e-Informatica Software Engineering Journal, Volume 15, Issue 1, 2021, pages: 133–162, DOI 10.37190/e-Inf210107

A Systematic Reuse Process for Automated Acceptance Tests: Construction and Elementary Evaluation

Mohsin Irshad*, Kai Petersen**

*Ericsson Sweden AB, Karlskrona, Sweden

**Blekinge Institute of Technology, Karlskrona, Sweden and University of Applied Sciences Flensburg, Germany

mohsin.irshad@bth.se, Kai.Petersen@bth.se

Abstract

Context: Automated acceptance testing validates a product's functionality from the customer's perspective. Text-based automated acceptance tests (AATs) have gained popularity because they link requirements and testing.

Objective: To propose and evaluate a cost-effective systematic reuse process for automated acceptance tests.

Method: A systematic approach, method engineering, is used to construct a systematic reuse process for automated acceptance tests. The techniques to support searching, assessing, adapting the reusable tests are proposed and evaluated. The constructed process is evaluated using (i) qualitative feedback from software practitioners and (ii) a demonstration of the process in an industry setting. The process was evaluated for three constraints: performance expectancy, effort expectancy, and facilitating conditions.

Results: The process consists of eleven activities that support development for reuse, development with reuse, and assessment of the costs and benefits of reuse. During the evaluation, practitioners found the process a useful method to support reuse. In the industrial demonstration, it was noted that the activities in the solution helped in developing an automated acceptance test with reuse faster than creating a test from scratch i.e., searching, assessment and adaptation parts.

Conclusion: The process is found to be useful and relevant to the industry during the preliminary investigation.

Keywords: Software components and reuse, software testing, analysis and verification, agile software development methodologies and practices, software quality

1. Introduction

Software testing provides information on the quality of the software product [1]. An essential aspect of software testing is to verify that newly developed features of a product work according to the agreed requirements, and existing functionality is still working correctly (i.e., regression testing) [2]. With the advent of new technologies and processes, software testing's importance has increased as well. Existing testing approaches, such as unit testing and test-driven development, verify that software is working and developed according to the requirements. However, these approaches do not take into account the testing of business requirements. Acceptance testing verifies the business-critical aspects of software product [3]. The business requirements of a product and the user needs are the focus of acceptance testing [4]. The objective is to validate these business requirements before accepting the software product [3]. These business requirements, in the form of acceptance tests, are automated so that these are tested repeatedly and frequently similar

Submitted: 01 February 2021; Revised: 07 August 2021; Accepted: 09 August 2021; Available online: 25 October 2021

to unit tests [5]. Such automated tests are called automated acceptance tests (AATs) [5]. Studies in AAT have identified that AATs are responsible for high development and maintenance costs when the product is still changing frequently [5].

In agile development methodologies (such as Extreme Programming), the acceptance tests are scripted (using any programming language source code) by the development teams to get continuous feedback on the product's quality, and completeness [6]. Such scripted testing is referred to as "automated acceptance testing", and the tests are known as "automated acceptance tests" (abbreviated as AAT in this study). In agile development methodologies development of AATs starts much early and these AATs are executed frequently [6]. Empirical studies on AATs have established that AATs facilitate the organization's knowledge transfer by describing business requirements as test cases [7].

Software reuse improves productivity and quality during software development [8]. Studies have suggested that software reuse is most beneficial when organizations have a "systematic reuse process" [9]. In a systematic reuse process, the reusable artifacts are developed to support "development for reuse" [10], and these reusable artifacts are easy to find and adapt for reuse purposes [9]. The new development utilizes these reusable artifacts, and this is called "development with reuse" in which the identification, evaluation, and adaptation effort and costs are considerably reduced [10, 11]. A reuse process starts with "reuse assessment", which is a two-step activity, i.e., identification and evaluation of software components for reuse [12]. The evaluation of components for reuse involves the evaluation of the technical aspects (i.e., how to reuse) and the economic aspects (cost of reuse vs. benefits of reuse) of the component [12].

Almeida et al. suggested that the systematic reuse process's decisions should be financially justified (i.e., costs vs. benefits of reuse) before the actual reuse takes place [13]. Studies in literature have suggested that in "development for reuse", the costs of developing the reusable artifacts are high because the artifact should have high quality and artifact reused easily [9]. However, after multiple reuse instances, the benefits become more than the costs [14]. To calculate the reuse costs and reuse benefits, several cost models and metrics exist in the literature. However, many of these metrics and models only capture the costs and benefits of artifacts consisting of source code [15].

In scientific literature, solutions/processes that are constructed to solve a specific business problem can be developed using various existing practices [16]. An integrated solution (or a process) combines multiple activities that were not previously related to each other to address a critical business problem, e.g., finding a defect, fixing the defective code, and verifying the new code change [17]. Method engineering is a discipline of engineering that helps construct new solutions/processes from existing methods [18]. In this study, we have utilized method engineering to support the systematic reuse process of AATs.

This idea of a systematic reuse process can be applied to AATs to address the high development and maintenance costs of AATs. Reusable AATs can be developed to support the development of several new AATs, thus reducing the time to develop, increasing maintainability (by decreasing redundancy) and increasing quality of the automated acceptance tests [19]. Reusability (what to reuse, how to reuse, how to calculate reuse costs, and when to reuse) of AATs is a new area, and very few studies have addressed the reuse in AATs. To the best of our knowledge, there is a lack of suitable methods that support the reusability of AATs. The contribution of this investigation are:

- to provide a systematic reuse process that supports economically justified "development with reuse" and "development for reuse" of text-based AATs.
- to evaluate the proposed systematic reuse process for AATs with the help of experienced practitioners.
- to evaluate the proposed process using an industrial scale example.

Section 2 describes the background and related work, Section 3 contains research approach and the results are described in Section 4. Section 5 contains a discussion on the results of this

investigation and Section 6 describes threats to the validity of this study. Important conclusions are described in Section 7.

2. Background and related work

This section describes the background and related work on AATs.

2.1. Automated acceptance tests (AATs)

An acceptance test is used to verify that the requirements are developed according to the contract/agreed specification [20][21]. The acceptance tests are derived from the customer requirements, and their focus is to verify the completeness of the implementation of requirements [22]. It was proposed that customers specify these acceptance tests, and these tests can be used as a form of requirements [7]. In a literature review [5], Haugset and Hanssen found the following benefits associated with AATs:

- AATs improve the understanding of the domain and product.
- AATs improve the communication with the customer by involving the customer in the test documentation process.
- AATs improve the system's quality by making it safe to make a code change in the product.

Several AAT formats (to write AATs) exist in literature such as source code, text-based domain-specific languages (DSL) and FIT tables [19, 23, 24]. From the existing literature, we have identified four types of AAT formats used by practitioners and researchers. These four formats are described below:

GUI-based formats. GUI-based acceptance tests are written to verify the functionality of GUI-based applications such as websites and mobile applications [25]. These tests emulate the user's actions (e.g., clicks, inputs) and validate the GUI-based application's output. These GUI actions are based on the scenarios described by the software requirements [25].

Code-based format. In the code-based AATs, programming language (e.g., Java, Python) is utilized to develop test execution logic. The test code implements and executes the scenarios in a programming language. The code-based AAT is triggered by a test framework such as JUnit[1]. These types of acceptance tests can be written using the existing practices and tools used for product development [26]. However, a drawback with this format is the difficulty for a customer to be part of the development of the acceptance test process [7].

Data-driven formats. In the data-driven formats, the acceptance tests are written so that input and output values are written in tabular form [27]. These input and corresponding output values are used with the same test execution logic. FitNesse[2] is the most commonly used framework for data-driven acceptance testing. A literature review on AATs identified that most of the studies present in literature are on the FitNesse (21 out of 26 identified studies) [7]. The data-driven formats are also useful when performing regression testing on systems with varying input and output values [28].

Text-based formats. Software practitioners prefer to write requirements in natural language text [29] and the text-based AAT frameworks (such as Cucumber[3], Robot Framework[4]) support the use of these requirements as the test-cases for software products, i.e., acceptance tests. Behavior-driven format, Keyword-driven format, Scenario by example, are popular text-based formats. The text-based AATs are gaining popularity because they help to promote a common understanding of requirements in the organization and increase collaboration among practitioners [30]. An experiment on comparing text-based AATs and FIT tables revealed that text-based AATs are easier to understand. These tests are developed in shorter duration as compared to FIT tables [31].

In behavior-driven format, the tests are written in template of *Given, When and Then* parts. Each behavior-driven test case validates a busi-

[1] https://junit.org
[2] http://fitnesse.org/
[3] https://cucumber.io/
[4] https://robotframework.org

ness requirement described by the customer [32]. Keyword-driven acceptance tests describe an executable requirement written using domain-specific keywords [33]. In specification by example, example scenarios are described in natural language by the stakeholders. These example scenarios are used to execute automated acceptance tests [34]. Each of these formats requires different tools and development frameworks. These formats allow the developers to write the tests in various styles and do not enforce any control over the vocabulary used in the AATs. Each statement of AAT (e.g., "the mobile is sent to customer" in the Scenario 1 below) is connected with a method in code called fixture/glue-code/hook that perform the actions associated with each statement. This method is written in any programming language (Python, Java, etc.) or a library publishes it (e.g., Selenium[5]).

Existing studies on AATs have discussed code--based and FIT tables (as an AAT format); little work is done on the text-based automated acceptance tests [7]. An example of a text-based automated acceptance test (a customer buying a mobile phone) is shown below using *behavior-driven development* format:

Scenario 1: A customer buys a mobile phone.
Given the customer accesses the web shop
And he enters his name and id-card number
When he selects a mobile phone
And customer pays the money using debit card
Then the mobile is sent to customers address

Another automated acceptance test in *keyword-driven* format is shown below verifying a use case of a customer buying a mobile phone.

Scenario 2: A customer buys a mobile phone.
Open Browser To Page example.com/shop
Input name Sam
Input id 939393
Submit information
Click button Buy to purchase mobile
Close Browser

Existing literature has described several approaches to generate automated test cases from manual acceptance tests automatically. Paiva et al. proposed a process where requirements are transformed into manual acceptance tests using a domain-specific language called Requirements Specification Language (RSL) [35]. Later, these RSL-based test cases are transformed into an automated test-scripts executable using Robot Framework. Later, their proposed process is evaluated using an example application. Soeken et al. provided a semi-automated approach to generate automated acceptance tests from natural language requirements [36]. The phrases from requirements are used to extract stubs executing the test scenarios. Later, the proposed approach was evaluated using an example case. Research studies have also utilized IDE-plugins and tools to generate automated acceptance tests from the use cases such as Fitclipse by Deng et al. [37] and Test-Duo by Hsieh et al. [38]. Such tools are often limited to a specific framework or acceptance test formats.

2.2. Systematic software reuse

Software reuse describes the practice of reusing existing software artifacts for developing a new product or maintaining an old product [39]. The field has been the subject of research for several years, and different aspects of software reuse such as costs, testing, artifacts, the level of reuse, stakeholders, and reuse processes have been investigated thoroughly by the researchers [40–44]. Software reuse can take place at any time during a project's life-cycle, starting from the requirements analysis phase to the maintenance phase [43, 45].

