Agile Clinical Decision Support Development

VAISHNAVI KANNAN, MS University of Texas Southwestern Health System
DUWAYNE WILLETT, MD, MS, University of Texas Southwestern Health System

Designing effective Clinical Decision Support (CDS) tools in an Electronic Health Record (EHR) can prove challenging due to complex clinical scenarios and changing requirements; application of agile principles and practices proves helpful in the development and delivery of effective CDS tools for use in clinical care. By applying agile project management methods such as short time-boxed iterations with demos of working software at end-iteration, and agile modeling techniques, we were able to deliver over 100 new CDS and EHR data collection tools in calendar year 2015, a marked increase over our prior capability.

1. INTRODUCTION

Healthcare initially may seem an odd setting in which to explore agile principles and practices, with its overlay of extensive regulation and a software environment dominated by large vendor commercial applications. However, these healthcare applications—particularly electronic health records (EHRs)—are often highly configurable, with the opportunity to create custom data structures, business logic, and end-user interface forms and screens, so that custom configuration work shares important properties with new product development. Frankly, we had struggled with the quality of some of our custom configuration work, which rarely survived rollout into production without needing significant re-work. Attempts to add further "sign-offs" up-front didn't seem to improve our ability to hit the mark. Costly delays often ensued. In this setting, we were asked to take on a project to deliver large numbers of custom EHR configuration items for a widely diverse set of clinical specialties and users, to support measurement of ambulatory quality of care. We knew we had to work differently to have a chance to succeed. Fortunately, we had colleagues with experience and success employing agile methods in our Data Warehouse development. So, we turned to these principles and practices to address our challenge.

2. BACKGROUND

A. What is Clinical Decision Support, and why is it important?

Clinical decision support (CDS) refers to a variety of "tools" employed to help guide and support clinicians in making optimal decisions, often in uncertain situations. The types of tools can include: alerts of various types (medication interaction alerts, advisories to promote best practices), order sets, order questions, reminders, feedback reports, and reference materials. As our healthcare system moves more to "value-based care", clinical outcomes and the costs incurred to achieve those outcomes are increasingly measured and incentivized. Clinicians' decisions still drive the majority of costs, and impact the quality of patient outcomes. Thus clinical decision support helping make "the right thing the easy thing" to do for the patient can play a key role in individualizing and optimizing care delivery.

B. Challenges in creating CDS tools

A number of challenges arise in creating CDS tools. Often multiple clinical stakeholders are involved—physicians of different specialties and/or levels of training, nurses, technologists, other staff. Even similar clinical stakeholders can have differing understanding of the intended solution. A robust clinical decision support solution can have multiple parts and rules that interact with each other in complex ways; in addition,

Vaishnavi Kannan, MS, University of Texas Southwestern Health System, email: Vaishnavi.Kannan@UTSouthwestern.edu
Duwayne Willett, MD, MS, University of Texas Southwestern Health System, email: duwayne.willett@UTSouthwestern.edu

the clinical environment into which these tools are deployed is inherently complex. Accordingly, release of CDS tools into the production EHR often yields new understanding of desired CDS tool behavior.

C. Issues encountered with CDS tools

Not surprisingly then, issues continue to be encountered with CDS tools. An important issue is so-called "alert fatigue", where clinicians become numbed by a high volume of alerts, and progressively "tune out" even important ones. In a recent study in the Boston area, the "override" rate for medication alerts increased over a 10-year period from 83% to 88% (Topax, Seger, et al, JAMIA 17 Nov 2015); even alerts about possible severe, life-threatening drug reactions or interactions were ignored about 3/4 of the time. A second important issue is the existence of defects in the CDS tools themselves. In a report from 28 Mar 2016, Wright et al state "CDS malfunctions occur commonly and often go undetected. Better methods are needed to prevent and detect these malfunctions." A third issue is frequent lack of confirmation that CDS tools are working as intended, or that they are achieving the intended results.

