Zur Spezifikation der Aufgabenstellung des Themas Koop-11 im SWT-Praktikum 2011

Eine E-Mail-Diskussion zwischen Matthias-Christian Ott (MCO) und Jörg Wittenberger (JFW) mit teilweider Einbindung von Prof. Hans-Gert Gräbe (HGG).

Zusammenstellung des Textes durch HGG mit folgender Vorbemerkung: Es ist von vornherein klar, dass eine Diskussion mit mehreren thematischen Fäden schwer in eine lineare Form eines Textes zu bringen ist.

JFW am 09.02. zum Verhältnis von „so etwas wie“ Diaspora und Askemos:

... die Idee mit Diaspora kann ich mir sehr gut kooperativ vorstellen:

- So eine Oberfläche muss in „Web 2.0“ gemacht werden. Ein Haufen AJAX, Browserabhängigkeiten und Hintertüren. Deswegen meide ich den Versuch/Aufwand, Experimente dazu unter dem Anspruch und der Marke Askemos zu machen. ... Und leide darunter, dass man meine Oberflächen wohl als „accessability optimized“ verkaufen kann und vielleicht noch „obervorsichtig gebaut“ – aber eben nicht als schön.
- Die Oberfläche sehe ich definitiv als Komponente, und zwar als eine, die wirklich nix unter dem Thema Askemos verloren hat.
- Die Funktionalität „social network“ (Funktionalität nach dem Motto „function first, sugar later“) ist natürlich genau das gegenwärtige Buzzword für die Herzensangelegenheit der Idee hinter Askemos.

Ergo: Für „so etwas wie“ Diaspora (im Sinne von: Askemos kümmert sich um die Interaktion zwischen den Nutzern und Diaspora strukturiert und malt das Nutzerinterface und die Integration mit anderen Webanwendungen) als UI würde ich mich schon etwas strecken. Aufgabe müsste nur etwa sein: der eigentlichen Prozess (bzgl. Personen, also Vertragsschluss, Zahlung etc.) wird irgendwie in Askemos abgebildet (mit Punkteabzug nur als die zu verwendende Datenbank, Plus: die Entscheidung wird im Askemos/BALL gefällt – das darf gerne auch auf einer isolierten Maschine als Netzwerkmodell geschehen).

Praktisch: Mit den letzten Entwicklungen, die nun auch so stabil sind, dass ich nicht mehr laufend in der Anwendungsentwicklung gestoppt werde, weil irgendwas im Kern nicht so funktioniert, wie es beabsichtigt ist, steht auch SQL mit zur Verfügung, HTTP, WebDAV, Scheme und etwas XSLT sowieso. Derzeit bastle ich noch ein paar Demos, u.a. für das Stadtteilarchiv der Neustadt. Es sollten sich also brauchbare Interfaces finden lassen.

Weiter:

- Ich wüsste auch gerne, wozu man Ruby vorteilhaft verwendet, also, was man davon wirklich gerne hätte.
- Schon eine Analyse auf Praktikumsniveau, was Diaspora an Vor- und Nachteilen hat, würde mich interessieren: spart mir die Zeit, denn meine Meinung dazu, was ich da seziere, kennst Du (HGG) ja schon.

- Ich fände es spannend, gelegentlich zu vergleichen, wie man denselben Effekt in Scheme (oder anderweitig anders) codieren würde. (Ich sehe Ruby-Blöcke als anonymes Lambda und frage mich, ob das zutrifft.) Braucht aber als „schön gemacht“ beurteilte Konstrukte, 0-8-15 kann jeder – oder nicht.
- C-Schnittstelle ist nicht so wild: Netzwerk geht doch auch.
- Bei Lesen von pastry ist mir aufgefallen: die führen ganz am Anfang in Continuations ein. Das Konzept
  a) ist die konzeptionell einzige Steuerstruktur in Scheme,
  b) ist für Web-Anwendungen auffallend gut geeignet,
  c) entspricht dem Konzept des Platzes in Askemos (abgesehen davon, dass zum Platz noch weitere Informationen gehören – insofern also genau der Funktionalität eines Platzes unter Vernachlässigung der topologischen Einbettung, zugewiesenen Capabilities u.a.).
     Mir scheint nur, dass die Continuations nicht replizieren, sondern nur lokal verwenden können.
  d) bringt im Falle der zustandsfreien Anbindung via HTTP eine interessante Frage mit sich: wie lange sollte so eine Continuation in Erwartung einer Antwort – die vielleicht nie kommt – stehen bleiben?

MCO am 21.02. zur Präzisierung der Themenstellung:

Im Nachhinein ist mir aufgefallen, dass wir uns heute beim Gespräch über das Softwaretechnikpraktikum zu sehr auf technische Details konzentriert haben. Ich hätte aber noch einige inhaltliche Verständnisfragen (auch wenn das sich vielleicht mit der ersten Projektphase überschneidet, aber da das Projekt ja, wenn richtig umgesetzt, Neuland und damit etwas schwieriger als vielleicht das Webseiten programmieren nach klassischer Herangehensweise ist, denke ich, es ist sinnvoll wenn ich mir schon ein paar Gedanken im Voraus machen kann).

So wie ich es verstanden habe, wollen Sie (HGG) mit dem Projekt die Kooperation und den Wissenstransfer zwischen kleinen Firmen fördern. Dies soll durch öffentliche Bereitstellung und Austausch von Informationen passieren, wobei es verschiedene Abstufungen der Öffentlichkeit gibt (einzelne Personen innerhalb der Firma, alle Personen in der Firma, Geschäftspartner, Kunden und so weiter).

