Test Case Selection in Pre-Deployment Testing of Complex Clinical Decision Support Systems

Geoffrey J. Tso, MD1,2, Kaeli Yuen, BS3, Susana Martins, MD, MS3, Samson W. Tu, MS4, Michael Ashcraft, MD3, Paul Heidenreich, MD, MS5, Brian B. Hoffman, MD6,7, Mary K. Goldstein, MD, MS2,3

1 Center for Innovation to Implementation, VA Palo Alto Health Care System, Menlo Park, CA; 2 Center for Primary Care and Outcomes Research (PCOR), Stanford University School of Medicine, Stanford, CA; 3 Geriatrics Research Education and Clinical Center (GRECC), VA Palo Alto Health Care System, Palo Alto, CA; 4 Biomedical Informatics Research (BMIR), Stanford University School of Medicine, Stanford, CA; 5 Cardiology, VA Palo Alto Health Care System, Palo Alto, CA; 6 VA Boston Health Care System, Boston, MA; 7 Harvard Medical School, Boston, MA

Abstract

Clinical decision support (CDS) systems with complex logic are being developed. Ensuring the quality of CDS is imperative, but there is no consensus on testing standards. We tested ATHENA-HTN CDS after encoding updated hypertension guidelines into the system. A logic flow and a complexity analysis of the encoding were performed to guide testing. 100 test cases were selected to test the major pathways in the CDS logic flow, and the effectiveness of the testing was analyzed. The encoding contained 26 decision points and 3120 possible output combinations. The 100 cases selected tested all of the major pathways in the logic, but only 1% of the possible output combinations. Test case selection is one of the most challenging aspects in CDS testing and has a major impact on testing coverage. A test selection strategy should take into account the complexity of the system, identification of major logic pathways, and available resources.

Introduction

Health information technology (HIT) has an overall positive impact on healthcare delivery and patient outcomes1,2. However, poorly designed and implemented systems can have detrimental effects3-5. Many studies have focused on usability of HIT; however, software defects that have high adverse consequences are common in many high-risk fields. Consequently, vigilance in identifying and correcting defects in HIT is essential6,7,8,9.

ATHENA-HTN is a knowledge-based clinical decision support (CDS) system designed to provide patient-specific recommendations based on guidelines from the Joint National Committee on Detection, Evaluation, and Treatment of High Blood Pressure (JNC) and the VA Diagnosis and Management of Hypertension in the Primary Care Setting10. Originally designed for delivery of recommendations on hypertension (HTN) management as a model, the system has been expanded to ATHENA-CDS, which includes knowledge bases encoding clinical practice guideline (CPG)-based recommendations for chronic kidney disease (CKD), heart failure (HF), diabetes (DM), and hyperlipidemia (HL). We have been reconfiguring the CDS for use within a clinical dashboard rather than within an electronic health record.

With the release of the Eighth Report of the JNC (JNC 8), the ATHENA-CDS-HTN knowledge base (KB) was updated to reflect the new guidelines11. While the changes to blood pressure targets were relatively easy to encode, the changes to prioritization of drug choices in JNC 8 required re-encoding of the ATHENA-HTN KB. As a result, regression testing, which is software testing used to detect defects from incremental changes to the code, was not suitable, and complete re-evaluation was necessary.

CDS system testing aims to ensure adequate coverage of the knowledge encoded as well as consistency and accuracy of results12. Unfortunately, software testing is a challenging task that is labor-intensive and expensive. Software engineering studies have found that quality assurance typically accounts for 50-60% of total software development costs13,14,15,16. Despite the amount of time and money devoted to testing, its effectiveness is lower than expectations, with only 30-60% of defects being found17.
Software testing is often constrained by limitations in resources, for example, time, labor, and computing power. There is often a trade-off between the testing coverage (amount of the logic tested) and the size of the test suite (number of test cases and testing cycles) utilized. Systematic techniques are necessary to make testing effective and efficient. In CDS, testing the accuracy of the system’s output requires human evaluation of test cases to establish a reference standard set of test cases with correct output. In complex clinical decisions with multiple associated clinical characteristics to consider, the number of possible case scenarios can make it impractical to generate reference standards for each. For example, in HTN management, the presence of chronic comorbid conditions greatly increases the complexity of medication recommendations (Figure 1).