D. Specialty Registry project (Ambulatory Quality Outcomes project)

At the end of 2014, we were asked to engage in an Ambulatory Quality Outcomes (AQO) project, as our leadership wanted to make concrete their assertion that in the current day we would need to prove the quality of care we deliver, not just assert it as an academic medical center. Our directive was to work with each of 30+ specialties to define the health condition(s) each was most interested in, then jointly select measures to prove we are providing good care (process measures) for each condition, and ultimately measure patient outcomes. Each such subpopulation would populate a registry of patients. Most registries would require CDS and/or other EHR data collection tools, as well as reports of calculated measures.

3. OUR STORY

A. Problems

As we began to tackle this project at the start of 2015, we faced significant problems. First, our CDS development track record historically was poor: the turnaround time from start of a project to delivery to production was long (months), and still our CDS tools commonly exhibited unintended behavior (defects) when finally released to production. Second, we needed to build large numbers of high-quality CDS tools to support the development of multiple specialty patient registries across the breadth of our multispecialty practice. Third, the timeline was so aggressive, with the need for working with multiple specialties simultaneously, that we couldn't rely on the small CDS specialist team alone—we needed all hands on deck to be building CDS tools. Finally, given the multiple developers working together, we needed a facile way to communicate about what we were building, why we were building it, and how it was being designed and constructed.

B. What we did:

i. Adopted agile project management principles and methods

Although we had used agile methods on our data warehousing team for several years and on our EHR team for the past year, we had to teach and demonstrate to the broader team (including people from clinical operations, quality, analytics, and information resources) the value of an iterative, incremental delivery approach. Some challenges we faced included a tendency to take a waterfall approach to iterations (an analysis iteration, then a design iteration, then one or more build iterations, then a testing and training iteration). Additionally, as it became clear we had changing/evolving requirements, the initial tendency was to add yet more up-front detailed design in advance of the actual iteration, rather than focus on getting Analysis-Design-Build-Test completed within an iteration, and learning from each cycle. We found we had to re-emphasize the value of delivering working product at the end of each iteration. Scheduling end-of-sprint demos every 2 weeks for the whole team was helpful in establishing a rhythm of delivery, and conveying the benefit of time-boxed iterations. Speaking of demos, though, another challenge was getting actual customers to attend the demos, as they were often busy clinicians. We had one team serve as the 'voice of the customer' at the demo, and meet with the real customer whenever they could. While this proved helpful, the lack of interactive feedback real-time during iteration-end demos continued to be a challenge for us.

ii. Adopted agile modeling principles and methods

With a large, diverse stakeholder group, and a large project team with varied backgrounds, we noticed we were re-hashing discussions and design decisions previously reached, and often re-discovering build approaches previously worked out. We needed a way to mutually understand the problems we were trying to address and the solution(s) being designed, and to communicate those discussions and designs decisions reached unambiguously. Yet with 2-week iterations, we didn't have a lot of time to spend on highly formatted and templated documentation. Consequently, we found Scott Ambler's "Agile Modeling" principles and practices provided just the approach we needed. Modeling fostered mental clarity about a number of issues we encountered both in understanding the "problem-space" (analysis of the clinical domain of interest), and the "solution space" or design of what we would be building in the EHR. We embraced and tried to employ many of Ambler's agile modeling principles and practices, especially "Use the simplest tools", "Model with others", "Apply the right artifacts", and "Create simple content". While we had access to two different modeling software packages, we created the majority of our models during group sessions, drawing on physical or large electronic whiteboards, with subsequent photos or screen captures of the drawings.

We found a subset of models consistently most useful across multiple projects—some standard UML model types, some non-UML but well accepted modeling types, and other lightweight scrum practices, such as User Stories.

