Crazy title

Murat Ozcan
Siemens Building Technologies CPS Software Hub
Chicago, USA
murat.ozcan@siemens.com

***Abstract*—The overall purpose of this study is to describe the new paradigms achieved in testing, in lieu of the transition of applications from the desktop, to web and to cloud. We study the Siemens Building Technologies Horizon Cloud Program, where products are enabled by the Horizon ecosystem with a focus on speed, independence and small complexity. Horizon Cloud provides widespread connectivity to legacy & new controls, 3rd party, IoT, edge devices and a step-wise covering of on-site, real-time requirements. Horizon Cloud features its own apps based on building integration framework, along with themes, with an open API for ecosystem and SSP. The program also provides a common data model and API to BT Building Integration Framework to map southbound data against model.**

We start by describing the architecture of the Horizon Cloud program; the components that make up the whole, how they relate, technologies being used in each layer and how they are being tested individually and as a whole.

We continue by describing the continuous delivery process; Dockerization of the continuous integration pipeline, the architectural components being built as Docker images and being run as Docker containers, being deployed and tested in the cloud at unit, API and UI levels. Later, we describe how the components that make the architecture are orchestrated in AWS hosted OpenShift.

We describe the abstraction and the structure of the test code, the page object model, how specifications are separated from the page object / service / logic layer and the test data.

Finally, we analyze three applications of combinatorial testing (CT) in this modern development environment, incorporating the CT model into automation using BDD style Jasmine framework's Protractor end to end UI tests. We go through the CT modeling process using newly released, open source, cloud based CTWedge: Combinatorial Testing Web-based Editor and Generator. We describe how the model translates into functions in the code and its utilization in behavioral driven tests.

Finally, we analyze a scenario where a sequence of actions are incorporated into a CT model; with a focus on verification of these sequences, compositions of the actions and streamlining the expected assertions, in line, per the data-manipulated test oracle.

***Index Terms*—Combinatorial testing, Input parameter model, Sequence, Angular, Protractor, UI automation**

TODO: Add comments from Rcik Rick : A number of things I think would be of particular interest. One would be your comments on how CT fits into container/Docker development and some of the other platforms that are used a lot today. Most researchers don't know a lot about modern development environments, so your experience and recommendations on how to use CT for these would be very valuable, e.g., what sort of extra tooling would be helpful, characteristics that are good or poor fits for existing CT tools.

Also of interest is the way you integrate covering arrays and sequence testing. This is something that isn't always handled well, and yet very important for an awful lot of applications. The measurements of coverage at the end of the presentation are also of interest, in particular how to use coverage to determine when it's cost effective to complete testing.

I. Introduction

Modern web development typically falls into a design pattern. In the front-end there can be a JavaScript (JS) framework, some popular ones being Angular, React and Vue. The back-end preferences are varied, there is a plethora of choices in language : JS (NodeJS), GoLang, Python, C#, Java to name a few. This is coupled by a database, examples include MySQL, MongoDB, Elasticsearch.

Previous to cloud technologies, such an application would be hosted on a dedicated web server, or a hosting company. In contrast, cloud computing adopts a concept called "virtualization," where hardware resources can be further optimized through software functionality. As a result, not only application performance is optimized but also hosting the application is more cost effective.

A challenge for cloud computing is the resource intensive operating system (OS) usage, where the size of the OS image can be in gigabytes while the application is much smaller. Consequently virtual machines, sinfce they have to host an OS, do not solve this problem. Containerization is one proposed solution, and Docker is one example of a program that performs operating-system-level virtualization. TODO: insert Docker reference In containerization, a layer between OS and applications is introduced to optimize resource usage and eliminate the need for an OS.

This is highly valuable for application development because it enables the application be hosted in a minimal, resource and cost effective "container" which allows the application to be built, deployed and tested faster. Throughout the paper we will refer to this paradigm as Cloud Computing.

In this paper, we will study how Combinatorial Testing (CT) fits in the front-end test automation and continuous deployment of cloud computing paradigm. Examples will include modeling of the input parameter model (IPM), how the model translates to fields and methods in page objects TODO: insert reference, and the utilization of it all in behavioral driven test specifications. One example will include a scenario where a sequence of actions will be incorporated into a CT model, a problem that has been addressed in a variety of ways in previous works.

II. Related Work

A. Introduction to Combinatorial Testing

TODO: @Ludwig: small intro to CT.

