Meng Weng Wong, Helena Haapio, Sebastiaan Deckers & Sidhi Dhir: Computational Contract Collaboration and Construction. In: Erich Schweighofer et al. (Eds.) *Co-operation. Proceedings of the 18th International Legal Informatics Symposium IRIS 2015*. Österreichische Computer Gesellschaft OCG, Wien 2015, pp. 505–512 (ISBN 978-3-85403-309-7). Also published in Jusletter IT 26 February 2015, http://jusletter-it.weblaw.ch/en/issues/2015/IRIS/computational-contra_dc05f86bf7.html (subscription required)

COMPUTATIONAL CONTRACT COLLABORATION AND CONSTRUCTION

Meng Weng Wong[1], Helena Haapio[2], Sebastiaan Deckers[3], Sidhi Dhir[4]

[1]Social Engineer, JFDI.Asia
71 Ayer Rajah Crescent #05-16, 139951 Singapore, SG
mengwong@pobox.com; http://www.mengwong.com/
[2]Business Law Teacher & Postdoctoral Researcher, University of Vaasa / International Contract Counsel, Lexpert Ltd
Pohjoisranta 20, 00170 Helsinki, FI
Helena.Haapio@lexpert.com; http://www.lexpert.com
[3]Frontend Cofounder, Cofounders Pte Ltd, Singapore, SG
sebastiaan@cofounders.sg; http://www.cofounders.sg
[4]Executive Director, The Indus Entrepreneurs
71 Ayer Rajah Crescent #05-16, 139951 Singapore, SG
sidhidhir@gmail.com; https://singapore.tie.org

Keywords: ***automated contract generation toolkit, contract visualisation, "do-it-yourself" law, document assembly, computational law, term sheets, venture financing***

Abstract: *In recent years, technology investors have published templates for investment deals, with the aim of reducing the costs and delays of startup financings by standardising common agreements and summarising complex contracts into shorter term sheets. These practices allow experienced investors and entrepreneurs to efficiently negotiate high-level business terms, involving counsel only later to review the expanded agreements.*

In previous research, we have explored contract visualisation – the use of charts, images, and interactive interfaces to help non-lawyers quickly grasp contract essentials. This paper proposes a family of complementary approaches informed by computer science, which offers methods and practices relevant to, but presently little adopted in, legal drafting. Tools and techniques familiar to software engineers offer opportunities to simplify negotiation, automate document assembly, visualise scenarios, verify correctness, and transform the way contracts are shared and edited. We review recent efforts to marry computing with "do-it-yourself" law, including Internet repositories of model contracts, document assembly wizard workflows, and opensource-style sharing of templates. We analyse these efforts using a four-part framework of automation, visualisation, collaboration, and formalisation. We then present our proof-of-concept prototype of an automated contract generation toolkit. It includes a collaborative online interface, a compiler which expands term sheets into long forms, and a document assembler which circulates signature-ready contracts, all without human involvement. We extrapolate a future where computational techniques are ubiquitous in contracting and in law, as they already are in entertainment, telecommunications, and finance.

1. Introduction: Applying Four Computational Themes

In the world of finance, mistakes are dangerous. Tibco Software Inc.'s management, shareholders and advisers recently learned this the hard way in the software company's sale to Vista Equity

Partners, which is paying about $100 million less in the deal because of a misinterpretation of a financing term. [Tan 2014] While the outcome of the resulting lawsuits is still unclear, one thing is clear: mistakes do happen. In contract drafting, mistakes "enter the legal documentation often and spread like memes thru copy and paste". [Ward 2015] This paper argues that many such mistakes are preventable through intelligent application of concepts borrowed from the software world.

Other thinkers have asked: What if investment agreements were written in a kind of code similar to computer code; something that can be checked for validity, evaluated, perhaps even unit tested? What if terms were captured in *formulas* familiar to those in the field, rather than language that is difficult to verify? For someone experienced in financing, "generating a spreadsheet of vesting events would be more precise and less ambiguous than writing paragraphs"; in their view, it seems strange that *mathematical expressions*, such as "vesting schedules, liquidation preferences, conversion ratios, and acceleration provisions, are defined in prose". [Ward 2015]

Computational ideas are now sweeping through the legal industry. In the words of Marc Andreessen, a computer scientist and technology investor, "software is eating the world". [Andreessen 2011] The legal industry is no exception. This paper chooses the field of startup investing as a fertile ground to investigate the applications of four major computational themes – *automation, visualisation, collaboration,* and *formalisation* – to the legal field.

