The Elements of Decision Alignment

Mark S. Miller[1] and Bill Tulloh[2]

1 erights@google.com
2 btulloh@gmail.com

Abstract

When one object makes a request of another, why do we expect that the second object's behavior correctly satisfies the first object's wishes? The need to cope with such *principal-agent problems* shapes programming practice as much as it shapes human organizations and economies. However, the literature about such plan coordination issues among humans is almost disjoint from the literature about these issues among objects. Even the terms used are unrelated.

These fields have much to learn from each other—both from their similarities and from the causes of their differences. We propose a framework for thinking about *decision alignment* as a bridge between these disciplines.

1998 ACM Subject Classification D.2 Software Engineering, F.3 Logics and Meanings of Programs

Keywords and phrases economics, law, contracts, principal-agent problem, incentive alignment, least authority, verification

Digital Object Identifier 10.4230/LIPIcs.ECOOP.2016.17

1 Extended Abstract

Many complex systems can be described as networks of principal-agent relationships, in which a requesting entity, the *principal*, sends a request to another entity, the *agent*, who is expected to perform the actions the principal intends. Of course, an agent may decide to do something else instead. Thus, these requests usually occur within a framework in which the principal uses various techniques to try to align the decision of the agents with the interests of the principal.

Economics, organizational theory and political science study principal-agent relationships among humans. Language designers, programmers, and software engineers deal with principal-agent relationships among computational objects. When both principal and agent are humans, the hazards and techniques are examined in terms of divergent interests, asymmetric information, contracts, and incentives[1, 4]. When both principal and agent are objects, the hazards and techniques are examined in terms of abstraction mechanisms, object design patterns, expressiveness, and reliability engineering. In the human case, agents are assumed to have their own interests, and must therefore be coaxed into performing what the principal wants. In the object case, the agent's intentions are generally assumed benign[1], so the goal is to minimize unintentional misbehavior.

A human principal makes requests of a software agent via a user interface. In *HCI*, the study of human-computer interaction, the agent is generally assumed benign and bug-free. The focus is on request expressiveness and user confusion. The hazard is that the agent does

[1] Except computer security, which does consider intentional misbehavior

30th European Conference on Object-Oriented Programming (ECOOP 2016).
Editors: Shriram Krishnamurthi and Benjamin S. Lerner; Article No. 17; pp. 17:1–17:5

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Table 1 The Elements of Decision Alignment.

	Human to Human	Human to Object	Object to Object
Explanation	Language	User interface	Abstraction
Rights	Law, Contracts	App permissions Powerbox	Security Protection patterns
"Incentives"	Economics	Objective functions	Machine learning Agorics
Static Inspection	Accounting controls	Trusted path URL bar	Types, Verification Open source eyeballs
Dynamic Monitoring and Feedback	Reviews, Complaints Word of mouth	Bug reports	Contracts, Testing Backprop
Trust and Reputation	Trademark Chain of custody	App stores White and black lists	Trusted developer Same origin

what the user asks for rather than what the user wants. HCI examines how to design an agent's interface to shape user behavior so that users ask for what they want.

Mixed cases are important. Humans increasingly use user-interfaces to express intent and interact with other humans. Indeed, human-to-human intentions are increasingly expressed, constrained, and transformed through software intermediaries, whether email, social networks, or ride sharing.

A unified view, that looks at the different techniques used to deal with each of these three cases, can provide important insights. We identify six broad categories of techniques principals use to align agent decisions with the principal's intent. A common theme throughout is that the principal and agent know different things. By virtue of specialization, they each embody information inaccessible to the other.

Explanation The principal must express its intent as a request that the agent can understand. Only the principal has detailed knowledge of *why* they make this request. Only the agent has detailed knowledge of *how* to fulfill the request. The request conveys *what* the principal needs in terms of concepts they share[5, 10]. A good abstraction boundary composes their separate knowledge to achieve a greater result[2, 8].

Rights All actions that the agent performs, whether it serves the principal's interests or not, must be among the actions that are possible for the agent to take. If the agent's range of possible actions is narrow, this limits its abilities to misbehave in ways that harm the principal. But if too narrow, then the agent cannot do what the principal asks[9]. We return to this issue below.

"Incentives" Some agents change their behavior in order to optimize some externally provided metric. These can be humans responding to prices, or machine learning systems optimizing an objective function. In both cases, the metric might be a proxy measure for a complex composition of goals. By constructing or influencing this metric, principals shape the behavior of such agents to serve goals that the principal cannot otherwise articulate[3].

