

## **Remixing Contracts**

This paper describes my initial research into the economic and social relationships that form the underlying structure of our highly connected online society. These relationships are built on expectations and agreements that I will refer to as *contracts*. Contracts are the links that bind us together, forming a medium of trust through which we channel individual ideas into community, and individual actions into collectivity.

Formalizing, generalizing, and experimenting with our approach to these relationships is key to understanding them. My explorations have led me to preliminary designs for an open-source contract programming system that I believe will empower us to collaborate and participate in designing, understanding, and controlling the many contracts—implicit and explicit—that shape how we live, socialize and work. The goal of this paper is twofold: to present for the first time my vision for such a networked contract system, and to review some theories that may offer insight into this system and indicate possible future directions.

### **A Practical Contract Theory**

In describing this system I will use a somewhat simple but easily applicable definition of contracts. Later in this paper I will point to the origins of this definition within traditional social and judicial contract theory, but for this initial task of designing a system a more straightforward definition is needed.

In this practical definition a contract is any relationship with predefined expectations and agreements. By relationship I mean interactions between any two or more agents—individuals, groups, states, or even machines. The interactions, whether social or business, may range from single events to ongoing connections. Predefined expectations held by one or more of the involved entities requires some level of trust, such that one or both parties must trust the other to fulfill these expectations. The expectations may be predefined explicitly, or may be assumed.

Given this definition, we can find examples of contracts all around us, from traditional contracted institutions such as investments, subscriptions and licenses, to everyday informal contracts like cab rides, tip jars, even friendships. Each of these relationships depends on trust between one or both parties to meet the predefined expectations that make the relationship harmonious. Actions that break out of the expectations breach the contract. Since these expectations are often not explicit, social context is extremely important, and will be an integral

part of the contract system design.

## Contracts Remixed

In designing this contract system there are two critical pieces of the process to consider: language and enforcement. This design decouples contracts from the traditional language of law and the traditional enforcement by the government's judicial system. In their place I am respectively substituting code and community. For the language of contracts in this system, instead of the language of law we will use code, which I argue is preferable for a variety of reasons, including its aesthetic of clarity, its accessibility, and its modularity. And in lieu of government enforcement, for this system enforcement of contracts will fall to communities.

Fundamentally the contract language and the encompassing system is intended to be accessible and useful as a generalized platform. The programming language will be a standard, modern functional language such as Ruby, augmented with a few key libraries to abstract complexities like commitment, validation, and duration.

The following pseudocode is a sample of a small contract between a client and a designer. Take a moment to read through the code; even non-programmers will hopefully glean some notion of the intent.

```
def my_contract(client, designer, document, instructions, fee)
  unless commit(client, designer, self)
    contract_failed 'Not all parties agreed to the contract.'
  end
  account_transfer :from => client.account,
                  :to => self.escrow,
                  :amount => fee
  designer_chances = 3
  while designer_chances > 0
    send :to => designer,
         :attachment => document,
         :content => instructions
    deliverables = nil
    while deliverables.nil?
      if time.now > deadline
        account_transfer :from => self.escrow,
                        :to => client.account,
                        :amount => fee
        contract_failed 'Designer didn't meet the deadline'
      end
      deliverables = receive_deliverables(designer)
    end
  end
  unless verify(client, deliverables)
    designer_chances -= 1
  else
    send :to => client,
         :attachment => deliverables
    account_transfer :from => self.escrow,
                    :to => designer.account,
                    :amount => fee
  end
end
```

```

        return
    end
end
# If we get here then it didn't happen.
account_transfer :from => self.escrow,
                 :to => client.account,
                 :amount => fee
contract_failed "Designer's #{designer_chances} chances all used."
end

```

This code first asks the participants to commit to the contract (potentially via email or a web site). Part of this commitment process would include actually showing each involved party this very contract code. It then transfers the money to escrow and gives the designer three chances to get it right. If the client verifies one of the attempts, then the final product is forwarded to the client and the fee is transferred out of escrow and to the designer. If the due date passes, or if the three chances are used up with out success, the fee is transferred back to the client and the contract is marked as a failure.

