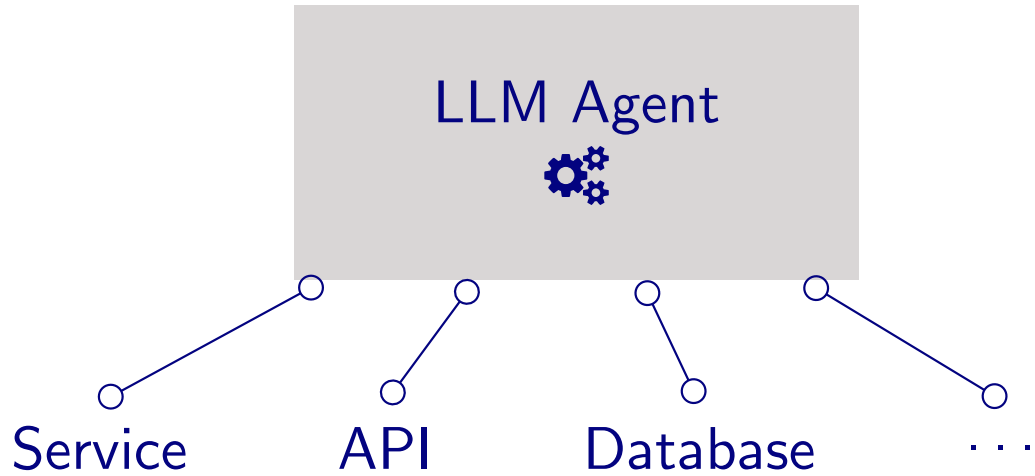
Agentic AI Needs Interaction-Oriented Programming

Amit K. Chopra
(Munindar P. Singh)