Organizations apply systematic software reuse to improve the quality, and productivity [41]. Various software development artifacts (requirements, test cases, code) can be systematically reused across the entire development process [46]. Lam et al. described a 10-step process to support the systematic reuse of software requirements [47]. The study suggested that a systematic reuse process can be successful when the organization produces and consumes reusable software artifacts. Research studies have identi-

[5]https://www.selenium.dev/

fied the following characteristics of a systematic software reuse process [12, 48]:

- *Development for reuse:* reusable artifacts are produced to support the organization's future software development needs.
- *Development with reuse:* new development occurs using reusable artifacts whenever possible.
- Guidelines (or techniques) to produce reusable artifacts exist and are practiced in the organization.
- Guidelines (or techniques) on reusing an artifact exist and are used in the organization.
- Development teams consider the extra costs and risks associated with the reuse of artifacts.

2.3. Related work: reuse in automated acceptance tests

The concept of text-based AATs is new, and their textual nature differentiates these tests from the conventional code-based test-cases, thus requiring a different approach for reuse.

In a study on reuse of AATs, Landhaußer and Genaid suggested an ontology-based approach (F-TRec) that enables reusing tests written in natural language. They found that approach lacked precision when retrieving test steps for reuse [49]. Crispin and House provided a tool to create AATs that can be reused across different modules [50]. They suggested that before reusing any AATs to develop a new AAT, practitioners should evaluate the effort spent writing and maintaining the new AATs. They also claimed that their proposed method could quickly create new test cases from the existing reusable tests, thus reducing development time and increasing quality. Rahman and Gao recommended an architecture that enables the reuse of Behavior-driven acceptance tests across multiple repositories [19]. They claim that their proposed approach can reduce maintenance costs, which are considered one of the pain points of AATs. Binamungu et al., in their study on the maintenance of BDD tests, found duplication as a critical challenge facing the practitioners [51]. Irshad et al. also found duplication among AATs in their study refactoring BBD specifications [52]. This duplication can be decreased with the help of an increase in reuse.

Liebel et al. identified costs related to writing the automated acceptance tests as one of the major problems of these acceptance tests [25]. In their industrial case study, Hanssen et al. suggested that the benefits of automated acceptance tests should be weighed against the costs associated with these tests [23]. They found that automated acceptance tests verifying the graphical user interface (GUI) are often hard to develop and maintain. GUI-tests are unstable and require more maintenance. Angmo and Sharma evaluated AAT tools and proposed that cost-benefits analysis should be done when considering a tool for automated acceptance tests [53]. Shelly and Frank conducted a literature review on story test-driven development. They found that cost of writing AATs is high, and many organizations do not have a budget to account for this high-cost [54].

Xie describes time as the measure of cost in development and maintenance of AATs [55]. Haugset and Stalhane claim that AAT can benefit organizations in two ways (i) increasing the correctness of requirements and (ii) automated test-ability of requirements. Borg and Kropp described a tool that helps maintain and refactor automated acceptance tests, reducing their maintenance costs [56]. To support reuse and reduce costs, Schwarz et al. introduced and evaluated a plugin (Eclipse[6] IDE plugin) that rapidly develops AATs. They claim that using this plugin can quickly develop automated acceptance tests quickly [57].

In short, we identified the following gaps in the literature, and our investigation attempts to provide a solution to these research gaps.

- **Lack of support to systematic reuse of AATs:** Only one study is identified that provides methods for "developing with reuse", and "developing for reuse". However, the study's approach is limited to the use of a specific tool provided by the authors of the study [50].
- **Lack of generic reuse practices supporting diverse AAT formats:** There is a need for a generic practice supporting the reuse of

[6]https://www.eclipse.org/

all types of AAT formats. Several formats to write text-based AATs exist in research and practice (BDD, keyword-drive, specification by example).
- **Lack of means to calculate benefits of reusing AATs:** AATs are costly to write and reuse, and costs should be considered when reusing AATs [7, 23]. An instrument to calculate the reuse costs of AATs may help perform cost vs. benefits analysis of reuse instances.

This study complements the existing work by providing a systematic reuse process that supports economically justifiable "development for reuse", and "development with reuse" of AATs, independent of any particular text-based AAT formats, independent of any particular tools and frameworks.

3. Research approach

This section describes the research questions, study execution, data collection, and data analysis performed during this investigation.

3.1. Research questions

In order to achieve the objectives of this study, we have devised the following research questions:
- **RQ 1:** How can the cost-effective systematic reuse of text-based AATs be achieved?
  This research question details the establishment of a systematic reuse process for AATs along with suitable activities and techniques used in the process. The proposed process incorporates the cost-benefit aspects of AATs when evaluating reuse opportunities.
- **RQ 2:** How does the systematic reuse process, from RQ 1, perform concerning performance expectancy, effort expectancy, and facilitating conditions in the industrial context?
  RQ 2 addresses the preliminary industrial evaluation of the systematic reuse process of AATs with the help of industry professionals and practical demonstration.

3.2. Study execution

The study is executed using method engineering that is a research framework to develop new tools and methods. Method engineering consists of individual method fragments that can combine to form a project-specific (or product-specific) customized method [58]. Method engineering is applied using the following phases (defined by Mayer [18]):
- *Document motivation:* the motivation to develop a new process is identified and documented.
- *Search for existing methods:* the existing methods are identified that may help in the new process.
- *Tailor existing methods:* the identified methods are adapted to suit the needs of the process.
- *Design method application technique:* a new process is formed using the modified existing methods.
- *Test candidate design elements:* the process (and its components) are evaluated to identify potential shortcomings and modifications.
- *Refine method design:* The process is modified/refined based on the evaluation.

The first four phases help in constructing a new process, i.e., see Figure 1. Later, the last two phases (see Figure 1) evaluates and improve the new process. The study approach and the research questions are described in Figure 1. The first step (construction of process) identifies the motivation, requirements and develops a new systematic reuse process for text-based AATs. The second step (Evaluation and refining of the process) evaluates the new process by assessing the performance expectancy, effort expectancy, and facilitating conditions of the process, as suggested by [61]. The evaluation consists of:
- qualitative feedback on the process from experienced software practitioners (also referred to as static validation in [62]),
- by demonstrating the usage of the proposed process in an industrial application by an author.

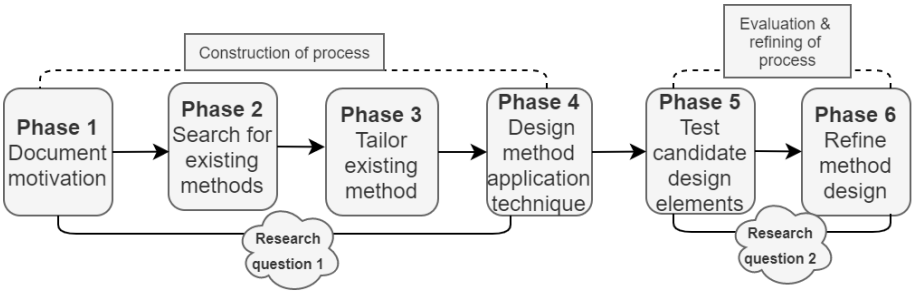

Figure 1. Research approach inspired by method engineering [18]

The sections below provide the details of the construction of the process and its evaluation (i.e., research method, the data collection, and analysis). The final version of the method is released after incorporating the feedback from the evaluation.

3.2.1. Construction of process – using method engineering

This step is used to construct a process supporting "development with reuse", and "development for reuse" of AATs also considering the costs vs. benefits of AATs, i.e., a systematic reuse process. The details of the four phases of this step are described below.

Phase 1: Document motivation. The existing studies that report the problems related to the reuse of AATs were identified and examined. The authors read the literature reviews and mapping studies on AATs to identify supporting methods for reusing AATs. We have focused on text-based (non-code) AATs only as the reuse of other types of AATs (code, FIT Tables) is already discussed in existing [7]. One of the authors conducted the following step using Google Scholar[7].

- Find and examine the literature reviews/ mapping studies on AATs. Table 1 describes keywords and identified studies.
- Examine the reference/citations in identified literature reviews.
- Manually analyzes the identified studies to find issues linked with AATs.
- Manually analyze the identified studies to find details on the reuse of AATs.

The identified literature reviews/mapping studies on the AATs [5, 7, 54, 59, 60] do not explicitly discuss the reuse of AATs. However, we identified studies using citations and references to review studies. The analysis (documented Section 2) concludes that a supporting process is needed that should address the three requirements/needs: AAT format, AAT reuse process, and AAT reuse costs. These three requirements/ needs are described below.

AAT format: The process should be independent of the AAT format. Different AAT frameworks support other formats, e.g., in Cucumber, BDD format is used, and the Robot Framework supports keyword-driven and BDD formats. The reuse process of AATs should be independent of formats dictated by different AAT frameworks.

AAT reuse process: The reuse process of AAT should support "develop for reuse" and "develop with reuse". The process should contain activities to search for a reusable AAT, assessment of AAT for reuse, and adaption for reusing AATs.

Table 1. Keywords and identified studies

Keywords	Identified review studies
Automated Acceptance Testing review	[5], [7]
Acceptance Testing review	[5], [59]
Story-driven review	[54]
BDD Review	[60]

[7]https://scholar.google.com/

AAT reuse costs: The process should incorporate activities for assessment of reuse costs of an AAT. Previous research has shown the high cost of writing and reusing AATs [53, 63].

Phase 2: Search for existing methods. In this phase, the existing methods are identified to match the needs/requirements of an AATs reuse process identified in the previous phase, i.e., AAT format, AAT reuse process, and AAT reuse costs.

AAT format: The following AAT formats are identified Behavior-driven tests, story-based test format, specification by example format, and keyword-driven tests. These formats were identified using the existing literature reviews on AATs ([7, 23, 59, 64]).

AAT reuse process: Mili et al. [65] suggested three stages for a reuse process: (i) finding the reusable artifact, (ii) assessing the relevance of reusable artifact, and (iii) adaptation of the artifact for reuse. For the first and second stage (finding and assessing relevance), techniques suitable for text-based artifacts were identified that could help in these stages. The two suggested approaches are (i) normalized compression distance (NCD) [66], and text classification using machine learning [67]. For the third stage (an adaptation of AATs for reuse), no existing study provides guidelines on adapting AATs for reuse. Since software requirements and AATs are text--based artifacts; therefore, we propose that the reuse methods used in software requirements can support reuse of text-based AATs [68].

AAT reuse costs: Software reuse cost models are categorized into three categories (by [14]):

1. Return-on-investment models (ROI) that measure benefits after an organization has invested in developing reusable artifacts.
2. Cost-benefit models that are used for making better decisions on reuse investments.
3. Cost-avoidance models that help in calculating costs avoided through reuse of artifacts [14].

We believe that cost-avoidance models are best suited for reusability assessment since these do not assume that an upfront reuse related investment was made, as required by ROI and cost-benefit methods [14]. The methods provided in the review study on cost-avoidance through reuse were identified for capturing reuse costs of AATs [15].