Gleichzeitig soll die Software auch noch eine Art „Enterprise Relation Management System“ sein (was auch immer darunter heutzutage verstanden wird; auf jeden Fall sind solche Systeme Großprojekte, die vermutlich unbenutzbar sind, oder gar nicht erst fertig werden), in denen irgendwie alle Kommunikation und alle Abläufe einer Firma abgebildet werden.

Als weiterer Aspekt wird an ein „Document Management System“ (gleiche Definitionsvielfalt wie beim Begriff des „Enterprise Relation Management System“ [1]) angeführt, in dem Dokumente sicher (gemäß eines mehrdimensionalen Verständnisses des Begriffs) gespeichert und verwaltet werden können.

> JFW 22.02.: Ich schreibe hier mal auf, was wie mit Askemos [6] zu tun haben könnte, denn mit den bisherigen Ausführungen könnte man typische Anwendungsgebiete für das Askemos-Konzept umreißen.

> Um die Sache mit den Großprojekten zu vermeiden, die auf one-size-fits-all hinauslaufen und das mit dem Aufwand der größten Nummer bezahlen, basiert das Konzept auf der Interaktion kleiner Anwendungen und diese sind frei programmierbar. Damit passt sich dann jeder an seine Bedürfnisse an.
>
> MCO 22.02.: Die Erfahrung habe ich auch gemacht. Anwendungen sollten komponentenbasiert sein. Das heißt, anstatt irgendwie sich das Enterprise-kann-alles-Framework zu holen installiert man sich nur in sich abgeschlossene Komponenten und verbindet die dann.

Gerade letzteres scheint widersprüchlich zum Konzept der Dezentralisation, denn Zugriffskontrolle mit nachträglicher Möglichkeit zum Widerruf von Rechten ist ja nur mit einer zentralisierten Überwachungsinstanz möglich (siehe dazu meine Ausführung in meiner Seminararbeit [7] über DRM und dort referenzierte Quellen).

> JFW 22.02.: Sehr wohl. Nachträglicher Widerruf von Rechten ist in der Regel ein juristischen Unding. Recht ändert sich höchstens „von nun an“.
>
> Zentrale Instanzen haben in diesem Sinne ein ernstes Problem: solange nicht auszuschließen ist, dass die zentrale Administrationsgewalt nicht missbraucht worden sein kann, sind die aus solchen Systemen gewonnenen Aussagen in der Regel einer Partei zuzurechnen und haben damit den Stellenwert einer Behauptung. Bestenfalls eines Indizes. Als Beweismittel taugen sie nicht.

So würde ein solches System stets zentralisiert in einem Unternehmen und einen Single-Point-of-Failure darstellen und somit teure Hardware, aufwendige Backupstrategien und dediziertes Personal erfordern, was eine kleine Firma sicher nicht im ausreichenden Maß erfüllen kann. Wenn ein solcher Dienst über mehrere Firmen verteilt wäre (beispielsweise bei Verträgen), wäre er zwar federated und das ganze wäre noch nicht so gravierend wie Cloud-Computing (wie zum Beispiel von salesforce angeboten), aber für kleine Firmen sicher nicht überzeugend. Wie sind hier die Begriffe „Identitätsmanagement“, „Rechtesystem“ und „dezentral“ zu verstehen?

> JFW 22.02.: Sehr gute Fragen! Insbesondere die Frage nach überzeugenden Argumenten interessiert mich. Da ich selbst überzeugt bin, hab' ich das Problem, dass ich die Fragen nicht gut sehe.
>
> In der Tat ist die Verteilung über mehrere, in der Regel etwa „gleichgroße“ Unternehmen ein zentraler Punkt im Askemos. Gerade beim Abschluss bzw. der Speicherung von Verträgen. Siehe dazu auch `http://idons.askemos.org/SRS`. Dort wird eine *contract language* zur Definition der Regeln zum Update eines dezentralen DNS vorgeschlagen. Nicht aus Zufall, das ist die Idee von Askemos: *gemeinsame* globale Zustände (hier die DNS-Datenbank) werden nach gemeinsamen Regeln in Abstimmung (byzantinisch) gepflegt.
>
> Gerade für Verträge ist das unabdingbar.
>
> Insofern ist Askemos die Idee der Cloud, was das „sei bitte verfügbar, auch wenn meine Hütte abbrennt“ angeht, macht aber nicht den Fehler, dafür Vertrauen

> in unbekannte Dritte zu fordern. Und schon gleich gar nicht, private Daten wie Verträge jenen – zu welchen Zwecken auch immer – verfügbar zu machen.
>
> Ganz ohne Vertrauen geht es aber nicht ...

> MCO 22.02.: Man könnte ja noch zusätzlich überlegen, sämtliche Computer des Unternehmens in Askemos-Knoten zu verwandeln (sofern es trotz Kryptographie natürlich keine Bedenken gibt), um die Redundanz zu erhöhen.

> d JFW 08.03.: Ja, auch das ist im Prinzip denkbar. So funktioniert ja derzeit die Praxis: ich habe zu fast allen Knoten administrativen Zugang. Ergo ist die Geschichte solange nur ein Modell, denn ich könnte ja fälschen, wenn es nicht so aufwändig wäre.

