Rule Governance, Social Coding and Social Modeling

Joost de Vries[1], Stijn Hoppenbrouwers[2]

[1] Everest b.v., j.de.vries@everest.nl
[2] HAN University of Applied Sciences, stijn.hoppenbrouwers@han.nl

Abstract. The current financial climate in the world forces organisations in government and finance to automate their operational decision making to the highest degree. The Dutch government is initiating an approach that facilitates quick, repeatable and correct implementation of new laws and thorough accountability of operational decisions that have been taken. The name of this approach can be translated as 'rule governance' or 'agile execution of law'. This article proposes the term social modeling and argues that a solution to facilitate rule governance modeling would benefit from being based on social modeling.

Keywords: Rule Governance, Enterprise Modeling; Accountability.

1 Background

1.1 Developments in Dutch government

In the Netherlands, where the authors are situated, the government has stated the vision that by 2017 citizens can completely *digitally interact* with the government [1]. A typical example of this interaction would be the process where a citizen requests a permit using the self-service internet channel. Two major goals of this vision are higher quality of governmental service and higher efficiency.

Examples of parts of government that this affects are the agency responsible for taxes, the agency responsible for immigration and the agency responsible for employment matters. These agencies have in common that their business processes need to implement the often complex, detailed and changing obligations as stipulated by the law.

The Dutch tax agency has started an initiative that can be translated as ´agile execution of law´[1] to be able to implement their execution of tax law in a more timely, efficient and accountable manner.

[1] ´Wendbare Wetsuitoefening´ in Dutch

1.2 Developments in the financial sector

The financial sector is another part of society where core business processes largely concern the processing of information. The financial crisis that started in 2008 has led to an increase in governmental regulation of the financial sector. At the same time there banks and insurance firms are increasingly feeling the pressure to interact with their customers digitally and reduce the role of local offices.

This article focuses on the government but the reasoning can be similarly applied to the financial sector.

1.3 Developments in distributed collaboration

In 2011 Marc Andreessen argued in the Wall Street Journal that *information systems are replacing physical business processes* to an uncommon degree [2]. His venture capital investment firm put their money on this global trend by investing 100 million dollar in github.com; a solution for software development collaboration that '[orients] around people instead of around [source code] repositories' [3].

GitHub is arguably the prime example of what is called *social coding*: a major change in software development that has quickly become the standard way of realising *distributed collaboration*.

2 Story

2.1 Rule governance modelling at the Dutch tax agency

Many of the business processes that a government agency like the tax agency performs are essentially *decision making* processes. For instance in handling a request for subsidy the majority of the work is not transferring the money but deciding whether the giving subsidy is warranted given the stipulations of the law, in the case of a specific request.

To be able to react adequately to changes in the law, governmental agencies have started using business rules as a single point of definition for the key decision logic.

The Dutch tax agency came to the conclusion that using business rules is not enough for them [4]. What they need is *accountability*: They want to be able to support at all times the outcome of their operational decisions by a reasoned description how the decision follows from the relevant legal sources and from their agencies policy, so that they can account for how their decisions comply with all relevant laws.

What is also needed is *impact analysis:* In the event of an upcoming change in law they want to be able to pinpoint where exactly the work procedures and IT systems need to be changed. That way they should be able to effectuate new law in a short time frame and with minimal cost.

The process of implementing a new law involves a chain of analyses:

1. Careful *modelling of the law*. This is often called annotation after the physical process of highlighting the concepts involved in the legal text which forms the starting point of the analysis.
2. Modelling the relevant agency *policy*. The tax agency will decide how they can reach their goals in the best way given the law.
3. Modelling of the right *portfolio of products and services*[2] to fulfil these demands. Maybe a new service (event type) is needed or the scope of an existing one needs to be changed. This step and the next basically amount to enterprise architecture modelling.
4. Modelling how to best *implement* this change in the relevant *business processes* in terms of work procedures and relevant IT systems: data repositories[3], business rules, process activities.
5. Modelling of the internals of mainly IT components. This is amounts to modeling of system design. This step results in a model of the functioning business processes. After which operations start.

The resulting chain of models can be seen as a traceability graph from law to operational decisions. The traceability graph consists of an acyclic transitive relation that we name ´supports´. The model of the law from step 1 is the starting point of this graph and has a ´represents´ or ´models´ relation with text parts of specific legal sources.

At all steps of the modelling chain design *discussions about* the modelling decisions taken should be included in the traceability graph. These motivations of design decisions are not required for impact analysis but are essential for the compliance chain.

