The SMART Platform: early experience enabling substitutable applications for electronic health records

Kenneth D Mandl,[1,2] Joshua C Mandel,[1,3] Shawn N Murphy,[4,5] Elmer Victor Bernstam,[6] Rachel L Ramoni,[1,2] David A Kreda,[7] J Michael McCoy,[8] Ben Adida,[9] Isaac S Kohane[1,2]

[1]Children's Hospital Informatics Program at Harvard-MIT Health Sciences and Technology, Boston, Massachusetts, USA
[2]Center for Biomedical Informatics, Harvard Medical School, Boston, Massachusetts, USA
[3]Department of Pediatrics, Harvard Medical School, Boston, Massachusetts, USA
[4]Partners HealthCare Systems, Information Systems, Charlestown, Massachusetts, USA
[5]Laboratory of Computer Science, Massachusetts General Hospital, Boston, Massachusetts, USA
[6]School of Biomedical Informatics, Department of Internal Medicine, The University of Texas Health Science Center, Houston, Texas, USA
[7]SMART Platforms Project, Center for Biomedical Informatics, Harvard Medical School, Boston, Massachusetts, USA
[8]Department of Preventive Medicine, Keck School of Medicine, University of Southern California, Los Angeles, California, USA
[9]Mozilla, Mountain View, California, USA

Correspondence to
Dr Kenneth D Mandl, Children's Hospital Boston, 300 Longwood Avenue, Boston, MA 02115, USA;
kenneth_mandl@harvard.edu

KDM and JCM contributed equally to this work.

Received 3 October 2011
Accepted 12 February 2012
Published Online First 17 March 2012

ABSTRACT

Objective The Substitutable Medical Applications, Reusable Technologies (SMART) Platforms project seeks to develop a health information technology platform with substitutable applications (apps) constructed around core services. The authors believe this is a promising approach to driving down healthcare costs, supporting standards evolution, accommodating differences in care workflow, fostering competition in the market, and accelerating innovation.

Materials and methods The Office of the National Coordinator for Health Information Technology, through the Strategic Health IT Advanced Research Projects (SHARP) Program, funds the project. The SMART team has focused on enabling the property of substitutability through an app programming interface leveraging web standards, presenting predictable data payloads, and abstracting away many details of enterprise health information technology systems. Containers—health information technology systems, such as electronic health records (EHR), personally controlled health records, and health information exchanges that use the SMART app programming interface or a portion of it—marshal data sources and present data simply, reliably, and consistently to apps.

Results The SMART team has completed the first phase of the project (a) defining an app programming interface, (b) developing containers, and (c) producing a set of charter apps that showcase the system capabilities. A focal point of this phase was the SMART Apps Challenge, publicized by the White House, using http://www.challenge.gov website, and generating 15 app submissions with diverse functionality.

Conclusion Key strategic decisions must be made about the most effective market for further disseminating SMART: existing market-leading EHR vendors, new entrants into the EHR market, or other stakeholders such as health information exchanges.

BACKGROUND AND SIGNIFICANCE

The structure, function, and cost of the US healthcare system are under ever-increasing scrutiny. But for the system to adapt to the impact of an aging population, growing expenditures, and a diminishing primary care workforce, innovation in medical practice will have to be supported by information technology (IT) that enables rather than hinders experimentation and innovation. The proprietary electronic health record (EHR) offerings currently on the market tend to be architected monolithically, making modification difficult for hospitals and physician practices. In 2009, we proposed that EHRs instead should be designed as platforms supporting a selection of 'substitutable' modular third party applications (apps).[1]

We drew an analogy with mobile phone platforms such as iPhone and Android, which lower the barrier to app development by providing a software platform with a published interface to a set of core services such as camera, address book, geo-location, and cell and wireless networks. The platform functionally separates the core system from the apps, and the apps are substitutable. Thus, for example, a consumer can download a calendar reminder system, reject it, and replace it with another one. Through substitutable apps, the iPhone and Android platforms now support myriad capabilities that the original platform designers never imagined.

A platform with substitutable apps constructed around core services is a promising approach to driving down healthcare technology costs, supporting standards evolution, accommodating differences in care workflow, fostering competition in the market, and accelerating innovation. With the cost of switching kept low, the platform enables a physician using an EHR, a Chief Information Officer running a hospital IT infrastructure, or a patient using a personally controlled health record (PCHR) to readily discard an underperforming app and install a better one. Competition on quality, cost, and usability is enabled, and the pace of innovation increases. This model stands in stark contrast with the largely monolithic and slow-to-evolve health information systems that have been designed and implemented to date, where one size has to fit most providers, customization is arduous and expensive, and only the few established EHR vendor developers can innovate.