> MCO 22.02.: Eine andere Idee wäre eine minimale Version von Askemos auch auf Smartcards zu implementieren, da diese nur sehr aufwendig kompromittierbar sind (neuere Karten haben auch schon sowas wie 1 MiB Speicher), oder für die Authentifikation Smartcards zu verwenden, ohne die man in einem Unternehmen nicht rein und nicht raus kann, damit die nicht vergessen werden.

Realistischer wäre ein dezentrales Konzept (wie es auch Askemos umsetzt). Bei diesem wäre die Zugriffskontrolle nicht so umfassend, und wie in der Realität (DRM und ähnliche Technologien sind ja realitätsfremd) müssten Vertrauensverhältnisse aufgebaut

> JFW 22.02.: Genau so funktioniert das. Mein Beispiel ist da gewöhnlich: meine Daten liegen (zwecks Nachweis, Backup etc.) da, wo sie hingehören. Erstens bei mir, zweitens bei denen, wo sie herkommen (Gesundheit: beim Arzt, Einkommen beim Arbeitgeber, Fotos bei Freunden), drittens, wo sie hingehen (Geld: Bank, Gesundheit: Apotheker). Und wenn's nötig wird (mehr backup), dann suche ich mir Personen meines Vertrauens.

und nicht Misstrauensverhältnisse versucht werden technisch durchzusetzen.

> JFW 22.02.: Das ist einer der besten Sprüche, die ich zur Charakterisierung der Vorgehensweise je gehört habe.

Das würde dann bedeuten, die Zugriffskontrolle würde sich darauf beschränken, Personen Dokumente auszuhändigen und dann gemäß dem Motto „Information wants to be free“ davon auszugehen, dass sie unbeschränkt verbreitet werden, aber gleichzeitig darauf zu vertrauen, dass die jeweilige Person sich an Abmachungen hält und von dieser Möglichkeit nicht Gebrauch macht.

> JFW 22.02.: Exakt! Informatiker sollten niemals versuchen, neue Verfahren zu erfinden. Jedenfalls nicht bei diesem Thema. Job der Informatiker ist es, die Welt zu modellieren, wie sie sie vorfinden.
>
> Zum Thema „Geheimnis“ gibt es den Begriff des „Verrates“ als Bruch eines bestehenden Vertrauens- oder Vertragsverhältnisses.

> MCO 22.02.: Manchmal kann Technologie die Welt, wie wir sie vorfinden, aber auch verändern.
>
> JFW 08.03.: Das will ich nicht bestreiten. Passiert aber doch eher als Nebenprodukt, sobald irgend ein zuvor schon real existierender (und vor allem interessanter) Vorgang so effektiv abgebildet wird, dass andere Alternativen überflügelt werden.

Ein solches Modell ließe sich ja sehr wohl mit minimalem Wartungsaufwand auf Commodity-Hardware und über das Web-of-Trust, das ja solche Vertrauenssituationen halbwegs realistisch abbildet, realisieren.

> JFW 22.02.: Das versucht Askemos.
>
> MCO 22.02.: Interessant fand ich in dem Zusammenhang auch RFC 6091.
>
> JFW 08.03.: Großartig. Den habe ich gar nicht mitbekommen. Sehr schön; jetzt muss das nur noch jemand implementieren. Warten wir also ab.
>
> MCO 08.03.: Also GnuTLS implementiert das schon seit Jahren. Seit 2007 ist das mit RFC 5081 auch schon standardisiert. Leider implementiert das OpenSSL nicht (ich denke, es ist auch nicht sehr warscheinlich, dass das je passiert).

Bezüglich des Kommunikationsansatzes ließe sich doch festhalten, dass dafür existierende Technologien genutzt werden sollten, das sind heutzutage E-Mail (RFC2822, SMTP, IMAP und so weiter) für zeitversetzte Kommunikation

> JFW 22.02.: Jetzt muß ich mal einen alten Notizzettel[1] anführen: Deswegen spricht `ball.askemos.org` HTTP/S und SMTP, bislang. IMAP ist derzeit etwas, was „nur“ Arbeitszeit braucht. DNS-Updates macht es auch schon.

und XMPP und SIP für Echtzeitkommunikation,

> JFW 22.02.: Echtzeit wird im Rahmen von „Dokumenten“ im Sinne von Verträgen nur selten benötigt (Stock-Exchange). Das kommt später.
>
> MCO 22.02.: Das hatte ich so aus der Praktikumsbeschreibung herausgelesen, dass auch Echtzeitkommunikation möglich sein soll.

wenn das Projekt praktikabel sein und Akzeptanz finden soll (irgendwelche Webinterfaces und Messaging-Protokolle sind nicht kompatibel und finden sicher keine Akzeptanz). Oder hatten Sie (HGG) sich hier etwas anderes vorgestellt (Diaspora geht ja mit dem Salomon Protocol [2] einen ganz anderen Weg)?

> JFW 22.02.: Nun, sowas ist aus Sicht eines Programmierers *auf* Askemos genau die Aufgabe: man implementiere jene Protokolle, die dem Zweck dienen. Das unterliegende System kümmert sich nur um die Ausführung und den Nachrichtenaustausch. Nachrichteninhalte bestimmt das Programm.