Phase 3: Tailor existing method. The identified methods from the previous phase are tailored to address the needs of a systematic reuse process of AATs.

AAT format: For AAT formats, no tailoring is needed.

AAT reuse process: In the AAT reuse process, to find and assess the AAT candidates for reuse, two techniques (Normalized compression distance and text classification using machine learning) are tailored to be used for AATs. The AATs are text-based, and the AAT content is structured so that each line acts as a single independent, reusable test statement (see examples in Section 2). The two techniques were implemented using scripts (available at [69]) to apply them on the AAT suite. Section 4 described the working of these two techniques.

For the AAT adaption template (to support "develop for reuse"), four templates (Structuring, Matching, Analogy, Parametrization) from the identified review study [68] were selected for their applicability over AATs. The reuse of software requirements inspires these templates. Section 4 describes the details of these templates.

AAT reuse costs: The study on reuse cost avoidance methods described four approaches to calculate reuse cost-avoidance [15]. Only two of the reuse cost methods ([70] and [15]) are applicable over non-code artifacts such as text-based AATs. These two methods and their corresponding metrics do not require any tailoring for AATs.

Phase 4: Design method application technique: In this phase, the identified activities concerning the components of the process are assembled to form a process.

The three identified requirements/needs (AAT format, reuse process, and reuse costs) and their identified methods are converted into activities of a process. As discussed in Section 2.3, AATs can be costly to write and maintain; therefore, a check is introduced in the process that lets practitioners decide when to write a new AAT and when to reuse an existing AAT. Furthermore, it was decided among the authors to divide the

activities into two levels, i.e., organizational level activities and test developer-level activities. These activities are described below:

Organizational/Team level activities and decisions: The organization-level activities are decided when setting up the reuse related process, and these are rarely changed. Some examples of these activities are the format used in the organization (e.g., BDD, keywords files), the support for the reuse approach, and the test framework used in the organization. The actor in these activities is the test architect.

Test developer-level activities and decisions: The activities performed during this level are the primary activities relevant to developing a new AAT. An example of such activities is selecting an ATT for reusing purposes. Test developers perform these activities. The activities are guided by the choices made during the organizational activities, e.g., which artifact to reuse, how to modify an artifact for reuse.

The constructed systematic reuse process is documented in Section 4.1.

3.2.2. Evaluation and refining of process

After developing a reuse process, the next step involves the evaluation and refinement of the process. The sections below describe the types of evaluation and refinement steps.

Phase 5: Test candidate design elements. After designing the process and its activities, the next step evaluates the process for its usefulness concerning the reuse of AATs. [62] has suggested that validation with practitioners helps assess the industrial usage of the process before applying it in the industry.

Petersen and Wohlin described six context facets for explaining the context of the evaluation [71]. The evaluation in this study is conducted in the context of large-scale software development. The complexity concerning the reuse of AATs is considered high for large-scale organizations with a considerably large test base. The development process is assumed to be an agile development method where new requirements, development, and testing are conducted in each iteration.

This survey questionnaire's participants belong to two large telecommunication organizations, and the industrial demonstration is conducted in one organization's product verification unit. The details of the evaluation are described below.

Qualitative feedback of software practitioners. In this evaluation, software practitioners provided feedback on the proposed systematic reuse process.

Objective: The objective of this evaluation was to identify generic findings on the applicability of the process and sanity-check the process for performance expectancy, effort expectancy and facilitating conditions as suggested in unified theory of acceptance and use of technology (UTAUT) by [61].

Wohlin et al. describe "exploratory surveys" as a way to validate the proposals before a thorough investigation is conducted [72]. As the first step, the proposed systematic reuse process in this study was evaluated using an exploratory survey to improve the process before implementing and evaluating it in the industry, which requires a longitudinal study. This longitudinal study is planned as the next step after this study because it requires resources, budget, and changes in the organization's current ways of working. Wohlin et al. suggested questionnaires and interviews as two data collection methods during the surveys. We developed a questionnaire and asked the subjects to fill the questionnaire during an online session (for direct interaction like an interview).

Questionnaire design: In this evaluation, a questionnaire (available at [73]) is developed that contained four parts: (a) description of reuse in automated acceptance testing, (b) description of our proposed process, (c) an example of explaining the usage of our proposed process, and (d) practitioners feedback on the proposed process. The questionnaire's design, as per Molléri et al. can be classified as "self-administrated", i.e., online form [74]. This questionnaire is used to execute the industrial evaluation. The details on designing the questionnaire, selecting participants, data collection, and data analysis are provided below.

The following questions (under each UTAUT construct) were part of the questionnaire.

Performance expectancy. This construct describes "the degree to which technology will benefit users" [61].

- Question 1: In your opinion, what are the benefits of using the proposed process?
- Question 2: What are the drawbacks/limitations of the process? How can we improve/revise the process?

Effort expectancy. This construct describes "the degree to which technology is easy to use" [61].

- Question 3: In your opinion, how easy is it to use this process?
- Question 4: Do you have any recommendations to improve the ease of use?

Facilitating Condition. This construct describes "the degree to which technology helps in performing a task" [61].

- Question 5: Are there any other steps that should be added to the process?
- Question 6: Are their steps that should be removed from the process? If yes, then kindly list those items here and also state why do you recommend removing them?

Subjects: Initially, two academic researchers (not authors) working on AATs were invited to review the questionnaire and process for sanity-check. They suggested improvement in the questionnaire (i.e., text, process flow). After incorporating the input of the researchers, we conducted an industrial evaluation.

During the industrial evaluation, five industry participants with knowledge of AATs in large-scale products and considerable working experience are considered. As the reuse of AATs is a relatively new area, it is difficult to find participants with relevant experience. Five participants evaluated the process and provided their feedback on the process. These participants were selected from two large-scale organizations with more than five years of experience with development. The background of the participants is provided in Table 2.

Study execution: The authors have presented the study's purpose, the working of the process, and its activities to the participants in an online meeting. The participating subjects asked questions about things they do not understand during the presentation of the process. The details were provided to the inquiring participants. This step was conducted to overcome the survey questionnaires' critique that subjects might not understand the concepts and processes by reading the questionnaire.

In the next step, the participants were provided the link to the survey questionnaire, and they were asked to fill in the information. The author remained online while the respondents filled out the form to have direct interaction and allow participants to ask follow-up questions. These sessions lasted between 45–65 minutes, as shown in Table 2. These online sessions helped improve the qualitative feedback and relevance of the feedback for the reuse of AATs.

All five participants responded to the survey questionnaire in the presence of an author. In the form of responses to the questionnaire, the feedback is evaluated using the "Constant comparison" method [75]. This method can help in the identification of common themes present in the qualitative data. During the analysis, the responses are divided into themes related to the process's benefits, usefulness, and completeness. Later, the final process is improved based on identified themes corresponding to performance expectancy, effort expectancy, and facilitating conditions.

Demonstrating industrial application of the proposed process – an example. Stous-

Table 2. Background of the software practitioners (P1–P5) participating in the evaluation

	Work experience	Experience in AAT	Role	Product type	Online session
P1	12 years	5+ years	Developer	Large-scale	60 minutes
P2	17 years	5+ years	Architect	Large-scale	45 minutes
P3	12 years	4 years	Test developer	Large-scale	55 minutes
P4	5 years	3 years	Test developer	Large-scale	65 minutes
P5	13 years	2 years	Developer	Large-scale	50 minutes

trup identified that lack of practicality or an excessive level of complexity results in the failure of seemingly successful research projects when these projects are applied in the industry [76]. A demonstration is conducted in an industrial setting using the proposed systematic reuse process to address this concern.

Objective: The purpose of this evaluation is to check the feasibility of implementing the process in the context of a real software development environment using the existing tools and knowledge present in the organization (e.g., metrics to identify reuse costs). This evaluation will help identify lessons regarding each activity before the process is applied and evaluated in the industry without authors' involvement.

The author (working in the organization) evaluated details related to:

– the effort to create an AAT using the proposed process,
– the number of tasks performed in each activity of the proposed process,
– details of tasks performed in each activity of the proposed process.

This demonstration took place in one of the system verification units of an organization that develops a large-scale product consisting of 28 micro-services. The unit of analysis is the AAT suite used by the end-to-end verification team. Four experienced test developers manage the AAT test suite. The AAT suite was introduced in the year 2016 and contained 87 system-level AATs. Each system level AAT is based on a complete use-case. The AATs are written in keyword-driven format (text-based), with fixtures written in Java. The AAT suite is extended when the test developers find a stable product no longer modified by the development teams. The AATs are executed each night using Jenkins to build, execute the AAT, and generate AAT reports.

Table 3. Activities in a systematic reuse process for automated acceptance tests (AATs)

Activity	Input	Output	Actor	Type
A1: Select keywords	Requirements in natural language	Keywords extracted from requirements to find reusable AAT .	Test developer	Manual
A2: Select AAT format	AAT formats (an organization can seek help from literature if it does not have a pre-decided format)	Selected AAT type, e.g., BDD	Test Architect	Manual
A3: Select artifact adaptation template	List of adaptation approaches to support "develop for reuse". See 4.1.3	Selected approach to adapt AAT for reuse and to help in search	Test Architect	Manual
A4: Search for reusable AATs	Keywords from A1 and a search technique. Details in 4.1.4	The list of AATs matching keywords	Test developer	Automated
A5: Assess relevance of AATs	The AATs matching keywords from output of A4	The potentially reusable AATs	Test developer	Manual
A6: Select reuse cost method and metrics	The available metrics, the reuse cost calculation method. Details in 4.1.6	A selected reuse cost method	Test developer	Manual
A7: Calculate cost of reuse	The value of metrics and the reuse cost method	Evaluation if reuse is beneficial or not	Test developer	Automated
A8: Develop new AAT by reuse	The selected reusable AAT and adaptation technique from activity A3	A new AAT supporting "development for reuse"	Test developer	Manual
A9: Develop a new AAT	Software requirements, adaptation technique from activity A3	A new AAT supporting "development for reuse"	Test developer	Manual
A10: Add new AAT to repository	A new (or reused) AAT test-case	A new AAT is added to the repository	Test developer	Manual

IntellJ[8] was used as an IDE for developing the AAT. The AATs are stored in a Git[9] repository. The system under test (SUT) is hosted on a remote server, and the development takes place on the local machine. The AAT is executed against SUT from the local machine, but after finalizing the AAT, the AAT execution is automated as part of the AAT suite that is executed frequently using a continuous integration server.

One of the authors (who is not a developer of the existing AATs) applied the process's activities to develop a new AAT with reuse. The new AAT is to verify that a REST interface's performance is within limits decided by the requirements engineers. The activities described in Table 3 are followed to evaluate the proposed process's flow.

According to the classification provided by Lethbridge et al. [77], third-degree data collection was utilized in this study (i.e., historical data on the case or using the compiled information). For the searching and assessment activities, a script (described in 4.1.4) is used to search in the repository and identify the relevant AATs. The search and assessment data is available at [78]. The cost model used in this demonstration needed historical information on the person-hours previously spent on similar AAT [15]. This information was extracted from the Git repository using the "git history <AAT-File-Name>" command. The time taken was noted for each activity by one of the authors.