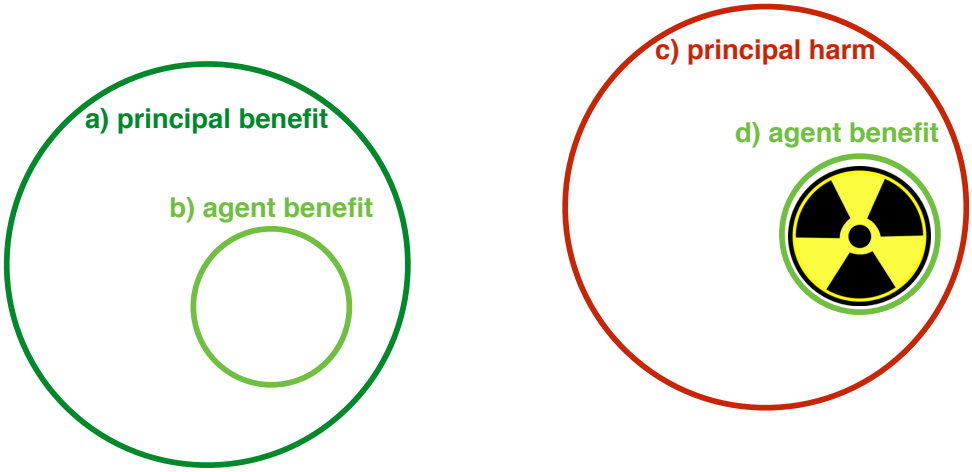

Figure 1 Incentive landscape.

Static Inspection Some agents are internally constructed to be unable to attempt certain behaviors. A business keeps *front-office* and *back-office* separate so that it cannot misbehave in certain ways without a conspiracy of two[2]. Within a statically type-safe language, a principal knows that an agent cannot violate its own interface type. Static verification of complex behavioral constraints is already practical and rapidly improving[7].

Dynamic Monitoring and Feedback The techniques above apply before the agent starts to act. Afterwards, the principal may look at what the agent is actually doing or has done. Depending on its judgement, the principal may change how it continues to employ the agent. The principal may send the judgement as feedback so the agent can learn or be fixed. It may share this feedback with other potential principals.

Trust and Reputation Even with all the safeguards, it can still be risky for the principal to rely on the agent's good behavior. Principals attempt to reduce this risk by putting their trust in the identity of the agent or the agent's origin. Companies trust their own developers. Consumers trust known brands. Feedback from past performance affects reputation, impacting future decisions to take such risks.

While each category provides useful lessons, the real payoff comes from how they interact and reinforce one another. For example, the co-design of an agent's rights and incentives can lead to better results than focusing on each individually.

Figure 1 illustrates a classic principal-agent dilemma. The space represents possible agent behaviors. Circle (a) contains agent behaviors that would benefit the principal. Circle (b) contains agent behaviors that the principal knows how to ask for and reward, benefiting the agent as well. Circle (c) contains agent behaviors that would harm the principal. Circle (d) is the region of greatest hazard, where the principal's harm coincides with the agent's benefit. Often, an incentive-only approach cannot eliminate this area at reasonable cost.

Figure 2 illustrates how practicing *POLA*, the *principle of least authority*[3], can improve the situation. POLA says that, ideally, a principal should grant the agent the narrowest rights that the agent needs to fulfill the principal's request.

[2] Barings learned this the hard way when it gave Nick Leeson both front-office and back-office privileges.
[3] The principle of least authority refines *the principle of least privilege*[9] to clarify what kinds of rights we need to minimize[6].