[1] `http://www.askemos.org/A849640f672ed0df0958abc0712110f3c/KommunikationsInfrastruktur`

Das Betreiben eines SMTP-Servers (bei XMPP und SIP sehe ich das Problem erst in einigen Jahren) in den eigenen Geschäftsräumen ist sicher sehr sinnvoll, bloß müsste auch funktionierendes SPAM-Filtering durchgeführt werden und es müsste sichergestellt werden, dass keine Open-Relays enstehen.

> JFW 22.02.: Deswegen stelle ich dann dem Kunden eine fertig konfigurierte Maschine hin und kassiere für die Arbeit der Pflege. Zumindest ist das der Plan. Unter meinem Schreibtisch werkelt derweil ein Guruplug. Der macht eine der 8 Maschinen aus, auf denen die Daten z. B. der askemos.org-Website liegen. Eine andere Maschine ist ein Seagate Docstar bei meinen Eltern. Ich muss erstmal sicher sein, dass das alles so geht, wie gedacht.

Bei einer dezentralen Kooperation und Wissenstransfer ist ja sicherlich das gemeinsame Bearbeiten von Dokumenten wichtig. Praktischerweise können diese Änderungen, sofern das Netzwerk fehlertolerant und verfügbar sein soll, nicht konsistent sein (Inhalt des CAP-Theorems).

> JFW 22.02.: Ja, es ist nicht 100 % verfügbar. Das war aber mein Rechner noch nie. Ihrer?

> MCO 22.02.: Ich meinte damit eigentlich eine Verfügbarkeit a la Amazon Dynamo.

Das bedeutet, es müssten auch Änderungen zusammengeführt werden, was nicht trivial sein kann (Askemos zieht hier ja Konsistenz vor und Inhalte müssten daher vor der Speicherung zusammengeführt werden oder im Konfliktfall ganz verworfen werden).

> JFW 22.02.: Ja, Zusammenführen ist aus Sicht des Kerns eine Aufgabe der Anwendungsprogramme. Grund: ich kenne keinen allgemeingültigen Algorithmus. Also hat es im Kern nix zu suchen.

Gerade bei nicht textbasierten oder komplexen Dokumenten (beispielsweise Dateien aus LibreOffice oder Microsoft Office) halte ich das Zusammenführen für unmöglich oder mit mehreren Jahren Entwicklungszeit verbunden. Dies würde dann auch direkt alle Daten in dem Dokumentenspeicher betreffen und nicht nur ein Wiki oder ähnliches. Wie ist hier die Dezentralität zu verstehen?

> JFW 22.02.: Siehe oben: Askemos hat einen beschränkten Scope. Wir versuchen nicht, die Welt vorwegzunehmen. Datenmanipulation ist Sache der Anwendung. Dafür gibt es problemorientierte oder allgemeine Programmiersprachen.

Die Software soll ja im Endeffekt auch Nachrichten signieren. Gemäß dem Signaturengesetz ist für Signaturen, die einer handschriftlichen Unterschrift gleichberechtigt sind, ein X.509 Zertifikat, das als Wurzel der Zertifikatskette die Bundesnetzagentur hat, notwendig. Diese Organisationsform ist natürlich streng hierarchisch. Wie soll das in dezentrales System integriert werden?

MCO am 21.02.2011 zum Verhältnis zur FreedomBox:

Wie stünde das Projekt zum von Eben Moglen vorgeschlagenem Konzept der FreedomBox [3,4,5]?

> JFW 22.02.: Das ist das Konzept des Askemos. Schon die ganze Zeit. (Zeugen für Sätze wie „ich stelle mir eine Box vor, die neben dem DSL-Modem angebracht wird. Da liegen meine Daten, die macht das gegenseitige Backup im Netz ...“ finden sich für das Jahr 2000 und bei sourceforge liegt der damalige Wiki-Inhalt noch rum.)

MCO am 22.02.2011 zur Rolle von Askemos:

> MCO 21.02.: Welche Rolle würde Askemos einnehmen? Vom Konzept her könnte man es nämlich sicherlich für die Dokumentenspeicherung verwenden (wenn auch mit deutlichem Anpassungsaufwand und vielleicht nur eingeschränkt nutzbar und eigentlich als virtuelle Maschine anstatt Datenspeicher viel zu allgemein und teilweise andere Ziele verfolgend).
>
> JFW 22.02.: Die VM versucht nur, die oben angeführten Anwendungen als frei programmierbar zu ermöglichen und so die Anwendung von der Infrastruktur zu trennen. `ball.askemos.org` ist der erste Versuch einer Implementation.
>
> Nicht nur Datenspeicher zu sein, ist nur in genau einer Anwendung nötig: beim Verwalten von Daten, welche gegen Manipulation geschützt werden sollen. Dann braucht es die *contract language*, welche bestimmt, welche Aktualisierungen rechtens sind und wie diese dokumentiert werden. Das müssen alle Parteien in gleicher Weise selbst durchführen, sonst: SPOF.
>
> MCO 21.02.: ... da ich schon möchte, dass wenigstens die Architektur (auch wenn der Quelltext nachher weggeworfen wird) ordentlich werden sollte, dass man, wenn sie erfolgreich funktioniert, vielleicht einen kleinen technischen Report darüber schreiben kann, um implementierungsunabhängig aufzuzeigen, dass sich praktische dezentrale Systeme auch heute schon realisieren lassen (wenn auch nicht allen theoretischen Anforderungen genügend, aber immerhin).
>
> JFW 22.02.: Auch hier sind wir einen halben Schritt weiter. Das System ist von vornherein darauf angelegt gewesen, implementierungsunabhängig zu sein.
>
> „Note that Askemos concerns the abstract specification exclusively. Data formats, protocols, service interfaces etc. – not the actual implementations.“[2]

Bezüglich der Programmierung ist es ja bei Askemos zur Zeit so, dass das irgendwie in XSLT eingebettes Scheme ist, das dann Ausgaben in Form von XML produziert.