Note that this sample contract code acts as a standalone contract, but could also be wrapped into a larger one. Suppose a design collective contracts with a firm, and each designer in the collective is to submit a design, then this code could be used as a subcontract within the greater firm-collective contract. This is an example of the *modularity* gained by using computer code. Large, sophisticated contracts can be constructed from preexisting small, unit-tested components. Modularity and generality fit with Virginia Postrel's dynamist assertions that “underlying rules apply to simple, generic units and allow them to combine in many different ways.”<sup>1</sup> This generality, in combination with emphases on open source code and documentation, makes contract creation, like open source software, a social/community process.

The other key design factor to consider is enforcement. The program may run, and the contract may succeed or fail, but how should we deal with failed contracts? Certainly there should be ramifications, an incentive to encourage people to create and participate in fair contracts.

For this system, rather than using government to enforce contracts, the enforcement will instead be community-based, with breaking of a contract resulting in blemish in the reputation of each participant. This is a particularly dynamist notion, falling under Virginia Postrel's description that dynamist systems “allow individuals (including groups of individuals) to act on their own knowledge.”<sup>2</sup> The argument is that a community knows what it needs, and is on its own responsive and well suited to create appropriate systems. The communities that would use these contract systems would decide if and how to enforce them in a manner most tuned for their

1 Virginia Postrel, “The Bonds of Life,” *Reason*, 1999. <http://www.reason.com/9901/fe.vp.thebonds.shtml>  
 2 Virginia Postrel, “The Bonds of Life,” *Reason*, 1999. <http://www.reason.com/9901/fe.vp.thebonds.shtml>

needs. Note that this is not a proposal to replace the existing judicial system, but simply a design for a new, non-government system.

### **Social Contract Origins**

Within the Media Lab I am a research assistant in the Physical Language Workshop (PLW). The PLW descends from the Aesthetics and Computation Group and the Visible Language Workshop, two groups that worked on designing interactive graphic and visual information systems.

The aim of the PLW is to outwardly expand these creative design approaches, to transition creativity in technology and design into innovations in life and society. With the Internet now such an integral part of our daily lives, a primary thread in our research is in leveraging the power of this interconnected, networked world. Among a number of other small projects and studies, our primary group project is OPENSTUDIO, a web-based online economy and community in which members can use free, simple tools to create artwork, and buy and sell it using a virtual currency.

OPENSTUDIO combines a number of modern web approaches in novel ways. From the beginning we designed the system to emphasize *openness of all member data*. Each economic transaction is logged and openly displayed, and these transactions are used to characterize business and creative relationships among members. Another feature is a tagging '*artsonomy*' system that allows users to attach words and phrases to pieces. Each phrase sticks to the art piece like small bits of semantic lint, describing not only taggers' views of the piece, but 'bubbling up' as collective reflections on the creator, owner, and tagger as well. We also added a *commissioning* feature that allows members to hire one another with specific instructions on what to draw. Commissioning is very simple: pick an available artist, offer a price, provide instructions, and set a due date.

The arguments for programmable social contracts stem in large part from this work in OPENSTUDIO. These relatively simple features integrate in interesting ways. Tagging provides a basic but effective feedback mechanism, allowing the community's judgement to reflect on transactions and commissions. Art pieces fetch a range of prices based not only on content of the piece, but also on tags on the piece and the creator.

The commission process is also extremely simple, and serves as an interesting instance of a contract between two people. Whereas all previous interactions within OPENSTUDIO were instantaneous, with the addition of commissioning we now had transactions that operated over time, the success of which required members to maintain trust in one another. It was the first

obvious instance of a social contract, and it was this limited feature that I was originally interested in generalizing. As it stands now, the commissioning in OPENSTUDIO represents only one possible instance of the enormously diverse set of possible contracts that we may actually want to allow. Rather than enumerating each possibility, I became curious as to how we could make the system flexible enough to accommodate all possible agreements, promises and relationships, even those that we as designers could not foresee.