This paper posits that processes and techniques originating in the software construction field can be applied to the contracting and legal worlds. **Automation** can speed up the process of constructing contracts and circulating them for signature. **Visualisation** can help parties understand the meaning and implications of a contract through graphical representation and interactive scenario exploration. **Collaboration** at the level of a particular contract helps parties negotiate an agreement efficiently. More broadly, collaboration enables users to share and improve templates, both within an organisation and more publicly, in ways that programmers would find familiar. **Formalisation** lies at the heart of computational thinking. We argue that the same formal methods used by computer scientists and programmers to develop software for computers to execute, can be used to develop agreements for people to execute.

After introducing our context, we begin Section 2 by surveying existing legal applications of our four themes. We then review recent commercial and opensource efforts to apply computational tools to early stage investments. We discuss the potential for other kinds of formalisation, from a computer science perspective, with examples. In Section 3 we present our proof-of-concept prototype of an automated contract generation toolkit along with screen shots and show how our prototype takes the themes of this paper beyond the prior art. Section 4 concludes the paper.

1.1. Context: Early-Stage Fundraising

In the world of early-stage startups, investment agreements are widely recognised as a source of friction and delay. Whether based in Silicon Valley, Singapore, or Salzburg, a startup faces administrative formalities and a lot of paperwork. First-time entrepreneurs and angel investors lack the expertise, time, and money required to understand every detail of a deal under negotiation. But millions of Dollars and Euros may ride on clauses whose implications are not easily appreciated.

For legal advisors, a typical drafting scenario starts with documents from an earlier financing. Documents go back and forth like tennis balls at a doubles match. A change in one document requires all related documents to be changed as well. Annually, thousands of financing rounds are closed, each consuming considerable time and effort on the part of investors, management teams and lawyers.

Considerable time and money are saved when parties focus on the high level issues and trade-offs of the deal at hand. Term sheets, which summarise the essentials of long-form documents, are

already an industry standard practice [e.g., Feld & Mendelson 2013]; this paper explores possible next steps. Recent research has started to build evidence that complex contracts can be made clearer and easier to use and act upon through visualisation [Curtotti n.d.; Passera et al. 2014; Passera et al. 2013]. In this paper we explore the opportunities offered by computational tools to summarize and visualise the core of complex agreements, using online, "smart" templates, to enhance collaboration and automate contract generation.

2. Survey of Earlier Initiatives and Recent Efforts in the Field

2.1. Document Assembly: Automation for Lawyers

Many legal practitioners produce new contracts by copying-and-pasting old contracts. This process is fraught with error. Since the days of WordPerfect, document assembly solutions have answered the need for more structured, streamlined approaches to contract generation. HotDocs, the industry leader, was introduced in 1993 to help counsel gain efficiency through automation. These solutions are spiritual heirs of the simple mail-merge, grown to support all manner of textual contortion.

Taken to the logical extreme, automation removes the lawyer from active participation in the contract generation process, by embedding the skill and experience of the lawyer into the rules and content of the system. In this vein, some law firms offer customised agreements in a web app. Wilson Sonsini Goodrich & Rosati's Term Sheet Generator[1], for instance, produces (after a 43-page questionnaire) a term sheet and investment agreement. Cooley offers a similar wizard.[2]

2.2. Contract Visualisation: Term Inputs and Financial Outcomes

Previous work [e.g., Curtotti n.d; Passera et al. 2014; Haapio 2012 and 2013] has demonstrated the use of visualisation in the context of contracts. Visualisations can also be used for early stage financings. Securities such as common stock, convertible notes, and preferred shares can be represented as icons. Events such as conversion of debt to equity can also be represented visually, using tools that borrow as much from the visual vocabulary of videogames as from spreadsheets.