The principle of substitutability is the central focus of the Substitutable Medical Applications, Reusable Technologies (SMART) Platforms project. By defining an app programming interface (API) that consistently presents well-specified data, we seek to (a) enable purchasers, users, and administrators of platform-based systems to be able to install and subsequently substitute apps from different vendors without software programming and (b) create a broad market for app developers

across multiple systems, including EHRs, PCHRs, and health information exchanges. The SMART Platforms project is funded by the Office of the National Coordinator for Health Information Technology as a part of the Strategic Health IT Advanced Research Projects (SHARP) Program.[2] Here we report on challenges and successes faced during the first year of the project, and describe the architecture of the system.

MATERIALS AND METHODS

We present a fictional scenario to convey the need for SMART Platforms.

A company—Medtastic—has designed an elegant medications-management app which needs access to a current medications list from the EHR. While the user-interface design is considered exemplary, and it market tests extraordinarily well with end users, the company finds that the necessity of having a long and involved sales cycle at each possible install location is draining their venture capital funds. They had hoped to exercise the 80:20 rule and focus on integration with the top five vendors, but: (1) only two of those are willing to entertain the proposition; (2) the technical teams of those vendors are not enthusiastic about prioritizing the work; (3) the Medtastic technical team has found that there is such variation across different instances of each brand of EHR—because of versioning and extensive local customization—that installation of their app will not be turnkey in any way.

The first year of the SMART Platforms project has focused on defining an API that would enable a company like Medtastic to succeed by providing apps with a common interface for working with health data. The interface must be a simplified and semantically precise abstraction of a medical record with well-structured, normalized data elements that app developers can understand. The approach is to harness standards and technologies behind successful web APIs and to employ open standards and specifications wherever possible.

SMART provides specifications allowing apps to run against existing health IT systems. Our intent is to specify, in detail, everything app developers need in order to create apps rapidly and independently of the SMART team. The scope of the specification encompasses user-interface integration, authentication, authorization, API access, and data payloads.

Definitions

We define 'SMART containers' as health IT systems, such as EHRs, PCHRs, and health information exchanges, that have implemented the SMART API or a portion of it. Containers marshal 'data sources' and present them consistently across the SMART API. 'SMART applications' consume the API and are substitutable.

Developer focus

The SMART architecture aims to reduce barriers that app developers face in building apps on health IT systems. By leveraging web standards, presenting predictable data payloads, and abstracting away many details of enterprise health IT systems, SMART allows app developers to focus on core tasks. In figure 1, a sample app called 'Got Statins?' illustrates the point. 'Got Statins?' is a complete 'hello world'-style SMART app that obtains a patient's medication list, iterates through each entry, and makes a simple determination. The entire app fits in 50 lines of HTML and JavaScript.

Containers

Any SMART container must present normalized clinical data to SMART apps in a reliable and consistent fashion, abstracting away details of the underlying health IT infrastructure. Thus, the same app could run inside our public reference container, an EHR, a PCHR, or a health information exchange.

To enable such substitutability, the SMART architecture imposes a substantial burden of normalization on any container, allowing apps to know upfront what data to expect and what each data element means. An important implication is that

```
<!DOCTYPE html>
<html>
<head><title>Got Statins?</title></head>
<body>
<h1>Got Statins?</h1>
<a id="TheAnswer">...</a>
<script src="http://sample-apps.smartplatforms.org/framework/smart/scripts/smart-api-page.js"></script>
<script>

    SMART.MEDS_get_all(function(meds) {
        var med_list = meds.where("?m rdf:type sp:Medication")
                           .where("?m sp:drugName ?n")
                           .where("?n dcterms:title ?drugname");

        var answer = false;
        for (var i = 0; i < med_list.length; i++) {
            if (is_a_statin(med_list[i].drugname.value)) {
                answer = true;
            }
        }
        document.getElementById("TheAnswer").innerHTML =answer ? "Yes." : "No.";
    });

    var is_a_statin = function(drug) {
        if (drug.match(/statin$/i)) return true;
        if (drug.match(/Advicor/i)) return true;
        // ... additional drug names cut for clarity
        if (drug.match(/Zocor/i)) return true;
        return false;
    }
</script>
</body>
</html>
```

Figure 1 'Got Statins?' is a complete SMART application (app) in 50 lines of HTML and JavaScript. The app includes an external SMART JavaScript library, then makes a call to obtain all medications for the in-context patient record. A list of drug names is created, and a loop checks each drug name against a list of known statin drugs. 'Got Statins?' is designed merely to illustrate the SMART API. A more robust approach would incorporate drug class data from a reference source such as NDF-RT.[3]

Figure 2 Example of a SMART data payload.

```
<?xml version="1.0" encoding="utf-8"?>

<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:sp="http://smartplatforms.org/terms#">

    <sp:Problem>
        <sp:onset>2007-06-12</sp:onset>
        <sp:problemName>
            <sp:CodedValue>
                <sp:code rdf:resource="http://www.ihtsdo.org/snomed-ct/concepts/37796009"/>
                <dcterms:title>Migraine (disorder)</dcterms:title>
            </sp:CodedValue>
        </sp:problemName>
    </sp:Problem>

</rdf:RDF>
```

some containers may need to reshape underlying data to support the SMART API.