[2] `http://www.askemos.org`

> JFW 08.03: Nun, ich hoffe, das wird jetzt langsam auch für Sie zum Mythos. XSLT ist völlig optional. Es gibt Leute, die nehmen das gar nicht. (Dabei kann es richtig leistungsfähig sein.) Scheme ist hochgradig einfach. SQL hatten Sie vergessen. XML ist nur als spezieller Datentyp im Kern bekannt, damit ein paar Optimierungen möglich sind. WebDAV kümmert sich beispielsweise gar nicht um den Inhalt des BLOB.

Für den vorliegenden Fall (und sicher auch andere Fälle) wäre es aber besser, wenn man irgendwie aus anderen Sprachen und vor allem nicht in XSLT und mit XML Ausgabe (XSLT kann auch Reintext, aber das zählt hier nicht) arbeiten könnte. So dass z. B. die Funktionen in eigenständigen virtuellen Maschinen und Betriebssystemprozessen sind (um Seiteneffekte und ähnliches muss sich der Programmierer dann selber kümmern), dann über REST, ØMQ oder ähnliches den Platz (Zustand) und den Request bekommen, dann herumrechnen und schließlich den neuen Zustand und die Reponse, wenn vorhanden, produzieren. So würde sich Askemos um die Speicherung und Konsistenz kümmern und die Anwendungen könnten trotzdem in anderen Sprachen (Heterogenität ist ja heute sehr wichtig) implementiert werden. Dann kann man auch gleich SOA drauf schreiben.

> JFW 08.03: Kann man. Wer schreibt den Code wann?
>
> JFW 22.02.: Derzeit läuft bereits ein Projekt, in dem eine Java-Implementierung entstehen soll. Teil dieses Projektes wird sein, das Abstimmungsprotokoll im Stile eines RFC zu beschreiben. (Derzeit: see the source Duke. Warum: stellenweise werden Werte „mal schnell so“ transferiert. Das wird sich ändern, sobald eine Alternative zum Kompatibilitätstest entsteht, denn ich werde das nicht dokumentieren, sondern die dokumentierte Form implementieren.)

Wenn das Protokoll soweit dokumentiert ist, dass es verständlich ist (muss ja nicht gerade RFC-Qualität haben), und es zeitlich passt, könnte ich mir vorstellen, dass eine Ruby-Implementierung im Rahmen des Softwaretechnikpraktikums entstehen könnte, wenn Askemos technologisch passend ist. Im Moment ist mir nämlich noch nicht klar, was außerhalb des im Whitepaper

> JFW 08.03: `http://ball.askemos.org/BALLFeatures` ???

beschriebenen wirklich passiert.

JFW am 22.02. zu Fragen der Zertifikate:

> MCO, 21.02.: Gemäß dem Signaturengesetz ist für Signaturen, die einer handschriftlichen Unterschrift gleichberechtigt sind, ein X.509 Zertifikat, das als Wurzel der Zertifikatskette die Bundesnetzagentur hat, notwendig. Diese Organisationsform ist natürlich streng hierarchisch. Wie soll das in dezentrales System integriert werden?

Zu eng gegriffen. Das ist aber ein problematisches Feld.

> MCO 22.02.: Das Gesetz sagt ohnehin erstmal nichts über Technologien, denn auch mit OpenPGP könnte man die Hierachien dort nachbilden.
>
> Die Bundesnetzagentur scheint zur Zeit aber nichts anderes als X.509 zu machen (konnte ich jedenfalls nicht auf der Seite finden).
>
> JFW 08.03: Natürlich nicht. Aber das hat dann wohl eher politische Hintergründe, über die ich nicht weiter spekulieren möchte.

X.509 ist nicht zwingend mit dem SigG. Allerdings sieht da Vieles so aus, als ob man sich neben X.509 nichts vorstellen konnte. Ich verzichte hier mal auf Ausführungen. Da haben wir hier Einiges an Texten rumzuliegen.

Faktisch ist ein Dokument im Askemos *immer* signiert. Und zwar mit einer Qualität, welche die Qualität der X.509-Signatur überschreitet. Wir sprechen von em multivalenter Signatur im Gegensatz zur *monovalenten*, wie sie mit PKI-Verfahren erreicht werden kann.

SigG sinngemäß: „Eine Signatur sind Metadaten, die mit einem Datum verknüft sind; fortgeschrittene Signaturen ist nur schwerlich zu fälschen.“ - Die Metadaten, welche u. a. Erstellungsdatum und Autor enthalten, werden im Askemos von *allen* Hosts mitgeführt, der Hashwert des Datensatzes ist der Objektidentifikator[3]. Damit ist es verdammt hart – gegeben, dass die einzelnen Hosts wirklich unter der Administration ggf. streitender Parteien stehen – diese Daten zu ändern. Härter als mit einer Kopie des privaten Schlüssels des Signierenden.