![Complexity of comorbid conditions affecting the prescribing of HTN medication](image)

Figure 1. Complexity of comorbid conditions affecting the prescribing of HTN medication

With an increased focus on CDS, methods are needed for test case selection that will allow for testing key components of the CDS while keeping the number of cases within the range feasible for establishing the reference standard. There are many methods of testing described in computer science, however there are few publications discussing testing techniques in CDS. In this paper, we explore one approach to test case selection for a complex CDS, using well-studied methods in computer science.

Methods

CDS System to be tested: ATHENA-CDS architecture

The ATHENA-CDS system uses the EON Guideline Interpreter system developed at the Stanford Center for Biomedical Informatics Research (BMIR), including a KB modeled in Protégé, a knowledge acquisition program.
also developed at BMIR (Figure 2)\textsuperscript{21}. Clinical data from the electronic health record is available in a relational database. EON processes the clinical data with knowledge in the KB to generate conclusions about the state of the patient and recommendations for next steps in therapy. In the current implementation, ATHENA-CDS’s output is designed for display in a clinical dashboard for primary care teams. Testing as described in this paper was conducted in a test environment that mimics the computing environment of the Veterans Integrated Service Network (VISN) 21 Clinical Dashboard, but is populated with completely de-identified patient data.

![ATHENA-HTN Architecture](image)

**Figure 2.** ATHENA-HTN architecture

**Hypertension management complexity**

The complexity of the HTN-KB was determined by three methods. The first method programmatically traversed the KB logic and determined the number of clinical decision branch points. The second method counted the total number of data elements (e.g., vitals, labs, medications, comorbidities) referenced by the ATHENA-HTN system. The third method determined the total number of unique outputs that could be generated by the ATHENA-HTN system as a surrogate for the total number of linear pathways in the logic\textsuperscript{18}. The total number of outputs from the ATHENA-HTN system was calculated from an analysis of major outputs (recommendations, messages) unique to HTN management. Free-text messages generated to provide supplemental information about recommendations were excluded for this stage of testing. Outputs generated from encodings shared by multiple knowledge bases were excluded (e.g. specific drug-related messages such as drug-drug interactions). Also, combinations of outputs that were not logically possible were excluded.

**Knowledge base encoding**

Three physicians reviewed the JNC 8 guidelines. During the review, they collaboratively discussed the guideline recommendations and mapped their consensus interpretation into an overview flow diagram (Figure 3). Eligibility criteria, exclusion criteria, logical flow, and border constraints were also determined. Included in the logic were medications that were directly referenced by the guidelines as well as medications and comorbidities with significant interactions. Laboratory tests were identified by standard LOINC\textsuperscript{22}. The KB was then encoded into the Protégé system by a physician knowledge engineer. As part of the encoding process, the knowledge engineer performed unit testing with sample data after each section of the guidelines was encoded, to check that the subset of knowledge just entered processed as intended. This unit testing was done in a test environment that is convenient to use during knowledge encoding; it tests the KB and the guideline interpreter (together these are the EON system), but does not test the larger system architecture that extracts patient data from the relational database and submits it to the EON system. A text-based “Rules” document was created to serve as a reference for how the guidelines were encoded into the knowledge base.
Figure 3, Overview flow diagram of hypertension management algorithm. Details of logic for adding or increasing dosage of anti-hypertensive drug classes not shown. HTN, Hypertension; BP, Blood pressure; SBP, Systolic blood pressure; DBP, Diastolic blood pressure; VA, Veteran Affairs; JNC, Joint National Committee; CKD, Chronic Kidney Disease; ACEI, Angiotensin converting enzyme inhibitor; ARB, Angiotensin receptor blocker

