (KQML)

- Somewhat along the same lines, but with some improved generality

Performative verbs : *request, inform,* and *promise*

should be limited to (beliefs, goals , intentions)

attributes.

```
(ask—one
:content (PRICE IBM ?price)
:receiver stock—server
:language LPROLOG
:ontology NYSE—TICKS
```

The recipient is assumed to 'understand' `LPROLOG/KIF)` (flexibility)

The database that the next attributes should interpreted upon it

- Keywords (preceded by :) can be in any order. Others reply-with, in-reply-to
- Message can be understood by agents – assuming language and ontology knowledge

** 1993 version contains a total of 41 performatives.

Example

```
(evaluate
:sender A :receiver B
:language KIF :ontology motors
:reply-with q1 :content (val (torque m1)))
```

```
(reply
:sender B :receiver A
:language KIF :ontology motors
:in-reply-to q1 :content (= (torque m1)
(scalar 12 N.m)))
```

Example 2

```
(stream-about
:sender A :receiver B
:language KIF
:ontology motors
:reply-with q1
:content m1)

(tell
:sender B :receiver A
:in-reply-to q1
:content (= (torque m1) (scalar 12 N.m)))

(tell
:sender B :receiver A
:in-reply-to q1
:content (= (status m1) normal))

(eos
:sender B :receiver A
:in-reply-to q1)
```

**q1 is query reference number

** eos: end of stream

KQML performatives

- Basic query (evaluate, ask-one, ask-all,…)
- Multi-response query (stream-in, stream-all,..)
- Response (reply, sorry,…)
- Generic info (tell, achieve, cancel, untell, unachieve, …)
- Generator (standby, ready, next, rest,…)
- Capability-definition (advertise, subscribe, monitor,…)
- Networking (register, unregister, forward, broadcast,…)

Drawbacks

- The basic KQML performative set was rather fluid — it was never tightly constrained.
- Transport mechanisms for KQML messages (i.e. ways of getting a message from agent (A to agent *B)* were never precisely defined, again making it hard for different KQML-talking agents to interoperate.
- The language was missing an entire class of performatives — *commissives*, by which one agent makes a commitment to another.
- The performative set for KQML was overly large

FIPA agent communication language

- Foundation for Intelligent Physical Agents(FIPA) 1999.
- Based on KQML
- Limit performative to 20 act.
- it does not mandate any specific language for message content.
- The FIPA ACL semantics is based on a formalization of the cognitive concepts such as the beliefs and intentions of agents.

FIPA performatives categories

accepts a proposal made by another agent

accepts a request to carry out the requested action

performative	passing info	requesting info	negotiation	performing actions	error handling
accept-proposal			x		
agree				x	
cancel		x		x	
cfp			x		
confirm	x				
disconfirm	x				
failure					x
inform	x				
inform-if	x				
inform-ref	x				
not-understood					x
propose			x		
query-if		x			
query-ref		x			
refuse				x	
reject-proposal			x		
request				x	
request-when				x	
request-whenever				x	
subscribe		x			

The performatives provided by the FIPA communication language are categorized

Evaluation of AI Approaches

Software Engineering		The AI approaches offer high-level abstractions, which is a positive. Restricted to mentalist of performative (all agents must have the same database)
Flexibility		Agents have little flexibility at runtime (Restricted to mentalist)
Compliance		It is impossible for an observer to verify the cognitive state of an Agent.

Commitment-Based Multiagent Approaches

- C(*debtor, creditor, antecedent, consequent*) ,

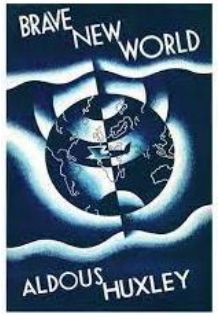
,

, if ,

student

Example: *(Commitment) C*(*EBook, Alice*,$12,*BNW*)