The compliance perspective has specific temporal requirements as well: obviously for any part of the model it should be specified what its validity range is. But also there's a requirement that it should be possible to reconstruct what the model at any point in time was. This last requirement is colloquially known as 'time travel'. Fowler [5] calls this a model with *multiple temporal dimensions*. The latter he calls 'time of record'. Snodgrass [6] calls this 'transaction time'.

To summarise; there are three modelling requirements that we want to address in this paper. First there should be *traceability graph* from legal source texts to operational execution of law. This traceability graph should support *impact analysis* of legal changes and *compliance analysis* of operational decisions. Second *discussions about modelling decisions* should be attached to the traceability graph for reasons of compliance analysis. Third it should be possible to *reconstruct what the traceability graph was* at a given moment in time.

2.2 The collaborative aspect of rule governance modelling

The whole process involves very different competences: legal analysts, civil servants responsible for policy, business architects, business process designers, IT system

[2] Produkt-Diensten Catalogus

[3] Gegevensadministraties

landscape architects and finally system designers and developers of the data stores, business rules and process activities involved. Also the latter steps involve a greater number of people. As a result it is unfeasible to have all the people involved work as one team in one room.

However, impact analysis and accountability can only function if the results of all the analysis steps form one *integral traceability graph* that describes the end-to-end links from legal texts analysis model on the one end down to the operational decisions on the other end. As a result the challenges of the required collaborative rule governance modelling have a lot in common with so-called *distributed collaborative modelling*. That is; collaborative modelling regardless of location and organizational affiliation.

2.3 Benefits of social coding

A study by Carnegie Mellon University in 2012 [7] found that the social coding solution Github allows users to understand "the activities of a large number of others regardless of location or affiliation." And that "this transparency [has a potential] to radically improve collaboration and learning in complex knowledge-based activities." They found that "people make a surprisingly rich set of social inferences from the networked activity information [offered by] Github." Such as "inferring someone else's goals and vision when they edit code, or guessing which of several similar projects has the best chance of thriving in the long term. Users combine these inferences into effective strategies for coordinating work, advancing technical skills and managing their reputation."

They cite research that shows that collaborators in knowledge work who work in the same room are aware of each other's activities "through overhearing, shared visual space and shared memory of discussions around artefacts." As a result knowledge co-workers are aware of each other's work state and expertise which helps them coordinate their activities.

GitHub is a system that lets people that cannot be together in the same room or department have the same type of awareness and mutual knowledge. "The GitHub site is unique in that it makes user identities, intern project artefacts and actions on them publicly visible across a wide community." "The record of all action information combined with user subscription allows activity updates to flow across the site. […] Developers can 'follow' other developers and 'watch' other repositories, subscribing them to a feed of actions and communications form those developers or projects with frequent updates for active projects."

By interviewing developers, the researchers found that people make a rich set of social inferences from this information. From recency and volume of activity developers got a sense of how '*live*' or active a project was by the amount of commit events showing up in their feed. But also, for instance, inferences were made as to who had expertise in which areas.

Another type of inference people made was that "visible information about community interest in the form of watcher and fork counts for a project seemed to be

and important indicator that a project was *high quality and worthwhile*." Developers would also *learn* from following so called 'rock star' developers: developers with a large number of followers that were "deemed to have special skill and knowledge about the domain."

The awareness and visibility created a "direct feedback and interaction between project owners and their user", "the owner could infer more clearly who their user base was, how they were using the project, and when they were having problems." The researchers describe this as a micro supply chain of projects depending on projects where projects *improve the quality of their support for depending projects* through better understanding of how their used.

3 Analysis and comments

3.1 Conceptual model of the data model of Github

To analyse Github it is helpful to consider its data model. The data model of Github can easily be reconstructed by looking at the extensive API [8]. The following UML diagram gives an impression of the underlying data on a conceptual level.