amit.chopra@lancaster.ac.uk

Autonomous Agents on the Web Community Group, August 2025

Agentic AI

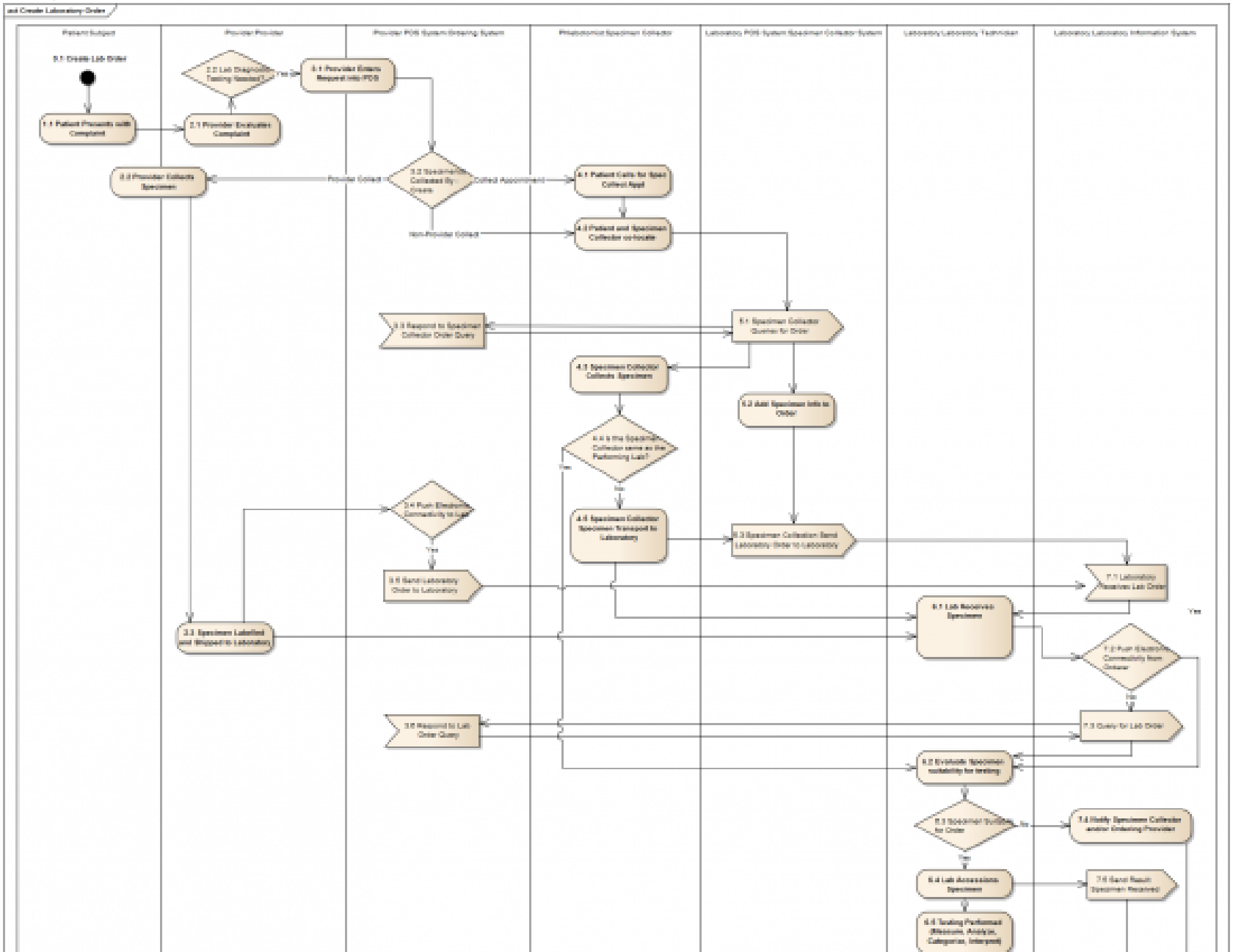


Large Language Models (LLMs) taking actions

(Model Context Protocol (MCP) promotes a standard way of describing, discovering, and using tools)

Workflows

Don't try to read!



Agentic AI as Workflow

Orchestrated execution of tasks, which may involve LLM agents and are written in programming languages such as Python

Framework	*Idea*
LangGraph, Autogen	Enables programming "multiagent" workflow
AFlow	LLM generates the "multiagent" workflow needed to solve the task

Exploit LLMs to the hilt!

No more programming! LLMs will plan and generate a custom multiagent system to put it into action

Stumbling Block: LLMs Are Unreliable

Actions occur in a sociotechnical system involving autonomous real-world principals and have normative consequences

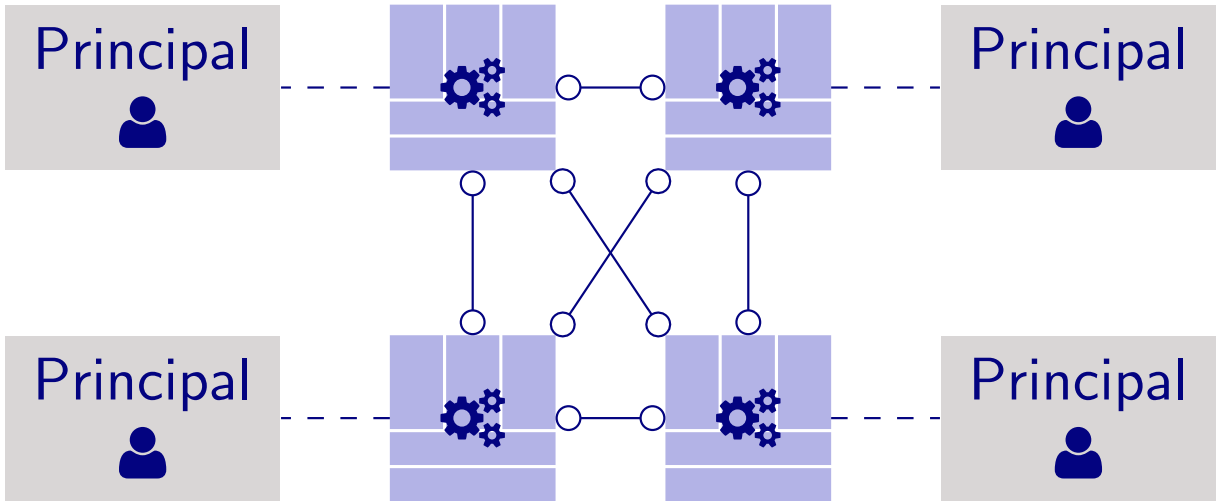
- May create, discharge, or violate commitments
- May affect trust in others
- Are accountable!