> MCO 22.02.: Askemos nimmt hier ja SHA-1. Die Bundesnetzagentur sieht das aber nur bis 2015 vor, wenn ich mich recht erinnere.
>
> JFW 08.03: Nicht Askemos, BALL. Askemos kümmert sich nur um „es gibt einen Hash aus ...“
>
> Schlimmer: MD5. An der Stelle. Ist aber nur begrenzt problematisch, denn es handelt sich nicht um binäre Daten, sondern um ein sehr fest definiertes Format. Da ist kein Platz für 56 beliebige Byte.
>
> Für den BLOB nehm’ ich derzeit SHA-256.
>
> MCO 22.02.: Wäre es nicht besser die Identifier in der Form
> Hash-Algorithmus:Hash,
> also sha1:A849640f672ed0df0958abc0712110f3c
> zu schreiben?
>
> JFW 08.03: Definitiv. Hab’ ich nur damals nicht überlegt. Das kann man ja noch weiter entwickeln.

Zu den Begriffen: Multivalent = an mehrere Beweismittel gebunden, Monovalent = an ein Beweismittel. Dazu gibt es ein anwaltliches Gutachten[4].

Eines der größten Probleme für uns mit SigG: dort wird von einem Geheimnis geredet, welches gehütet werden muss und das Gerät nicht verlassen darf ... Hier muss man den Schutz

[3] `http://www.askemos.org/A849640f672ed0df0958abc0712110f3c/OID`
[4] `http://www.softeyes.net/A0e80fdd97a7b6e7af87c5d294f39a96c`

nachweisen. Wir haben aber gar kein solches nötig. Jedenfalls nicht, nachdem wir das Dokument signiert haben, und auch im Signaturvorgang wird es von der Implementation zwar verwendet; könnte aber jederzeit durch andere Verfahren ersetzt werden.

> MCO 22.02.: Beispielsweise Smartcards via PKCS#11 oder OpenPGP.

> JFW 08.03: Mit einem Quorum von Host jeweils in der Hand der beiden Geschäftspartner und dazu einem Notar (so wie im richtigen Leben) kann ich eine Signatur auch prüfen, ohne überhaupt PKI zu benötigen!
>
> Das macht Askemos. Deswegen kann ein Objekt darin länger als signiert gehalten werden, als der ursprünglich verwendete Algorithmus. Die Zeugen bringen die Wahrheit, nicht die Annahme der Schwierigkeit der Manipulation.

> MCO 08.03.: Übrigens wäre das Signieren ohne vertrauenswürdigen Dritten auch unmöglich [10,11].
>
> Wie würde eine Vertragsunterschrift dann laufen? Mein derzeitiges Verständnis ist so:
>
> 1. Der erste Vertragspartner lässt eine Applikation auf seinem Askemos-Knoten den Vertrag unterschreiben (das kann ja mit einem kryptografischen Algorithmus passieren).
> 2. Der Askemos-Knoten des ersten Vertragspartners repliziert die Änderungen auf die Askemos-Knoten des zweiten Vertragspartners und des Notars. Diese bestätigen die Änderung.
> 3. Der zweite Vertragsparter führt ebenfalls Schritte 1 und 2 aus.
>
> Wenn überprüft werden soll, ob der Vertrag von beiden Vertragspartnern unterschrieben wurde, wird gemäß einem Byzantinischen Fehlerprotokoll ein Quorum durchgeführt, ob der Vertrag von beiden unterschrieben wurde (wegen einer $3\,f+1$ Mehrheit würde das in dem Beispiel natürlich nicht gehen, wenn $f \geq 1$ sein soll).

Nunja. SigG wird langsam aktualisiert [12] und schwupps können wir an wichtiger Stelle auf den Kram verzichten. Das dürfte helfen.

> MCO, 07.03.: Eine weitere Sache betrifft das Signieren von Verträgen (im weiteren Sinne), wo ich ja heute nicht weiter wusste und nochmal nachschauen wollte, ob das überhaupt geht. Sie (HGG) hatten ja gesagt, die Meilensteine sollen ein Einvernehmen der Teilnehmer ausdrücken mit dem vorgeschlagenen Identitätsmanagement (ein anderes sehe ich übrigens nicht) würde das bedeuten, dass die Teilnehmer den Meilenstein gleichzeitig signieren. Wie in [10] aufgegriffen und in [11, Lemma 2] bewiesen (ich sehe mich nicht im Stande den Beweis selbst zu verifizieren, aber bei 102 Zitationen sollte der wohl stimmen) kann es dafür keinen deterministischen Algorithmus ohne vertrauenswürdige Partei geben, wodurch das System wieder zentralisiert werden würde. Und angenommen, es gäbe einen nicht deterministischen Algorithmus für das Problem und man könnte nicht deterministische Algorithmen effektiv berechnen, dann wäre auch eigentlich alle

Kryptographie nutzlos und man müsste sich gar nicht erst dem Problem annehmen.

Selbst mit einem Byzantinischem Fehlerprotokoll (mit ACID Eigenschaften), das bei $\frac{2}{3}$ Mehrheit noch einen konsistenten Zustand wiederherstellen kann, kann dann eben nur ein konsistenter Zustand gehalten werden, aber kein multilateraler Vertrag geschlossen werden.

MCO am 22.02. zum CAP-Theorem und Fragen der Verfügbarkeit:

MCO am 21.02.: Bei einer dezentralen Kooperation und Wissenstransfer ist ja sicherlich das gemeinsame Bearbeiten von Dokumenten wichtig. Praktischerweise können diese Änderungen, sofern das Netzwerk fehlertolerant und verfügbar sein soll, nicht konsistent sein (Inhalt des CAP-Theorems).