Given the large numbers of registries and associated CDS tools we were developing in short order, we also found adopting an overall framework for applying models to our development cycle useful. Shown below is what we adapted from Rosenberg's ICONIX process:

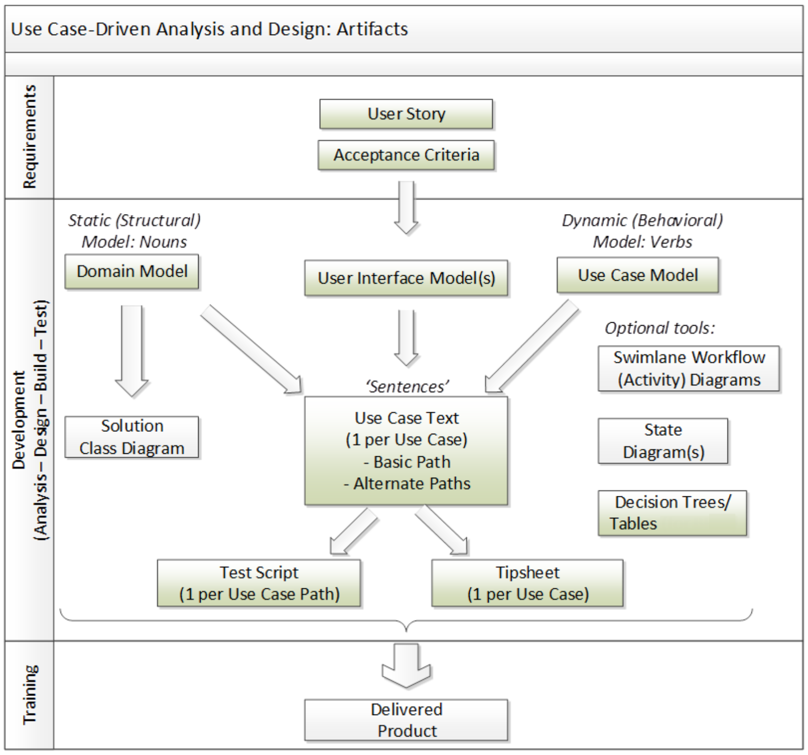

Use Case-Driven Analysis and Design: Artifacts
Requirements
User Story
Acceptance Criteria
Development (Analysis – Design – Build – Test)
Static (Structural) Model: Nouns
Domain Model
User Interface Model(s)
Dynamic (Behavioral) Model: Verbs
Use Case Model
Optional tools:
Swimlane Workflow (Activity) Diagrams
State Diagram(s)
Decision Trees/ Tables
'Sentences'
Use Case Text (1 per Use Case) - Basic Path - Alternate Paths
Solution Class Diagram
Test Script (1 per Use Case Path)
Tipsheet (1 per Use Case)
Training
Delivered Product

Figure 1. Model-Driven Development Method Employed

Examples of key model types we found useful for developing registries and associated CDS tools follow:

a. User Stories (and Acceptance Criteria)

Purpose: Define who the product is for, what it should do, and why (for what benefit). Specify what success (being "done") looks like.

Example User Story for an Osteoporosis Registry: [Note: this is an "Epic", later broken down into smaller user stories scheduled into 2-week iterations]:

"As a specialist caring for patients with osteoporosis and/or osteopenia,
I want to develop a registry of all patients seen with either of these conditions (a) in my clinic and (b) across the Health System,
so that I can more easily identify and rectify care gaps in appropriate use of medications for these conditions in order to improve bone health in the patients we serve."

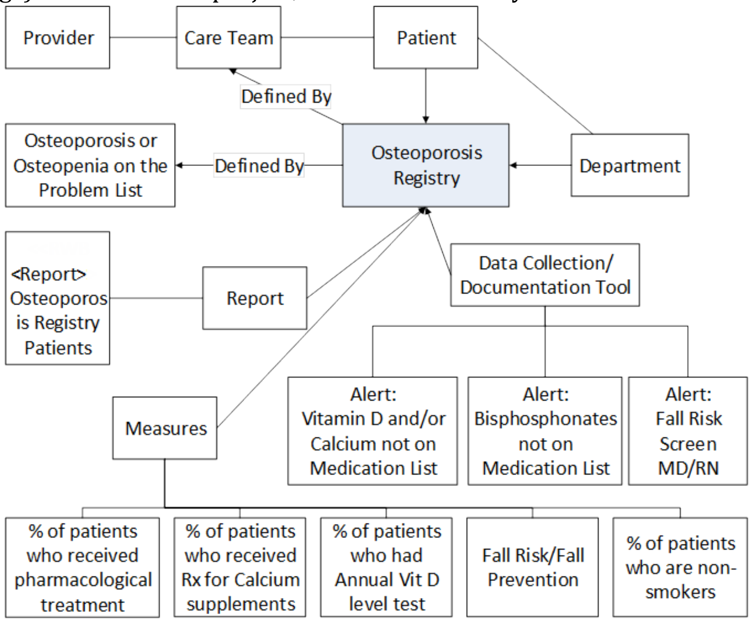
Acceptance Criteria for a User Story can be a simple bulleted list:

