# Presentation

## Introduction

Hi everyone,

Thank you for coming. My name is Bjorn Harvold. I am the founder of Health XCEL, an e-health technology startup and I also work for Adobe Systems as a software architect. I am very excited to be here with you today because I think e-health is poised to radically change the world both on a technological level but most importantly on a human level in the way we receive and administer care to a global populace.

I like to compare e-health to Japan during the shogunate period. From the 17th century to the 19th century, Japan had reverted back to swords as the honorable way of fighting because there were no major wars left to be fought. However, Japan went from a culture of sword baring samurais to become an organized militarized army with machine guns over a very short period of time, largely due to the fact that they were defenseless against western military battleships. It was a cultural upheaval but by their embracing of new technology, they were victorious in their war against Russia in the early 20th century.

Health care informatics is playing catch-up to the technological revolution and phenomena of the Internet, mainly, because the Internet generation has had the luxury to start with a clean slate, while health IT has had to fight three decades of legacy systems and political and organizational red tape.

While the Internet learned how to crawl and take its first baby steps, E-health is now in a position where it can actually skip the linear technological progression we’ve witnessed over the last 15 years and jump into the for-front of technical innovation by leveraging all the great accomplishments and lessons learned since the advent of the Internet. It’s no longer about experimenting and seeing what will fly in a dot com economy. We know what needs to be done and we have the tools to do it. I think the next decade will be a defining one for e-health and how we put the proverbial heart back into the Internet.

## Synopsis

### Interoperability

Imagine if you only had to create one profile for all your favorite social networking sites (openID anyone). Now let’s take it a step further. Let’s say you create an account with Facebook and Facebook wants to know all about you and all the data you entered in for MySpace, Hi5, LinkedIn, Plaxo and Last.fm. Not only that, once you’re signed up with Facebook, all the latter sites want all the data you enter in on Facebook and each other’s data as well. As a person tasked with something of this magnitude you would say, “We need a STANDARD!” That’s an excellent start, but a standard for what? Do you want to create a standard that you maintain and impose on these sites and that limits their ability to innovate? Your standard only covers X and Y but the site has implemented Z as well. How will another site that also knows about Z access this data through your standard. If someone is forced into a standard, the standard will many times be misused and extra data will be squeezed into unorthodox places, such as comment fields. SWIFT is a good example of this.

What we need is something a bit more lenient. If Facebook has implemented a feature Z and wants to export it, it should be able to. Likewise, if LinkedIn knows about feature Z it should be able to consume it from Facebook or any other site that offers this service. A standard is only as good as the standards body governing it.

This scenario will be the task for the Health Information Exchanges of the world; dealing with huge disparate data sets from new and legacy systems. We need meta-data to describe our meta-data. Enter RDF (aka Symantic Web, aka Web 3.0).

Paul’s stuff goes here!

Then mention HL7 messaging and Mirth Project

### Platform

#### Lessons learned

We started building a platform for creating and sharing PHR and EHR data back in 2004, only to realize that we weren’t part of the solution but just another part of the problem. We had created another great EHR system that assumed it would take over the world so there was no need to support interoperability. We would be IT! That was a costly albeit invaluable lesson and it made us sit down and rethink how we built software on a very fundamental level. These lessons might be obvious to some. We had to learn it the hard way.

#### Standards, Standards, Standards

* HIPAA & E.U. compliancy
* Internationalization (UTF-8)
* The portable user (openID)
* Security
  + Encryption (AES-256)
  + SSL
  + WS-Security
  + Access Control
* Enterprise Service Bus (ESB)
  + Web Services / SOAP
  + AMF
  + RDF
  + HL7 v2/3
* Medical
  + ICPC-2
  + ICD-10
  + SNOMED
  + LOINC
  + DSM-IV
  + DICOM
  + HL7 EHR
  + HL7 PHR
* iCalendar
* VoIP

#### Putting it all together

Our first goal was to have the ability to support a large amount of users spanning the globe, who could interact with their PHR, EHR and share this information with their doctors. This involved supporting internationalization (i18N) and support for customization of features based on the regulations within the specific countries. We had to integrate translations of medical standards such as ICPC-2 and ICD-10 for all the different countries we wanted to operate in. We also had to be able to support updates to the medical standards themselves.

Our second goal was to become a Health Information Exchange. This is a huge undertaking and involves supporting large corporate entities at several different integration points within our system. In our system, organizations can exist internally or externally, or both. An organization can be a hospital, an insurance company, another HIE, or any other entity for that matter. An organization has users with roles such as “Patient”, “Doctor” and “Administrator” etc. An organization has access to widgets/services, such as an EHR, either through inheritance, licensing or creation. If the organization created the widget, it can license it to other organizations, which could recap its ROI. A widget can also be hosted internally or externally which adds another layer of security complexities.

A user within an organization is the widget end user. A user’s role within the organization or through inheritance (Organizations can have a hierarchy) governs how he interacts with the widgets and what he is able to access.

Our third goal was to easily be able to create new and enhance existing widgets. This brought on the “widget marketplace”. Widgets are applications, such as a PHR or an EHR; functionality that can easily be grouped and without too many dependencies. For every widget there is back-end code and front-end code. We embraced RIA and have standardized the front-end part on the Adobe Flex/AIR platform for its proven technology, great messaging protocol (AMF, Web Services, XML HTTP) and a component-based architecture that enables developers to finally write solid code for the front-end. The widget back-end uses an Enterprise Service Bus to produce several data channels to consume such as AMF, Web Services, HL7 v2/3, XML and RDF XML. The interesting part is that our back-end services run inside a modular micro-container that supports adding and removing of services at runtime without restarting the application server using OSGi. That allows us to easily host internally created applications as well as external ones. (Think Facebook applications on steroids).

We have accomplished all this by first building a platform that can support users, organizations and their widgets. Now we can easily build new widgets, extend the functionality of existing ones and remove deprecated ones. And so can everyone else.

The most important thing to notice here is the combination of right back-end and front-end technologies. Anyone can still build a non-interoperable widget on our platform if they so choose. However, the technology is in place to support the development of HIE quality applications that can be used by external parties as well as internal ones.

### Showcase demo

### Looking forward…

* Cradle-to-Grave application suite
* Large scale data mining capabilities
* Leveraging more of Adobe LCDS for push-messaging, real-time collaboration through Connect and use PDF/H as a way for doctors to make secure copies locally. \*plug\*

# Scenario: Sharing data

## End-user solution

1. Patient wants access to all her medical records but she can’t because they are either paper-based or stored locally on each physician’s computer.
2. Patient creates an account with an HIE and fills out her PHR profile.
3. Patient connects/invites her doctors to join her HIE
4. Patient shares her PHR records with her doctors
5. Doctors enter in her medical records so Patient has access to it

## Enterprise solution

1. Hospital wants access patient’s medical records but can’t because patient’s record is stored at another hospital
2. Both hospitals connect their patient repositories with an HIE (doesn’t have to be the same) and the HIEs make sure the same patient data can be accessed across any hospital connected with either HIE.