Toward leveraging Gherkin Controlled Natural Language and Machine Translation for Global Product Information Development

Morgan O'Brien
McAfee, Building 2000 Citygate, Mahon, Cork
mobrien@mcafee.com

Abstract

Machine Translation (MT) already plays an important part in software development process at McAfee where the technology can be leveraged to provide early builds for localization and internationalization testing teams.

Behavior Driven Development (BDD) has been growing in usage as a development methodology in McAfee. Within BDD, the Gherkin Controlled Natural Language (CNL) is a syntax and common terminology set that is used to describe the software or business process in a User Story.

Given there exists this control on the language to describe User Stories for software features using Gherkin, we seek to use Machine Translation to Globalize it at high accuracy and without Post-Editing and reuse it as Product Information. This enables global product information development to happen as part of the Software Development Life Cycle (SDLC) and at low cost.

1 Credits

This document is based on the understanding that commercial Machine Translation systems perform well when used in conjunctions with Controlled Language rules (Roturier, 2004). It uses Gherkin CNL as written by developers and testers of McAfee products. The Machine Translation system used are from the Microsoft Translator Hub. The paper takes input from Information Development teams in McAfee based on the style and standards that they have in place.

2 Introduction

BDD is fast becoming a standard in software development, especially where the User Interface is primarily web based. It aims to satisfy needs of customers in software design by representing the behavior of the user as part of the plan. Using the Gherkin CNL, which is designed to work with BDD frameworks, a business manager who is not a developer can quickly describe how the software should function using examples.
Example based learning has advantages for understanding and retention of information. It enhances the effectiveness of User Stories in Agile software development by reducing ambiguity and enabling non-technical personnel to be involved. It also enhances the ability for a person to retain the information better by the application of a cognitive load to the user as they read. Gherkin could possibly be used as a superior learning asset to traditional information development for complex software processes by virtue that the reader employs more mental effort as they read which helps them retain the information better.
In McAfee, software design is managed through JIRA, a tool for planning, tracking and managing Agile software development. Gherkin descriptions are not currently part of that system and are rarely shared outside of the software Development and Testing teams. By accessing the test code, we get access to the Gherkin which in turn can facilitate the future possibilities such as Information Development. In turn, the ability to quickly leverage this content for international markets at low cost is an attractive possibility for the business.

In this paper, we explore how Gherkin can be organized and stored in JIRA. We test a baseline on how effective Gherkin is with Product Based Machine Translation engines. We then optimize the Gherkin as a better information asset, and in

Pérez-Ortiz, Sánchez-Martínez, Esplà-Gomis, Popović, Rico, Martins, Van den Bogaert, Forcada (eds.)
Proceedings of the 21st Annual Conference of the European Association for Machine Translation, p. 293–296
Alacant, Spain, May 2018.

turn optimize the MT to ensure accuracy of message by training glossaries of product and Gherkin terminology.

3 Process

Here we will explain a little about Gherkin and how it will be stored for access. Then we process it through our Product MT engines which are trained on the latest User Interface (UI) translations before running two types of tests on the outputs. We then modify the Gherkin source and re-run the process to test for improvement.

3.1 Gherkin

Gherkin is a ubiquitous language designed to be simple and effective at explaining behaviors carried out on software. Behaviors refer to things the user will do in the Graphical User Interface (GUI). The Gherkin CNL is based on the following concepts:

- A Feature
- A Scenario
- A Background
- A Scenario outline
- The Steps (Given, When, Then, And)
- Examples

There is a specific set of steps to be used for a Gherkin feature or scenario using the "Given", "When", "Then" declarations:

- **Given** I experience a specific state
- **And** I experience another starting state
- **When** I do something
- **And** I do something else
- **Then** I will experience an outcome

To reuse the descriptions in Gherkin more effectively, the use of Data Tables is popular. Data Tables allow the test case to be run with a set of variables in the input. The Data Tables we used in our testing are indicative of typical software development content:

- User Interface text
- Usernames and Passwords
- Server Names and Descriptions
- Currency, Numbers and Amounts

Gherkin authoring standards don't exist in a structured way within the company to date, but a minimal approach is taken; adhering to the syntax and GUI accuracy. There are 2 types of Gherkin that can be used for different purposes; Imperative and Declarative. Imperative is a detailed description of the behavior expected which has enough specifics to allow test automation code to be written for it, while Declarative is a less detailed higher-level description of the business goals of the software design without thought about the specifics or test code.
Consistency and reusability is of great importance in Gherkin authoring and management to reduce the amount of scenario writing needed.