The container's job is best understood through an example. Health IT systems take a variety of approaches to representing medical problems or diagnoses. Some represent a problem as an entity with a start date and a resolution date, persisting over time; other systems represent problems as a series of observations over time. To allow substitutability, SMART takes the stance that a problem has an onset date, resolution date, and a SNOMED CT disorder code. The container must reshape existing data, and the details of transformation will vary case-by-case, but we expect that, in many instances, simple transformations can be computed on-the-fly. For example, an EHR may maintain a discrete list of dates and ICD9 codes indicating that a patient experienced migraine headaches. To format the appropriate SMART problem list, the EHR would need to bundle up related ICD9 diagnoses into a SMART problem element with an appropriate onset date, resolution date (which may be null for ongoing problems), and a SNOMED CT disorder code for migraine. Figure 2 provides an example.

We emphasize that reshaping data in such ways may not always be a straightforward or even well-defined task. Some systems may be unable to implement the complete SMART API—for example, if underlying data models are too divergent from the SMART specifications. Such limitations arise because substitutability is a high bar.

Data models, coding, and normalization

In practical terms, substitutability is a high bar because it imposes a need for 'semantic interoperability' between apps and containers. To achieve this interoperability, SMART defines a highly normalized abstraction of a medical record that is designed to be intuitive and easy to learn. Key features include: (a) developers work directly with concrete types such as allergy, medication, or problem, not abstract types such as entity, actor, or role; (b) the SMART specification is 'opinionated,' making upfront choices about how data are represented so that developers know what to expect—for instance, every problem in a SMART record is associated with a SNOMED CT[4] disorder code; (c) SMART defines a limited set of broadly applicable data types, rather than permitting a proliferation of interface-specific definitions.

We take this approach because poorly normalized data require developers and apps to expend tremendous effort just 'making sense' of the payloads they receive. For example, consider an app that obtains a list of medications from a container to assess for polypharmacy. If some of the medications are coded with National Drug Codes (NDC),[5] others with RxNorm[6] codes, and still others with codes from a local dictionary, the app must first go through the considerable effort of remapping these codes into some common vocabulary, reducing the integrity of the medication list while creating additional work for an app developer.

Concretely, SMART represents a medical record as a series of statements or 'triples' according to the Resource Description Framework.[7] Multiple statements together form a 'graph' of patient data. The meaning of each element in this graph is precisely defined by the SMART ontology in OWL2 DL,[8] a web standard for representing knowledge based on formal description logics. Thus, each SMART medical record has an explicit, formally defined meaning—but app developers performing simple operations with SMART medical records do not need any deep understanding of OWL2.

Importantly, SMART data models are still a work in progress, and they are limited in scope: the intention is not to provide a detailed model for every possible aspect of a patient's medical history. Rather, at this stage, SMART attempts to provide highly consistent views for the most common data elements. The SMART data models are freely available.[9]

App programming interface

The current version of the SMART API provides a read-only view of the patient record. An app can access the API through two distinct routes: SMART Connect and SMART REST. SMART Connect is the browser-based JavaScript interface illustrated in the 'Got Statins?' app (figure 1), designed to offer a lightweight approach for developers building apps with rich client-side functionality. SMART REST provides a representation state transfer interface to the medical record, allowing an app's back-end component to communicate directly with a container.

Regardless of whether data are accessed through SMART Connect or SMART REST, the set of API calls and payload formats are the same. Each patient, and each clinical statement about that patient, is represented as a resource with a URI. The structure of these URIs is specified with respect to a container's 'base URI.' For example, a public SMART sandbox container is hosted with the base URI: http://sandbox-api.smartplatforms.org/. (The user interface associated with this public sandbox is presented at http://sandbox.smartplatforms.org.) An individual patient in the sandbox might be identified by: <http://sandbox-api.smartplatforms.org/records/123>. A single medication on that patient's medication list is represented as: <http://sandbox-api.smartplatforms.org/records/123/medications/456>.

