Requirements Modeling for the Verifiable Claims Task Force

By Joe Andrieu, *First Person Consulting* <joe@joeandrieu.com>

At the recent ID2020 workshop[1], a group of us reviewed the Verifiable Claims Use Cases currently under development by the W3C Verifiable Claims Task Force. http://w3c.github.io/webpayments-ig/VCTF/use-cases/

This paper shares our initial engagement and proposes a few requirements models from my own work based on something I call Fluid Development.

As I led our discussion, it was challenging to exactly pin down what makes a good use case. So this report is as much an exploration of that issue as it is feedback for the VCTF. If, in some small way, we can improve the quality of the output from the VCTF, our work will have been a success. We offer this commentary in that vein. Perhaps it may be of use to those of you working on the challenge of clarifying good requirements for identity systems.

First Impressions

Manu Sporny msporny@digitalbazaar.com introduced the VCTF architecture to the workshop in a brief presentation to kick off the morning's work session. My initial thought was "What about the use cases?" It was hard for me to evaluate the merit of the architecture without a firm grounding in the value it was supposed to provide to the humans using it. I offered to spend some time working on that topic and Manu pointed me to the excellent work already done in this area.

So my first impression was, "Awesome", they've already done a solid amount of work.

Since the team hadn't yet read the work to date, we did so as a collaborative exercise, verbally discussing what we could glean from the document in the limited time that we had. These initial observations are primarily just a reflection of how easy or challenging we found that initial processing. Most of the team had also not worked together before, so we didn't have a foundational lexicon or sense of each other's strengths and focal issues. It was an intriguing exercise in quickly assimilating a group understanding of a complex piece of technical writing.

Flow from Human Value

First, it was hard to tease out the human value from the use cases document. We would have liked the document to flow from the human throughout, but it was structured around technical function. While scenarios illustrate da human interaction that enabled by verified claims, e.g., "Susan looking to send remittances," there was little or no continuity or resonance across different scenarios. The overall effect was that the scenarios—and the use cases—came across as mere illustrations of the technical capabilities of the system, rather than the drivers for the requirements of that system. This may be, and often is, the historical progression of the system, but a good set of use cases should hold together as an independent, compelling story.

[1] Allen, Christopher, et al., ID 2020 Design Workshop. Workshop. Microsoft Technology Center, NYC, NY. May 2015. http://www.weboftrust.info/ and https://github.com/WebOfTrustInfo/ID2020DesignWorkshop

Title and Distinctions

Second, we found it useful to have a simple, recallable "title" for each use case or scenario so it was easy to talk about it. Later, we included clarifications of the technical distinctions for each. Even as early as our presentation back to the workshop, we found value in these short, value-focused titles. In my own experience, I've found that simple titles reduce the cognitive load and increase "meme-ability", by which I mean the ease of repeating the use case across conversations. The simple short hand of a good title greatly increases both the ability to recognize and to discuss the value of each use case.

Later, I'll list the titles we came up with in our review.

Less is More

It was a bit hard to see the trees because of the forest. In part, because of our time pressure, we needed to quickly grasp the core value proposition and we found it hard to get into the work. In part, this is an inevitable consequence of the rigor sought for a standards-track conversation. The explanation of how to read the document, the glossary, and the explanation of the importance… these elements all have their role. But several participants expressed frustration at how hard it took to get to the "meat" of the use cases. Even the examples felt "in the way". It is a bit unclear why there are examples rather than the core work. I'll introduce some ideas about how we might improve on that in the section on requirements models, but the key take away from our rapid assimilation is that fewer use cases would have been easier to absorb.

Terminology is Hard

Several participants found the glossary to be confusing, introducing unfamiliar terms, such as "consumer" rather than "holder", "issuer", or "inspector". On the other hand, I know that the lexical exercise is necessary because we have so many differing ways to think about and work with identity. My own personal soap box—as discussed in my own topic paper submitted for the workshop[2]-- is that discussing identity as a property independent of the observer is confusing. My mathematical training taught me to be parsimonious with my axioms, to start with the least amount of definition needed. Perhaps the glossary could be reduced. What I think is more important is that the terminology make sense across the audiences reading the use cases and related W3C documents. As a newcomer to W3C work in this area, it is hard to comment further.