3.2 JIRA and XRAY

JIRA is a software development tool used by Agile teams. It is designed to streamline the process of Issue and Feature creation and allow global teams to collaborate in their software release process. XRAY is a plugin for JIRA that focuses on the test process by managing the test cases and reporting on their validation in an easy to use dashboard. XRAY allows for the support of Gherkin language in JIRA in multiple ways.
1. Gherkin language is highlighted for known keyword declarations and Data Tables.
2. Gherkin is managed and exportable via a manual or API process in an XML format.
3. The automation code that is bound to the Gherkin test cases can report back on validation again via an API or an importable XML file.

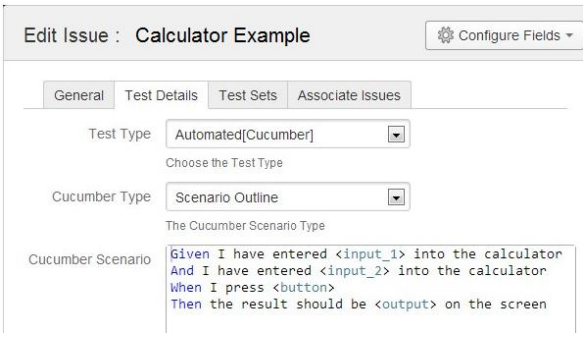

Fig 1. Gherkin scenario in XRAY for JIRA

The exportable XML format is key as this offers the opportunity to manage the Gherkin scenario and process it within a localization workflow.

3.3 Translation Quality Tests

To evaluate the success of Machine Translation applied to Gherkin we envisaged tests that are in line with how we currently rate non-Post-Editing translation jobs using MT. The languages we have chosen for this test are Italian, French and Brazilian Portuguese due to the availability of resources to help with the testing. There are 5 complete Gherkin scenarios used in each of the tests, making it 10 scenarios used in all (before and after). It is important to have different

scenarios for the before (baseline) and after (future) as familiarity with the process can affect the ability to understand the content during the second set of tests. We chose these tests as they are not exhaustive and provide a quick and useful baseline before pursuing further testing with larger datasets and project participants.

Usability Feedback - from a linguist familiar with products and terminology. The task is to rate the usability and fluency of the sentence in terms of how it can be understood. Usability Feedback is rated from 1 to 5 where 5 is highest quality and 1 is lowest (unusable) quality. We will test this on the source language (English) as well as Brazilian Portuguese, Italian and French. The number generated per language is then the total score divided by the number of strings reviewed (92 Strings were used in the tests).

Cognitive Usage - from an Engineer unfamiliar with the specific product usage but competent in general enterprise software usage. The Cognitive Usability study tests the participant's ability to complete a task with no prior knowledge of the software and is measured on how long it takes to complete the task in minutes. It is measured against the time needed to perform the task using English source. For example, if the task takes 5 minutes in English and 5 minutes in the target language, then the ratio is 1:1 (Same time needed).

Term Type	Example Content
Gherkin	When
Mfe Term	TIE Server
Action	receives a new
Mfe Term	MWG report
none	for the file with
Mfe Term	"Known malicious"
Mfe Term	MWG reputation
none	in
Object/UI	TIE Reputations

Table 1. Term identification - Gherkin segment

3.4 Gherkin Information MT Optimization

Initially our baseline MT systems are not optimized for Gherkin and ultimately the goal is to create an optimized engine. While fluency is sometimes important to understanding, the main goal of Gherkin is quick "In-Process" information development which is cheap to globalize. We focus then on the Keywords and lesser so on the fluency.

The main Action Keywords observed during this test are: Login, Go, Search, Click, See, Set, Accept, Wait, Open, Request, Have, Run, Receive, Request, Reject, Discard.

Gherkin as an information asset must speak in the imperative to direct the user actions. The Gherkin Keywords must be removed to make this possible. This can be done through simple regular expressions (RegEx) on the patterns. This then transforms Gherkin from a User Story description into an instructional asset:

Gherkin as User Story	Gherkin as Instruction
When I override the file reputation to "Known Malicious"	Override the file reputation to "Known Malicious"
And I go to "Overrides" tab in the "TIE Reputations" section in ePO	Go to "Overrides" tab in the "TIE Reputations" section in ePO
And I search for the file in the table	Search for the file in the table
And I click on the file in the table	Click on the file in the table

Table 2. Transform Gherkin to an instruction

The process then to move Gherkin from a test asset to an information asset for Machine Translation is like this:

1. Train MT engine with Product Terms
2. Export Gherkin in XML
3. RegEx replaces to the Imperative
4. Machine Translate
5. Publish

3.5 Test Results (before and after)

3.4.1 Usability Study

The Usability Study showed improvements on the understanding and language accuracy for most languages. However, there was a slight drop in accuracy on Italian after optimization was completed.