B. Previous works on sequences

The method of modeling sequenced parameter groups applied in this paper borrows ideas from a combination of the modeling patterns found in [1]. These modeling patterns include:

1) *Optional and conditionally-excluded values:* the use of N/A value for parameters that are not a part of the parameter group in the current sequence.
2) *Ranges and boundaries:* the reduction of parameter values for certain parameters with over 100 possibilities
3) *Multi-selection & Order and padding:* a variation of these ideas was used for control parameters which enable/disable the parameter groups per the sequence.

C. Previous works on page objects

There has been hundreds of papers on BACnet since early 90s. While many papers have touched on testing. Generally, the testing of Event Enrollment Objects has not been a topic of focus. To get a better understanding on BACnet and Event Enrollment, some of the previous work can be reviewed in [2], [3], [4] and [5].

D. Previous works on cloud computing

There has been hundreds of papers on BACnet since early 90s. While many papers have touched on testing. Generally, the testing of Event Enrollment Objects has not been a topic of focus. To get a better understanding on BACnet and Event Enrollment, some of the previous work can be reviewed in [2], [3], [4] and [5].

III. The System Under Test

A. Description of the Architecture

Horizon Cloud program is composed of many teams and microservice architectural applications. The application / system under test in this study will be Building Operator IC (BOIC). BOIC is used for TODO: describe BOIC , also insert what IC means . BOI is being development by the Chicago team at Siemens Software Hub.

Figure 1 represents the BOIC architecture. The front-end is an Angular framework, in TypeScript. The back-end is an ExpressJS application on top of NodeJS platform. At the time of this study, there is an effort to move some of this functionality to other microservices - implemented in GoLang (Go) - to reduce operating costs. The Protocol Adapter (in C#) and Gateway (in NodeJS and Java) serve the purpose of exposing Siemens or third party edge-devices to the cloud. This enables the hardware and the gateway at a customer site to be controlled from a web browser, anywhere in the globe.

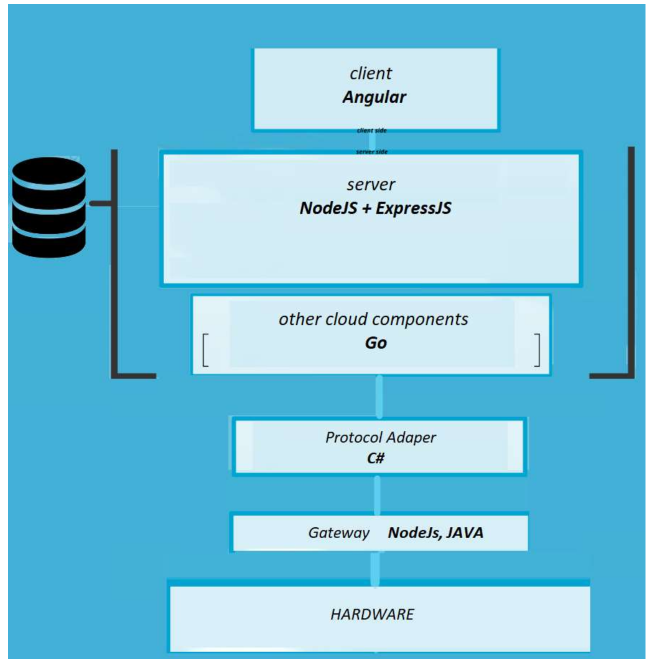

Fig. 1. BOIC architecture

B. Description of the Continuous Deployment Environment

Refer to figure **??**. Related parameter groups are henceforth referred to by color groups. In the figure the green path is described as when the Object Type is either analog input (AI), analog output (AO), analog value (AV), binary output (BO) or multisate output (MO). When this is the case the related fields are enabled and the others are disabled. The teal path signifies another group of related parameters: binary input (BI), binary value (BV). The blue path with multistate value (MV) works with the same logic.

The 'enabling' of related parameters occurs as the parameter used for control of that parameter group is set to TRUE. Since only a single parameter group should be enabled at a time, through constraints, the other control parameters which control other groups are set to FALSE. In turn, the parameter groups that are controlled by those control parameters are forced to have values equal to N/A. Here the use of N/A value for parameters that are not a part of the active parameter group in the current sequence is an idea inspired from the modeling pattern referred to in section II C 1. The use of control parameters with values TRUE & FALSE, the use of the control logic to identify the sequence in which the parameter group is occurring are ideas inspired from the modeling patterns referred to in section II C 3.