Having for a long time been interested in the expressivity and precision of both human and machine languages, I was at the time also studying symbolic programming and the use of Domain Specific Languages (DSLs). DSLs are typically add-ons to existing programming languages, making programming for some certain specific domain more natural, and abstracting some of the underlying complexity of the domain. I realized that much of the language of contracts could be captured as a DSL, allowing integration of contract-specific functionality into general programming.

This project, and subsequently this paper, are direct results of applying these language design approaches to the economic models within the PLW's work. The intent of the project is in line with the intent of OPENSTUDIO as “a declaration of the new post digital ideals: transparency, community, and cooperation, yet grounded in economic reality.”<sup>3</sup> The aim of this contract system is to extend this grounding in reality by adding economic and social sophistication and expressiveness, while simultaneously furthering the fluidity and community of the open system.

### **Industry & Real World Contexts**

There are many existing systems in which implicit social and business contracts play an important role. By *implicit* I mean that the rules of the contract are embedded within the code for these systems, and while perhaps described by a legal license or within company documents, the actual logic and flow of the contract is conflated with the logic and control of the interface. Note how many of these systems appear quite dissimilar in all aspects from user community to intended function. Yet beneath these differences, each depends on an implicit built-in contract mechanism.

Online auction models are perhaps some of the most sophisticated contractual systems. Google's AdSense and Ebay are two prime examples. AdSense uses two general types of contracts, one for affiliate publishers and one for advertisers. Google acts as a clearinghouse for these auctions. Ebay's contracts are more social and extremely dynamic, with sellers initiating

---

<sup>3</sup> The Physical Language Workshop, *OPENSTUDIO: An Experiment in Creativity, Community & Capitalism*. [http://plw.media.mit.edu/openstudio/openstudio\\_book\\_sm.pdf](http://plw.media.mit.edu/openstudio/openstudio_book_sm.pdf)

the contract, and each bidder as an additional participant. Ebay also uses a reputation-based enforcement that works very well, and is the inspiration for many recent second generation p2p sites. This reputation-based economy is also fundamental to the enforcement of my proposed contract system.

More extreme examples of p2p systems are on the horizon, and these too make heavy use of implicit social contracts and community-driven enforcement. SwapTree is a startup that assists in finding people to freely swap with one another. Acting as a central dispatcher for such swaps, the contracts required for each transaction are quite static, simply requiring confirmation and mailing. Two new p2p finance companies, Prosper.com and Zopa.com, are offering services that allow members to borrow and lend money from one another. As with Ebay and my own proposed system, enforcement comes from the community itself, such that defaulting on a loan reflects poorly on a member's credit rating. Prosper takes the idea a step further by allowing users to form groups and alliances, such that the group itself has a cumulative credit rating that reflects on its members.

It is no coincidence that these are the sorts of examples used by Chris Anderson to support his Long Tail economic theory. These are companies with business models that depend on the massive connectivity of the internet to exploit and market niche content and products. Within these web and media industries we are seeing massive transition to many-to-many systems that depend on implicit contracts as the bond between participants.

Beyond these companies, there are a number of collectives and communities in which contracts play a vital role. The Deck is a “premiere” blogger advertising network that only accepts ads from select companies relevant to The Deck's audience, predominantly web developers and business tech nerds. Their criteria make for an interesting informal contract, with their advertisers but also with their readers, who are expected to trust the Deck to select appropriate advertisers. From their online materials: “We’re picky about the advertising we’ll accept. We won’t take an ad unless we have paid for and/or used the product or service. Sell us something relevant to our audience and we’ll sell you an ad.”<sup>4</sup>

SecondLife is a prime example of a community in which a new form of social and economic contracts are of huge import. Consider denizens such as Anshe Chung, who runs a virtual real estate company within the SecondLife metaverse.<sup>5</sup> Chung has built a business that earns over \$150,000 USD per year by investing in virtual pieces of land. The many deals she engineers are all forms of socio-economic contracts. Her business is wholly dependent on the community of