Capitalization tables (or cap tables) are widely used by entrepreneurs and investors to analyze founders' and investors' percentage of ownership and other important information which can become quite complex due to different stock classes, convertible notes, stock options, and so on. When cap tables are visualised, it becomes easier to keep track of such information and subsequent issuance of stock, options, and financing events.[3] Early stage investments hold the promise of outsize capital gains. Agreements are scrutinised by both founders and funders to understand profit potential under different scenarios. These, too, can be visualised for ease of understanding.[4]

Figure 1 illustrates a possible dashboard interface to negotiating a term sheet. As options are chosen, implications are visualised.

[1] https://www.wsgr.com/WSGR/Display.aspx?SectionName=practice/termsheet.htm
[2] http://www.cooleygo.com/documents/y-combinator-safe-financing-document-generator
[3] For an illustration, see "Captables made easy as pie" at https://captable.io/home.html. Captable.io offers a platform which can be used to create graphical representations of cap tables. A demonstration is available at https://captable.io/demo. According to the web page, "Captable.io keeps everyone on the same page, provides transparency and helps avoid costly mistakes."
[4] An illustration of how exit scenarios can be modeled and visualized is provided by Captable.io at https://www.captable.io where one of the images ("Understand outcomes") shows possible shareholder returns based on various exit valuations.