- Population registry report of all osteoporosis patients

- Clinical decision support tools to prompt providers to order Vitamin D, calcium supplements, and/or bisphosphonates for osteoporosis patients
- Report of potentially-eligible patients not enrolled in registry
- Training on use of decision support tools and reports

b. Class Diagram (Domain Model)

Purpose: Define "nouns" of system—depict the major terms and concepts ("objects" or "classes" of things) relevant to the project, and show how they are related to each other.

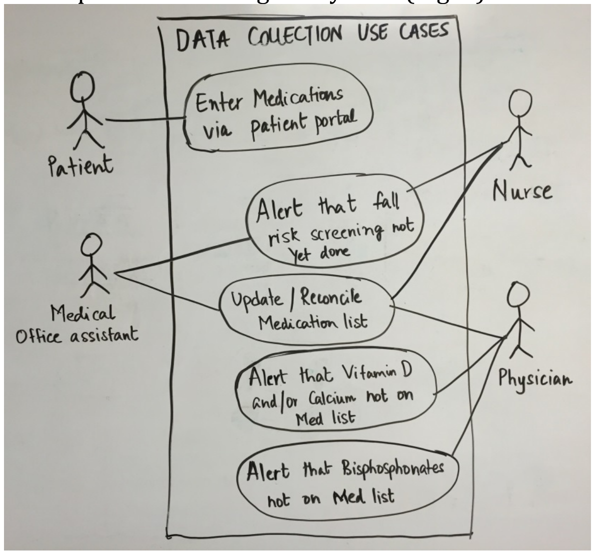


Provider
Care Team
Patient
Defined By
Osteoporosis or Osteopenia on the Problem List
Defined By
Osteoporosis Registry
Department
<Report> Osteoporosis Registry Patients
Report
Data Collection/ Documentation Tool
Measures
Alert: Vitamin D and/or Calcium not on Medication List
Alert: Bisphosphonates not on Medication List
Alert: Fall Risk Screen MD/RN
% of patients who received pharmacological treatment
% of patients who received Rx for Calcium supplements
% of patients who had Annual Vit D level test
Fall Risk/Fall Prevention
% of patients who are non-smokers

Figure 2. Domain Model (Class Diagram) for an Osteoporosis Registry

c. Use Case Diagram

Purpose: Define "verbs" of system—depict the major activities people in various roles can accomplish when using the system (Fig. 3).