Pre-Pilot Testing
When the initial KB encoding and unit testing were complete, pre-pilot tests were performed with patient data from the relational database to verify system integrity and identify major defects in KB encoding. The pre-pilot set was run in a test environment using the EON Guideline Interpreter to generate messages and recommendations from the encoded KB and electronic health record patient data stored in a relational database. Expected outputs for the test patient data were manually determined by two clinical experts. Recommendations from the two clinical experts were compared and modified to create a reference standard. This reference standard was then compared with the ATHENA-HTN output to identify defects.
100 Case Pilot Testing
The major objective of the pilot testing phase was to detect errors in order to identify improvements needed to the ATHENA-HTN system to ensure that its recommendations were in accord with guideline-based clinical decision-making.

Test case selection
Analysis of the logical flow diagram of the guidelines was used to partition the test cases into major categories for testing. For each category, clinical characteristics (e.g. blood pressure, comorbidities, presence or absence of medications on active medication list) were determined for use as test case selection criteria. Test cases that met criteria for each test category were then selected randomly from a dataset of 5000 patients seen at VAPAHCS in 2009-2013 that had been fully de-identified prior to use for this study.

Testing procedure
100 patient cases were selected from the dataset using the criteria outlined above. Guideline-based recommendations were made on the test cases by a physician (MA), who was very familiar with the Rules document and the intended encoding of ATHENA-HTN, and by a pre-clinical medical student (KY) who used the Rules document to guide recommendations. Patient data was presented to these raters with a custom-designed clinical user interface generated in Microsoft Access and populated with data from a SQL Server database. The human testers recorded correct conclusions and recommendations for each case. They both rated 20 cases and resolved any discrepancies. The medical student then completed the remaining 80 cases. The resulting set of 100 test cases with human-generated correct answers constituted the reference standard against which the CDS would be judged. The ATHENA-HTN system processed the 100 cases and the output was stored in the Access database for comparison. Differences between recommendations by the CDS and the reference standard were compiled by the medical student and reviewed with the physician. The development team then analyzed and classified discrepancies between the ATHENA-HTN output and the human reference standard. We report here the extent to which the test cases provided coverage of the logic flow and potential recommendations.

Results

Patient selection characteristics
With analysis of the encoded guidelines and logic flow, the HTN management algorithm was divided into seven pathways. Six of the pathways had distinct clinical characteristics as inputs into the logic that did not result in drug recommendations. The last pathway included all of the drug recommendations for patients with blood pressure above the target. This pathway was sub-divided into five pathways based on the absence of a preferred antihypertensive drug class in their active medication list. Eighty test cases were selected to meet criteria in one of these 11 major pathways (6 without drug recommendation, 5 with drug recommendation). Another 20 cases who met eligibility requirements were randomly selected from the dataset (Table 1).

Table 1. Patient Test Selection Group Characteristics.

| Patient Selection Groups                                      | # of Patient Cases |
|---------------------------------------------------------------|--------------------|
| Not eligible to be considered for HTN recommendations          | 5                  |
| SBP >= 220 or DBP >= 110                                      | 5                  |
| Ischemic heart diseases and DBP < 60                          | 5                  |
| BP above target set by the existing VA clinical dashboard but below BP target set by JNC 8 guidelines | 5                  |
| BP below target set by the existing VA clinical dashboard and the JNC 8 clinical guideline | 5                  |
| On 4 or more BP medications                                   | 5                  |
| Not on ACEI                                                   | 10                 |
| Not on ARB                                                    | 10                 |
| Not on thiazide diuretic                                      | 10                 |
| Not on dihydropyridine CCB                                    | 10                 |
| Not on non-dihydropyridine CCB                                | 10                 |
| Patients eligible for HTN recommendations                     | 20                 |
| Total                                                         | 100                |
Hypertension management complexity

The updated ATHENA-HTN guidelines were encoded into the KB with 915 Protégé knowledge frames that include information about medication and patient conditions. These knowledge frames referenced another 2166 subclasses of medical information in the knowledge base. The encoded HTN guideline included 26 clinical decision branch points. Thirty-nine drugs or drug classes and 43 patient data elements (clinical characteristics) were referenced in the guideline.