The data analysis was performed on the collected quantitative and qualitative data. The quantitative data is the time taken during activity, the number of steps performed in each activity (ease of use), and qualitative data involves the test developer's observations (i.e., author). The results from the data analysis are described in Section 4.2.2.

Phase 6: Refine method design. In this phase, the proposed method is modified based on the feedback from the evaluation. The identified themes related to the method's improvement are considered, and the method is modified. The feedback from the evaluation and the changes suggested during the evaluation are described in Section 4.2. The final version of the proposed process is present in Section 4.3.

4. Results

This section describes the outcome of the two research questions and the final systematic reuse process. The first research question described a cost-effective systematic reuse process, its activities, and techniques applicable to the activities. The second research question describes the industrial evaluation of the systematic reuse process. Later, the final version of the systematic reuse process is described.

4.1. Constructed solution: A systematic reuse process for AATs

The systematic reuse process for AATs supports activities and techniques to (i) develop for reuse, (ii) develop with reuse, and (iii) methods and metrics to calculate the reuse costs of AATs. Figure 2 shows the constructed process to support systematic reuse of AATs and the details of each activity is described in Table 3.

In the first activity, the practitioner analyzes the requirements and identifies keywords relevant to the new AAT. Next, an AAT artifact format is selected, e.g., BDD. After selecting the artifact, an approach to facilitate "development for reuse" is selected. The next two activities search and assess the relevance of AAT artifacts for reuse. If no suitable artifact is found, then a new AAT is developed from scratch. If existing relevant AATs are found, then the next activity is to calculate the cost of reuse. If the reuse is presumed cheaper, a new AAT is developed by adapting the existing artifact according to the approach selected for development for reuse. In the last activity, a new reusable AAT is added to the repository. Table 3 provides the inputs, outputs and actor of each activity.

In the sections below, each activity in the process is described below with an example.

[8]https://www.jetbrains.com/idea/
[9]https://git-scm.com/

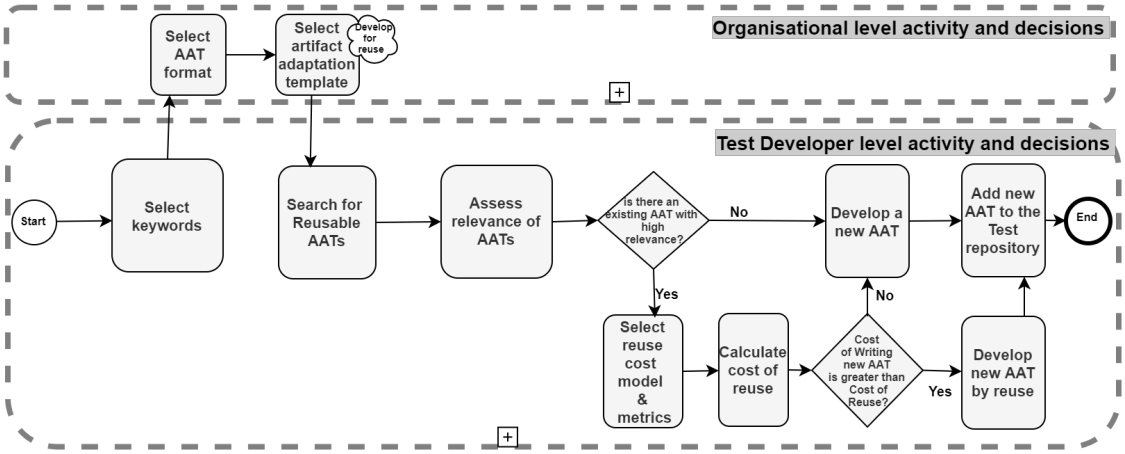

Figure 2. A systematic reuse process for automated acceptance tests (AATs)

4.1.1. A1: Select keywords

Software requirements are often written in natural language [29], and in this activity, relevant keywords are selected that represent the requirements. These keywords will help in searching for reusable AATs in the existing test-base. This is a manual activity performed by a test developer. This activity may be performed in several iterations before finalizing the keywords; e.g., an initial selection of keywords may not provide good search results; therefore, it needs several revisions (or discussion with experts) before reaching the concluding choices.

Example: The test developer has the requirements: (i) "As a user I should be able to register my new account", and (ii) "As a user, I am able to view the products using my account". In this example, the relevant keywords selected from the requirements are "register account", "view products".

4.1.2. A2: Select AAT format

The text-based AATs are written in a variety of formats. Each of these formats has a unique way of writing AATs, e.g., stories, specifications, behaviors, features. Therefore, it is necessary to select the format in which the new AATs are written or reused, e.g., a selection from Behavior-driven tests, story-based tests, feature files, keyword-driven tests. This activity is performed manually.

Example: As an example, we can assume that the organization writes AAT in a Behavior-driven format; therefore, BDD is selected as the format of AAT.

4.1.3. A3: Select artifact adaptation template

In this activity, a reuse adaptation template is selected for writing a new AAT from existing artifacts. The objective of this activity is to support "development for reuse". The new AAT will be modified and saved according to the artifact adaptation template. The four templates supporting development for the reuse of AATs are structuring, matching, analogy, and parameterization as described in Table 4. This selection of artifact adaptation template is a manual activity. A detailed discussion on the reuse adaptation approaches for text-based software requirements is provided by Irshad et al. [68].

Example: In this case, "Structuring" (from Table 4) is selected as reuse adaptation template because BDD test cases are written in structured format of "Given, When, and Then" [32].

4.1.4. A4: Search for reusable AATs

A vital activity of the constructed process is to search for reusable AATs in the repository. This search is conducted using the keywords from activity A1. The search operation can be implemented using an automated script that searches for AATs matching the keywords.

Table 4. Templates supporting "development for reuse". Inspired by [68]

Approach	Description
Structuring	Reusable AATs are saved in specific/pre-defined format to reuse, e.g., directory structure.
Matching	Reusable AATs are saved in formats such AATs are retrieved using matching, e.g., supports search using lexical or semantic matching.
Analogy	Reusable AATs can be storied in languages/formats that support retrieval using analogy-based approaches, e.g., special languages supporting analogical matching of AATs.
Parameterization	AATs can support parameterization for reuse, e.g., the use of variables in keywords. Many AAT frameworks (Robot, Cucumber) support this adaptation approach.

We have identified two techniques that are useful for searching for text-based reusable AATs. These techniques are applicable over the text content of AATs only and do not consider the fixture/hooks/glue-code of AAT. We have provided an automated script for each of these techniques as part of the reuse process. These techniques are described below.

Normalized compression distance (NCD) With this technique, the similarity between an AAT and keywords is calculated using a compression algorithm [79]. In this study's context, NCD helps calculate the pair-wise distance between all the AATs present in the test base against the keywords identified. Later, this compression distance helps in the assessment of similar and dissimilar AATs. NCD can be defined by the following equation [79]:

$$\mathrm{NCD}(s1, s2) = \frac{Z(s1s2) - \min\{Z(s1), Z(s2)\}}{\max\{Z(s1), Z(s2)\}}, \tag{1}$$

Here, $s1$ is an automated acceptance test present in the test suite, and $s2$ is a keyword used to search and assess reusable automated acceptance tests. Z represents the compressor used for the calculation of NCD. $Z(s1)$ represents the compressed size of AAT $s1$, $Z(s2)$ represents the compressed size of AAT $s2$ and $Z(s1s2)$ represents the compressed size of the concatenation of $s1$ and $s2$. NCD values lie between 0 and 1, where 0 means that the BDD specifications are similar, while 1 represents that they are entirely different. A script implementing NCD is provided as part of the proposed process [69].

Text-classification using machine learning Text-classification can help in identifying reusable AATs by training a classifier using a supervised machine-learning algorithm [67]. For text-classification commonly used machine learning-based algorithms are Naive Bayes and Support Vector Machines [80]. A machine learning-based algorithm improves the classification process because it considers the domain-specific language used in the AAT instead of using Wikipedia or large-text databases present over the internet [67]. The existing AAT suite is used to train the text classifier using each AATs title as a category. Later, this text classifier helps in suggesting the closest matching reusable AAT cases to the selected keywords. The following sequence of steps is followed when using machine-learning based text classification to search for and assess a reusable AAT.

Step 1: Place each AAT in a separate file where each file's name is unique. This is needed to allow ML-algorithm to assign a category to each AAT in the training set.

Step 2: Load the training AAT files into memory.

Step 3: Extract features from AAT files using Bag of Words.

Step 4: Train a classifier from these features.

Step 5: Use the search keywords to query the classifier to identify the reusable AAT.

It is important to note that Step 1 to Step 5 are executed only if new changes are introduced in the AAT suite. A script providing the implementation from Step 1 to Step 5 is provided as part of this proposed process (available at [69]).

Table 5. NCD values generated by the automated-script

Selected keyword	Scenario 1 NCD value	Scenario 2 NCD value
view products	0.620	0.290
register account	0.720	0.850

Example: For the sake of simplicity, we assume that there are only two AAT scenarios in the repository and a test-developer wants to use the keywords (from A1) to search for the closest matching scenario to the keywords. The test-developer uses the automated-script (See [69]) to perform this activity.

Scenario 1: *A user deletes a product in the system*
Given A product is configured in the system
When User sends a Delete request to delete the products
Then User is able to delete the product
Scenario 2: *A user view a product in the system*
Given A product is configured in the system
When User sends a Get request to fetch the products
Then User is able to view the product

The distance measures (such as NCD) represent a mathematical concept of similarity [81, 82]. The similarity is high when the distance between objects (in comparison) is low, i.e., a value closer to "0". An advantage of distance measure is that they can classify the similarity and dissimilarity of two objects on a numerical rating scale by suggesting how closely similar or how different objects are from each other, i.e., 0.90 means very dissimilar, 0.75 means dissimilar, 0.248 means similar, 0.05 means very similar. The text-classification approaches often classify in the binary format, i.e., two objects are alike or different.

The test developer selects "Normalized Compression Distance (NCD)" to retrieve the reusable AATs with help of keywords "register account", "view products". The search activity is performed using the script implementing NCD, and a pairwise comparison of each keyword and scenario is conducted. For each comparison, a value between "0" and "1" is produced. The values are shown in Table 5 e.g., Scenario 2 and keyword "view product" has lower NCD value (0.290), showing higher similarity because keywords "view", "products" are found in Scenario 2 but not in Scenario 1.

4.1.5. A5: Assess relevance of AATs

In this activity, an assessment is made on the relevance of the identified reusable AATs from the searching activity. This assessment involves how closely the identified AATs are related to the requirements. This activity requires domain knowledge and understanding of the existing AATs; therefore, it is a manual activity, e.g., if several closely matching reusable AATs are identified, a manual assessment to select the most suitable reusable AAT.