Authentication and authorization

Installing a SMART app on a container is a statement of trust in the app about its functions and handling of privacy and security of data. In this respect, installing a SMART app is no different than installing any other clinical IT system. The SMART architecture is flexible, allowing apps to be hosted within an institutional intranet, or remotely 'in the cloud'. Existing health IT

systems have heterogeneous methods of authentication and authorization. To allow a single app to run against heterogeneous systems, the SMART specification standardizes the authentication of API access but allows an underlying container to apply its own authorization mechanisms. This means that, when an app makes an API call, the container can be certain that the call has not been forged. But the container retains full power to determine whether the app is, in fact, authorized to retrieve the data it has requested. Two authentication schemes are employed, one for SMART Connect and one for SMART REST calls.

Since SMART Connect calls happen in-browser, they are by definition made in the context of an established user session. In this case, no additional authentication is explicitly required by the SMART specification. SMART REST calls, on the other hand, are made server-to-server and are signed using a token and shared secret via the OAuth 1.0a protocol—an open protocol for secure API authentication.[10] The scope of access tokens is kept narrow so that, for instance, an app requires a separate access token for each patient record it wishes to query. Figure 3 illustrates the SMART API functionality.

User-interface apps

SMART user-interface apps are user-facing, browser-based apps written as HTML5 web apps, which are platform-independent and run in all modern web browsers on desktops or mobile devices. They integrate into an existing health IT system via the HTML inline-frame element. If an existing health IT system is web-based, SMART apps may be integrated by adding an IFRAME to an existing web interface. If an existing health IT system runs a 'thick client,' inclusion of a SMART app may require augmenting the client with a 'web view' widget or launching a separate browser instance.

Background apps

Background apps employ the same API calls as user-interface apps but perform data processing and analysis without the need for user interaction. Since these background apps do not run in the context of a web-browsing session, they must access patient data via SMART REST, not SMART Connect. In addition, SMART provides a preliminary interface for background apps to loop through the patients in a container one-by-one, fetching and processing data for each. Hence, background apps can be used for batch-processing tasks such as executing clinical rules or computing quality measures. Background apps might also be used to simulate population-level queries by serially evaluating one patient at a time, although this approach is unlikely to scale to large datasets.

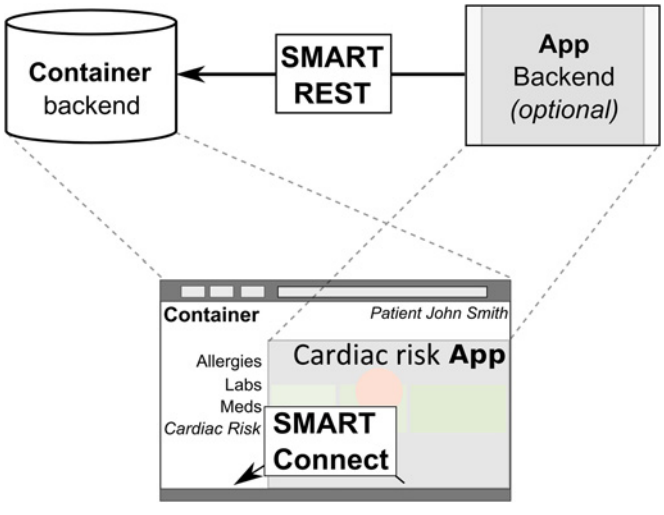


Figure 3 At the center, a web browser window presents two principal components: (1) a white border region belongs to the SMART container, displaying a list of available applications (apps) as well as patient context; (2) a larger gray block belongs to a single 'Cardiac Risk' app. As illustrated, the 'Cardiac Risk' app can request data from the container directly inside the browser via SMART Connect; an app with a back-end component may request data with a server-to-server call via SMART REST. In either case, the SMART-enabled electronic medical record or personally controlled health record responds with the same SMART RDF data payload.

RESULTS

In addition to development of the API, the first-year efforts centered on SMART-enabling select containers and creation of a small set of charter apps to demonstrate functionality and inform system design.

Apps challenge

A focal point of the first year was the SMART Apps Challenge. Sponsored by the Office of the National Coordinator for Health Information Technology, we launched a developer-focused challenge on the Administration's Challenge.gov website.[11] The contest, offering a US$5000 prize and judged by a blue ribbon panel,[12] was announced by the President's Chief Technology Officer, Aneesh Chopra, at the 2010 mHealth Conference during his shared Keynote with Bill Gates,[13] and on his White House blog.[14]