[2] Andrieu, Joe "Identity is a Phenomenon, not a Property", *ID 2020 Design Workshop Topic Papers*. May 2016. Online. https://github.com/WebOfTrustInfo/ID2020DesignWorkshop/blob/master/topics-and-advance-readings/Identity%20is%20a%20Phenomenon%20Not%20a%20Property.Andrieu.2016.pdf

Requirements Modeling

Requirements change.

That's the foundation for requirements modeling.

We know requirements change, whether because they were initially incomplete or wrong, because the team learned more about the problem and solution, or because the market changed.

Requirements modeling treats requirements as an ongoing process instead of a static specification. Rather than "requirements engineering" that results in a rigorous specification, requirements modeling focuses on a living "current best consensus" in the form of models. These models can be created and updated "just-in-time" to meet the evolving needs of the project.

As mathematician George E. P. Box put it

All models are wrong. Some are useful.[3]

Of course, you have to pick the right models, but when models can be quickly created and easily updated, the cost of choosing poorly is minimized. It also vital to understand that, inherently, all models are wrong. Requirements—as rigorous specifications—can never be complete nor completely accurate.

Requirements modeling accepts that the current best consensus is the best you can do, avoiding the conceit of the "Market Requirements Document" and other formal specifications. Requirements modeling allows work to proceed before exhaustive specifications are nailed down, allows the project to evolve as the team learns, and gives permission to be wrong, to sketch out incomplete ideas, to explore possible solutions, to fail during prototyping, and perhaps most importantly, to focus on the work rather than slavishly following a static, out-of-date document based on what a few people *thought* were the requirements at the beginning of development.

For the VCTF work, this means accepting that not all possible use cases need to be illustrated and. Sometimes the best way to quickly communicate the core fundamentals is by limiting the details and scope to allow a clearer picture of individual interactions. A combination of high-level overview with minimal detail and one or two deep dives can more quickly provide an understanding of a system than an encyclopedic tome of all possible scenarios, interactions, features, capabilities, and performance criteria.

The Model

Requirements modeling builds, in part, on Constantine & Lockwood's work[4] to focus on the model as the bridge between the problem domain and the solution domain.

Problem -> Model -> Solution

A good Model allows experts in both domains to understand what problem is being solved and how the solution will work. In the terminology of Fluid Development, the requirements "Model" for a system

[3] Box, G. E. P., and Draper, N. R., (1987), *Empirical Model Building and Response Surfaces*, John Wiley & Sons, New York, NY. Cited on Wikiquote https://en.wikiquote.org/wiki/George_E._P._Box

[4] Constantine, Larry and Lockwood, Lucy A. D., *Software for Use*. ACM Press, NY NY. 1999.

(with a capitol "M") is the best current consensus in the form of a collection of models (lower case "m"), each with their own particular view of the system.

Some models, such as personas, scenarios, user need maps, and lifecycle engagement models, are closer to the problem domain, capturing the needs of humans and businesses to be addressed by the system. Other models, such as user task maps, detailed use cases, and component, communication, and sequence diagrams, are closer to the solution domain, describing how an implementation would accomplish a given task.

For the VCTF use cases document, the problem domain elements (the scenarios) were separated and subordinate to the solution domain elements (the technical use cases). I'll suggest some models that may help present a simpler, clearer picture. But first, a note about use cases.

Use Cases

"Use Cases" are a popular technique for capturing requirements in both problem and solution domains. A use case captures a single value-creating interaction with a system. They have become so ubiquitous in software development that the term is often used without a concise definition.[5] Rather than focus on defining this term, the group at the ID 2020 Design Workshop first reviewed all of the use cases and scenarios in the Verifiable Claims Use Case document in an attempt to distill the key human values and give each use case and scenario a recallable name.

After the list of potential use case titles that follows, I present User Needs, User Tasks, and the User Roles involved in Verified Claims.

Extracted Use Case / Scenario Titles

Titled scenarios with the individuals' names and the Use Case it came from.