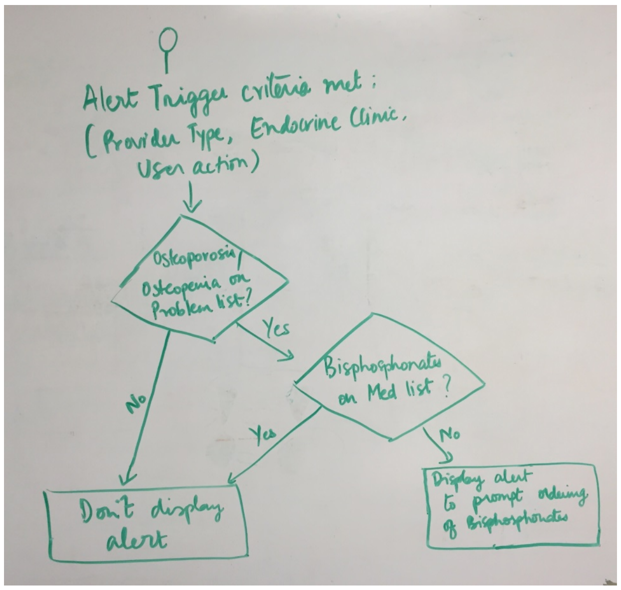


DATA COLLECTION USE CASES
Patient
Enter Medications via patient portal
Nurse
Alert that fall risk screening not Yet done
Medical Office assistant
Update / Reconcile Medication list
Physician
Alert that Vitamin D and/or Calcium not on Med list
Alert that Bisphosphonates not on Med list

Figure 3. Use Case Diagram for an Osteoporosis Registry

d. Decision Tree

Purpose: Unambiguously define the logic ("business rules") involved when evaluating one or more input conditions to derive a result, such as whether or not to display an alert to a physician or nurse (Fig. 4).

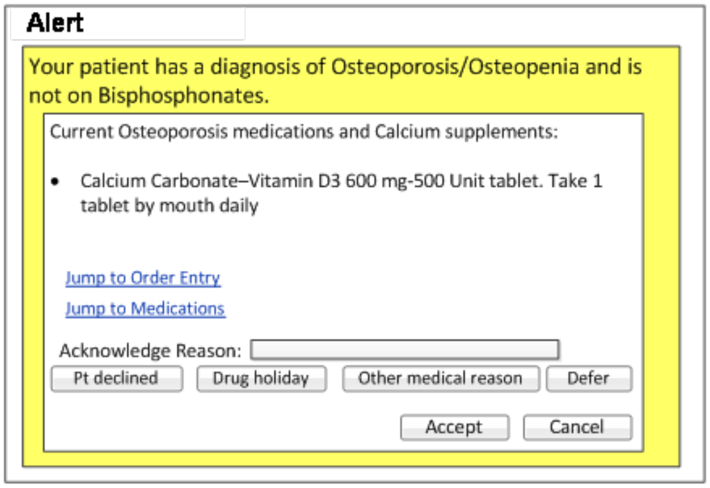


Alert Trigger criteria met : (Provider Type, Endocrine Clinic, User action)
Osteoporosis/ Osteopenia on Problem list?
Yes
No
Bisphosphonates on Med list ?
Yes
No
Don't display alert
Display alert to prompt ordering of Bisphosphonates

Figure 4. Decision Tree for a Clinical Alert

e. Graphical User Interface (GUI) Storyboard

Purpose: Provide a chance to anticipate what the system will look like, and to visualize interacting with it when reading a Use Case Text, described next (GUI mockup of EHR Alert: Fig. 5).



Your patient has a diagnosis of Osteoporosis/Osteopenia and is not on Bisphosphonates.

Current Osteoporosis medications and Calcium supplements:

- Calcium Carbonate–Vitamin D3 600 mg-500 Unit tablet. Take 1 tablet by mouth daily

Figure 5. Graphical User Interface (GUI) Storyboard

f. Use Case Text

Purpose: Define the "sentences" of the system. Describe how the end-user will interact with the system's objects in the context of a given user interface, typically in a "user does this, system does that" format.

Example:

Scenario/step	User action	System action
Basic Path:		
1.a	Clicks Jump to Order Entry hyperlink	Opens up the Order Entry activity for the provider to prescribe Bisphosphonates, silently files an Acknowledge reason of 'Action taken', and locks out the Alert for 168 hours for this provider, this encounter.
Alt Paths:		

2.a	Clicks Jump to Medications hyperlink	Opens up the Medications activity for the provider to review the pt's meds and order Bisphosphonates if desired, silently files an Acknowledge reason of 'Action taken', and locks out the Alert for 168 hours for this provider, this encounter.
3.a	Clicks [Pt declined]	Displays "Not done – patient reason" in the Acknowledge Reason: box. Enables the [Accept] button
3.b	Clicks [Accept]	Files an Acknowledge reason of "Not done – patient reason", and locks out the alert for 6 months, all providers, all encounters
4.a, 4.b. thru 6.a, 6.b	<Clicks other Ack button>	<similar to 3.a. and 3.b. for each, with different Ack reasons and possibly different lockout settings>
7.A	Clicks [Cancel]	Closes Alert window, no lockout set

g. Solution Object Diagram

Purpose: Depict the software objects (records) to be built or employed in our EHR, and how they relate to one another, as a road-map for understanding the detailed design