A better way to represent this sequence logic would be to separate them in blocks. Figure **??** pictures how this can be structured. These blocks represent different UIs the user is presented with based on their parameter choices for the base block. The base block contains the parameters ObjectType and EventType, which are common to all paths. ObjectType's parameter values AI, AO, AV, BO, MO enable block 1 highlighted in green. ObjectType's parameter values BI and BV enable to block 2 highlighted in teal. ObjectType's last parameter value MV relates to block 3 highlighted in blue.

Block 1 (in green) contains the parameters SetPointOrFeedbackPropType, SetPointOrFeedbackPropProperty.

Block 2 (in teal) contains the parameters ObjectPropertyReferenceMV, PropertyStateType, ValueUnsigned.

Block 3 (in blue) contains the parameters ObjectPropertyReferenceBVBI, PropertyStateType, ValueBinary.

Notice PropertyStateType being common in both blocks. Depending on the choices in ObjectType of the base block, a default parameter value gets selected for PropertyStateType: binary when ObjectType is binary-input or binary-value, Unsigned when ObjectType is Unsigned. We will exemplify the 3 different sequences that may occur based on the initial choices of the user in the base block.

Example 1: User selects ObjectType = 'AO', EventType = 'floating-limit' or 'out-of-range', the next set of configurations can only be for SetPointOrFeedbackPropType (over 2 choices under it) and SetPointOrFeedbackPropProperty (25 choices under it). All other parameters set to N/A utilizing constraints.

Example 2: User selects ObjectType = 'MV', EventType = 'change-of-value', the next set of configurations can only be for ObjectPropertyReferenceMV (True or False), PropertyStateType (only Unsigned), ValueUnsigned (4 configurations). All other parameters set to N/A utilizing constraints.

Example 3: User selects ObjectType = 'BV', EventType = 'change-of-state', the next set of configuration con only be for ObjectPropertyReferenceBVBI (True or False), PropertyStateType (only Binary), ValueBinary (00, 01, 10, 11). All other parameters set to N/A utilizing constraints.

A better understanding can be achieved by observing the verbal constraints. The Javascript code will be revealed in a later section while analyzing the pros and cons of the method.

If ObjectType = AI,AO,AV then EventType = floating-limit OR out-of-range

If ObjectType = BO,MO then EventType = command-failure

If ObjectType = MV then EventType = change-of-value

If ObjectType = AI,AO,AV,BO,MO then ObjectPropertyReferenceAIAOAVBOMO = TRUE ObjectPropertyReferenceMV = FALSE, ObjectPropertyReferenceBVBI = FALSE, PropertyStateType = N/A, ValueBinary = N/A, ValueUnsigned = N/A

If ObjectType = BI,BV then EventType =change-of-state, ObjectPropertyReferenceBVBI = TRUE, PropertyStateType = Binary, ObjectPropertyReferenceAIAOAVBOMO = FALSE, SetpointOrFeedbackPropType = N/A, SetpointOrFeedbackPropProperty = N/A, ObjectPropertyReferenceMV = FALSE, ValueBinary != N/A, ValueUnsigned = N/A

If ObjectType = MV then EventType = change-of-value, ObjectPropertyReferenceMV = TRUE, PropertyStateType = Unsigned, ValueUnsigned != N/A, ObjectPropertyReferenceAIAOAVBOMO = FALSE, SetpointOrFeedbackPropType = N/A, SetpointOrFeedbackPropProperty = N/A, ObjectPropertyReferenceBVBI = FALSE, ValueBinary = N/A

C. Description of the Test Code Structure

All t-way coverage measurements were made using NIST's coverage measurement tool Combinatorial Coverage Measurement (CCM) command line version [6].

For IBM method, the coverage with constraints factored in and the coverage without constraints are examined in figures **??** and **??**. Figure **??** shows the 2, 3, 4-way coverage measurement of the IBM method, respectively figure **??** shows the coverage measurement without constraints.

As mentioned in Section III B, the use of constraints are at the foundation of the IBM method. The 2-way coverage without constraints is at 100% as seen in figure **??**. The application of constraints, which make the method possible, reduces the 2-way coverage measurement to 58.1%. Because constraints are an integral part of the method, in this evaluation we believe that the coverage measurement of the IBM method is inconsequential.

For SBA method, the coverage with constraints factored in and the coverage without constraints are examined in figures **??** and **??**. Figure **??** shows the 2, 3, 4-way coverage measurement of the SBA method, respectively figure **??** shows the coverage measurement without constraints. The 2-way coverage without constraints is at 100% as seen in figure **??**. The minimum 2-way coverage without constraints is also at 100%.