From the analysis of the major outputs generated by the hypertension module, there were five case scenarios that resulted in messages without further drug recommendations. When drug recommendations were made, each drug class could be started (Add), started after some action (Contingent Add), increased in dose (Increase Dosage), increased in dose after some action (Contingent Increase Dosage), or stopped. Considering these outputs and excluding messages not unique to hypertension management, we calculated a total of 3120 possible CDS output combinations.

Testing output

After processing of the 100 test cases by ATHENA-HTN, all of the eleven major pathways were tested (Table 2). The testing evaluated the rule-in logic for each of the six non-drug recommending pathways and all five pathways recommending one of the preferred drug classes in the HTN guidelines. A total of 31 unique output combinations were generated by the test cases out of the 3120 logically possible combinations. Out of the 3113 possible drug recommendation combinations, only 20 different combinations were generated from the test cases (Table 3).

Table 2. Testing output coverage from 100 patient cases.

| Testing Coverage                          | Total Possible | # of Tested | Percentage |
|------------------------------------------|----------------|-------------|------------|
| Major pathways                           | 11             | 11          | 100%       |
| Messages                                 | 12             | 12          | 100%       |
| Drug classes evaluated                   | 5              | 5           | 100%       |
| Drug classes recommended                 | 5              | 5           | 100%       |
| Drug classes increased                   | 5              | 2           | 40%        |
| Drug recommendation combinations         | 3113           | 20          | 0.03%      |
| Major output combinations                | 3120           | 31          | 1.0%       |
**Table 3. Unique Drug Recommendation Combinations from 100 Test Cases**

| Recommendation No. | ACEI | ARB | Thiazide | CCB DHP | CCB NDHP |
|--------------------|------|-----|----------|---------|----------|
| 1                  |      |     |          |         |          |
| 2                  |      |     |          |         |          |
| 3                  |      |     |          |         |          |
| 4                  |      |     |          |         |          |
| 5                  |      |     |          |         |          |
| 6                  |      |     |          |         |          |
| 7                  |      |     |          |         |          |
| 8                  |      |     |          |         |          |
| 9                  |      |     |          |         |          |
| 10                 |      |     |          |         |          |
| 11                 |      |     |          |         |          |
| 12                 |      |     |          |         |          |
| 13                 |      |     |          |         |          |
| 14                 |      |     |          |         |          |
| 15                 |      |     |          |         |          |
| 16                 |      |     |          |         |          |
| 17                 |      |     |          |         |          |
| 18                 |      |     |          |         |          |
| 19                 |      |     |          |         |          |
| 20                 |      |     |          |         |          |

The 20 unique drug recommendation outputs from running ATHENA-HTN on the 100 test cases. ACEI, Angiotensin converting enzyme inhibitor; ARB, Angiotensin receptor blocker; CCB, Calcium channel blocker; DHP, Dihydropyridine; NDHP, Non-dihydropyridine; Add, Consider adding the drug class to the medication regimen; Contingent Add, Recommendation of adding the drug class is contingent on the result of a laboratory test; Increase dosage, Consider increasing the dosage of medication; Contingent Increase Dosage, Recommendation for increase in dosage of the drug class is contingent on the result of a laboratory test.

**Discussion**

This study underscores the complexity of CDS and the challenges that can be encountered in ensuring quality control. We developed a test case selection strategy with a goal of broad testing around critical logic in the CDS. The strategy utilized a combination approach of well-studied methodologies from software engineering including adaptive random testing generation and model-based testing that are described below. The testing achieved our goal of selecting test cases that traversed all of the major pathways in the HTN algorithm flow diagram. However, in this phase of testing, only a small percentage of the total possible pathways in the logic were tested. Also, many of the test cases were effectively redundant in the logic that they tested. Here we discuss the numerous challenges in effective testing and describe methodologies to improve the testing and test case selection.