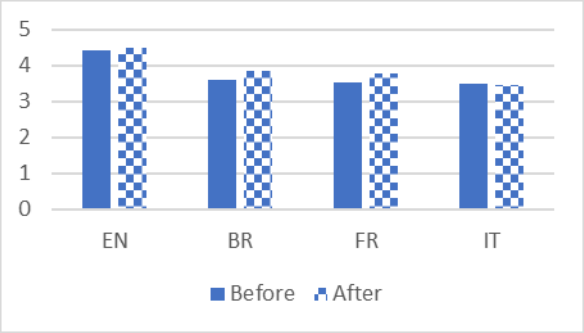

Fig 2. Usability Study averages before/after.

3.4.2 Cognitive Usability

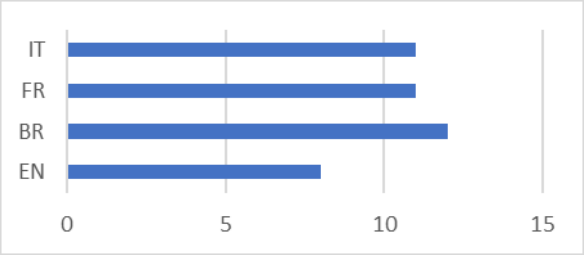

Fig 3. Time to complete tasks baseline.

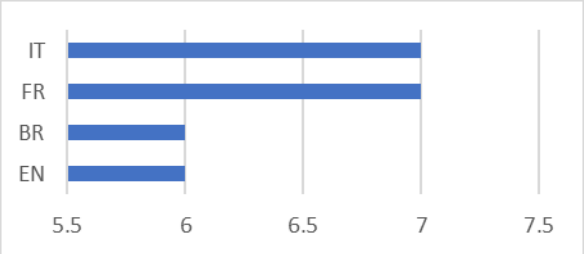

Fig 4. Time to complete after improvements

The Cognitive Study showed improvements across all languages when compared against the source (English) baseline. Before optimization it took between 1.37 to 1.5 times as long to perform the task when compared to following the English source. After optimization this was reduced to 1 to 1.17 times the times, showing that optimization has improved the ability for the user to perform the task in the target languages.

4 Discussion and conclusions

We proposed a method to further leverage an asset currently in use for software development by leveraging Machine Translation and NLP tools such as Regular Expressions. This was done by pre-processing content and optimizing MT engines quickly with one optimization training specifically focused on compliance to terminology. The result is promising showing improvements in many areas based on the analysis of the MT output, and in some cases, is fit for purpose for publishing directly to a customer. In cases where we see drops in usability, the issues stem from the quality of the MT and training/tuning sets. In Italian the UI did not translate well even though the same bitext training content was used in training of all languages. We are confident that training more iterations of the engine to address some of the issues directly would prove useful as issues were predominantly terminology based and may require more weight in the training corpora and or an adjustment of the tuning set.

5 Next Steps

We plan now to expand this test to other languages and improve the current trained MT engines further. In addition, we aim to work with the software development teams to apply more uniform standards to the authoring of Gherkin scenarios and how it is managed as an asset.

Acknowledgements

Acknowledgement is made to the assistance given from Mariana Rolin, Maddalena Benedetto, Sandro Castelletti, Rouslan Placella and Hernan Rey Corvalan from McAfee LLC.

References

Adzic, G. 2009. *Bridging the Communication Gap: Specification by Example and Agile Acceptance Testing*. Neuri, London.

Beck, K. 2002. *Test Driven Development: By Example*. Addison-Wesley, Boston.

Chelimsky, D., Astels, D., Dennis, Z., Hellesoy, A., Helmkamp, B., North, D. 2010. *The RSpec Book: Behaviour Driven Development with RSpec, Cucumber, and Friends*. Pragmatic Programmer, New York.

Roturier, J. 2015. *Assessing a set of Controlled Language rules: Can they improve the performance of commercial Machine Translation systems?* Centre for Translation and Textual Studies, Dublin City University.

Sern, L., Salleh, K., Sulaiman, N., Mohamad, M., Yunos, JBM. 2015. *Comparison of Example-based Learning and Problem-based Learning in Engineering Domain*, Universiti Tun Hussein Onn Malaysia, Johor, Malaysia.

SpecFlow. 2010. *Pragmatic BDD for .NET* http://specflow.org.

Yagel, R., Sarig, O. 2011. *Can executable specifications close the gap between software requirements and implementation?* Proceedings of SKY 2011 International Workshop on Software Engineering, SciTePress, France.