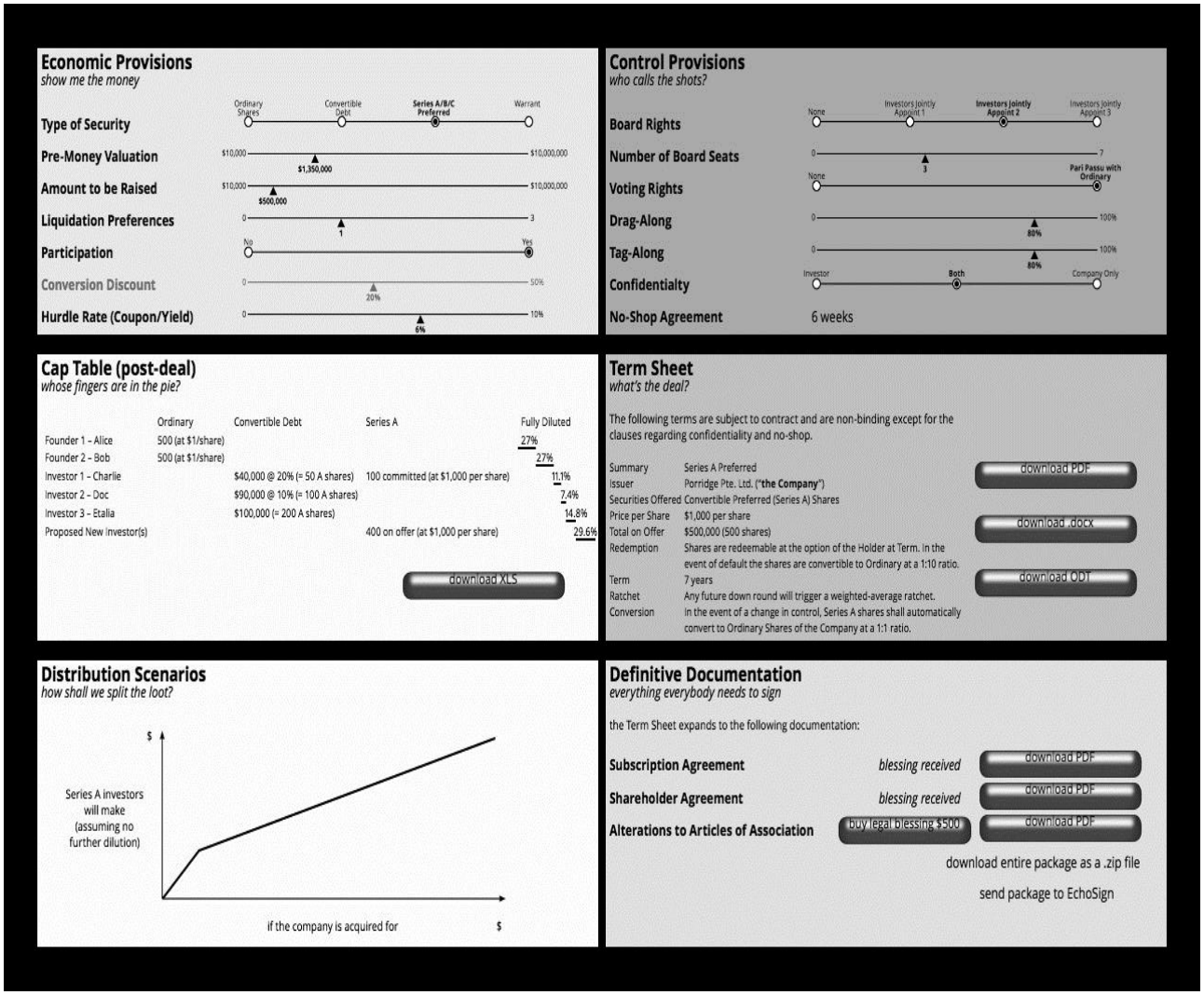

Figure 1: Draft Interface for Visual Termsheets

2.3. Contract Collaboration: Editing and Negotiation

Many contracts are negotiated by emailing Word documents between parties, with *track changes* turned on and redlines debated point by point. When will this change? "Born online" systems such as Google Docs include revision history features and offer the promise of realtime collaborative text editing between geographically dispersed parties, though social business practices have not yet evolved the rituals and customs needed for such technologies to go mainstream.

In an early-stage financing, valuation looms largest but other terms shape a deal in crucial ways. Novice entrepreneurs and investors often are willing to accept "industry standard" terms – but, being novices, they are ignorant of the prevailing standards! Guidance may be found in books [e.g., Wilmerding 2006], which offers different versions of common clauses: an investor-friendly version, an entrepreneur-friendly version, and a middle-of-the-road version. Contract visualisation can illustrate these alternatives vividly. In Figure 2, a Google Spreadsheet can help evaluate the entrepreneur-friendliness of a set of terms, giving each term a green light – or red.

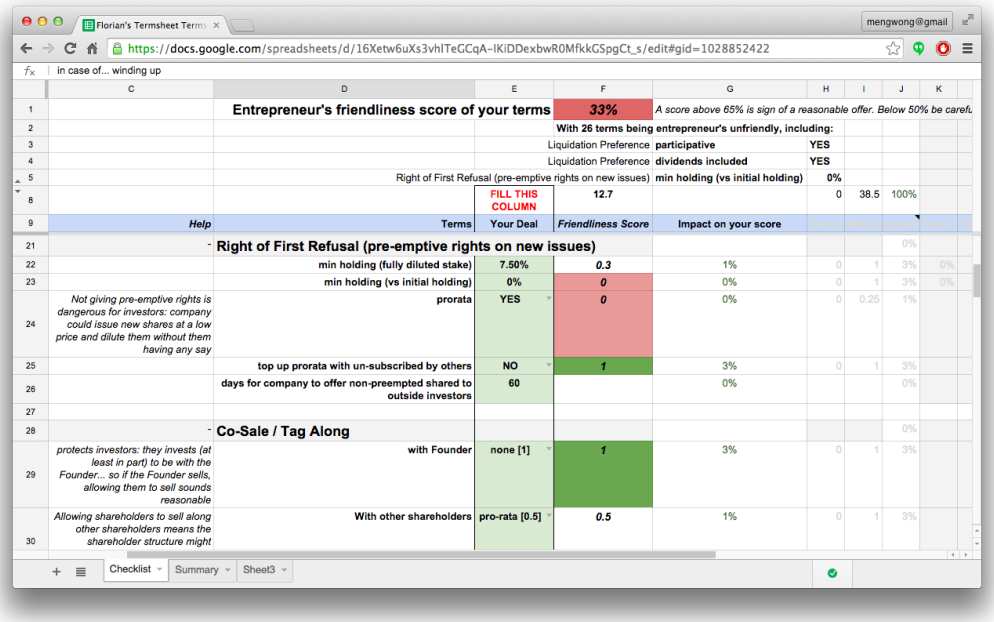

Figure 2: Visual Guide to Friendliness of Terms. Image reprinted with the permission of Florian Cornu.

2.4. Template Collaboration: Online Document and Clause Repositories

Software engineers package standard solutions to recurring problems into code libraries, for their fellows to incorporate into applications. In the field of contracts, online equivalents exist: for example, KM Standards (formerly KIIAC)[5] offers a library of standard clauses for public download.