**Human Testing Constraints**

As is generally the situation for evaluating clinical recommendations, the amount of testing we could perform was constrained by the need for a human to establish a reference standard for each test case. This time-intensive task required a physician to review the clinical characteristics of a new patient and record their recommendations for every possible output that the system could generate. Due to this constraint, we limited this round of testing to only 100 patient cases. As a result, we could ideally test only 3.1% of the 3120 possible unique logic pathways. Unfortunately, this limitation can only be overcome with the addition of more test cases in future rounds of testing. The constraint on the number of test cases emphasized the importance of test case selection. For our initial pre-deployment testing, we chose to test each major logic pathway with multiple cases to ensure that indirectly associated inputs for other pathways did not vary the output in unanticipated ways. For future rounds of testing, selection criteria of test cases can be narrowed further to reduce redundancy and limit the number of cases for decision nodes that have already been well-tested. With the establishment of an optimized library of test cases with reference standards, future regression testing can be performed with fewer resources. Automation of testing, after
establishment of reference standards, can also be performed in regression testing and has been shown to clear improvement in testing costs and efficiency\textsuperscript{13}.

**Test Data Limitations**

The clinical dataset used in this study was large and randomly selected. These characteristics are considered positive attributes for many medical studies. However, for CDS testing, a clinical dataset can have both positive and negative results. Large randomized datasets often have characteristics that are normally distributed.\textsuperscript{23} This distribution likely contributed to the selection of test cases that repeat the same essential test. Test case generation, where test cases are altered or manufactured to produce non-normalized data, is one method to improve a clinical dataset. Complete creation of a test case library is also an option and is the most common source of test cases in software engineering. However, methods used to create test cases in non-medical software engineering might require substantial revision for application in medical informatics. Creating medical test cases can be complex because not all possible combinations of clinical data are realistic or physiologically possible. Generating clinically-possible test cases would require extensive programming to ensure that, for example, all laboratory values are chemically-consistent. Testing for improbable cases has its utility, however, detecting highly recurring defects is a higher priority in initial pre-deployment testing.

**Test Case Selection Challenges**

Even with an effective method for obtaining or creating test cases, determining and testing all of the possible testing scenarios is difficult in complex software. With the limitations imposed on testing a CDS, it is often not possible to test all of the scenarios, and thoughtful test case selection becomes vital in order to reduce testing burden and to increase coverage. Many approaches to test case selection have been described in computer science literature. Three of the more common approaches are random testing, input testing, and control flow testing.

*Random testing*

The use of random test sets is commonly utilized. Random testing is simple to execute and can often find unanticipated defects in complex logic. With a large number of test cases, complete testing coverage can be theoretically achieved. However, studies have shown that random inputs have a tendency to test the same logic repeatedly, leaving parts of the software untested\textsuperscript{24-26}. Adaptive random testing is an effective enhancement to random testing that improves the testing coverage through application of test selection criteria in repeat testing to cover case scenarios not covered in initial testing. An example of an adaptive random testing approach would be to apply selection criteria to exclude cases already in a test case library. Another example is partitioning test cases into groups and then focusing on poorly tested partitions in subsequent testing\textsuperscript{13}.

*Input testing*

Input testing is an approach that samples a subset of the inputs of a software program. Combinatorial testing is a type of input testing that samples a subset of the inputs of a software program. Test cases are selected from the possible set of input combinations\textsuperscript{13}. Inputs that are numbers such as serum potassium (K), are considered in the context of ranges (for example: $K \leq 3.0$ mEq/L, $3.0 < K < 5.5$, $K > 5.5$). One major limitation to input testing is that the number of combinations quickly increases as the number and complexity of each input increases. For example, a program that tests a patient with five binary clinical characteristics (e.g. gender, systolic blood pressure $> 140$ mmHg, has diabetes, serum potassium $< 5.5$ mEq/L, presence of microalbuminuria) would have $2^5$ or 32 possible input combinations to test. A program with seven inputs, each with three possible values, has $3^7$ or 2187 possible input combinations. Software that has complex inputs such as clinical data can have an extremely large number of input combinations. The pairwise method, a modification to combinatorial testing, was developed to reduce the number of input combinations tested while retaining quality of coverage. This method focuses on testing interactions between inputs and reduces the number of tests required by focusing only on pairs of input combinations\textsuperscript{27}.