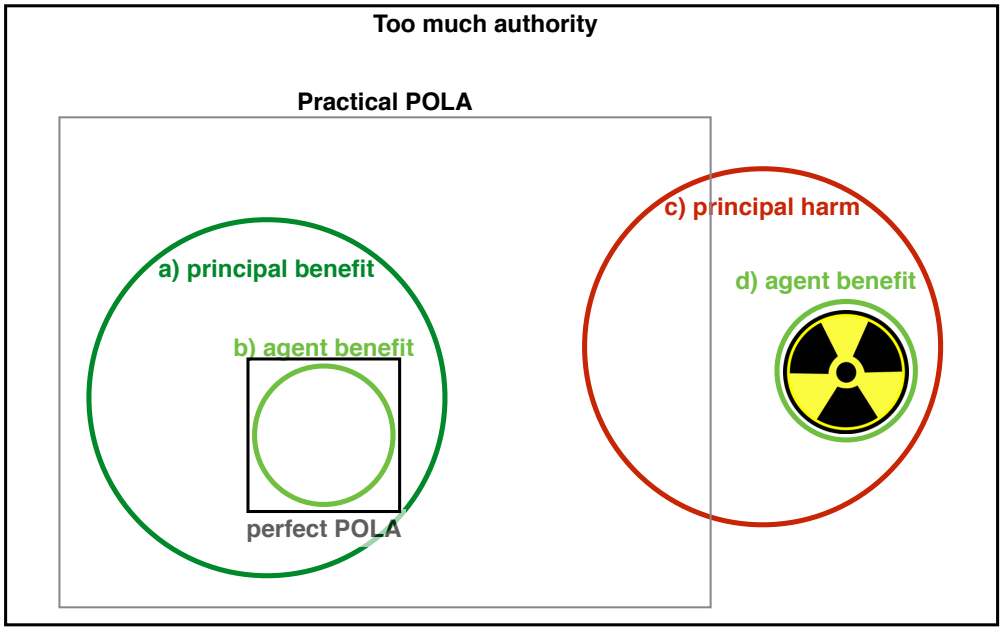

Figure 2 Co-design of incentives and rights.

The outer "too much authority" box represents the rights given to the agent in common practice, such as when programs and imported libraries run with the full privileges of their user. By taking "least" literally, the POLA alternative is often misunderstood as requiring the tiny box labeled "perfect POLA" tightly surrounding circle (b). This requirement is correctly dismissed as too hard to achieve in practice. The perfect is the enemy of the practical.

The box labeled "practical POLA" is large enough to achieve with reasonable effort. It does include agent behaviors that would harm the principal. However, it excludes the region of greatest hazard, where principal harm coincides with agent benefit. When thinking about incentives by themselves, we miss the option to avoid this region by restricting the agent's rights. When thinking about computer security by itself, we miss the differences among agent actions that cause harm to the principal. By thinking about both together, we can co-design principal-agent arrangements in which each technique fills in for weaknesses in the other.

Other surprising opportunities are found examining combinations of the other techniques. By reasoning across both rows and columns, we can help achieve a more cooperative world among diverse human and software, composed together in ever denser networks of principal-agent relationships.

Acknowledgements. We thank Terry A. Stanley and K. Eric Drexler.

References

1 Kathleen M. Eisenhardt. Agency theory: An assessment and review. *The Academy of Management Review*, 14(1):57–74, 1989. URL: `http://www.jstor.org/stable/258191`.

2 Friedrich A. Hayek. The use of knowledge in society. *The American economic review*, pages 519–530, 1945.

3 Friedrich A. Hayek. Competition as a discovery procedure. *New Studies in Philosophy, Politics, Economics and the History of Ideas*, 1978.

4 Laffont Jean-Jacques and David Martimort. The theory of incentives. the principal–agent model. *Princeton, NJ: Princeton Univ*, 2001.

5 Ludwig M. Lachmann. *Capital and its Structure*. Ludwig von Mises Institute, 1956.

6 Mark Samuel Miller. *Robust Composition: Towards a Unified Approach to Access Control and Concurrency Control*. PhD thesis, Johns Hopkins University, Baltimore, Maryland, USA, May 2006.

7 Toby S. Murray, Daniel Matichuk, Matthew Brassil, Peter Gammie, Timothy Bourke, Sean Seefried, Carmen Lewis, Xin Gao, and Gary Klein. sel4: from general purpose to a proof of information flow enforcement. In *Security and Privacy (SP), 2013 IEEE Symposium on*, pages 415–429. IEEE, 2013.

8 David L. Parnas. On the criteria to be used in decomposing systems into modules. *Communications of the ACM*, 15(12):1053–1058, 1972.

9 Jerome H. Saltzer and Michael D. Schroeder. The protection of information in computer systems. *Proceedings of the IEEE*, 63(9):1278–1308, 1975.

10 Bill Tulloh and Mark S. Miller. Institutions as abstraction boundaries. *Social Learning: Essays in Honor of Don Lavoie*, 2002.