In the opensource tradition of software, entire programs are made available, not just as compiled executable code, but as human-readable source code. Similarly, services like Lawdepot, Lawcanvas, Docracy, and Peppercorn[6] offer repositories of templates and "other people's contracts". However, these repositories tend to offer little guidance as to fitness for purpose: *caveat emptor* prevails.

Software programs are often managed in version control systems. Versioning has long been a part of legal drafting, but house styles seldom go beyond putting a date-stamp in the filename. Programmers by contrast have for decades trained on sophisticated systems like RCS, CVS, SVN, and, most recently, Git, which offer tools and social practices to report bugs, release new versions, and share code – all, crucially, without the need for the original author's permission or cooperation. Inspired by those practices, one of the authors, Sebastiaan Deckers, published Legal.cf.sg and initiated the Cofounders GitHub repository, the first ever GitHub-hosted set of legal templates.[7]

2.5. Publicly Available Term Sheets and Investment Agreements

In recent years, technology investors and their Associations have published templates for investment agreements. In 2006, the U.S. National Venture Capital Association kicked off this movement with its model venture capital financing documents[8], followed by Y Combinator with its Series AA documents (prepared with WSGR).[9] In 2009, TechStars open sourced their model

[5] http://www.contractstandards.com/
[6] http://www.lawdepot.com/; https://lawcanvas.com/; http://www.docracy.com/; http://www.peppercorn.it/en
[7] http://legal.cf.sg/; https://github.com/cofounders/legal
[8] http://www.nvca.org/index.php?option=com_content&view=article&id=108&Itemid=136
[9] https://www.ycombinator.com/documents/

documents (prepared with Cooley).[10] In 2010, SeriesSeed.com appeared (by Fenwick & West.).[11] Soon, the field was so crowded that commentators started publishing comparative analyses of the documents. As the saying goes: "A man with a watch knows what time it is. A man with two watches is never sure."

2.6. The Promise of Formalisation

Contracts and programs have much in common. Both are collections of statements that specify desired and prohibited behaviours. Both attempt to express assertions in precise language. Both can be written with clarity or with a confusion of cross-references. Of course, law predates programming; but how might contracts be different if programming had been invented before law?

Surden (2012) describes *Computable Contracts* and their relevance to computerised finance. His description emphasise machine-readable *data* elements in computerised contracts. Later research has developed these ideas further. [e.g., Flood & Goodenough 2014]. We assert that ultimately, computer science promises to do for law what mathematics does for physics. Whereas NLP (natural language processing) methods attempt to parse existing legal texts for semantic structure, "strong-form" formalisation insists that contracts be constructed from the start in a formal language – a language of a higher Platonic order and greater mathematical clarity than English, German, or any other natural language. Programming languages like ML and JavaScript are formal.[12] Compare the English-language version "the principal paid by the Investor shall be no less than $100,000" with the machine-readable expression `(investor.principal.paid >= 100000)`. Computers can do much more with the latter: they can visualise it, they can test it for validity, they can run simulations to explore scenarios of breach. Indeed, they can turn the latter into the former with complete confidence; crucially, the reverse is not true.

In computing, programs in a high-level formal language like JavaScript are *compiled:* parsed into an intermediate syntax tree, then transformed to an output language. Usually, that output language is native machine code suitable for an Intel, AMD, Motorola, or other processor. But the output language can also, with some work, be a natural language – English, German, French. Given a formal-language specification, a "contract compiler" could conceivably produce legalese in French, English, and German – all three of which are *provably* identical (to the extent it is ever possible).

2.7. Combinations of Themes in Recent Commercial Initiatives

Commercial products often combine the above themes. For example, Shake[13] offers a combination of contract templates, customisation, and execution on mobile. In the early stage investment arena, three startups have recently emerged to streamline investment agreements and support the parties involved in those financings. Termsheet.io implements document automation as a service, offering a limited number of templates for the seed-funding use case. Captable.io helps a company record and represent its register of shareholders, noteholders, and other stakeholders. eShares[14] approaches the same problem from an additional perspective – that of an employee managing a portfolio of option and equity grants accrued over a career at multiple companies.

