Interaction-Oriented Programming: Intelligent, Meaning-Based Business Processes

Amit K. Chopra, [1] **Samuel H. Christie V,** [2] **Munindar P. Singh** [2]

[1] Lancaster University
[2] North Carolina State University
amit.chopra@lancaster.ac.uk, schrist@ncsu.edu, singh@ncsu.edu

Abstract

Traditional representations of business processes such as workflows specify a rigid ordering of actions or events. Such representations are incompatible with intelligent decision making by the principals involved in the business process. Interaction-Oriented Programming (IOP) is an approach that accommodates intelligent decision making by modeling the *business meaning* of interactions. IOP addresses every architectural element of business processes, from communication services for business messaging to business protocols and business contracts. In this demo, we showcase the tools developed over the last decade that enable specifying, verifying, and implementing meaning-based, decentralized business processes.

Interaction-Oriented Programming

Our research program, titled *Interaction-Oriented Programming* (IOP), concerns software abstractions for *decentralized business processes*, that is, systems that support interactions between multiple *autonomous* business principals. Autonomy refers to intelligent, flexible decision making. IOP stems from a single question

> How can we represent autonomy in software for business processes?

Current business process modeling technologies are typically based either on workflows (a centrally executed program) or choreographies (constraints on message ordering). Such approaches yield unduly rigid models of processes and are therefore incompatible with intelligent, decentralized decision making..

IOP unifies decentralized decision making with business process enactment. IOP has a single solution concept at its heart.

> Representing autonomy in software requires representing the *business meaning* of communications.

E.g., in an ebusiness transaction between a buyer and a seller, an *Offer* message specifying an item and a price means the corresponding real-world offer (a domain object), which itself means a real-world commitment (a higher-level domain object) from the seller to the buyer for *Delivery* of the item if *Payment* of the price occurs. The meaning of a *CancelOffer* is to rescind some *Offer*; additionally, it means a commitment from the seller to *Refund* the buyer's *Payment* if already made. When principals make decisions, they do so based on such meaning.

Communication meaning has a rich history in AI, marred though by early conceptual errors (Singh 1998). Our purpose with this demo is to show that the promising approach of *social meaning* has come of age; that it enables engineering business processes as loosely-coupled, decentralized multiagent systems. We demonstrate software for the following purposes.

Demo: Specifying Business Contracts

We will demonstrate how a business processes may be specified in terms of declarative *norms*, which may be understood as elements of business contracts. We show how a traditional information store, e.g., a relational database may understood in terms of norms, thus supporting decision making (Chopra and Singh 2015, 2016).

Listing 1: A commitment specification.

```
base events
  quote(S, B, ID, item, price, t)
  accept(S, B, ID, item, price, addr, t)
  pay(S, B, ID, item, price, amt, t)
  deliver(S, B, ID, addr, status, t)
  refund(S, B, ID, amt, rAmt, t)

commitment PurchaseCom S to B
  create quote
  detach (accept and pay) within quote + 5d
      where amt >= price
  discharge deliver within detached PurchaseCom
      + 10d

commitment Compensation S to B
  create quote
  detach violated PurchaseCom
  discharge refund within violated PurchaseCom
      + 2d where rAmt >= amt
```

We demonstrate Cupid, a compiler that outputs SQL queries given norm specifications such as the one in Listing 1. The specification specifies the atomic business events (e.g., *quote*), each with a key (the underlined attributes) for identifying its instances and a distinguished attribute `t` for timestamping purposes. The atomic events maps to relations in relational databases (the compiler in fact generates 'Create table...' statements.)

On top of the atomic events, commitments that capture business relationships may be specified. Listing 1 shows two commitments from S (seller) to B (buyer). `PurchaseCom` captures that S will make timely deliveries to B upon timely payment; `Compensation` captures that S will compensate B for violated instances of `PurchaseCom`.

The generated SQL queries enable principals to query databases in terms of commitment events, e.g., discharge, violation, expiry, and so on; in other words, in terms of *key performance indicators* (KPI). The queries run into hundreds of lines, illustrating that as simple and declarative as the commitments are, writing the relevant SQL queries by hand—the only alternative today—would be highly cumbersome and error-prone.

Demo: Business Contracts on Blockchain

Smart contracts as representations for business contracts are a nonstarter for several reasons, but two stand out. One, they can't actually guarantee outcomes; two, they are antiautonomy (Singh and Chopra 2020).

We demonstrate Hercule, software that interprets business contracts based on norms over blockchains (Christie V, Chopra, and Singh 2021). Inspired from Cupid, Hercule enables laying out norms-based contracts on Hyperledger Fabric (Androulaki et al. 2018) and tracking norm states on the blockchain. Hercule is adapted to document-oriented databases, specifically CouchDB, the underlying database of the Hyperledger Fabric blockchain.

Hercule demonstrates that you can deploy realistic business contracts on blockchain. Hercule contracts, in contrast to smart contracts, support flexible decision making, since they don't update the blockchain themselves, being just queries. Yet, they enable tracking the states of the contracts, thereby supporting compliance monitoring and accountability processes.