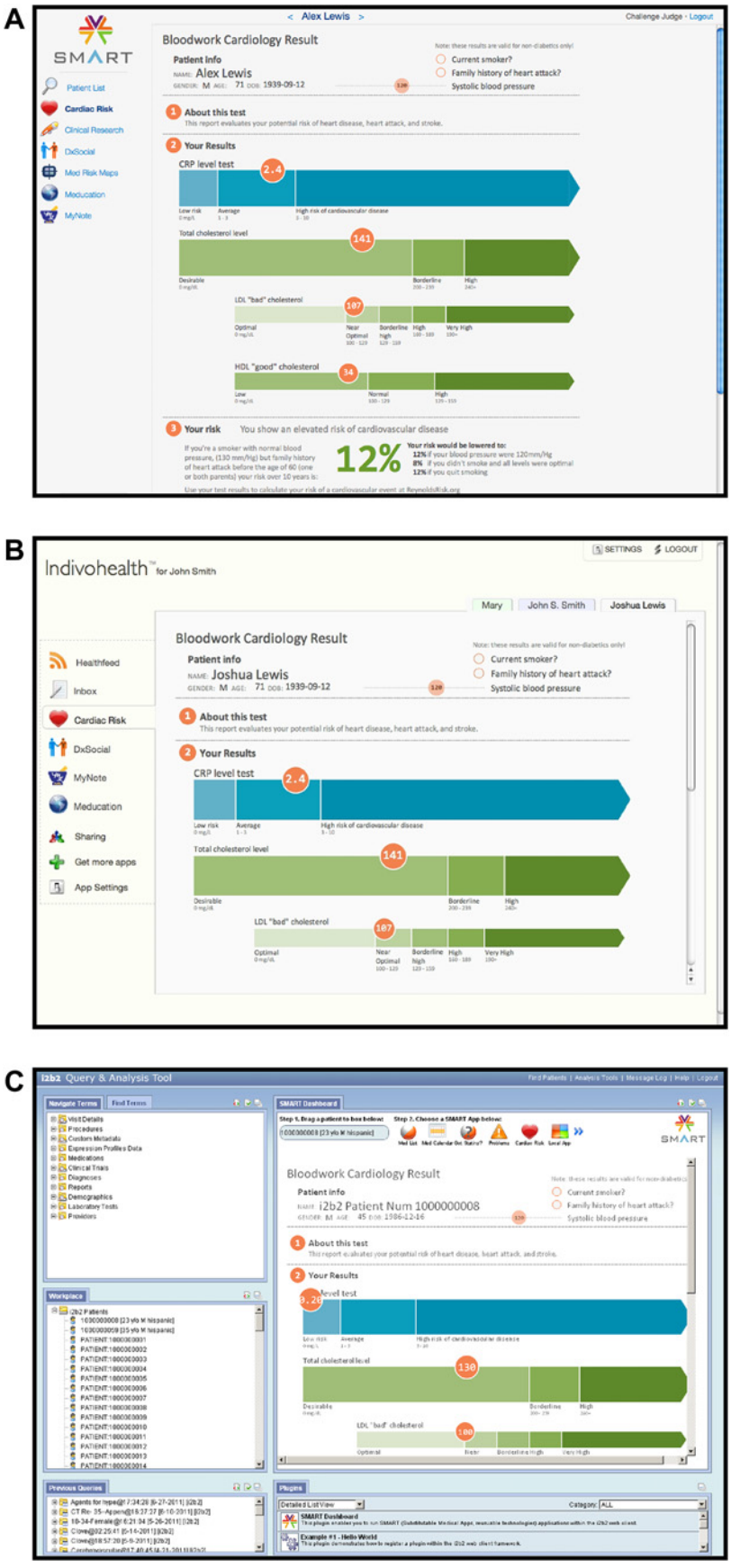
The SMART team created a reference container deployed in a 'sandbox' environment for challenge entrants populated with a hybrid of anonymized and synthesized clinical data for 50 sample patients, published for open use and redistribution.[9] The challenge was to build a SMART app that provides value to patients, providers, or researchers, using patient-level data delivered through the SMART API.

There were 15 entries with functionality such as generating multi-lingual patient-facing medication instructions, or providing a public health dashboard that links EHRs with immunization registry and syndromic surveillance data. The winner was the Meducation app by Polyglot. Meducation pulls the patient's medication list across the API and joins it to simplified patient-friendly instructions for the individual medications in 12 languages. Polyglot is a small company whose business model would clearly be advanced by SMART API access to the wider IT infrastructure. The challenge achieved several objectives: (1) forcing an early (10 months into the project) release of the API and sandbox; (2) necessitating early development of charter apps (see below); (3) requiring extensive documentation and creation of website and promotional materials; (4) widely publicizing the platform; (5) for only a US$5000 investment, generating 15 intriguing apps; (6) demonstrating a key property of SMART—the challenge apps were able to run, unmodified, across multiple platforms.