*Control flow testing*

This method selects test cases that cover sequences from a control flow graph. The symbolic execution method constructs a flow graph of the logic through analysis of the software code. This approach has limits to real-world applications as the amount of effort required for the analysis and for the identification of appropriate test cases increases enormously with the complexity of the code\textsuperscript{13,26}. Model-based testing is a control flow method that constructs the control flow from an abstraction or model of the logic in a software program\textsuperscript{13,28}. Models can be derived from sources such as scenario, state, or process diagrams of the logic. Inputs and outputs are
specified for each step in the diagram. Test cases are selected from criteria that cover paths in the diagrams. Due to limitations in testing discussed above, sometimes only a subset of the paths in the model can be tested, and some criteria must be established to determine paths that are most important, likely, or critical.

There are many test selection methodologies including others not discussed in this paper. Each method has its own strengths and limitations and some studies have shown that employing a combination of techniques can improve test case selection. Software testing experts suggest that the choice of methodology be individualized for each software project. The complexity and composition of the software and the resources available are two major factors that should be considered when choosing a test case selection strategy. In testing ATHENA-HTN, these factors have had a great impact on our testing process.

While most implemented CDS provides clinicians with reminders or alerts, with increasing prevalence of multiple comorbidities and polypharmacy, there has been interest in the development of sophisticated systems that handle more intricate logic and data. As discussed in this paper, testing a complex CDS can be quite challenging. Although CDS systems are designed to supplement clinical decision making, incorrect information can potentially cause harm. Sufficient testing of CDS is required prior to deployment; however, there have been few publications on testing techniques and the determination of sufficient testing in CDS. To our knowledge, there is no authoritative guidance on standards or best practices for quality control in CDS and we hope that this paper will contribute to this topic and lead to further investigation.

**Conclusion**

Test case selection is an important and challenging process in CDS testing. The complexity of the system, identification of major logic pathways, available resources, and the need for a reference standard can have major influences on the extent and type of testing performed. We plan to continue validation and verification of ATHENA-HTN through future rounds of testing and further development of our test selection strategy.

**Acknowledgements**

This work was supported in part by Department of Veterans Affairs Health Services Research and Development (HSR&D) grant IIR 11-071-1 (PI: Goldstein). Dr. Tso's time was supported by an Advanced Fellowship in Medical Informatics funded by the Veterans Affairs Office of Academic Affiliations, Health Services Research and Development Service, and Office of Informatics and Analytics. Ms. Yuen's time was supported in part by the Alpha Omega Alpha Carolyn L. Kuckein Student Research Fellowship. The authors thank Gilbert Doan, BS, for contributions in managing the testing environment. Views expressed are those of the authors and not necessarily those of the Department of Veterans Affairs or other affiliated organizations.

**References**

1. Buntin MB, Burke MF, Hoaglin MC, Blumenthal D. The benefits of health information technology: a review of the recent literature shows predominantly positive results. *Health Aff (Millwood)*. 2011;30(3):464-471.
2. Greenes RA. *Clinical decision support: the road ahead*. Amsterdam ; Boston: Elsevier Academic Press; 2007.
3. Middleton B, Bloomrosen M, Dente MA, et al. Enhancing patient safety and quality of care by improving the usability of electronic health record systems: recommendations from AMIA. *J Am Med Inform Assoc.* 2013;20(1):e2-8.
4. Abookire SA, Teich JM, Bates DW. An institution-based process to ensure clinical software quality. Paper presented at: Proceedings of the AMIA Symposium1999.
5. Technology IoMCoPSaHL. *Health IT and patient safety: Building safer systems for better care*. National Academies Press; 2012.
6. Ash JS, Berg M, Coiera E. Some unintended consequences of information technology in health care: the nature of patient care information system-related errors. *J Am Med Inform Assoc*. 2004;11(2):104-112.
7. Waghofikar KB, MacLaughlin KL, Kastner TM, et al. Formative evaluation of the accuracy of a clinical decision support system for cervical cancer screening. *J Am Med Inform Assoc*. 2013;20(4):749-757.
8. Berner ES. *Clinical decision support systems: theory and practice*. Springer Science & Business Media; 2007.
9. Goldstein MK, Hoffman BB, Coleman RW, et al. Patient safety in guideline-based decision support for hypertension management: ATHENA DSS. *Journal of the American Medical Informatics Association*. 2002;9(Supplement 6):S11-S16.
10. Goldstein M, Hoffman B, Coleman R, et al. Implementing clinical practice guidelines while taking account of changing evidence: ATHENA DSS, an easily modifiable decision-support system for managing hypertension in primary care. Paper presented at: Proceedings of the AMIA Symposium 2000.