Can we rely on LLMs to take actions on behalf of its user?

ChatGPT, via Operator, can fill forms, but requires users to take actions

Challenge: LLM Agents Help Principals Exercise Autonomy

In *real* multiagent systems



Agents

- Are heterogeneous in construction
- Encode decision making of their respective principals
- Interact via arms length communications
- Cannot generate another!
- Decentralized, not orchestrated

Interaction-Oriented Programming (IOP)

Empower stakeholders and programmers

Method

- Model a multiagent system in terms of interactions
- Compose and verify models
- Implement agents independently on the basis of models

High-level abstractions that

- Reflect stakeholder intuitions and
- Let programmers focus on the business logic

Synthesis

Current Agentic	**Current IOP**
Centralized	Decentralized
Minimal knowledge engineering	Declarative means low effort
Unreliable	Formal models capture interaction meaning and guide agents

Engineering is a science of the artificial (Herb Simon)

Models important because they precisely capture requirements of the engineered artifact

Information Protocols in BSPL (MPS, 2011)

Declarative, no control flow, no message ordering

Protocol specifies the computation of a decentralized information object (a tuple) via messages

- Messages contain information and convey meaning
- Keys identify enactments (*business transactions*)
- Causality adornments determine when a role may send a message in an enactment

Think of Message Meaning

Toward specifying a protocol for conducting ebusiness

```
Ebusiness {
  roles Buyer, Seller, Bank
  parameters ID key, Item, Price, Status

  Seller -> Buyer: offer[Id key, Item, Price]

  Buyer  -> Seller: accept[Id key, Item, Price, Decision]

  Buyer  -> Bank: instruct[Id key, Price, Details]

  Bank   -> Seller: transfer[Id key, Price, Details,
      Payment]

  Seller -> Buyer: shipment[Id key, Item, Price, Status]

  Seller -> Bank:  refund[Id key, Item, Payment, Amount,
      Status]
}
```

Think of Causality

accept, instruct, and shipment (anytime) after offer; transfer after instruct; refund after transfer and mutually exclusive with shipment

```
Ebusiness {
  roles Buyer, Seller, Bank
  parameters out Id key, out Item, out Price, out Status

  Seller -> Buyer: offer[out Id key, out Item, out Price]

  Buyer  -> Seller: accept[in Id key, in Item, in Price,
      out Decision]

  Buyer  -> Bank: instruct[in Id key, in Price, out
      Details]

  Bank   -> Seller: transfer[in Id key, in Price, in
      Details, out Payment]

  Seller -> Buyer: shipment[in Id key, in Item, in Price,
      out Status]

  Seller -> Bank:  refund[in Id key, in Item, in Payment,
      out Amount, out Status]
}
```

Flexibility: Ebusiness has 658 distinct enactments

```
>bspl verify all_paths Ebusiness.bspl --verbose
(Seller!offer, Seller!shipment, Buyer?shipment,
   Buyer!accept, Buyer!instruct, Bank?instruct,
   Buyer?offer, Bank!transfer, Seller?accept,
   Seller?transfer)
(Seller!offer, Buyer?offer, Buyer!instruct, Buyer!
   accept, Bank?instruct, Bank!transfer, Seller?
   accept, Seller!shipment, Buyer?shipment, Seller
   ?transfer)
...
```

Agent Programming Model Based on Protocols (with Samuel H. Christie)