The significance and the impact of minimum t-way coverage on branch coverage conditions in the code is discussed in paper [7]. Label M as the minimum 2-way coverage; i.e., the lowest proportion of settings covered for all t-way combinations of variables. For example, 2-way combinations of binary variables have four possible settings: 00, 01, 10, 11. Some variable pairs may have all four settings covered, but others may have less. M is the smallest proportion of coverage among all of the t-way variable combinations. M is also viewable as the rightmost line in the coverage strength meters in Figure 1 and 3.

Applying the constraints to the SBA method has a miniscule impact on the overall 2-way coverage: down from 100% to 99.76% as seen in figure **??**. The effect on 3 and 4-way coverage is similar. TODO: Ludwig, do you want to insert a sentence on why this is? The impact of constraints on the minimum 3-way and 4-way coverage is non-existent, however on minimum 2-way coverage the difference is 40%. TODO: Ludwig, do you want to insert a sentence on why this is?. Overall, t-way coverage analysis is valid on the SBA method.

1) Vertically juxtapose the juxtaposed CAs in step 4 to the result of step 3. This guarantees that the final result is again a 2-way covering array: : We simply add the 2-way CA under the 1-way CA as shown in Figure **??**. Here the first seed has 925 tests while the others have 8 and 4 respectively, this may seem challenging at first. For seed 2 and 3 we simply, repeatedly keep adding in blocks (of 8 and 4 respectively) until we cannot add anymore. If the final addition is not a multiple of 925, then we add the remainders.

IV. Test Methodology

In the following sections the coverage measurements, the pro's and con's of the IBM and the SBA methods will be evaluated.

A. Input driven testing of hardware

Considering t-way coverage measurement, SBA method provides evidence of the measurement while coverage measurement in IBM method is inconsequential as explained in the previous section.

Regarding the Application of the methods, IBM method is straightforward once the constraints are figured out. SBA method has elaborate steps as outlined in section III D. In step 3 of the method (plug-in construction), there is a need for a script that replaces integers with real values. Currently a freely available tool does not exist for such a task.

Constraints are important for both methods, but they are vital for IBM method. They can be manageable in a small scale, yet as the SUT gets larger they get more complicated. Figure **??** shows the Javascipt code for constraints used in the IBM method.

Considering the above factors, IBM method may be preferred for smaller systems under test while SBA method may be preferred for larger systems. However, the significance of the above factors for the system and/or the test team may effect the preferences for the method of use.

B. Filtering Points by Tag Combinations

C. State Transitions Between Combinations of Alarm Data, With a Variable Oracle

D. Sequenced Filtering, Searching, Sorting of Geo-located Sites

V. Conclusion and future work

The initial challenge in this study was how to best test EEO configurations in the most efficient manner. During the study it became apparent that the methods applied can be used to test any system where the parameter groups for the Combinatorial Input Parameter Model (IPM) are not simultaneously available, and instead may appear sequentially. Future work on this front may incorporate system test ideas where each seed (in the SBA method) or different color (in the IBM method) being a different module of functionality.

Acknowledgments. This research was carried out in the context of Austrian COMET K1 program and publicly funded by the Austrian Research Promotion Agency (FFG) and the Vienna Business Agency (WAW).

References

[1] I. Segall, R. Tzoref-Brill, and A. Zlotnick, "Common patterns in combinatorial models," in *Software Testing, Verification and Validation (ICST), 2012 IEEE Fifth International Conference on*. IEEE, 2012, pp. 624–629.

[2] S. T. Bushby, "Bacnettm: a standard communication infrastructure for intelligent buildings," *Automation in Construction*, vol. 6, no. 5-6, pp. 529–540, 1997.

[3] ——, "Bacnet today: Significant new features and future enhancements," *ASHRAE journal*, vol. 44, no. 10, p. S10, 2002.

[4] R. Nussbaumer and G. Waters, "Bacnet support for epics."

[5] H. M. Newman, "Bacnet celebrates 20 years," *ASHRAE Journal*, vol. 57, no. 6, p. 36, 2015.

[6] D. R. Kuhn, I. D. Mendoza, R. N. Kacker, and Y. Lei, "Combinatorial coverage measurement concepts and applications," in *Software Testing, Verification and Validation Workshops (ICSTW), 2013 IEEE Sixth International Conference on*. IEEE, 2013, pp. 352–361.

[7] D. R. Kuhn, R. N. Kacker, and Y. Lei, "Measuring and specifying combinatorial coverage of test input configurations," *Innovations in systems and software engineering*, vol. 12, no. 4, pp. 249–261, 2016.