Example: The search results of activity A4 (in Table 5) contains results from the search operation. Table 5 shows that the NCD values between the keywords and two scenarios. As stated earlier, an NCD value closer to 0 means higher similarity, and a value closer to 1 means lower similarity. Scenario 2 and keyword "view product" has value 0.290, showing higher similarity and relevance as a candidate for reuse.

4.1.6. A6: Select reuse cost method and metrics

In this activity, a method is selected to calculate and compare costs of developing a new AAT by reusing an existing AAT and costs of creating a new AAT from scratch. The metrics required to apply the method are selected in this activity. This activity is necessary because the development of text-based AATs is known for higher costs, and in some cases developing a new AAT from scratch can provide more savings than reusing an AAT. The reuse cost calculation methods and metrics used in the code-based artifacts are not applicable over the text-based AATs because they use "lines of source code" (or indirect metrics such as complexity and function points) as an essential metric to calculate the

cost. The selection of the cost model depends on the following factors:

1. Maturity of the existing reuse process in the organization, i.e., ad-hoc or systematic reuse.
2. Easiness to collect the required metrics: Each model uses various metrics to calculate reuse-related costs. Therefore, a key consideration when selecting a model is the availability of the required metrics in the organization.

Manual estimation can be used if the needed metrics are not available or time-consuming to collect these metrics. The manual estimate can be based on (i) the size of a similar task completed previously, (ii) the complexity of the task, and (iii) the experience level of the software practitioners.

We identified two methods and their metrics that could help in calculating the reuse related costs of AATs. These methods are described below.

Amar and Coffey's reuse cost calculation method [70] Amar and Coffey attempted to provide a unified unit of measure capable of calculating the costs that occurred during a reuse instance [70]. They claim that the reuse-related expenses are directly related to the time spent on reuse activities. They claim that the time spent during each of these activities should be considered. The method and metrics proposed by Amar and Coffey are described below.

$$\% \text{ of reuse cost avoided per AAT} = \left[S - (T+U) \times \frac{\frac{N}{i}}{B} - S \times \frac{M}{B} \right] \times 100 \quad (2)$$

where S is the search hit rate (i.e., the number of attempted searches yielding a reuse instance is divided by the number of total searches), T is time to locate a reusable AAT, U is time to understand the AAT, N is a number of AATs analyzed, i is a number of reuse instances of AAT, M is time to integrate (reuse) the (e.g., after adapting a reusable artifact), and B is time to develop an AAT from scratch. Further details on the working and evaluation of this method are found in study [70].

Irshad et al.'s reuse cost calculation method [15] According to Irshad et al., their proposed metric can calculate cost avoidance by reusing any software artifact [15]. Their provided instrument considers the effort spent reusing an artifact vs. effort spent on developing it from scratch. The basic formula is described below in the context of this study.

$$\text{Reuse Costs} = (O - I) \times H \quad (3)$$

where O is the personnel hours when an AAT is developed from scratch, I is the personnel hours spent on the adaptation of reusable AAT (i.e., changing an artifact to match the needs) and, H is the cost of one personnel hour. According to Irshad et al., historical data or expert opinion can estimate the personal hours when AAT is developed from scratch. Further details on the instrument and its evaluation can be found in the study [15].

Example: For the sake of this example, we can assume that the test developer selects a reuse cost model [15] i.e., Equation 3.

4.1.7. A7: Calculate cost of reuse

In this activity, the selected cost-avoidance method and its metrics are applied to calculate the cost of reusing the identified AATs. The reuse costs are only calculated for the artifact that is deemed relevant. The activity can help in decisions like should a new test be developed or existing ones are modified. This activity can be performed with the help of automated scripts or manually.

Example: Using the metrics and method, the test developer calculates that the new development cost is 40 person-hours, and the cost of Reusing by Adaptation Scenario 2 is 30 person-hours. Therefore, it makes sense to reuse Scenario 2 to develop a new AAT as it saves ten person-hours.

4.1.8. A8: Develop new AAT by reuse

In this activity, a new AAT is developed by reusing the reusable AAT. This activity takes place if the reuse is deemed as cost-effective in

activity A7. During this activity, one of the adaptation templates is applied to develop the new AAT that supports future reuse opportunities (present in Section 4.1.3).

Test developer performs this activity manually. When developing a new AAT by reuse (using the proposed process), the following pre-requisites are needed:

– The format in which the new AAT will be written so that it becomes a reusable asset for the future.
– The cost-efficient reusable test case(or test cases) which will be used to develop a new AAT (or parts of a new AAT).

Example: When the reuse is considered beneficial, we assume that the test developer uses "Structuring" as a reuse approach to developing a new AAT scenario that test registration of a user account and viewing products below. The grey colored lines show the reuse from the existing scenario (Scenario 2 described in activity A4).

Scenario 3: *As a user I should be able to create account and view product*
Given **A product is configured in the system**
AND A login system is present for the product
AND A user is able to access the GUI Registration system
When User Registration is successful
AND **User sends a Get request to fetch the products**
Then **User is able to view the product**
AND a new user account is created

4.1.9. A9: Develop a new AAT

In this activity, the development of a new AAT from scratch takes place. This activity happens if the reuse from existing AATs is not possible, i.e., no relevant reusable AAT or reuse has unfavorable cost vs. benefits. While developing the new AATs, the vocabulary and existing rules of writing an AAT are considered. An adaptation templates is selected to develop the new AAT that supports future reuse opportunities (present in Section 4.1.3). A test developer performs this activity manually.

The example from activity A8 (Scenario 3) shows a new AAT if developed from scratch.

4.1.10. A10: Add new AAT to repository

The final activity is to add the newly developed (or reused) AAT into the existing test repository. These steps help in improving and developing test-base. This activity is performed manually.

An example of the repository is a Git repository[10], which is used by the majority of the software development organization to version-control the software artifacts.

4.2. Evaluation and refining of systematic reuse process

We first evaluated the proposed process with experienced practitioners who have first-hand experience working with AATs. Later, the authors assessed the process using it in an AAT suite from a large-scale software product.

4.2.1. Qualitative feedback of software practitioners

The industrial evaluation with five participants is conducted with the help of a questionnaire. The results from each section, based on UTAUT [61], are described below:

Performance expectancy: According to practitioner one, combining the reuse process and the reuse cost is very beneficial. Other participants also found this solution beneficial for the reuse of AATs. The quotes below describe the specific statements from the respondent of the survey questionnaire.

> "It guides a systematic approach, and it will help in optimizing the AAT process. Provided if it does not involve additional execution cost and the process is automated." (P1)
>
> "The possible benefit of the process is the reduction in the effort to write the new test case." (P2)
>
> "In my point of view, this process will really help the software practitioner to select the test cases and use it for automated acceptance

[10] https://git-scm.com/

tests. The use of this process can be cost effective in sense of saving time by selecting the test cases with respect to the score of each test." (P3)

"In case of low assess relevance and less cost of reuse, it will improve the quality of the tests by having already reliable tests. Also, it'll reduce the time to test a functionality that is closer to already existing and selected TC." (P3)

Effort expectancy. The participants, overall, seems happy with respect to the effort expectancy of the proposed process. They suggested that activities in the process are easy to follow. However, they posed a few interesting questions (in quotes below) that we have attempted to resolve in the final design of the proposed process. Some of the quotes from the practitioners are given below:

"The process seems to be simple and easy to use, provided if process tasks are automated." (P3)

"The only thing I have found its hard to implement this process in the existing project because it takes time to change the process and this process has involved many steps but this one time cost of implementation can save a lot in future." (P4)

"The relevance and cost estimation should be automated. The increasing number of reusable artifacts will affect the efficiency of search. So some mechanisms should be introduced to speed up the search, e.g indexing etc." (P3)

When asked about ease of use, four practitioners marked the process as easy or very easy to use. One practitioner suggested it as "Normal" to use. The results are shown in Table 6.

Facilitating Condition. The participants stated that the solution is right to have, but the organization-level activities should be reconsidered because they add an over-head in the process. The respondents of the survey suggested the following improvements with respect to facilitating conditions.

"Organizational level activities should not be the part of the process, rather these should be the pre-requisites of the process." (P1)

"Steps A2 and A3 can cause possible delay, so they should be a pre-requisite to the process, and not a part of the process." (P3)

"Reassessment after cost evaluation with other closely related test cases." (P4)

From the evaluation, we found that:

- The systematic reuse process saves the time to write new AATs.
- The organizational level activities are one-time activities, and test developers should skip these activities.
- There can be more reuse candidates than one. A new AAT can use parts of multiple existing reusable AAT.
- The scripts implementing search and assessment activities should be part of the systematic reuse process. In the future, the focus should be to automate many of these activities.

4.2.2. Demonstrating industrial application of the proposed process

An industrial AAT was implemented using the activities of the proposed process. A summary of quantitative data captured during the evaluation is shown in Table 7 and qualitative information mentioned in the sections below. The context and details of the industrial demonstration are discussed in the research approach (Section 3.2.2 Phase 5: Test candidate design elements).

Details of tasks in each activity. Each activity inside the proposed process consists of one

Table 6. How easy it is to use the process

Participant	Very Easy	Easy	Normal	Difficult	Very difficult
Participant 1	✓				
Participant 2		✓			
Participant 3		✓			
Participant 4			✓		
Participant 5	✓				

or more tasks that are performed in the activity. The tasks performed in each activity are assessed, and important lessons are documented. It was noted that A4, A5, A6, and A8 activities take more time and contain multiple tasks per activity. The tasks under each activity are described in Table 7 and in the sub-sections below:

A1: Select Keywords: The test developer (author) selected 3 keywords (a) REST (b) <Interface Name> (c) a description of performance requirement.

Select AAT format: From the available choices (BDD or keyword-driven), the test developer selected keyword-driven as AAT format. The existing AATs were also written in keyword-driven format.

Select artifact adaptation template: "Parametrization" is selected as artifact adaptation template because parameterization is by default supported by the keyword-driven frameworks, and existing AAT is also based on keyword-driven tests.

Search for reusable AATs: The searching of AATs was executed using the script provided as part of the proposed process (See [69]). Before the search is executed, libraries required for the script are installed. The existing AATs are checked-out from the repository. Three files, each containing one keyword, are created in the same directory. The NCD script is executed to identify the AATs closely matching the keywords. The first search did not yield AATs that were closely matching with keywords. Later, keywords were changed (mentioned in the activity "select keywords" above) that produced better results. The output of this activity is available online [78].

Assess Relevance of AATs: The output of search activity is sorted in ascending order. The output of the five pairs with lowest NCD values (i.e., similar) is shown in Table 8. The pairs with the lowest NCD values are selected for the analysis. After manually analyzing the content of selected AATs (from pairs with the lowest NCD), two AATs (testcase27, testcase26) are selected for reuse purposes.

Select reuse cost method and metrics: The verification team keeps track of tasks in a ticketing system. The time spent on each task in each phase (in backlog, in development, in Done) is present in the system. Therefore, it was decided to use a person-hour based metric and model. The reuse cost calculation method proposed by Irshad et al. [15] was selected to calculate reuse costs.