---

4 The Deck: <http://www.coudal.com/deck/>

5 Robert D. Hof, “My Virtual Life,” *BusinessWeek*, May 1, 2006  
[http://www.businessweek.com/magazine/content/06\\_18/b3982001.htm](http://www.businessweek.com/magazine/content/06_18/b3982001.htm)

SecondLife residents maintaining a shared belief that these contracts—and all the other rules within the virtual world—are valid. Her transactions and contracts are not enforced by any governmental judicial system, but are implicitly enforced by a combination of the code that runs SecondLife and the community that inhabits it. This is just one example of thousands of small contract relationships within the SecondLife world. These economic and contract models are rapidly evolving into more complex and sophisticated forms as the community grows and changes.

## **Two Traditions of Contract Theory**

A huge body of contract theory already exists. These theories fit within roughly two categories: *social contract theory* (or *contractarianism*) and *contract theory*.

*Social contract theory* is the study of implicit social contracts between and among states, groups and individuals. The approach is based in the political-philosophical tradition of liberalism, defined in large part by the work of philosophers such as Locke, Hobbes, and Rousseau<sup>6</sup>. This work is relevant to my proposed contract-as-code design because it treats human social systems as sets of implicit agreements. These agreements underlie legal, social and cultural systems.

In contrast, *contract theory*, is concerned with the judicial and economic theories of contracts in a legal, government, and market settings. Though the system I am advocating is expressly intended to be disassociated from law and government, judicial contract theory does address issues which are relevant to such a system, in particular questions of contract enforcement, defaulting and completion of contracts, and economic incentives for contracts.<sup>7</sup> As mentioned previously, for this programmable social contract system I seek to decouple the enforcement from the government judicial system, using instead social capital (community standing, reputation, and perception) to provide enforcement and incentives.

Each of these theories of contracts are huge, mature, and well delineated areas of study. I intend to simply dip into each when needed, using pieces of each to inform a patchwork, practical theory of networked contract systems.

## **Participation & Beyond**

In his 1999 book *Code and Other Laws of Cyberspace*, Lawrence Lessig touches on issues very relevant to this discussion. He is primarily concerned with law and regulation of the internet—an area which this project in social contracts aims to avoid. However, Lessig presciently

<sup>6</sup> Wikipedia entry on Social Contract: <http://en.wikipedia.org/wiki/Contractarianism>

<sup>7</sup> Wikipedia entry on Contract Theory: [http://en.wikipedia.org/wiki/Contract\\_theory](http://en.wikipedia.org/wiki/Contract_theory)

emphasized the importance of open source code in the democratic formation of the internet:

In a way that the American founders would have instinctively understood, “free software” or “open source software”—or “open code,” ... —is itself a check on arbitrary power. A structural guarantee of constitutionalized liberty, it functions as a type of separation of powers in the American constitutional tradition. It stands alongside substantive protections, like freedom of speech or of the press, but its stand is more fundamental. ...the first intuition of our founders was right: structure builds substance. Guarantee the structure (a space in cyberspace for open code), and (much of) the substance will take care of itself.<sup>8</sup>

This initial structure of a system, Lessig argues, is the foundation that later determines the character of the substance of the system. Open source embodies values of transparency and community, and so a system that is fundamentally open source inherently fosters democratic and participatory use. Tim O'Reilly touched on these ideas as well in his definition of “The Architecture of Participation”:

I've come to use the term "the architecture of participation" to describe the nature of systems that are designed for user contribution. Larry Lessig's book, *Code and Other Laws of Cyberspace* ... made the case that we need to pay attention to the architecture of systems if we want to understand their effects.<sup>9</sup>

Both O'Reilly and Lessig's analysis can be aptly applied to this programmable contract system. By emphasizing open source, modularity, and transparency in its contract processes, this project will hopefully contribute to this architecture, making possible the participatory open-style culture they describe.

