Programming MAS without Programming Agents

Munindar P. Singh and Amit K. Chopra

North Carolina State University, Università degli Studi di Trento

May 2009
Architecture

How a system is organized

- Primarily its ingredients
  - Components
  - Interconnections

- An architecture is motivated by
  - Stakeholders’ requirements
  - The environment in which it will be instantiated as a system
Open Architecture

- **Openness**: specifying the interconnections cleanly
  - Physical components *disappear*
  - Their *logical* traces remain

- **Protocols**: interconnections in open information environments
Architectural Style

- A style identifies
  - (Architectural) Constraints on components and interconnections
  - Patterns on components and interconnections
- A style yields a language (possibly also a notation) in which we present the architecture of a particular system
Requirements for a MAS Architecture

- The components are *agents*
  - Autonomous
  - Heterogeneous
- The environment provides
  - Communication: inherently *asynchronous*
  - Perceptions
  - Actions
  - For IT environments, we can treat all as communications
- The stakeholders require the agents to *interoperate*
Criteria for Judging Interconnections

The purpose of the interconnections is to support interoperation of the components

- **Loose coupling**: support heterogeneity
- **Flexibility**: support autonomy, enabling participants to extract all the value they can by exploiting opportunities and handling exceptions
- **Encapsulation**: promote modularity
- **Compositionality**: promote reuse
Interoperation

- Two or more components interoperate when each meets the expectations that each of the others places on it
  - Neither about control flow, nor about data flow [Parnas, 1972]
- Most, if not all, subsequent software engineering research considers only control or data flow
  - Ill-suited for MAS
- Challenges for expectations
  - How may we characterize them except via flow?
  - How may we verify or ensure they are met?
Protocols, Generally
Consider networking or even power

- Protocols encapsulate the allowed interactions
  - *Connect*: conceptual interfaces
  - *Separate*: provide clean partitions among logical components
- Wherever we can identify protocols, we can
  - Make interactions explicit
  - Identify markets for components
- Protocols yield standards; their implementations yield products
Specifying MAS Protocols

In light of the above criteria

- Procedural, which specify the *how*
  - Finite state machines, Petri nets
  - Generally over-specify the interactions, thus limiting flexibility and coupling the components tightly

- Declarative, which specify the *what*
  - Logic in its various forms
  - Not necessarily higher level than the procedural approaches
  - Most valuable when the conceptual abstractions promote loose coupling and flexibility
Proposed Approach
Agent communication done right

- **Syntax:** documents to be exchanged as messages
- **Semantics:** formal meaning of each message
  - For business applications, expressed via *commitments*
  - For other situations, potentially expressed via other suitable constructs in a like manner
- **Minimal operational constraints**
  - It is surprising how few constraints are truly needed
Compliance means not violating a commitment

- \( C(\text{debtor}, \text{creditor}, \text{antecedent}, \text{consequent}) \)
  - \( C(\text{EBook}, \text{Alice}, \$12, \text{BNW}) \)

- **DETACH:** \( C(x, y, r, u) \land r \rightarrow C(x, y, \top, u) \)
  - \( C(\text{EBook}, \text{Alice}, \$12, \text{BNW}) \land \$12 \rightarrow C(\text{EBook}, \text{Alice}, \top, \text{BNW}) \)
  - \( C(\text{debtor}, \text{creditor}, \top, \text{consequent}): \text{unconditional commitment} \)

- **DISCHARGE:** \( u \rightarrow \neg C(x, y, r, u) \)
  - \( \text{BNW} \rightarrow \neg C(\text{EBook}, \text{Alice}, \$12, \text{BNW}) \)
  - \( \text{BNW} \rightarrow \neg C(\text{EBook}, \text{Alice}, \top, \text{BNW}) \)
Ensuring Interoperation: 1

Traditional representations

- Traditional SE approaches handle only control and data flow
  - A poor notion of interoperation, but at least they have it
- Traditional AAMAS approaches make onerous demands
  - The beliefs—similarly, desires or intentions—of the parties involved must be
    - *Determinable*: impossible without violating heterogeneity
    - In *mutual agreement*: impossible without violating autonomy and asynchrony
  - Hence, no viable notion of interoperation
Ensuring Interoperation: 2
Commitment alignment

- Two sources of asymmetry in MAS
  - Communications are directed: direction of causality is from sender to receiver
  - Commitments are directed: direction of expectation is from creditor to debtor

- Our definition of *alignment* is asymmetric
  - Whenever a creditor computes a commitment, the debtor computes the same commitment
    - Finesse in “whenever”
Ensuring Interoperation: 3
Alignment and asynchrony

► When a debtor autonomously creates a commitment, it sends a corresponding message, which lands at the creditor
  ► Here the debtor is committed before the creditor learns of it
► When a creditor detaches a commitment, thereby strengthening it, a message corresponding to the detach eventually arrives at the debtor
  ► Here the debtor is committed when it receives the detach
  ► This motivates a treatment of *quiescence* wherein we only consider well-formed points in executions where each message has landed
► When a debtor or creditor learns that a commitment is discharged or detached, respectively, it must immediately notify the other (*integrity*, which ensures no quiescence until the information has propagated)

[Chopra & Singh, AAMAS 2009]
To program a MAS, define a protocol

- Specify roles
- Specify messages
- Specify meaning messages in terms of commitments
- Specify any additional constraints

The above is valid because we have formalized interoperability based on commitments

We can compose protocols
To instantiate and enact a MAS, identify agents to play roles in its protocol
  - Could be preexisting agents proceeding on their own initiative
  - Could be new agents instantiated from preexisting code-bases
  - Could be custom agents
Offer primitives encapsulated as programming abstractions or middleware by which agents can
- Communicate with each other
- Maintain their commitments as debtors
- Maintain their commitments as creditors
- Verify each other’s compliance: are any commitments not being discharged?
- Ensure that the constraints required for interoperability are applied

[Chopra & Singh, ProMAS 2009]
Beady Eye

_Not for the architecture guy_

Also known as BDI

- Violates heterogeneity: presumes knowledge of agent internals
- Prevents alignment in settings involving asynchrony
- Tightly couples the agents
- Leads to invalid assumptions such as sincerity in communication
Aside: Notation

Important but secondary to concepts

When we describe an architecture

- What matter most are the concepts using which we do so
- Notation is important, but less so
  - Existing notations are not complete for our purposes
  - A contribution of MAS research is to invent suitable notations
Conclusions

- In attempting to develop practical MAS, AAMAS approaches
  - Adopt traditional software ideas wholesale, thus neglecting the key features that characterize MAS
  - Seek to differentiate themselves from traditional SE, mainly through BDI
- Approaches that ignore asynchrony, autonomy, heterogeneity are unacceptable
- Higher-level concepts yield interconnections that support MAS applications
  - Provided we realize them correctly to yield interoperation
Directions

- CSOA, Commitment-based service-oriented architecture [Singh, Chopra, Desai, IEEE Computer 2009]
  - An architecture style that treats business (not technical) services as agents, and includes patterns for service engagements
- Business modeling language [Telang & Singh, SOCASE 2009]
  - A way to express a MAS in terms of the business relationships among agents as conglomerates of commitments
  - A way to verify the computations realized with respect to business models