publisher



Commitment-Based Multiagent Approaches

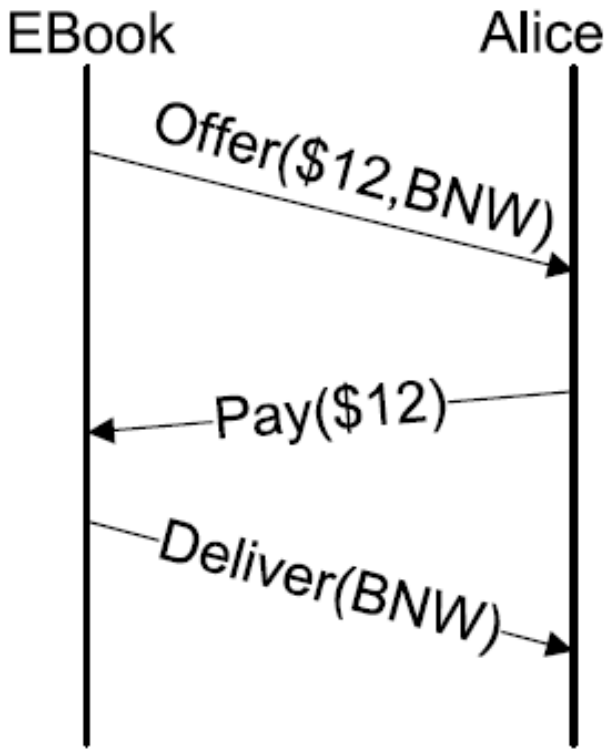
- C(*debtor, creditor, antecedent, consequent*) ,

Example: *(Commitment) C*(*EBook, Alice,*$12*,BNW*)

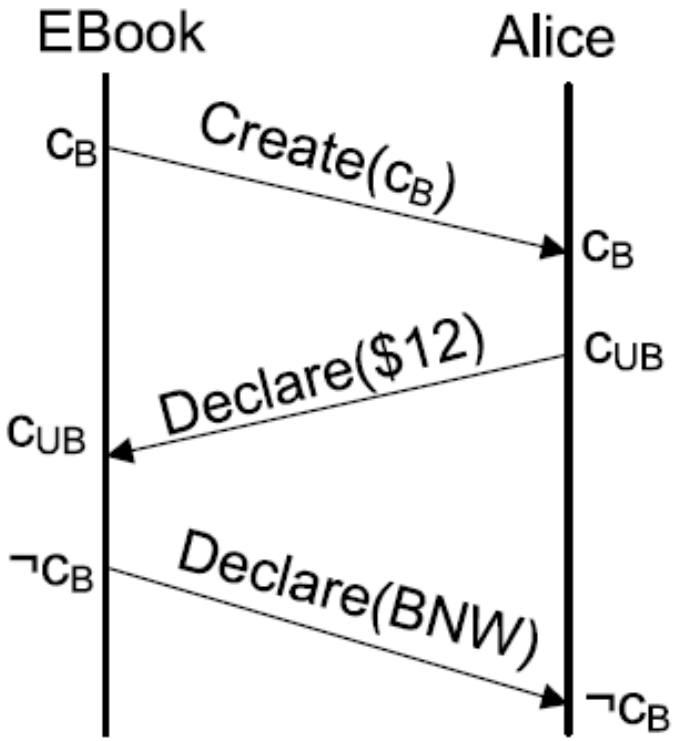
$$C(EBook, Alice,\$12, BNW) \wedge \$12 \Rightarrow C(EBook, Alice, T, BNW)$$

$C(EBook, Alice,\$12, BNW) \wedge \$12 \Rightarrow C(EBook, Alice, T, BNW)$

Sequence Diagram



Messaging



Meaning

imposes penalties on parties that violate their commitments

Evaluation of commitment Protocols

Software Engineering		Commitments offer a high-level abstraction for capturing business interactions
Flexibility		enhances flexibility over traditional approaches by expanding the operational choices for each party
Compliance		Protocol enactments can be judged correct as long as the parties involved do not violate their commitments.

	Traditional SE	**Traditional AI**	**Commitment Protocols**
Abstraction	control flow	mentalist	business relationship
Compliance	lexical basis	unverifiable	semantic basis
Flexibility	low	low	high
Interoperability	message level	integration	business level