Demo: Specifying Business Protocols

Enacting a Cupid or Hercule business contract requires a declarative business protocol that captures the relevant information causality and integrity constraints. We will demonstrate how a business contract may be enacted in a decentralized manner on the basis of *information protocols* (Singh 2011). Listing 2 shows a protocol over which the commitments in Listing 1 could be operationalized.

Business protocols may be erroneously specified. For example, they may be specified in such a way that the agents enacting the protocols may make inconsistent decisions. Such protocols would not be *safe* (Singh 2012). Alternatively, a protocol may be such that on some branch, a decision required for termination can never be made. Such protocols would not be *live* (Singh 2012). We demonstrate tools for efficiently verifying the liveness and safety of protocols, in addition to properties such as refinement (which captures whether a protocol can substitute for another) (Christie V, Chopra, and Singh 2020) and atomicity (which is the idea

Listing 2: An information protocol.

```
Purchase {
  role B, S
  parameter out ID, out item, out amt, out
      status

  S ↦ B: quote[out ID, out item, out price]
  B ↦ S: accept[in ID, in item, in price, out
      addr]
  B ↦ S: pay[in ID, in item, in price, out
      amt]
  S ↦ B: deliver[in ID, in addr, out status]
  S ↦ B: refund[in ID, in amt, out rAmt, out
      status]
}
```

that composite protocols not have any unterminated constituent protocols) (Christie V, Chopra, and Singh 2018).

A significant challenge for workflow-based business processes is relating to the content of communications. We demonstrate how our approach unifies coordination with content and in that supports business meaning, including in terms of business contracts.

Demo: Decision-Based Programming

In a business process, each communication represents a principal's decision. We will show how to implement a principal's software, that is, its *agent* in accordance with its own decision making policies (Christie V, Chopra, and Singh 2022). We will demonstrate flexibility in decision making that is not possible with alternative technologies, e.g., workflow-based.

Specifically, we introduce a decision-oriented programming model in which one implements an agent by writing *decision handlers*. The idea is that in any state, an agent can make a set of decisions. A decision handler picks some of those potential decisions, fleshes them out with the missing information by applying some internal business logic and communicates them via messages. Our programming model guarantees compliance and supports automatic correlation of decisions, multiprotocol decision making, and atomically making composite decisions that consists of several decisions-as-messages, among other things. Additionally, agents are fully asynchronous and operate purely on the basis of information and meaning rather than message ordering, as is commonplace in alternative software implementations of business processes. This enables our agents suitable to the IoT and they can be essentially be deployed as loosely-coupled microservices.

Related Software

Cupid is available at https://github.com/akchopr/Cupid/.

All the other software, along with the relevant documentation is available at https://gitlab.com/masr/bspl/.

We will offer all of the above demonstrations to attendees; they can pick the demos they are interested in.

References

Androulaki, E.; Barger, A.; Bortnikov, V.; Cachin, C.; Christidis, K.; Caro, A. D.; Enyeart, D.; Ferris, C.; Laventman, G.; Manevich, Y.; Muralidharan, S.; Murthy, C.; Nguyen, B.; Sethi, M.; Singh, G.; Smith, K.; Sorniotti, A.; Stathakopoulou, C.; Vukolic, M.; Cocco, S. W.; and Yellick, J. 2018. Hyperledger Fabric: A distributed operating system for permissioned blockchains. In *Proceedings of the Thirteenth EuroSys Conference*, 30:1–30:15. ACM.

Chopra, A. K.; and Singh, M. P. 2015. Cupid: Commitments in Relational Algebra. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence*, 2052–2059.

Chopra, A. K.; and Singh, M. P. 2016. Custard: Computing Norm States over Information Stores. In *Proceedings of the 15th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 1096–1105. Singapore: IFAAMAS.

Christie V, S. H.; Chopra, A. K.; and Singh, M. P. 2018. Compositional Correctness for Multiagent Interactions. In *Proceedings of the 17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 1159–1167. São Paolo: IFAAMAS.

Christie V, S. H.; Chopra, A. K.; and Singh, M. P. 2020. Multiagent protocol refinement. In *Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*. IFAAMAS. 258–266.

Christie V, S. H.; Chopra, A. K.; and Singh, M. P. 2021. Hercule: Representing and Reasoning about Norms as a Foundation for Declarative Contracts over Blockchain.

Christie V, S. H.; Chopra, A. K.; and Singh, M. P. 2022. Mandrake: Multiagent systems as a basis for programming fault-tolerant decentralized applications. *Autonomous Agents and Multi-Agent Systems*, 36(16): 30.

Singh, M. P. 1998. Agent Communication Languages: Rethinking the Principles. *IEEE Computer*, 31(12): 40–47.

Singh, M. P. 2011. Information-Driven Interaction-Oriented Programming: BSPL, the Blindingly Simple Protocol Language. In *Proceedings of the 10th International Conference on Autonomous Agents and MultiAgent Systems*, 491–498.

Singh, M. P. 2012. Semantics and Verification of Information-Based Protocols. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 1149–1156. Valencia, Spain: IFAAMAS.

Singh, M. P.; and Chopra, A. K. 2020. Computational Governance and Violable Contracts for Blockchain Applications. *IEEE Computer*, 53–62.