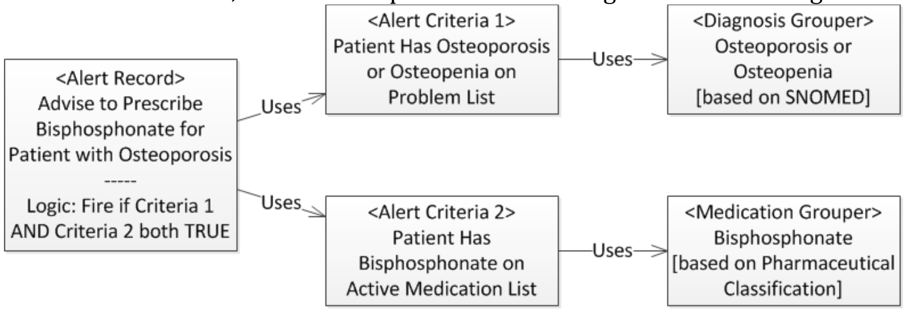
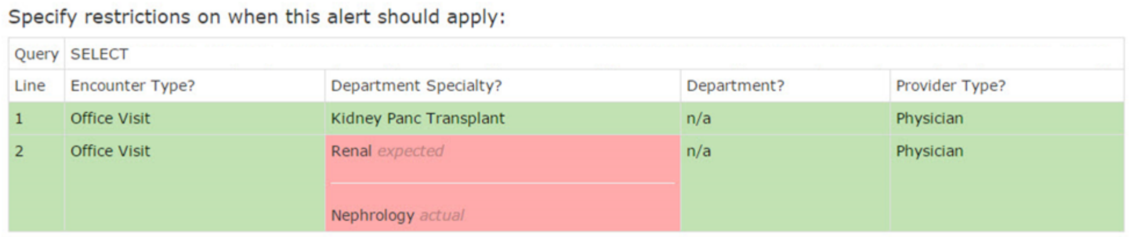

<Alert Record> Advise to Prescribe Bisphosphonate for Patient with Osteoporosis ----- Logic: Fire if Criteria 1 AND Criteria 2 both TRUE
Uses
<Alert Criteria 1> Patient Has Osteoporosis or Osteopenia on Problem List
Uses
<Diagnosis Grouper> Osteoporosis or Osteopenia [based on SNOMED]
Uses
<Alert Criteria 2> Patient Has Bisphosphonate on Active Medication List
Uses
<Medication Grouper> Bisphosphonate [based on Pharmaceutical Classification]

Figure 6. Solution Object Diagram

iii. Began gaining experience with additional practices associated with agile development:

As the projects continued and went into production, in several cases significant changes were requested, either for additional functionality, or to re-do an approach that proved impractical in actual practice. Having "living documentation" of the actual current build to facilitate evolving the design struck us as highly useful. We already had an automated acceptance testing tool in place (FitNesse, along with dbFit for querying databases). We have begun creating "executable specifications for CDS tools as FitNesse tables. In this approach (also known as Acceptance Test-Driven Development), the same table can serve as (a) requirements specification, (b) automated acceptance test, (c) automated regression test (once in Production), and (d) living documentation of how the software is actually built/configured.