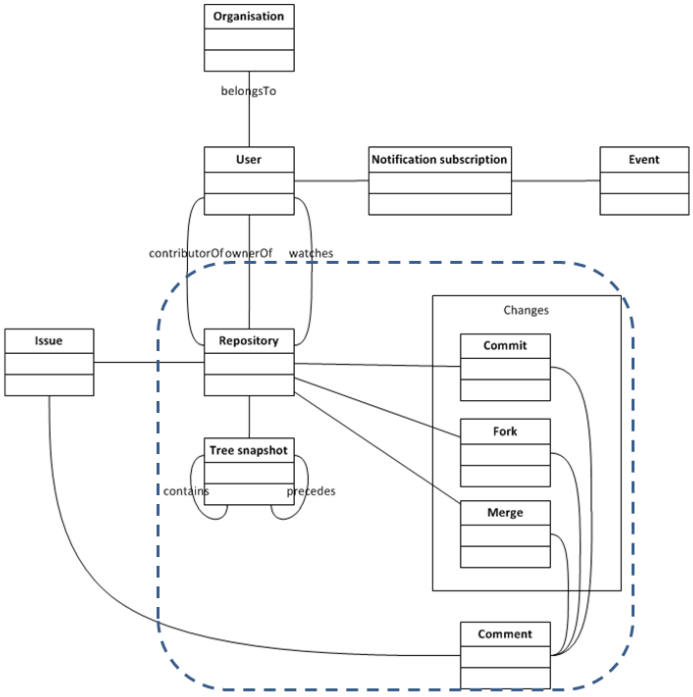

Fig. 1.Github simplified data model

The main notable aspects are that:

- Everything is an event to which a user can subscribe. An example is that a user can 'watch' what is happening with a repository. The resultant personal 'feed' of recent events largely provides the aforementioned awareness of what collaborators are doing.
- Chunks of work that need to be done can be tracked through 'issues', which double as feature requests and other units of work that need to be done.
- Not all comment relations are shown. Mainly, comments and discussions relate to specific (parts of) changes and issues. One could say that discussions pertain to work that needs to be done (or not) and to work that has been done and how to proceed from that.
- Everything is centred around changes and change proposals (so-called 'pull requests').

3.2 Github compared to other collaborative software

From the conceptual data model we observe that coding solutions like Github differ from other collaborative software in that:

1. the work is done in discrete steps and
2. specific work steps are
    a. subject to individual discussion and
    b. can individually be accepted or not or reversed at any time.

This is crucial to collaborative work processes where the work done by different collaborators needs to 'fit' exactly and where it is important to know exactly at what point in time which work results fulfil which tasks or targets. These requirements certainly apply for rule governance modelling.

3.3 Modelling decisions and the traceability graph

Github has an essential feature where it is very easy to submit a change to a model. Other actors have the opportunity to discuss the model in detail, suggest changes and finally accept the suggestions into the final group result. This workflow is called 'forking', 'providing a pull request' and 'accepting and merging the pull request' in Github parlance. From the conceptual model we learned that discussions about potential model changes are available in relationship to the changes themselves. This makes it possible to query these modelling decisions in relationship to the traceability graph and thus fulfull that requirement of agile execution of law.

3.4 Models and temporal dimensions

The Github software can be divided into the open source Git version control foundation and the commercial browser based collaboration software product built on top of it. Git is an example of Distributed Version Control (DVC) software. It is this

version control core that fulfils the requirement of being able to reconstruct what the state of any model was at a given moment in time.

The part of the Github conceptual data model that is managed by Git is marked in the diagram by a blue dashed rectangle.

4 Lessons learned

We found that a social coding solution like Github offers three distinctive features that make it a suitable foundation for a solution for rule governance modelling:

- It offers support for a dependency graph of versions of models
- It offers collaborative awareness to knowledge workers who cannot be physically collocated in the same physical space at all times.
- Progress of modelling work is tracked in discrete units of work that can individually be discussed and accepted or reversed at any time
- It makes it possible to reconstruct the state of the model at previous moments in time

Environments like Github can serve as a source of inspiration for 'collaborative rule modelling environments' of the sort envisioned by the Dutch Government, and can help define requirements and patterns for the realization of such environments.

References

1. Plasterk, Ronald. Visiebrief digitale overheid 2017, http://www.rijksoverheid.nl/documenten-en-publicaties/kamerstukken/2013/05/23/visiebrief-digitale-overheid-2017.html
2. Andreessen, Marc. Why software is eating the world. *The Wall Street Journal* 20 august 2011, http://online.wsj.com/article/SB10001424053111903480904576512250915629460.html
3. Levine, Peter. Software eats software development. http://peter.a16z.com/2012/07/09/software-eats-software-development/
4. Belastingdienst. Gestructureerd Nederlands voor Wetsanalyse. Manuscript april 2013
5. Fowler, M. (1996). *Analysis Patterns*. Addison-Wesley
6. Snodgrass, R.T. (2000). *Developing time-oriented databases in SQL*. Morgan Kaufman
7. Dabbish, L., Stuart, C., Tsay, J., & Herbsleb, J. (2012, February). Social coding in github: transparency and collaboration in an open software repository. In *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work* (pp. 1277-1286). ACM. http://www.cs.cmu.edu/~xia/resources/Documents/cscw2012_Github-paper-FinalVersion-1.pdf
8. Github API, http://developer.github.com/