JFW 22.02.: CAP-Theorem? Ich bin aus der täglichen Wissenschaft wohl etwas raus.

Das CAP-Theorem besagt eben, dass man nur zwei der Eigenschaften Consistency, Availability und Partition Tolerance haben kann. Klassische relationale Datenbanken haben sich für Consistency und Availability entschieden, das heißt wenn ein Datenbankserver ausfällt, funktionieren alle anderen Server bestenfalls noch im Lesemodus, Schreiben ist aber nicht mehr möglich. NoSQL Datenbanken nehmen dann die anderen Möglichkeiten. Amazon Dynamo Klone (Riak, Project Voldemord und so weiter) nehmen Availability und Partition Tolerance, das heißt egal wie viele Server ausfallen, man kann immer schreiben und lesen, aber es ist keine Konsistenz garantiert, das muss dann die Anwendung irgendwie lösen (beispielsweise durch Byzantine Agreement oder andere Mechanismen). Die dritte Kategorie, in die auch Askemos fällt, setzt auf Consistency und Partition Tolerance, beispielsweise durch Paxos (siehe keyspace), so dass wenn durch ein Qourum keine Mehrheit festgestellt werden kann, die Daten eben praktisch nicht verfügbar sind (macht Askemos ja auch so).

JFW 22.02.: Die Verfügbarkeit mal theoretisch zu betrachten ist noch ein spannendes Feld, aber ich habe keine Zeit dafür. Grundsätzlich berechnet sie sich ziemlich einfach: $\frac{2}{3} + 1$ Netzwerkknoten müssen verbunden sein und korrekt arbeiten. Unter der Annahme, dass es keine böswilligen Störungen gibt, rechnet sich das mit der Ausfallwahrscheinlichkeit der Knoten und Netzverbindungen auf irgend eine Funktion der Anzahl der Hosts, die einen bestimmten Prozess bearbeiten.

Das ist ja eigentlich ein Alleinstellungsmerkmal von Askemos. Askemos könnte man ja, wenn ein weiteres Geschäftsmodell haben will, um eine NoSQL-Datenbank erweitern, die eben Konsistenz, Integrität und Verschlüsselung bietet,

JFW 08.03: Konsistenz und Integrität macht die Implementation im „consensus protocol“, zur Verschlüsselung nehmen wir zumindest auf der Leitung nur SSL (plugin).

> MCO 08.03.: Der Rest ließe sich ja über ein verschlüsseltes Dateisystem oder auf Applikationsebene abbilden.
>
> Selbst Verschlüsselung in der Transportschicht haben heute kaum Datenbanken (kann bei REST-Schnittstellen über SSL-Proxy emuliert werden). Z.B. alle Erlang basierten Datenbanken benutzen dessen kaputten Cookie-Mechanismus zur Authentifizierung (selbst wenn man SSL nutzt, bleibt der einem erhalten).

gleichzeitig aber verteilt ist, so dass man eben die Grundfunktionen Read, Write und Compare-and-Swap implementiert,

> JFW 08.03: Hier ist der Unterschied: es gibt read, wobei dabei ggf. eine abschließende Transformation des Views stattfinden kann, und write, wobei i.d.R. eine Transformation aus Daten des Request und internem Zustand in einen neuen Zustand und ausgehende Nachrichten vorgeschaltet wird. Diese beiden Transformationen bestimmen die Logik der Anwendung.
>
> D.h. - und das ist m.E. das eigentliche Alleinstellungsmerkmal - die Anwendung „läuft auf der replizierten Maschine“.
>
> Während also keyspace et al „von der Anwendung Daten erhalten“ (und deswegen der Anwendung *vertrauen müssen - spof*) berechnen die Konten diese im Askemos alle selber (WOT).

> MCO 08.03.: Dann kann man da ja auch gleich Map-Reduce usw. implementieren (nur mit der Ausnahme, dass eben alle Maschinen die Operationen durchführen). Das Berechnen im Knoten geht nur, weil die Anwendungen Teil der virtuellen Maschine sind. Das ist aber nicht bei jeder Anwendung möglich, weil z.B. Altsoftware.

die Daten dann in einer Berkeley DB oder so speichert

> JFW 08.03: Derzeit „oder so“: ein File per BLOB eines per Metadatensatz.

und dann den Benutzer noch Map-Reduce-Funktionen in der Sprache seiner Wahl

> JFW 08.03: Dafür gibt es jene Transformationen. Vom Prinzip habe ich weiter nix gemacht, als in das Abstimmungsprotokoll „user code hooks“ einzubauen. Mappe die Funktion (Transformation) über Input und State, dann sende Daten an den Reduce-Knoten.
>
> Fakt ist: die Sprache ist ziemlich frei wählbar. Scheme hab’ ich eigentlich hauptsächlich genommen, weil es so schön viele Implementationen gibt. (Wobei sich die einzelnen Implementationen intern ungefähr so sehr unterscheiden wie Java von FORTRAN und Python von Perl - will sagen gravierend.)
>
> Die Compiler, die ich zur Implementation verwendet habe, haben ein nettes Foreign Function Interface zu C. Damit ist die Sache der Wahlfreiheit dann nur noch eine Frage des Aufwandes, den man treiben möchte.