11. James PA, Oparil S, Carter BL, et al. 2014 evidence-based guideline for the management of high blood pressure in adults: report from the panel members appointed to the Eighth Joint National Committee (JNC 8). Jama. 2014;311(5):507-520.

12. Anchala R, Di Angelantonio E, Prabhakaran D, Franco OH. Development and validation of a clinical and computerised decision support system for management of hypertension (DSS-HTN) at a primary health care (PHC) setting. PLoS One. 2013;8(11):e79638.

13. Anand S, Burke E, Chen TYC, et al. An orchestrated survey of methodologies for automated software test case generation. 2013;86:1978-2001.

14. Ramler R, Wolfiniaier K. Economic perspectives in test automation: balancing automated and manual testing with opportunity cost. Paper presented at: Proceedings of the 2006 international workshop on Automation of software test 2006.

15. Myers GJ, Sandler C, Badgett T. The art of software testing. John Wiley & Sons; 2011.

16. Engel A, Barad M. A methodology for modeling VVT risks and costs. Systems engineering. 2003;6(3):135-151.

17. Farooq S, Quadri S. Empirical Evaluation of Software Testing Techniques - Need, Issues, and Mitigation. Software Engineering: An International Journal. 2013;3(1):41-51.

18. Menzies T, Cukic B. Adequacy of limited testing for knowledge based systems. International Journal on Artificial Intelligence Tools. 2000;9(01):153-172.

19. Martins SB, Lai S, Tu S, et al. Offline testing of the ATHENA Hypertension decision support system knowledge base to improve the accuracy of recommendations. AMIA Annu Symp Proc. 2006:539-543.

20. Zulman DM, Asch SM, Martins SB, Kerr EA, Hoffman BB, Goldstein MK. Quality of care for patients with multiple chronic conditions: the role of comorbidity interrelatedness. J Gen Intern Med. 2014;29(3):529-537.

21. SHAHAR Y. EON: a component-based approach to automation of protocol-directed therapy. Journal of the American Medical Informatics Association. 1996;3(6).

22. Forrey A, McDonald C, DeMoor G, et al. Logical observation identifier names and codes (LOINC) database: a public use set of codes and names for electronic reporting of clinical laboratory test results. Clinical Chemistry. 1996;42(1):81-90.

23. Altman DG. Practical statistics for medical research. Boca Raton, Fla.: Chapman & Hall/CRC; 1999.

24. Duran JW, Ntafos SC. An evaluation of random testing. Software Engineering, IEEE Transactions on. 1984(4):438-444.

25. Gutjahr WJ. Partition testing vs. random testing: The influence of uncertainty. Software Engineering, IEEE Transactions on. 1999;25(5):661-674.

26. Ding Z, Zhang K, Hu J. A rigorous approach towards test generation. Information Sciences. 2008;178:4057-4079.

27. Nie C, Leung H. A survey of combinatorial testing. ACM Computing Surveys (CSUR). 2011;43(2):11.

28. Utting M, Legeard B. Practical model-based testing: a tools approach. Morgan Kaufmann; 2010.

29. Chu H-D. An evaluation scheme of software testing techniques. Springer; 1997.

30. O'Sullivan D, Fraccaro P, Carson E, Weller P. Decision time for clinical decision support systems. Clinical Medicine. 2014;14(4):338-341.