Yet, as Lessig has pointed out, structure is only the starting point, the foundation on which this cultural substance is built. The most exciting prospect of programmable contracts is their potential for creating as-yet unforeseen new ways of living and working, of forming new collectives and collaborations. These possibilities stem not only from the structure of open source, modular code, but also from the open social systems that will accept the system and enforce these contracts. Cultural theorist Grant McCracken describes the process by which game worlds have incubated and fostered innovative new social behavior:

...we are looking at a virtual world that resembles the most dynamic thing in our real world: the economy. It is precisely when individuals engage in unmediated, undeliberated behaviors, that extraordinary social and cultural patterns begin to emerge. To use the once fashionable language of French structuralism, the economy comes not from "structure" (the shared, preconceived, conventionalized ideas) but from "event" (in this case, behaviors, their aggregations and concatenations).<sup>10</sup>

---

8 Lawrence Lessig, *Code and Other Laws of Cyberspace* Wiki Chapter 1, <http://codebook.jot.com/Book/Chapter1/Ch1Part3>

9 Tim O'Reilly, “The Architecture of Participation,” June 2004, [http://www.oreillynet.com/pub/a/oreilly/tim/articles/architecture\\_of\\_participation.html](http://www.oreillynet.com/pub/a/oreilly/tim/articles/architecture_of_participation.html)

10 Grant McCracken, “How Virtual Worlds Discovered Dynamism,” [http://www.cultureby.com/trilogy/2006/05/how\\_virtual\\_wor.html](http://www.cultureby.com/trilogy/2006/05/how_virtual_wor.html)

McCracken's events as “behaviors, their aggregations and concatenations” are Lessig's cultural substance that follows from the system structure. These behaviors are the atomic actions of individuals within an open and transparent “unmediated” society. It is this sort of society in which the contract system applies. It is a means of creating connections, sparking new collaborations and community processes through its openness, yet relying only on the enforcement of the community rather than the restrictions of a legal system.

Chris Anderson's forthcoming book *The Long Tail* offers a vision of the emerging niche economy in which the cheap, easy flow of information offers us all far greater choice in the media and products we consume, and as a result a much wider choice of cultural identifications that we may choose to adopt. Anderson theorizes that these choices will lead to a fundamental shift from mass culture to microculture:

The same Long Tail forces and technologies that are leading to an explosion of variety and abundant choice in the content we consume are also tending to lead us into tribal eddies. When mass culture breaks apart, it doesn't reform into a different mass. Instead, it turns into millions of microcultures, which co-exist and interact in a baffling array of ways.<sup>11</sup>

Within this coexistence and mass interaction, I envision this contract system as an open, generalized, universal tool that would enable these tribal eddies and microcultures to create their economies and societies in forms beyond that we are at this point capable of imagining. In dynamist fashion, the openness and generalizability of such a system not only enables but *encourages* these niche cultures and niche-oriented businesses to evolve correspondingly niche contracts that suit their needs.

## **Moving Forward**

In perhaps typical Media Lab fashion, my next step will be to stop writing and to start building. Though I can explain how it should all work, and I can support these ideas with some theories, until I actually try to build it I will not be fully convinced of its merit. This is the flip flop from design to theory and back to design. The first design was on paper, and the next will be in code.

The two fundamental features of this contract system—language as code and enforcement from community—can for now be treated separately. The first, designing the contract DSL, is much harder to implement, and depends on some real hard programming to figure out how to best abstract the durations that must pass for a contract system to work.

The second feature, community-based enforcement of contracts, does not depend on new

---

11 Chris Anderson. “Chapter 11: Niche Culture,” *The Long Tail* galley version.

technology so much as new ideas. This community system will be my first step. As a lightweight testbed for these ideas, I plan to build a small standalone promise service in which people can author and commit to simple promises within an open public, community setting. It will be extremely spare, with just enough functionality to test the idea. Interesting extensions like economics and folksonomies, as well as the contract DSL, I will plan to add in later. The creation of this first system, while small, will be an exciting development. We are rapidly learning the importance of designing technology that aids human relationships and fosters formation of communities. I hope that my contract approach can offer a general tool in the pursuit of this goal.