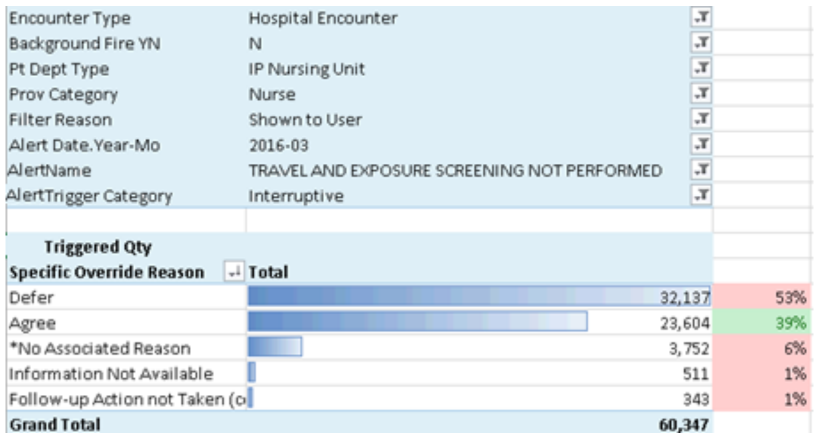
Example of some passing and failing executable specifications for an Alert:


Specify restrictions on when this alert should apply:

Query	SELECT			
Line	Encounter Type?	Department Specialty?	Department?	Provider Type?
1	Office Visit	Kidney Panc Transplant	n/a	Physician
2	Office Visit	Renal *expected* Nephrology *actual*	n/a	Physician

In addition, to promote rapid incorporation of feedback on CDS tool usage, we have begun creating and examining automated monitoring tools, to detect when a CDS tool may not be working as intended (an alert that's over-firing or under-firing, for instance), and to study actual user interactions with the CDS tools to further optimize their design.

The following Excel PivotTable queries a cube containing 30 million user interactions with our CDS alerts, filtered to display how nurses respond to a particular alert:



Encounter Type	Hospital Encounter
Background Fire YN	N
Pt Dept Type	IP Nursing Unit
Prov Category	Nurse
Filter Reason	Shown to User
Alert Date.Year-Mo	2016-03
AlertName	TRAVEL AND EXPOSURE SCREENING NOT PERFORMED
AlertTrigger Category	Interruptive

Triggered Qty Specific Override Reason	Total	
Defer	32,137	53%
Agree	23,604	39%
*No Associated Reason	3,752	6%
Information Not Available	511	1%
Follow-up Action not Taken (o	343	1%
Grand Total	**60,347**	

C. Results

i. Quantitative Results:

During calendar year 2015, the co-development team:

- developed 58 total EHR-based registries, with 1 or more for each of 33 specialties included in the project's initial scope
- defined 134 clinical process and outcome measurements
- built 111 EHR-based data collection tools
- placed over 16.000 patients on registries.

As of this writing, there are now over 60,000 patients on the registries.

The average cycle time per project was 14 weeks (seven 2-week iterations), which was longer than our estimate going-in of roughly five 2-week iterations per project. Causes for adding iterations included both value-added reasons such as additional CDS features desired by the customer, but also non-value added reasons, such as re-work of delivered features due to misunderstood requirements. Informally, factors that seemed to increase the likelihood of re-work included breakdowns in our creating shared models of scope and design leading to building the wrong thing, inaccessibility of the real customer during the iteration(s) to provide feedback, and discovery of unanticipated problems with a design following implementation in a real clinical setting. We held retrospective sessions to learn from these experiences how to better adapt our practices (such as rescheduling projects until the customer is available, and holding ourselves more accountable for creating and validating Acceptance Criteria for our user stories prior to beginning work). We do not believe the issue of discovery of unanticipated problems can be reduced to zero. However, the use of relatively short (2-week) iterations reduces the time delay and costs associated with rework. We have not quantified the improvement in cost of delay associated with our move to applying time-boxed iterative development on CDS projects, but believe it be substantial. For comparison, the initial 10 registry projects we had done prior to adopting the agile methodology took 6-12 months from initiation until move to Production, and most failed to achieve widespread clinical use when finally released due to persistent defects in their construction from a clinical perspective.

ii. Qualitative Results:

<u>Specific benefits by diagram type:</u>

- User story: rapidly gaining consensus on the purpose and scope of a given registry project. Along with the Acceptance Criteria, served as a useful point of reference when questions about original intent of a registry project's scope arose.
- Domain model: identifying clearly what clinical conditions or procedures define a given registry, and what data collection tools and reports are envisioned.
- Use case diagram: clarifying which EHR-based clinical alerts should display for which roles.
- Decision tree: removing ambiguity about the exact conditions under which a clinical alert should "fire" (display to the end-user).