Building SMART containers

A three-step process converts an existing health IT system into a SMART container: exposing data through the SMART API, providing a place for apps within or alongside the existing health IT system's user interface, and implementing appropriate authentication. In addition to the SMART reference container, we are SMART-enabling three open source systems: the Indivo PCHR[15–17]; the Informatics for Integrating Biology and the Bedside analytic platform—i2b2[18 19]; and the Open Medical Record System (OpenMRS).[20 21]

Using published Cerner APIs, we are working to implement a portion of the SMART API on top of the Children's Hospital



Boston Cerner Millennium installation, with a goal of deploying the 'Pediatric Blood Pressure' app described below. Our initial approach involves loose coupling on the front-end, allowing clinicians to click a link within Cerner's PowerChart patient view to launch a SMART app in a new browser window. On the back-end, we have built a thin translation layer that exposes a few key

Figure 4 The 'Cardiac Risk' app, based on David McCandless' design (released under a Creative Commons license) shown running unmodified in (A) the SMART reference container, (B) the Indivo PCHR, and (C) the i2b2 analytic platform.

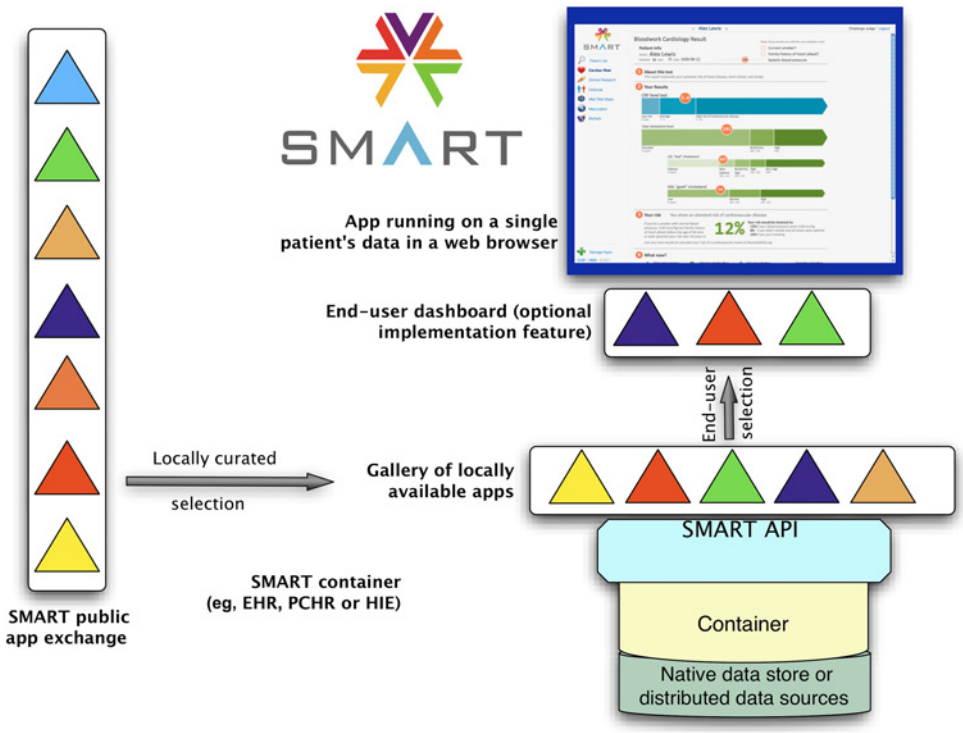


Figure 5 The envisioned SMART ecosystem. Health IT systems, such as electronic health records (EHR), personal health records (PCHR), and health information exchanges (HIE), that use the SMART application (app) programming interface (API) or a portion of it marshal data sources and present data simply, reliably, and consistently to apps. Apps are made available, under a business model to be specified, in one or more app exchanges. Administrators of individual SMART container installations (eg, vendors, chief information officers, practice leaders) can choose which apps are made available to their end users. End users can create a 'dashboard' of apps that they use in their workflow.

data elements including patient demographics, encounters, and vital signs, by issuing queries against the Cerner published 'Millennium Objects' SOAP interface, and translating results to SMART RDF on-the-fly. Early experience indicates that on-the-fly translation is feasible, but pre-fetching and caching results may be required for acceptable performance on larger datasets. The source code for the translation layer, including hooks into the SMART Reference EHR, is available from https://github.com/chb/smart_grails_proxy. In addition, Microsoft has produced a proof-of-concept SMART-enabled version of their HealthVault PCHR.

Charter apps

We have created several charter apps, briefly described here.

The 'Cardiac Risk' app is based on a conceptualization by David McCandless of a consumer-friendly presentation of cardiac risk based on laboratory information about cholesterol, demographics, and risk factors. The image appeared in *Wired* magazine[22] and is posted on McCandless' website for use under the Creative Commons license.[23] The SMART team, programming against the SMART reference container, created a functional, interactive app faithful to McCandless' aspiration. Figure 4 shows the app running in the SMART reference container, on the Indivo PCHR and the i2b2 analytic platform.