1. **Reuse Know Your Customer** (Jane 4.11 and Anna 4.4.1) – Portable claims
2. **Signed Transcript** (Jolene 4.1.1)
3. **Digital Driver's License** (Asako 4.1.1) – Real-time presentation & validation
4. **Prescribing** (Barney 4.1.1) – Qualification to Work
5. **Send Money** (Susan 4.1.2) – Compliance & secure transactions
6. **Finding a Doctor** (Jason 4.1.2) – Qualification to work
7. **Taking a Test** (Eunice 4.1.2) – Attaining qualifications
8. **Closing Account** (John 4.2.1)
9. **Lazy Doctor** (Barney 4.2.1)
10. **Bad University** (Jane 4.2.1)
11. **Trying out a new service** (Nikita 4.3.1)
12. **Transferring Schools** (Rocky 4.3.1)
13. **New Employer** (Jessica 4.3.1)
14. **Social Authority** (Josie 4.4.1, Paula 4.4.3) – Minimal Disclosure Credentials

Following the same general guidelines, here are possible titles for the remaining scenarios:

15. **Online courses** (Nick 4.4.1)

[5] As we see if the Verifiable Claims Task Force Use Cases May 20 2016 Report.

16. **Address Verification** (Francis 4.4.2)
17. **New Bank Account from Home** (Alice 4.4.2)
18. **Online pharmacy** (Bob 4.4.2)
19. **Job Application** (Cindy 4.4.2)
20. **Adult Beverages** (June 4.4.3) – Minimal disclosure
21. **Traveling Illness** (John 4.4.3) – Minimal disclosure

Some of these titles are much more memorable than others, e.g., Lazy Doctor and Bad University are a bit more intriguing, perhaps because the colorful adjective invites further inquiry and provides and emotionally charged handle.

Requirements Models

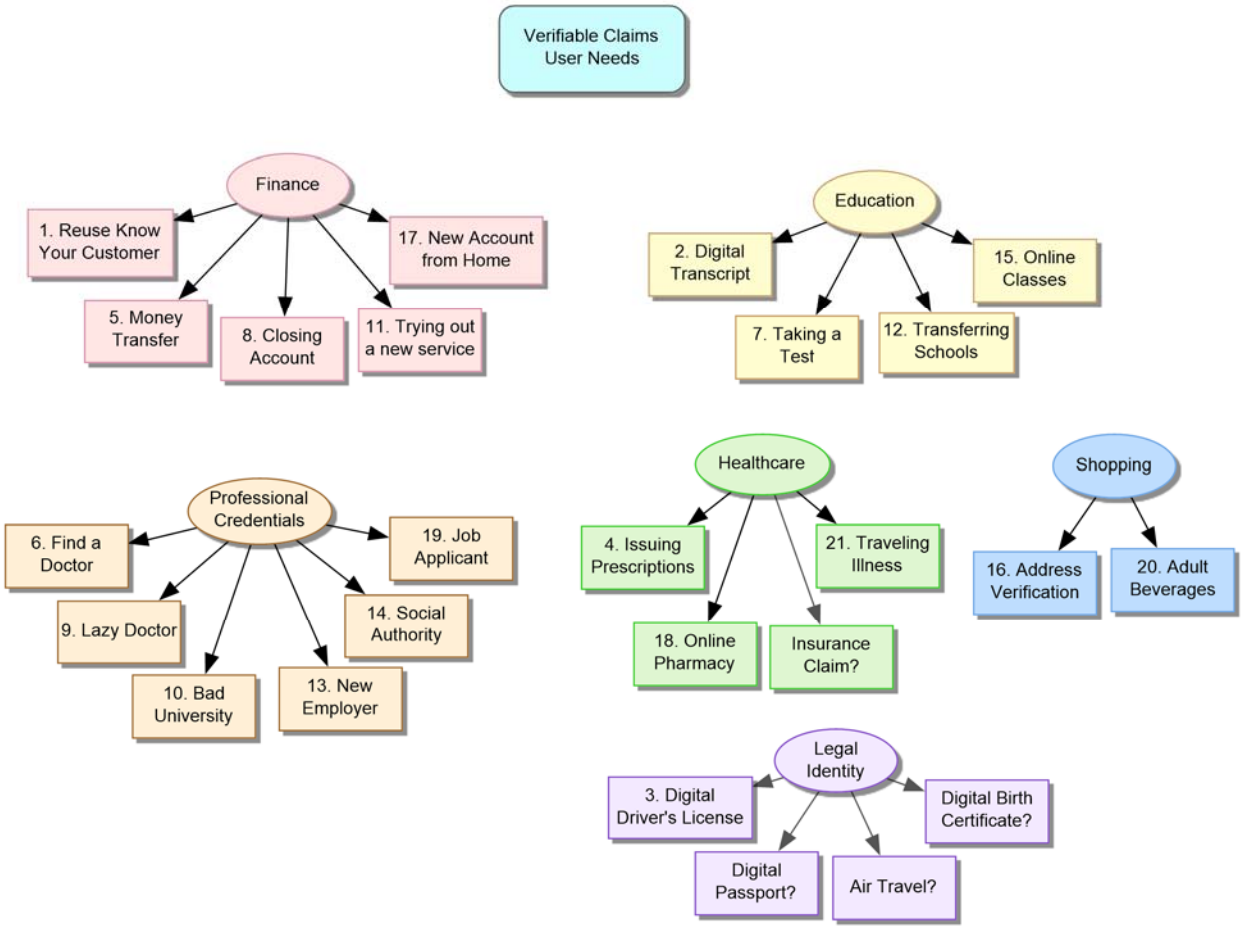
What follows are a few proposed requirements models for more concisely communicating the scope of the Verifiable Claims use cases. If they are useful for the upcoming W3C work, I'd be happy to collaborate to produce something suitable for publication in that process. These are "draft" quality models: rough, quick, and needing feedback and co-creative engagement with the rest of the team. Hopefully there is enough there to give a sense of what I was aiming for and how they might be of use. Also, I think the titles listed above could be improved with feedback from the group. It would be great if more of those scenarios had titles as evocative as Lazy Doctor and Bad University.