A number of interesting recent innovations illustrate other ways to apply computational techniques to contracting and law, for example in the context of legal and contract analytics. Companies like

[10] http://www.techstars.com/docs/
[11] http://www.seriesseed.com/
[12] Legal XML is a markup for natural language, but is not in itself a formal, Turing-complete language.
[13] http://www.shakelaw.com/
[14] https://www.esharesinc.com/

Seal Software[15] and eBrevia[16] apply Natural Language Processing and Machine Learning to automatically read and understand all the agreements to which a business is party. Lex Machina[17] applies NLP, ML, case-based reasoning, and Big Data Analytics to read case law, extract legal reasoning, and identify relevant precedents for litigation, particularly IP litigation. These, however, are not directly aligned with the themes of this paper.

3. Our Prototype of an Automated Contract Generation Toolkit: Legalese.io

Three of the authors are active participants in the startup community of Singapore. As investors and entrepreneurs, they have been repeatedly involved in term sheets involving dozens of investors. They report that it can take months to negotiate financial terms, choose a specific form of agreement, get the term sheets finalized, collect investor particulars, generate long-form agreements for review, correct errors in cross-references and elsewhere, and collect signatures on paper.

Our prototype addresses these problems, shortening the workflow from months to days. Such efficiencies are well known to other sectors: Salesforce.com, for example, boasts that with Adobe EchoSign integration, salespeople can issue contracts for signature in minutes.[18]

How is our prototype different from the prior art surveyed above? To help the user choose from a variety of templates, the system visualises the differences between templates both statically and in an interactive model. Having imported all publicly available templates, it fills in every template that fits the deal at hand. The interface is not a wizard but a spreadsheet, preferred by power users. That spreadsheet lives on Google Docs, a collaborative web application. The spreadsheet also visualises the investor vs founder-friendliness of the terms. The system is itself opensource.[19] This allows users to fork the project, create private templates, correct the main public templates, and submit new templates for public use, all without the involvement of the initial project owner. A single master can be inflected for different legal jurisdictions and slightly different variants. The deal terms spreadsheet can be shared between investor and startup, and changes can reflect negotiation in real time, until both sides are satisfied. Once the terms are agreed, a few clicks produce and circulate PDFs for electronic signature. Figure 3 shows screenshots of the web interface and the source XML. The resulting signature-ready PDF is excluded due to lack of space.[20]

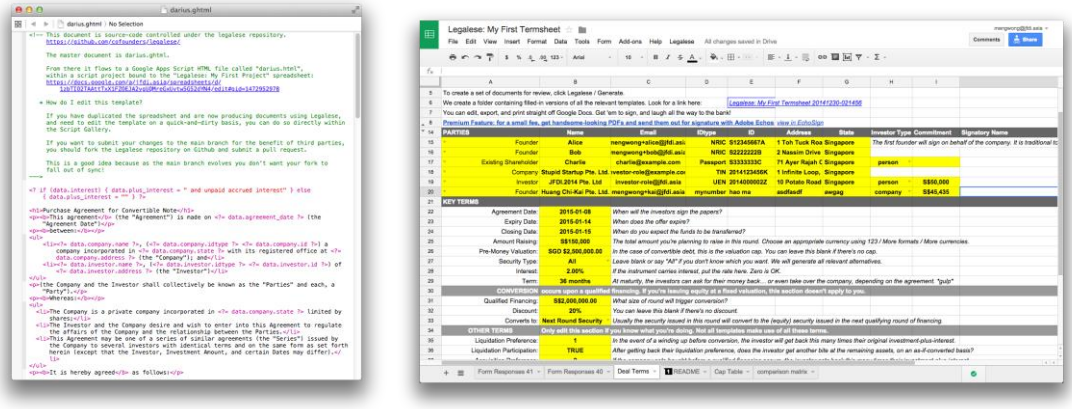

Figure 3: Screenshots of the Web Interface and the source XML of the Automated Contract Generation Toolkit

[15] http://www.seal-software.com
[16] http://ebrevia.com/
[17] https://lexmachina.com/
[18] https://www.echosign.adobe.com/en/integrations/echosign-salesforce.html
[19] Both software and templates are available on GitHub at https://github.com/cofounders/legal/.
[20] These images, and the resulting signature-ready PDF, are available at http://legalese.io/demo/.