The 'Adherence' app accepts medication fulfillment histories, displays gaps in medication possession, and predicts future non-adherence.[24] The 'Blood Pressure' app is the first SMART app designed for a defined population of real-world clinical users; developed according to clinician-derived specifications at the Children's Hospital Boston, it connects to a SMART-enabled Cerner EHR and monitors trends in blood pressure, flagging hypertension in pediatric patients by applying NIH guidelines that incorporate a child's age, gender, and height.

Upcoming research areas and strategic decisions

App distribution and access

While the full business model for distribution of SMART apps is still emerging, a few principles are clear. First, while the SMART API will remain open source and available under the Apache 2.0 license, there is no such obligation for externally developed apps, which may be open or closed source code. Second, we are not committed to a single iTunes-like app store, but rather envision that there may be one or several app exchanges. Assessing quality of apps will be a multi-input process, no doubt involving local opinion leaders, professional organizations, and possibly certification bodies. Administrators of individual SMART container installations (eg, vendors, chief information officers, practice leaders) can choose which apps are made available to their end users (figure 5). End users can create a 'dashboard' of apps that they use in their workflow. Security concerns may inform deployment decisions at a given site. For example, a hospital may want to install all SMART apps on locally hosted servers within the hospital intranet to help ensure the proper treatment of protected health information. By contrast, a PCHR may have much greater tolerance for running cloud-hosted apps, allowing individual patients to determine with which apps they wish to share data.

Write API

To date, the SMART API provides a read-only view of the patient record. We have constrained the API in this fashion in order to lower the barriers for existing health IT systems to adopt SMART and benefit from a growing community of apps. Allowing apps to write data back to a container considerably increases the complexity of implementation. We plan to add write capabilities gradually, as support for the read-only API grows.

Standards

With a goal of maintaining an open stack, we use web standards extensively (eg, HTML, JavaScript,[25] OAuth, RDF) and medical standards for coding systems (eg, RxNorm, LOINC,[25] SNOMED). The space of open clinical data models is underdeveloped. There is no widely implemented, developer-friendly open standard, for example, for what a medication, fulfillment,

or blood pressure looks like. Hence, we have been defining a simple set of abstractions, which we are refining over time.

We believe that the SMART standards and a modern web-based app platform are strong initial steps toward a universal exchange language suggested by the Presidential Council of Advisors on Science and Technology in their report to the President on Health Information Technology.[26] The report recognizes lack of interoperability as a major barrier to health IT adoption and recommends abandoning traditional health data standards and allowing the market to drive semantic harmonization. SMART puts forth a clear model for common health data, starting with medications and fulfillments, problems, allergies, and simple blood laboratory results. The model will continue to evolve, especially over the next year, as the apps market requirements become clear. We have not abandoned current standards, and rely heavily on SNOMED, RxNorm, and LOINC. Our modeling approach allows expression of atomic data points such as single allergies, problems, or a single blood laboratory value, exactly as recommended by the Presidential Council of Advisors on Science and Technology. Further, it allows for the addition of additional semantics over time, without the existing data being affected, or the existing software that relies on these data.

CONCLUSION

Over a period of less than 14 months, we have gone from the point of defining an approach to building a working API which has been used as the basis of several EHR apps developed by groups with no relationship to the SMART team. Nonetheless, we recognize several important challenges and questions. First, where is the most effective market for further disseminating SMART? Is it with the existing market-leading EHR vendors, with the new entrants into the EHR market, or with other stakeholders such as health information exchanges and accountable care organizations? At this time it remains unclear where the major adoption is going to occur. We are currently experimenting with three models of technology diffusion: (1) integration with legacy systems (as described above in our Cerner integration); (2) deployment of apps in health information exchanges; (3) deployment of apps running on a parallel platform (i2b2) with real-time extraction, transformation, and loading from the EHR.

Acknowledgments We are indebted to Dr Charles Friedman and Mr Wil Yu for their dedicated and enabling programmatic support of the project. We are grateful to Aneesh Chopra and Todd Park for their extraordinary commitment to promoting the SMART Platforms project and concept.

Contributors KDM wrote the first and final draft of the manuscript and is the co-PI of SMART. JCM co-wrote the manuscript and is lead architect of SMART. SNM provided valuable sections of the manuscript and is a core member of the SMART leadership team. EVB provided valuable comments on the manuscript and is a core member of the SMART leadership team. RBR provided valuable comments on the manuscript and is the Executive Director of SMART. DAK is a core member of the SMART leadership team, and contributed substantially to the SMART architecture. JMM is a core member of the SMART leadership team, contributed substantially to the SMART architecture and provided substantial input to the manuscript. BA is a core member of the SMART leadership team, contributed substantially to the SMART architecture, and provided valuable input to the manuscript. ISK co-wrote the manuscript and is the PI of SMART.