In general, each of these models should have coherent numbering and a brief prose description for each element for easy cross-referencing.

User Needs

Based on the scenarios, as entitled above, what follows is a suggested "User Needs Map" describing the User Needs being addressed by Verifiable Claims. Below the image, I describe a few we might consider adding. The goal of a needs map is to quickly present a snapshot of the entire scope of needs met by the proposed system. I've grouped them into topic areas so it is easier to browse.

User Needs are problem domain Use Cases. They use the language of the human domain, describing at a high level how real human needs are addressed by the proposed solution. Scenarios are illustrations of Use Cases (either User Needs or User Tasks for specific people in specific situations).

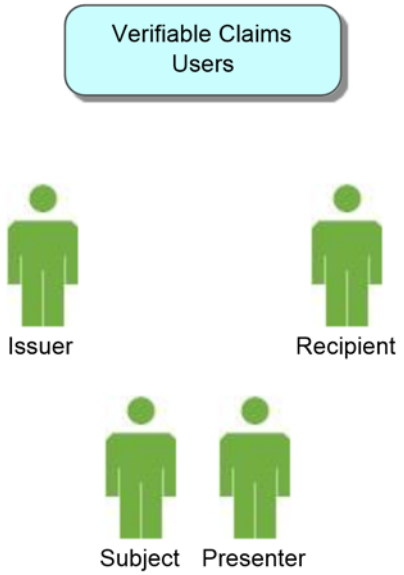


Insurance Claim – Presenting credentials at a healthcare provider for insurance coverage of health care services.

Digital Passport & Digital Birth Certificate – The ID2020 Summit demonstrated a real-world need for an improve form of legal identity for both transnational situations and record of birth. I'd like something more specific as the Digital *** use case is less evocative than "Unexpected Twins" or "Untimely Death".

Air Travel – A use case that emerged in my lunch break-out session at ID2020. When will we be able to use a digital identity system for boarding a flight?

User Roles



Issuer – The author of a particular claim; also capable of revoking it.

Recipient – The recipient of a particular claim about a subject

Subject – The focus of a claim. May or may not be involved in the use of that claim, especially in birth and death use cases.