This system embodies the themes of automation, visualisation, and collaboration. What about formalisation? At present, the document templates are authored in XML format by humans. A future version of the system will include the JavaScript-to-natural-language compiler. At that time, the authoritative master will move upstream to the formal JavaScript version of the contract, which will be compiled to one or more natural languages. Formal verification can prove contract completeness, and testing methods from unit testing to fuzzing can evaluate the robustness of a contract. Post-execution, a "smart contract" can go into active mode: as others have envisioned [Szabo 1994], it can evaluate itself for compliance and it can trigger business events.

4. Conclusion

In this paper we have applied processes and techniques originating in the software construction field to the contracting and legal worlds. Merging tools and practices familiar to software engineers but presently little adopted in the legal field with term sheets, we have explored opportunities to simplify and automate document assembly and transform the way contracts are generated. In addition to the idea of an automated contract generation toolkit, we have also provided a proof-of-concept prototype. This prototype combines our four computational themes of automation, visualisation, collaboration, and formalisation in a piece of opensource, online software, ready for public use by lawyers and non-lawyers alike. It is our hope that the friction of financings will be greatly reduced, making life easier for entrepreneurs, investors, and their advisers. While our initial toolkit is specialized for venture financings, it can be extended to other contract areas.

5. References

Andreessen, Mark, Why Software Is Eating The World. The Wall Street Journal, August 20, 2011, http://www.wsj.com/articles/SB10001424053111903480904576512250915629460 [Accessed January 5, 2015] (2011).

Curtotti, Michael, Computational Tools for Reading and Writing Law http://cs.anu.edu.au/people/Michael.Curtotti/ and Contract Tools at http://buttle.anu.edu.au/contracts/ [Accessed January 5, 2015] (n.d.).

Feld, Brad & Mendelson, Jason, Venture Deals. Be Smarter than Your Lawyer and Venture Capitalist. Wiley (2013).

Flood, Mark D. & Goodenough, Oliver R., Contract as Automaton: The Computational Representation of Financial Agreements, http://papers.ssrn.com/sol3/papers.cfm?abstract_id=2538224 [Accessed January 5, 2015] (2014).

Haapio, Helena, Making Contracts Work for Clients: towards Greater Clarity and Usability. In E. Schweighofer et al. (eds): Transformation of Legal Languages. Proceedings of the 15th International Legal Informatics Symposium IRIS 2012. Band 288. Österreichische Computer Gesellschaft, Wien, pp. 389–396 (2012).

Haapio, Helena, Next Generation Contracts. A Paradigm Shift. Doctoral dissertation. Lexpert Ltd, Helsinki (2013).

Passera, Stefania, Haapio, Helena & Barton, Thomas, Innovating Contract Practices: Merging Contract Design with Information Design. Proceedings of the IACCM Academic Forum on Contract and Commercial Management 2013, Phoenix, pp. 29–51 (2013).

Passera, Stefania, Haapio, Helena & Curtotti, Michael, Making the Meaning of Contracts Visible – Automating Contract Visualization. In E. Schweighofer et al. (eds): Transparency. Proceedings of the 17th International Legal Informatics Symposium IRIS 2014. Band 302. Österreichische Computer Gesellschaft OCG, Wien, pp. 443–450 (2014).

Surden, Harry, Computable contracts. *UC Davis Law Review,* Vol. 46, pp. 629–700. http://lawreview.law.ucdavis.edu/issues/46/2/Articles/46-2_Surden.pdf [Accessed January 5, 2015] (2012).

Szabo, Nick, Smart Contracts, http://szabo.best.vwh.net/smart.contracts.html [Accessed January 5, 2015] (1994).

Tan, Gillian, Spreadsheet Mistake Costs Tibco Shareholders $100 Million. The Wall Street Journal, October 16, 2014, http://blogs.wsj.com/moneybeat/2014/10/16/spreadsheet-mistake-costs-tibco-shareholders-100-million/ [Accessed January 5, 2015] (2014).

Ward, Henry, Broken cap tables, January 2, 2015 https://medium.com/@henrysward/broken-cap-tables-bbf84574a76a [Accessed January 5, 2015] (2015).

Wilmerding, Alex, Term Sheets & Valuations – A Line by Line Look at the Intricacies of Term Sheets & Valuations. Aspatore Books (2006).