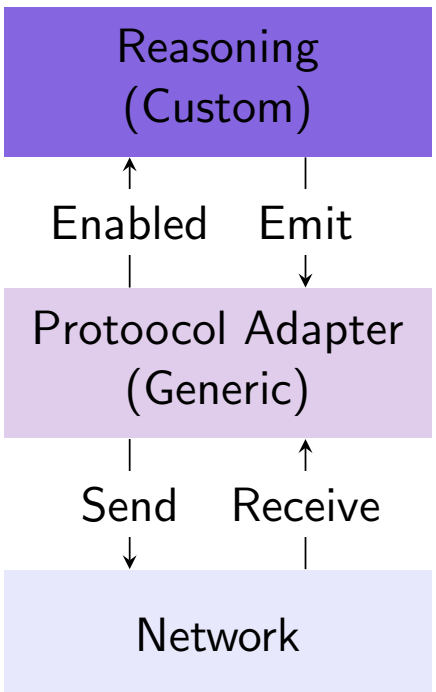
Agent's reasoning selects an information-enabled messages, fleshes it out, and emits it

offer(1, fig, £5)

Seller's Local State

offer(Id, Item, Price)
shipment(1, fig, £5, Status)

Seller's Enabled Messages




```
@adapter.schedule(* 17 * * *)
def decision_shipment(enabled):
    es = enabled["shipment"]
    for s in es
        s["Status"] = Nextday
```

Specifying Normative Meaning Explicitly

```
1: commitment OfferCom Seller to Buyer    //if transfer,
       then shipment
2:    create offer
3:    detach transfer[, created OfferCom + 5]
4:           where "Payment>=Price"
5:    discharge shipment [, detached OfferCom + 5]

7: commitment AcceptCom ... //if shipment, then transfer
8: commitment RefundCom ... //if violated OfferCom, then
       refund
9: commitment TransferCom ... //if instructed, then
       transfer
```

Programming Model (with Matteo Baldoni & SHC)

Upon detachment of RefundCom, Seller sends Refund

```
1: +ev_now_detached_RefundCom(Seller, Buyer, Id, Item,
       Price, Bank, Payment, Timestamp)
2:   <- !send_refund(Id, Item, Payment, Bank).
```

Seller sends shipment if OfferCom is nearing violation

```
1: +!handle_form([shipment(Id, Item, Price, out)[
       receiver(Buyer)]|_])
2:   : in_stock(Item) & violated_OfferCom(Id,..., T) &
       system_time(Now) & T <= Now + 10
3:  <- !send_shipment(Id, Item, Price, Buyer).
```

IOP is Heliocentric

Geocentrism of Traditional Approaches Based on Interactions

- No consideration of meaning
- Protocols as state machines, workflow, UML, etc.
- Inordinate focus on message formats
- Programmer implements interactions in low-level code
- Complex message delivery assumptions
- No flexibility

FIPA ACL & KQML

Of historical importance only

- Inspired from Austin's speech acts
- Handful of act types, e.g., inform, promise, achieve, ...
- KQML informal, intended for closed settings
- FIPA ACL semantics in terms of mental states
  - To promise something means speaker intends to do that thing
  - To assert something means speaker believes it
- Adopted in agent programming languages and frameworks, e.g., Jason and JADE

Limitations

- Types of communicative acts in practice are virtually unlimited
- Impossible to determine compliance in open systems

IOP Fixes Their Limitations

- Every message in a protocol is a communicative act
- Social meaning of messages is specified separately
  - No reference to mental states of agents
- Decentralized operations
- Better, more faithful realization of Austin's ideas

Time for New Agent Communication Standards based on IOP

Running code we have lots, just need rough consensus

`https://gitlab.com/masr`

Agentic Directions

1. Methodologies for exploiting LLMs to specify norms and protocols (with the help of IOP verification tools)
2. Exploit LLMs to enable an agent to act intelligently by using IOP programming models that support normative reasoning and protocol enactment
3. ...

Synthesis

Current Agentic	**Current IOP**
Centralized	Decentralized
Minimal knowledge engineering	Declarative means low effort
Unreliable	Formal models capture interaction meaning and guide agents

Engineering is a science of the artificial (Herb Simon)

Models important because they precisely capture requirements of the engineered artifact