> Das Problem ist[5], wenn so eine eingebundene Bibliothek irgendwelche Daten verwendet, die auf einem anderen Knoten anders aussehen, dann geht's nicht weiter.
>
> Das ist theoretisch klar und praktisch kann es zum Problem werden. Stories: Scheme ist absichtlich vage in der Zahlendarstellung, Lesen muss man fast alles, was `write` oder gar `display` erzeugt kann variieren. Und wenn der eine `2e5` schreibt und der andere `20000`, dann ... (nun `2e5` war's nicht, aber in der Datumsarithmetik ist schon Einiges auf diese Weise schief gegangen). Oder gar SQLite. Die Replikation habe ich so gelöst, dass ich das vfs des SQLite verwendet habe und die Blöcke via Askemos verteile. Ergebis[6]: jetzt linke ich statisch gegen eine feste Version, damit das nicht wieder überraschend zuschlägt.

direkt auf dem Server ausführen lässt (siehe auch [8]) und eine REST-Schnittstelle bietet.

> JFW 08.03: Ja, genau so funktioniert das in der Praxis!!!

Zur Zeit gäbe es mit Keyspace da nur einen schlechten Konkurrenten. Gerade für Finanztransaktionen oder ähnlich kritische Daten könnte ich mir vorstellen, dass es großen Zuspruch finden würde, wenn man gute Language-Bindings, eine übersichtliche Website und eine schnelle Installation hätte.

> JFW 08.03: Nun ja. Bislang hat wirklich jede verwendete Sprache, RScheme, Chicken, SQLite, FramerD (früher mal) irgendwann Ärger gemacht, wie in den Archiven zu erkennen ist.

Gerade auch die Stabilität im Sinne von der Entwicklungszeit wäre ja auch noch ein Glaubwürdigkeitskriterium.

Basho, die Firma hinter Riak, und andere verdienen zur Zeit nämlich scheinbar gut Geld damit, einfach nur eine Datenbank, die mehrere Computer umfasst und skaliert, und damit verbundene Dienstleistungen zu verkaufen. Die Sicherheitsansprüche, die Askemos hat, werden damit noch gar nicht abgedeckt.

MCO am 07.03. zu Hardwareanforderungen:

... habe ich noch vergessen zu fragen, wie die Hardwareanforderungen für das Projekt seien sollen. Können Sie (HGG) dazu noch etwas sagen.

> HGG, 07.03.: Üblicher Rechner, sagen wir Desktop PC.

> MCO, 07.03.: Leider weiß ich nicht, was heute so üblich ist. In meinem Bekanntenkreis sind teilweise noch Workstations aus den 90er Jahren in Benutzung. Nennen Sie (HGG) mir lieber absolute Zahlen (wichtig ist hier wahrscheinlich die Größe des RAMs). Auf der FreedomBox Mailingliste war dies kürzlich auch Thema und es wurde auf den „hohen" Speicherverbrauch interpretierter und JIT-kompilierter Programmiersprachen hingewiesen, was sich mit meinen Erfahrungen virtuellen

[5] `http://www.askemos.org/A849640f672ed0df0958abc0712110f3c/WhatIsTime`
[6] `http://ball.askemos.org/SQLite`

Servern < 128 MiB RAM deckt, wo Arbeitsspeicher das Hauptproblem ist. Übliche Leistungsdaten von Consumer-Routern finden sich im OpenWRT Wiki [9] und den Arbeitsspeicher eines Plug-Computers würde ich mit 128 MiB oder 256 MiB einkalkulieren. Das wäre vor allem für die Auswahl der Programmiersprache entscheident.

JFW am 08.03. zu NoSQL-Datenbanken und Askemos:

NoSQL-Datenbank? Nun, ist ein Key-Value-Store eine Solche?

> MCO 08.03.: Ja, alles was eben keine relationale Datenbank ist, vor allem Datenbanken, die andere Eigenschaften aus dem CAP-Theorem wählen.

Im Askemos wird jedes Objekt unter einem Schlüssel (OID) gespeichert. An einem Objekt kann bei der derzeitigen Implementation hängen:

a) ein Datenkörper (BLOB)

a1) Sollte der als xml-parseable erkannt werden, dann greifen ggf. ein paar Optimierungen (Vorgriff auf unten)

b) Links (Key→Value: String→OID) zu anderen Objekten. Was verlinkt ist erhält der Garbage Collector.

c) eine SQL(ite) Datenbank.

> MCO 08.03.: Meines Wissens nach hat b) nur noch Riak. Damit wäre Askemos dann auch eine primitive Graphdatenbank.

Zitierte Quellen

[1] `http://vimeo.com/2723800`

[2] `http://www.salmon-protocol.org/`

[3] `http://www.isoc-ny.org/?p=1338`

[4] `http://video.fosdem.org/2011/maintracks/political.xvid.avi`

[5] `http://www.freedomboxfoundation.org/`

[6] `http://www.askemos.org/`

[7] Matthias-Christian Ott: Seminararbeit zu DRM
`http://bis.informatik.uni-leipzig.de/de/Lehre/Graebe/Wissen`

[8] `http://vimeo.com/6614042`

[9] `http://wiki.openwrt.org/toh/start`

[10] `http://www.semper.org/sirene/publ/BaWa_00AbuseFreeMP.ps.gz`

[11] http://ftp.cs.technion.ac.il/users/wwwb/cgi-bin/tr-get.cgi/1980/CS/CS0175.pdf

[12] http://www.heise.de/newsticker/meldung/Zwang-zur-Rechnungssignatur-faellt-Ende-Juni-1183747.html