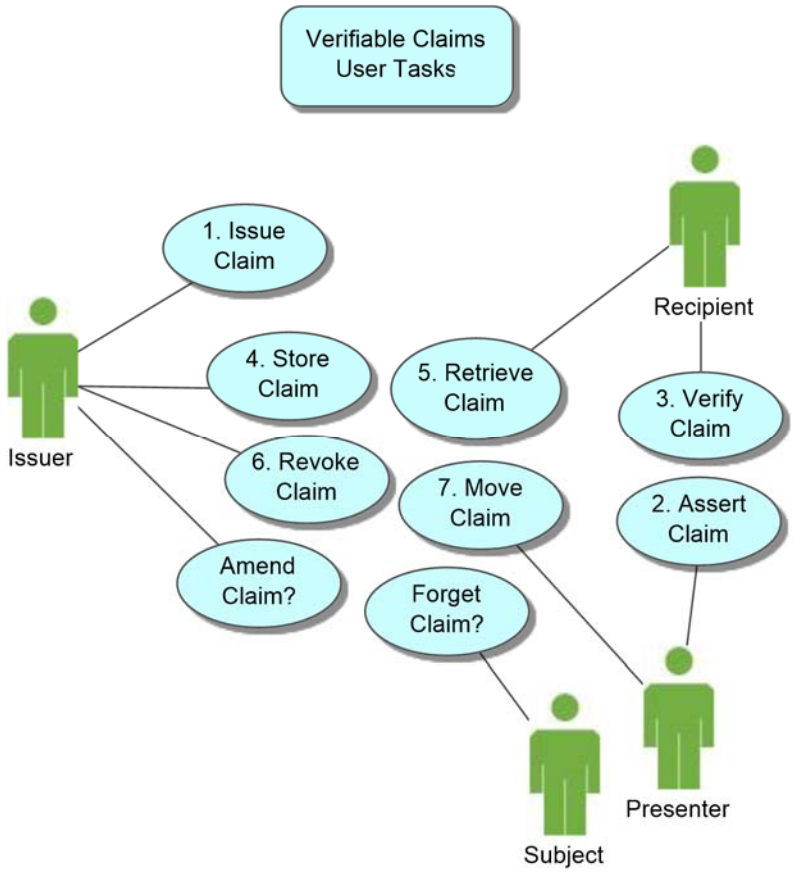
Presenter – The person asserting a given claim to a recipient.

I've taken the license to adjust this terminology to what was clearer to me. I don't want to re-open a rathole, so feel free to dismiss these suggestions without debate.

There were a few concerns raised at the workshop with "consumer" and based on the rest of the scope, "recipient" seemed to fit. The term "holder" was confusing for use cases when the claim is stored in a repository "in the cloud" or "on the blockchain"; the "holder" would seem to be the storage entity. Also, I found defining "holder" as who controlled the claim was a bit of a challenge, since issuance, amendment, and revocation are forms of control; so I separated the presentation of a credential (typically on the authority of the subject) from its revocation (on the authority of the issuer).

User Tasks

Mapped almost directly from the Verifiable Claims use cases, I propose the following User Task model. User Tasks are solution domain Use Cases, focused on a specific transaction with the system. A scenario based on a User Task would focus on a specific interaction with the system.



There are two additions I thought might be useful.

First, in light of the E.U.'s right to be forgotten is the task of a subject triggering the forgetting of a claim. I'm not sure if Verified Claims as proposed is capable of this, but it may not be too far removed from revoking claims, so I wanted to propose it as a potential user task.

Second, amending a claim – or perhaps replacing a claim – may be useful for situations where the issuer needs to update the status of the subject, but the Presenter and/or Recipient still use the same reference to assert or retrieve & validate that claim. For instance, my status as student may evolve from undergraduate to graduate; amendment or replacement would allow that updated claim to be invoked without re-asserting a new claim. Clearly amending (with a history) and replacing (complete novation without history) have different technical and policy consequences. However, since I'm not sure what the consensus of the task force might be, I simply proposed one of those options as a possibility.

Future Work

This is the current state of my draft. If the group is interested, I would be keen to add at least one Lifecycle Engagement model based on one of the proposed scenarios and a single Detailed Use Case of a single step in that Engagement model. A lifecycle engagement model tells a concise story of a specific individual as they interact with the verifiable claims system throughout the life of a verifiable claim. A detailed use case is a technology-free narrative of the user intention and system responsibility; it is typically a powerful bridge between the human language narrative and the technical language of system components and APIs.

Also, as stated earlier, if any of this feels close to something worth including in the W3C submission, I'd be happy to collaborate on revising it for both improvements and better alignment with the task force's consensus.

FWIW, I also think the use case document itself would strongly benefit from incorporating the implied/proposed architecture as that will help ground some of the models closer to the solution domain.