Calculate cost of reuse: The development time of testcase27 and testcase26 was four weeks for each test case. We estimated that by reusing (without change) some parts of the testcase26 (related to SUT configuration and test data generation and cleaning), we could save three weeks of development effort. Other parts of testcase27 (related to validation) support parametrization. These parametrization supporting parts were also reused using different values for the parameters. An example of parametrization is found on a link here [83].

Develop new AAT by reuse: The new AAT was developed using the parameterization approach (See example of parametrization [83]. The reusable lines from the testcase27 already supported parametrization. Seven out of twenty-five new AAT lines were reused (by using different parameter values) from the testcase27.

Add new AAT to repository: Once the AAT is ready and approved by the reviewers, the AAT is pushed to the central repository. This task triggers the build on the build server, executing the AAT.

Number of tasks performed in each activity. The number of tasks in an activity may show the effort required to perform the activity. An activity with a large number of tasks may require more effort from the practitioners. The test developer (one of the authors) kept a record of the number of tasks performed in each activity, e.g., searching, changing code. The tasks varied from 1 task to 3 tasks in activity. The number of tasks performed in each activity is described in Table 7.

Effort to create an AAT. To capture this construct, we measured the time taken during each activity. The time taken by each activity is shown in Table 7. The development (writing, testing, refactoring) of the AAT took the most time (1 week). Other activities in the process took less than 30 minutes each. The practitioner tracked the time spent on the task, and the practitioner used minutes to track the precise time. In the work management tool used by the organization,

Table 7. Evaluation: Time spent, no of tasks, the tasks performed during each activity of the process, description of the tasks and comparison with existing (manual) activity.

Activity	Time spent	No. of tasks	Description of tasks in the activity	Comparison with existing (manual) activity
A1: Select Keywords	5 minutes	2	(a) Reading requirements description. (b) Deciding suitable keyword.	Not needed in manual process
A2: Select AAT format*	1 minute	1	Selecting "Keyword-driven" as AAT format.	No such activity exists, a practitioner decides the format he has previous experience with.
A3: Select artifact adaptation template	1 minute	1	Selecting "Parametrization" as adaptation template to support develop with reuse.	No such activity exists.
A4: Search reusable AATs*	27 minutes	3	(a) Configuring libraries for NCD script, a one-time task (20 minutes). (b) Writing keywords from activity A1 in separate files (5 minutes). (c) Executing script in the repository (takes 2 minutes).	A practitioner uses his/her experience from test suite.
A5: Assess relevance of AATs*	30 minutes	3	(a) Sort and analyse the output of NCD script (excel file) 10 minutes. (b) Select top relevant AATs and analyse for relevance (10 minutes). (c) Select one AAT most suitable for a new AAT (10 minutes).	A practitioner uses his/her experience of domain.
A6: Select reuse cost method and metrics	11 minutes	2	(a) Analysis of the metric present in the organization/unit (10 minutes). (b) Select suitable cost model (1 minute).	No such activity exists.
A7: Calculate cost of reuse	15 minutes	1	Apply cost model to AAT (15 minutes).	No such activity exists.
A8: Develop new AAT by reuse*	1 week	4	(a) Selecting reusable parts of AAT from task "C" in activity A5 (20 minutes). (b) Adapting reusable parts to fit the newly developed AAT (24 hours). (c) Testing the new AAT (10 hour). (d) Refactoring the new AAT (6 hours).	A test case by reuse is developed using activities, A1–A7.
A9: Develop a new AAT*	0	0	Not performed.	A test case from scratch is developed using activities, A2 and A3.
A10: Add new AAT to repository*	1 minute	3	Using commands: git add <filename> and git commit -m "<Message>" andgit push	Similar to manual process.

* shows activity exists in manual and automated process.

the development of the new AAT took nearly 5-working days.

The time spent on a similar existing AAT was identified from the organization's archived data (using Git history, we found the development task and determined the time spent in the task development phase). A similar AAT was developed in four weeks, as per the development phase of the task. This difference is because, in the newly developed AAT, the test data setup,

Table 8. NCD values of comparison between Keywords and existing AATs

AAT 1	Keywords	NCD value
testcase27	keywords.txt	0.169
testcase26	keywords.txt	0.174
testcase84	keywords.txt	0.294
testcase25	keywords.txt	0.299
testcase83	keywords.txt	0.307

the SUTs state configuration, and the test data deletion parts were reused from an existing AAT. **Magnitude of reused statement.** Utilizing the reuse cost calculation activity is essential to recognize benefits before reusing any AATs. The cost savings through AAT's reuse depends on (i) the number of reusable AAT statements and (ii) the functionality corresponding to the reused statement. For example, only seven out of 25 lines were reused in the demonstration. However, these seven reused lines perform functionality that is time-consuming to develop, so the cost savings were almost 75% (1 week when reusing vs. 4-weeks of development time from scratch). Another example can be a case where many statements are reused, but these statements require a small amount of development time when developing from scratch; in that case, cost savings may not be a lot.

The evaluation from the industrial assessment identified the following lessons:

– Finding suitable keywords may require multiple iterations before finding the most useful reusable AAT.
– There were more than one AATs identified as reusable during the assessment of reusable AAT. There should be a guideline on how to select the best out of the possible reusable artifacts.
– The automated scripts provide a mechanism to search and assess the existing AATs.

4.3. Final version of systematic reuse process

Following changes were introduced in the proposed process, based on the feedback from practitioners and application in an industrial setting:

– A condition is introduced to skip organizational-level activities if these are already defined, as per the participants' suggestions.
– A condition is modified to select multiple candidates and perform reuse cost calculations on each of these candidates.
– A new activity is introduced to evaluate the most suitable candidate from a list of candidates having lower costs and requiring fewer changes. The activity takes input on a list of

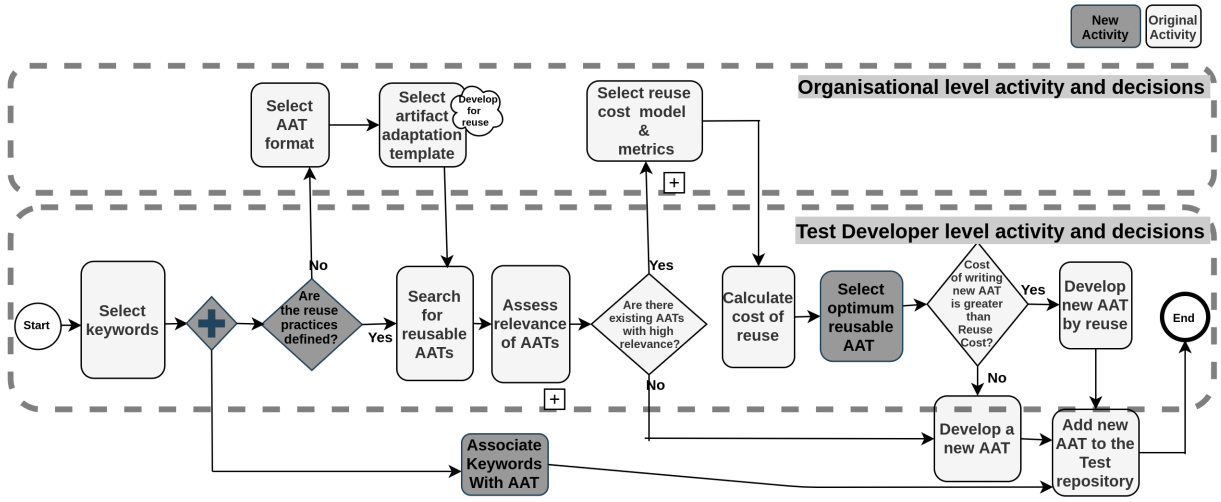

Figure 3. Final Version: A systematic reuse process for automated acceptance tests (AATs)

AATs with high relevance and low costs and lists the most feasible reusable AAT.
- A new activity, Associate Keywords with AAT, is introduced that stores the selected keywords when storing the new test case in the repository. These keywords help in categorizing the test cases and optimize the search functionality.
- The activities that can be automated with a script's help are mentioned in Table 3.

The final version of the process is shown in Figure 3.

5. Discussion

This section provides a discussion on the analysis of the evaluation, characteristics and benefits of the proposed process.

5.1. Analysis of industrial evaluation

This section describes the findings and analysis of industrial evaluation.

Analysis on performance expectancy. Performance expectancy has implications on using the proposed reuse process if the process is advantageous for the software practitioners [12]. If the process is perceived as advantageous, then it is likely that other software practitioners and the software industry embrace the proposed process. In the evaluation, practitioners provided positive feedback with regards to performance expectancy. The practitioners listed the following advantages regarding performance expectancy of the process:
- A systematic process to support reuse of automated acceptance tests.
- The activities of searching and assessment of a reusable test case can help software practitioners.
- New tests can be developed with less effort.

This feedback was re-confirmed during the industrial demonstration of the process where the test developer (an author) followed the activities and techniques proposed in the systematic reuse process and successfully developed the test case.

Analysis on effort expectancy. Effort expectancy identifies the level of ease to use the proposed reuse process. The easiness of using the process has direct implications on the adaptability of the process in the industry. Software practitioners found the process easy to use. However, to increase the effort expectancy, practitioners suggested that the majority of the process activities should be automated with the help of scripts. Practitioners also suggested that there are many activities involved in the process, and implementing these activities in their existing process can be challenging. They suggested automating these activities to reduce the impact of a large number of activities.

During the industrial demonstration, it was noted that the time spent on the activities of the process and the number of tasks in each activity are low in numbers. The activity with the highest number of tasks (4) and time is taken (1 week) is developing the new AAT. Overall the activities in the process were easy to use for the test developer well familiar with the process (an author).

Analysis on facilitating conditions. The process's impact on the development of AATs is evaluated using the facilitating condition construct. The practitioners, during the evaluation, suggested changes that can help improve the construct of facilitating conditions. Practitioners believed that the proposed process could become better by:
- making few activities pre-requisite to execute (only once) when the process is applied in any organization,
- allowing assessment of more than one reusable AATs before selecting the final reusable AAT.

The tool support and the competence needed to use the process was evaluated in the industrial demonstration. It was noted that tools and libraries need to be installed before running the provided scripts for searching and assessment. The calculation of reuse cost required metrics that were already available in the organization, i.e., better-facilitating conditions.

5.2. Guidelines for AATs reuse

The activities of proposed process act as guidelines for reusing the AAT. In the existing litera-

ture, limited studies have discussed the reuse of AAT, and this study provides step-by-step guidance for developing with reuse and developing for reuse. The input, the output, the actor and techniques relevant for each activity are described in Table 3 and Section 4. Practitioners and researchers can use this information as guidelines for supporting the reuse of AATs.

5.3. Compare reuse opportunities

The two activities (i.e., assess the relevance of AAT and reuse cost analysis) in the process can help the practitioners evaluate a reuse opportunity's effectiveness. With these two activities, the practitioners have an instrument to evaluate and compare the value of reusing different AATs. Based on their comparison, they can select the AAT, which is more suitable for their purpose.