- UI Mockup and Use Case Text: gaining agreement on data collection tool design and selection options. Specifying exactly what should happen when selecting a given option after a clinical alert is presented, including any lockout times to prevent the alert from re-firing prematurely.

General benefits of modeling:

- recognizing useful design patterns for re-use on subsequent projects.
- speeding the learning curve for cross-training team members on EHR tool capabilities new to them.
- providing a "mental model" to start from when needing to quickly adapt a design to clinical feedback.

D. Next Steps:

To-date we have primarily worked on developing clinical decision support tools and process measures for promoting delivery of optimal care for each specific clinical condition. Our ultimate goal is to collect outcomes data, to show that by promoting best-practice care, patients are achieving better quality of life and improved clinical outcomes. Thus, our project has already transformed into an ongoing program, intended to extend over the next several years. Encouraged by the benefits we have seen from applying agile principles and practices to date, we hope to extend them further, and get better at them through further experience.

4. WHAT WE LEARNED

A. Clinical Decision Support (CDS) tools built within Electronic Health Records (EHRs) combine internal complexity with deployment into the complex healthcare delivery environment. Commonly this complexity can yield initial unexpected or unwanted CDS tool behavior when deploying in active real-world clinical practice. Thus, requirements evolution is to be expected and embraced, to help ensure new EHR tools work in clinicians' actual workflows.

B. Agile development with 2-week time-boxed iterations can be highly effective in developing and evolving CDS tools that met clinicians' needs and expectations.

C. An "agile modeling" approach helps foster shared understanding of CDS requirements and design, removing ambiguities and promoting a shared mental model. A working set of structural and behavioral models promotes rapid-cycle evolution of EHR tool design in response to clinical feedback. A subset of diagram types proved most useful for constructing EHR-based specialty registries and associated CDS tools.

D. An open source automated testing tool (FitNesse) can be used to specify desired CDS tool design as a form of test-driven development, and used for regression testing following release to production.

5. ACKNOWLEDGEMENTS

We would like to recognize the expert contributions of the following individuals: Sue Burk—Our shepherd for writing this Experience Report, Sue's masterful insights and guidance proved invaluable in selecting, organizing, and presenting our thoughts. This was our first experience working with a shepherd, and Sue made it a delightful one! Margaret Parks, who helped in document preparation and submission, and—as always—in keeping us sane. Our UT Southwestern colleagues on the Ambulatory Quality Outcomes project. This project came to successful fruition through the talent and dedication of numerous individuals working together, drawn from our quality, information resources, clinic operations, and analytic teams. We are indebted to them all. We specifically want to acknowledge the vision and guidance of Dr. Jason Fish on this project, as well as the key leadership roles played by Deepa Bhat, Angela Carrington, Stacy Clark, Lisa Davis, Ki Lai, Jacqueline Mutz, and Adrian White. We also extend our heartfelt thanks to leaders of UT Southwestern's Health System Information Resources for their active support of this project: Mark Rauschuber, Kathryn Flores, Josh Youngblood, and Scott Minnerly.

REFERENCES

Ambler, Scott. "Agile Modeling: Effective Practices for eXtreme Programming and the Unified Process", Wiley 2002
Rosenberg, Doug and Stephens, Matt: "Use Case Driven Object Modeling with UML: Theory and practice", Apress 2013
Topaz, Maxim, et al. "Rising drug allergy alert overrides in electronic health records: an observational retrospective study of a decade of experience", Journal of the American Medical Informatics Association, first published online 17 Nov 2015
Wright, Adam, et al. "Analysis of clinical decision support system malfunctions: a case series and survey", Journal of the American Medical Informatics Association, first published online 28 March 2016