Funding This work was funded by the Strategic Health IT Advanced Research Projects Award 90TR000101 from the Office of the National Coordinator of Health Information Technology.

Competing interests None.

Provenance and peer review Not commissioned; externally peer reviewed.

REFERENCES

1. **Mandl KD,** Kohane IS. No small change for the health information economy. *N Engl J Med* 2009;**360**:1278–81.
2. **Office of the National Coordinator for Health Information Technology.** *Strategic Health IT Advanced Research Projects (SHARP) Program*. http://healthit.hhs.gov/portal/server.pt/community/healthit_hhs_gov__sharp_program/
3. **U.S. National Library of Medicine.** *2011AA National Drug File—Reference Terminology Source Information*. 2011. http://www.nlm.nih.gov/research/umls/sourcereleasedocs/current/NDFRT/
4. **U.S. National Library of Medicine.** *SNOMED Clinical Terms (SNOMED CT)*. http://www.nlm.nih.gov/research/umls/Snomed/snomed_main.html
5. **U.S,** Food and Drug Adminstration. *National Drug Code Directory*. http://www.fda.gov/Drugs/InformationOnDrugs/ucm142438.htm
6. **U.S. National Library of Medicine.** *RxNorm*. http://www.nlm.nih.gov/research/umls/rxnorm/
7. **W3C.** *Resource Description Framework (RDF): Primer*. http://www.w3.org/TR/rdf-primer/ (accessed 2 Sep 2011).
8. **W3C.** *OWL 2 Web Ontology Language: Primer*. http://www.w3.org/TR/owl2-primer/ (accessed 2 Sep 2011).
9. **SMART Platforms Team.** *SMART Development Site*. http://www.smartplatforms.org/development
10. **OAuth Community.** *OAuth*. http://oauth.net/
11. **Challenge. Gov.** *SMART Apps for Health Challenge*. http://challenge.gov/HHS/134-smart-apps-for-health
12. **SMART Platforms Team.** *SMART Platforms Challenge*. http://www.smartplatforms.org/challenge/
13. **Foundation for the National Institutes of Health.** *2010 mHealth Summit: Meeting 21st Century Health Goals Through Mobile Technology Research & Innovation*. http://www.fnih.org/events/2010-mhealth-summit
14. **Chopra A.** *SMART Prize for Patients, Physicians, and Researchers*. http://www.whitehouse.gov/blog/2011/03/10/smart-prize-patients-physicians-and-researchers
15. **Mandl KD,** Simons WW, Crawford WC, *et al*. Indivo: a personally controlled health record for health information exchange and communication. *BMC Med Inform Decis Mak* 2007;**7**:25.
16. **Mandl KD,** Kohane IS. Tectonic shifts in the health information economy. *N Engl J Med* 2008;**358**:1732–7.
17. **Adida B,** Sanyal A, Zabak S, *et al*. Indivo x: developing a fully substitutable personally controlled health record platform. *AMIA Annu Symp Proc* 2010;**2010**:6–10.
18. **Murphy SN,** Weber G, Mendis M, *et al*. Serving the enterprise and beyond with informatics for integrating biology and the bedside (i2b2). *J Am Med Inform Assoc* 2010;**17**:124–30.
19. **Murphy SN,** Mendis M, Hackett K, *et al*. Clinical research chart from informatics for integrating biology and the bedside. *AMIA Annu Symp Proc* 2007:548–52.
20. **Wolfe BA,** Mamlin BW, Biondich PG, *et al*. The OpenMRS system: collaborating toward an open source EMR for developing countries. *AMIA Annu Symp Proc* 2006:1146.
21. **Mamlin BW,** Biondich PG, Wolfe BA, *et al*. Cooking up an open source EMR for developing countries: openMRS—a recipe for successful collaboration. *AMIA Annu Symp Proc* 2006:529–33.
22. **Leckart S.** *The Blood Test Gets a Makeover*. Wired, 2010.
23. **McCandless D.** Visualizing Blood Tests. *Information is Beautiful: Ideas, Issues, Knowledge, Data—Visualized!* http://www.informationisbeautiful.net/2010/visualizing-bloodtests/
24. **Jonikas MA,** Mandl KD. Surveillance of medication use: early identification of poor adherence. *J Am Med Inform Assoc*. Published Online First: 19 November 2011. doi:10.1136/amiajnl-2011-000416
25. **The Regenstrief Institute.** *Logical Observation Identifiers Names and Codes (LOINC®)*. http://loinc.org/
26. **Executive Office of the President,** President's Council of Advisors on Science and technology. *Report to the President: Realizing the Full Potential of Health Information Technology to Improve Healthcare for Americans: the Path Forward*. 2010. http://www.whitehouse.gov/sites/default/files/microsites/ostp/pcast-health-it-report.pdf