5.4. Flexible techniques

Section 4 provides different techniques that apply to the activities of reuse process. The implementation of some of these techniques is also provided to support the practitioners. However, an organization can add its own techniques to search, assess, or calculate reuse costs if it wants to use customized techniques. The reuse process is not bound to fix a set of searching and calculation techniques.

5.5. Support for automation

Several steps in the proposed process are either automated already or have the potential (e.g., selecting keywords) to be automated. This can result in cost savings for the practitioners by (i) reducing time to develop an AAT and (ii) reducing the time to analyze, search, and assess the AATs. In future work, we want to provide automated scripts for activities A1, A6, and A7 from Table 3.

5.6. Verdict on the diversity of AAT-suite

The activities of the process can also be used to assess the diversity in an organization's AAT suite. A higher diversity means that the test suite has more test coverage. The search and assessment using NCD provides pair-wise comparison values of all the AATs. These values can be a good indicator of diversity in the AAT suite. A suite with low diversity could have many pairs with low NCD values (i.e., very similar to each other), indicating that refactoring is needed to diversify the AATs or remove the duplicates.

5.7. Tool support for reuse of AATs:

AATs are text-based artifacts different from traditional code-based test cases. IDE features often support the code-based reuse process to detect duplicates, detect similar usage, provide modularization of code snippets, etc. These basic reuse features are not yet mature enough for non-code artifacts. Therefore, we have provided easy to use techniques and scripts that can be applied to support the reuse of text-based AATs.

5.8. Increased coupling

Existing research literature has described the issue with decreased maintainability among AATs [7]. A key concern when developing by reusing parts of different reusable AATs is an increase in coupling (dependency between test cases) in the test base [84]. This increase in coupling decreases the maintainability of the test cases. Therefore, during the activity A5 (Assessing the relevance of reusable AATs), it is vital to consider the increase in coupling between the reused and reusable AATs.

5.9. Comparison with existing literature

Park and Maurer proposed three strategies that can be used to develop reusable assets in software product lines [85]. These three strategies are (i) proactive mode in which organization makes upfront investment in developing reusable assets, (ii) reactive mode in which reusable assets are developed when needed, and (iii) extractive mode in which existing product is reused [85]. Our proposed process can be classified as a reactive

model, in which we develop new reusable assets when there is a need for writing a new test.

In their study on variability management in software product lines, Kiani et al. proposed a method in which reusable assets are developed on demand when the need arises [86]. This is similar to our proposed approach in which reusable AATs are created when there is a need to write a new test, i.e., no upfront costs are required. In addition, de Silva proposed a software product line approach that uses automated acceptance tests to link scoping of requirements, implementation, and testing [87]. Our proposed process compliments this SPL-based study by suggesting a reuse-based approach to derive the reusable AATs along with requirements, implementation, and testing.

In a study by Mink et al., software practitioners suggested that specifying the granular details of automated acceptance tests, i.e., format and details of AATs is cumbersome and requires more time from them [3]. In another study, Mink et al. investigated executable acceptance testing. They found that AATs help in (i) preserving the domain knowledge and (ii) improve the communication among the developers [3]. Our study identified similar findings during the evaluation of the proposed process using experienced software practitioners. Thus, our approach may help the software practitioner specify the details of the automated acceptance tests by providing a specific AAT format and suggesting existing reusable AATs.

5.10. Scalability of Approach

Searching for reusable software artifacts is known as a time-consuming process (with high costs) during software reuse [43]. The cost of searching and retrieving a reusable software artifact grows when new artifacts are added to the repository [43]. Therefore, the scalability of the searching techniques is a vital characteristic to support future reuse opportunities. We evaluated performed an evaluation of the search approach. In an examination (by the authors) with a specification base of 500 AATs, reusable candidates were identified in less than 5 minutes using the scripts provided as part of the proposed process (See script [69].)

6. Threats to validity

Runeson et al. [88] classified the validity threats into four types (reliability, construct validity, internal validity, and external validity). These threats to the study's validity and the measures to address these validity threats are discussed in this section.

Internal validity deals with the case when the factors studied by the researchers are affected by any other factors unknown to the researchers. This threat applies to the design and development of the questionnaire for industrial evaluation. The questionnaire design can be classified as "self-administrated," i.e., online form. As the proposed process is developed for software organization, we evaluated the process using the constructs suggested by the unified theory of acceptance and technology use (UTAUT) [61]. These three constructs (performance expectancy, effort expectancy, and facilitating condition) help the authors to develop evaluation questions related to the proposed process systematically. Each question in the questionnaire is mapped to a construct that it addresses. The details are provided in Section 3. We believe that we have addressed this threat to this investigation's internal validity by following a systematic method to design and develop the questionnaire.

External validity concerns with the generalization of results. The proposed process developed in this study is considered useful for large-scale software organizations. Hence, we evaluated the proposed process using an industrial use-case for large-scale systems. During the evaluation, the process's completeness and usability for large-scale product organizations are evaluated. Furthermore, we involved five experienced practitioners from two large-scale organizations to evaluate the proposed process and provide feedback.

The experienced practitioners involved in this study worked in different roles in large-scale organizations. They were selected because they have a prior understanding of automated acceptance testing and reuse. As reuse of AATs is still a new area, it is difficult to find practitioners who understand these concepts. Furthermore, we involved practitioners from two large-scale

organizations to improve the generalizability of the proposed process. However, the authors believe that a reuse process should be applied and evaluated in the industry before the results of this study are considered generalizable, which is part of our future work.

Reliability deals with how the data collection and analysis are dependent on the researcher. The researcher independently conducted the first evaluation to validate a process in the industry setting. This practice is called lab validation by Gorscheck et al. [62]. Since this evaluation was conducted by a software practitioner (an author), the threat to the validity of data collection and analysis exists. To mitigate this threat to the evaluation's validity, in the second part of the evaluation (using experienced practitioners), the data collection was done in an online form without the author's active involvement in filling the form. A critical threat to the reliability of the study is related to the response bias of survey respondents. The responses from the practitioners reflect the belief of those practitioners, and these beliefs may be contrary to real-world contexts. We believe that this threat is relevant to this study, and in future evaluations of the proposed process, work is needed to address this concern. The usage of online form reduces the chance of losing any valuable feedback from practitioners. For data analysis, we used a systematic method called constant comparison to interpret the online questionnaire's feedback.

Construct validity deals with how well the study captures the intended constructs. During the evaluation, each subject involved presented the motivation, background, and walk-through of the proposed process during online sessions. The subjects asked questions about the study, the process, and the questionnaire during these presentations. Furthermore, one of the study authors has considerable experience working in the same domain and industry. Hence, he was able to explain the concepts in the language/terms understood by the subjects. This step helps in reducing confusion or ambiguities related to the reuse process.

Furthermore, the questionnaire (available at [73]) provides a detailed description of concepts, the activities used in the process, techniques used in these activities, and a working example of the proposed process. These details were provided to the respondents to make sure the study captures the intended constructs.

7. Conclusion

The reuse of automated acceptance tests help develop new tests cheaply, quickly, and with high quality. However, the textual nature of these tests makes the reuse of these tests different from code-based tests. In this investigation, we describe a systematic reuse process for text-based automated acceptance tests. We constructed this reuse process using the method engineering and performed an initial evaluation of the reuse process before applying it to the industry.

RQ 1: How can the cost-effective systematic reuse of text-based AATs be achieved? The construction of systematic reuse process starts with the identification of the motivation and requirements of the new process. The identified requirements are (i) the process should consider development with reuse and development for reuse, (ii) the process should be independent of several text-based formats (e.g., BDD, keywords) and frameworks of automated acceptance tests (e.g., Cucumber, Robot Framework), and (iii) the cost of reuse should be calculated before developing a new test by reusing existing tests. For these three requirements, we identified and tailored existing methods present in the literature. The final outcome is a systematic reuse process that supports the reuse of various types of text-based automated acceptance tests. The process activities are divided into two types, i.e., organizational level activities and test developer-level activities. We provided expected input, expected output, actor, examples, and techniques (automated using scripts) suitable for the process's activities.

RQ 2: How does the systematic reuse process, from RQ 1, perform concerning performance expectancy, effort expectancy, and facilitating conditions in the industrial context? After constructing the process, an elementary industrial evaluation assesses the performance expectancy, effort

expectancy, and facilitating conditions concerning the process. This evaluation is performed to sanity-check and improves the process before it is ready for a long and detailed industrial evaluation. Initially, five participants with considerable experience in automated acceptance testing provided qualitative feedback on the systematic reuse process. They found that the process can save time and reduce the effort to write and maintain automated acceptance tests. The practitioners suggested that activities are easy to use, and the reuse cost metrics are easy to find and apply using the proposed techniques. They suggested changes in the process, and these changes were incorporated in the final version of the process. This evaluation shows promising results concerning the processes' performance expectancy, effort expectancy, and facilitating conditions.

Later an illustration of the usage of a process in the industry is conducted. During the evaluation, a new test case is developed by reusing existing automated acceptance tests. This evaluation's objective was to identify and sanity-check the tasks in the process's activities and evaluate their complexity. One of the authors, from the same organization, conducted this demonstration. The evaluation helped identify several different granular tasks required to perform in each activity of the process. The number of tasks varies from 1 to 4 between different activities. These identified tasks can act as guidelines when using the process in the industry. The evaluation also recorded the time spent in each activity. It was noted that most of the time is spent developing the new test by reusing existing automated acceptance tests. The development time with the reuse process was 4-times quicker than developing a new artifact from scratch.

In the future, we want to have a longitudinal investigation on the process's transfer and usage in the industry, as application and evaluation of the process may take a long duration and resources. The current study enables the researchers to sanity-check the process before evaluating it in industrial settings. Secondly, in the next study, we want to automate most of the proposed process activities and evaluate the process using automated activities. Furthermore, we want to evaluate the precision and recall of the search and assessment functionality proposed in this process.

References

[1] M.J. Harrold, "Testing: A roadmap," in *Proceedings of the Conference on the Future of Software Engineering*, 2000, pp. 61–72.

[2] W.E. Wong, J.R. Horgan, S. London, and H. Agrawal, "A study of effective regression testing in practice," in *Proceedings., The Eighth International Symposium on Software Reliability Engineering*. IEEE, 1997, pp. 264–274.

[3] G. Melnik and F. Maurer, "Multiple perspectives on executable acceptance test-driven development," in *International Conference on Extreme Programming and Agile Processes in Software Engineering*. Springer, 2007, pp. 245–249.

[4] "Standard glossary of terms used in software testing," International Software Testing Qualifications Board, Standard 3.5, 2020. [Online]. https://www.istqb.org/downloads/glossary.html

[5] B. Haugset and G.K. Hanssen, "Automated acceptance testing: A literature review and an industrial case study," in *Agile Conference*. IEEE, 2008, pp. 27–38.

[6] M. Huo, J. Verner, L. Zhu, and M.A. Babar, "Software quality and agile methods," in *Proceedings of the 28th Annual International Computer Software and Applications Conference, 2004. COMPSAC 2004.* IEEE, 2004, pp. 520–525.

[7] J. Weiss, A. Schill, I. Richter, and P. Mandl, "Literature review of empirical research studies within the domain of acceptance testing," in *42th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*. IEEE, 2016, pp. 181–188.

[8] W.B. Frakes and K. Kang, "Software reuse research: Status and future," *IEEE Transactions on Software Engineering*, Vol. 31, No. 7, 2005, pp. 529–536.

[9] R. Capilla, B. Gallina, C. Cetina, and J. Favaro, "Opportunities for software reuse in an uncertain world: From past to emerging trends," *Journal of Software: Evolution and Process*, Vol. 31, No. 8, 2019, p. e2217.

[10] W.B. Frakes and S. Isoda, "Success factors of systematic reuse," *IEEE software*, Vol. 11, No. 5, 1994, pp. 14–19.

[11] D. Rombach, "Integrated software process and product lines," in *Software Process Workshop*. Springer, 2005, pp. 83–90.

[12] M. Ramachandran, "Software re-use assessment for quality," *WIT Transactions on Information and Communication Technologies*, Vol. 9, 1970.

[13] E.S. de Almeida, A. Alvaro, D. Lucrédio, V.C. Garcia, and S.R. de Lemos Meira, "Rise project: Towards a robust framework for software reuse," in *Proceedings of the International Conference on Information Reuse and Integration*. IEEE, 2004, pp. 48–53.

[14] J.S. Poulin, *Measuring software reuse: principles, practices, and economic models*. Addison-Wesley Reading, MA, 1997.

[15] M. Irshad, R. Torkar, K. Petersen, and W. Afzal, "Capturing cost avoidance through reuse: systematic literature review and industrial evaluation," in *Proceedings of the 20th International Conference on Evaluation and Assessment in Software Engineering*. ACM, 2016, p. 35.

[16] A. Davies, T. Brady, and M. Hobday, "Charting a path toward integrated solutions," *MIT Sloan management review*, Vol. 47, No. 3, 2006, p. 39.

[17] W.E. Wong, "An integrated solution for creating dependable software," in *Proceedings 24th Annual International Computer Software and Applications Conference. COMPSAC2000*. IEEE, 2000, pp. 269–270.

[18] R.J. Mayer, J.W. Crump, R. Fernandes, A. Keen, and M.K. Painter, "Information integration for concurrent engineering (IICE) compendium of methods report," Knowledge Based Systems Inc., Tech. Rep., 1995.

[19] M. Rahman and J. Gao, "A reusable automated acceptance testing architecture for microservices in behavior-driven development," in *Symposium on Service-Oriented System Engineering (SOSE)*. IEEE, 2015, pp. 321–325.

[20] G. Meszaros, "Agile regression testing using record and playback," in *Companion of the 18th Annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications*. ACM, 2003, pp. 353–360.

[21] A.K. Onoma, W.T. Tsai, M. Poonawala, and H. Suganuma, "Regression testing in an industrial environment," *Communications of the ACM*, Vol. 41, No. 5, 1998, pp. 81–86.

[22] P. Hsia, D. Kung, and C. Sell, "Software requirements and acceptance testing," *Annals of Software Engineering*, Vol. 3, No. 1, 1997, pp. 291–317.

[23] G.K. Hanssen and B. Haugset, "Automated acceptance testing using fit," in *42nd Hawaii International Conference on System Sciences*. IEEE, 2009, pp. 1–8.

[24] E. Pyshkin, M. Mozgovoy, and M. Glukhikh, "On requirements for acceptance testing automation tools in behavior driven software development," in *Proceedings of the 8th Software Engineering Conference in Russia (CEE-SECR)*, 2012.

[25] G. Liebel, E. Alégroth, and R. Feldt, "State-of--practice in GUI-based system and acceptance testing: An industrial multiple-case study," in *39th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA)*. IEEE, 2013, pp. 17–24.

[26] H. Munir and P. Runeson, "Software testing in open innovation: An exploratory case study of the acceptance test harness for Jenkins," in *Proceedings of the International Conference on Software and System Process*, 2015, pp. 187–191.

[27] G. Melnik and F. Maurer, "The practice of specifying requirements using executable acceptance tests in computer science courses," in *Companion to the 20th Annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications*, 2005, pp. 365–370.

[28] M. Hayek, P. Farhat, Y. Yamout, C. Ghorra, and R.A. Haraty, "Web 2.0 testing tools: A compendium," in *International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT)*. IEEE, 2019, pp. 1–6.

[29] P. Gandhi, N.C. Haugen, M. Hill, and R. Watt, "Creating a living specification using FIT documents," in *Agile Development Conference (ADC'05)*. IEEE, 2005, pp. 253–258.

[30] D. North, "Introducing behaviour driven development," *Better Software Magazine*, 2006.

[31] E.C. dos Santos and P. Vilain, "Automated acceptance tests as software requirements: An experiment to compare the applicability of fit tables and gherkin language," in *International Conference on Agile Software Development*. Springer, 2018, pp. 104–119.

[32] C. Solis and X. Wang, "A study of the characteristics of behaviour driven development," in *37th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA)*. IEEE, 2011, pp. 383–387.

[33] R. Hametner, D. Winkler, and A. Zoitl, "Agile testing concepts based on keyword-driven testing for industrial automation systems," in *IECON 2012-38th Annual Conference on IEEE Industrial Electronics Society*. IEEE, 2012, pp. 3727–3732.

[34] E. Bache and G. Bache, "Specification by example with gui tests-how could that work?" in

International Conference on Agile Software Development. Springer, 2014, pp. 320–326.

[35] A.C. Paiva, D. Maciel, and A.R. da Silva, "From requirements to automated acceptance tests with the RSL language," in *International Conference on Evaluation of Novel Approaches to Software Engineering*. Springer, 2019, pp. 39–57.

[36] M. Soeken, R. Wille, and R. Drechsler, "Assisted behavior driven development using natural language processing," in *International Conference on Modelling Techniques and Tools for Computer Performance Evaluation*. Springer, 2012, pp. 269–287.

[37] C. Deng, P. Wilson, and F. Maurer, "Fitclipse: A fit-based eclipse plug-in for executable acceptance test driven development," in *International Conference on Extreme Programming and Agile Processes in Software Engineering*. Springer, 2007, pp. 93–100.

[38] C.Y. Hsieh, C.H. Tsai, and Y.C. Cheng, "Test--Duo: A framework for generating and executing automated acceptance tests from use cases," in *8th International Workshop on Automation of Software Test (AST)*. IEEE, 2013, pp. 89–92.

[39] C.W. Krueger, "Software reuse," *ACM Computing Surveys (CSUR)*, Vol. 24, No. 2, 1992, pp. 131–183.

[40] D.M. Rafi, K.R.K. Moses, K. Petersen, and M.V. Mäntylä, "Benefits and limitations of automated software testing: Systematic literature review and practitioner survey," in *Proceedings of the 7th International Workshop on Automation of Software Test*. IEEE Press, 2012, pp. 36–42.

[41] W. Frakes and C. Terry, "Software reuse: metrics and models," *ACM Computing Surveys (CSUR)*, Vol. 28, No. 2, 1996, pp. 415–435.

[42] W. Tracz, "Where does reuse start?" *ACM SIGSOFT Software Engineering Notes*, Vol. 15, No. 2, 1990, pp. 42–46.

[43] T. Ravichandran and M.A. Rothenberger, "Software reuse strategies and component markets," *Communications of the ACM*, Vol. 46, No. 8, 2003, pp. 109–114.

[44] P. Mohagheghi and R. Conradi, "Quality, productivity and economic benefits of software reuse: A review of industrial studies," *Empirical Software Engineering*, Vol. 12, No. 5, 2007, pp. 471–516.

[45] V. Karakostas, "Requirements for CASE tools in early software reuse," *ACM SIGSOFT Software Engineering Notes*, Vol. 14, No. 2, 1989, pp. 39–41.

[46] J.L. Cybulski, "Introduction to software reuse," *Department of Information Systems, The University of Melbourne, Parkville, Australia*, Vol. 11, 1996, p. 12.

[47] W. Lam, J.A. McDermid, and A. Vickers, "Ten steps towards systematic requirements reuse," *Requirements Engineering*, Vol. 2, No. 2, 1997, pp. 102–113.

[48] R.G. Fichman and C.F. Kemerer, "Incentive compatibility and systematic software reuse," *Journal of Systems and Software*, Vol. 57, No. 1, 2001, pp. 45–60.

[49] A. Genaid et al., "Connecting user stories and code for test development," in *Third International Workshop on Recommendation Systems for Software Engineering (RSSE)*. IEEE, 2012, pp. 33–37.

[50] L. Crispin and T. House, "Testing in the fast lane: Automating acceptance testing in an extreme programming environment," in *XP Universe Conference*. Citeseer, 2001.

[51] L.P. Binamungu, S.M. Embury, and N. Konstantinou, "Maintaining behaviour driven development specifications: Challenges and opportunities," in *25th International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE, 2018, pp. 175–184.

[52] M. Irshad, J. Börster, and K. Petersen, "Supporting refactoring of BDD specifications – An empirical study," *Information and Software Technology*, 2022.

[53] R. Angmo and M. Sharma, "Performance evaluation of web based automation testing tools," in *5th International Conference – Confluence The Next Generation Information Technology Summit (Confluence)*. IEEE, 2014, pp. 731–735.

[54] S. Park and F. Maurer, "A literature review on story test driven development," in *International Conference on Agile Software Development*. Springer, 2010, pp. 208–213.

[55] Q. Xie, "Developing cost-effective model-based techniques for GUI testing," in *Proceedings of the 28th International Conference on Software Engineering*. ACM, 2006, pp. 997–1000.

[56] R. Borg and M. Kropp, "Automated acceptance test refactoring," in *Proceedings of the 4th Workshop on Refactoring Tools*. ACM, 2011, pp. 15–21.

[57] C. Schwarz, S.K. Skytteren, and T.M. Ovstetun, "AutAT: An eclipse plugin for automatic acceptance testing of web applications," in *Companion to the 20th Annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications*. ACM, 2005, pp. 182–183.

[58] B. Fitzgerald, N.L. Russo, and T. O'Kane, "Software development method tailoring at motorola,"

Communications of the ACM, Vol. 46, No. 4, 2003, pp. 64–70.

[59] P. Raulamo-Jurvanen, M. Mäntylä, and V. Garousi, "Choosing the right test automation tool: a grey literature review of practitioner sources," in *Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering*, 2017, pp. 21–30.

[60] A. Egbreghts, "A literature review of behavior driven development using grounded theory," in *27th Twente Student Conference on IT.*, 2017.

[61] V. Venkatesh, J.Y. Thong, and X. Xu, "Consumer acceptance and use of information technology: Extending the Unified Theory of Acceptance and Use of Technology," *MIS quarterly*, 2012, pp. 157–178.

[62] T. Gorschek, P. Garre, S. Larsson, and C. Wohlin, "A model for technology transfer in practice," *IEEE software*, Vol. 23, No. 6, 2006, pp. 88–95.

[63] M. Finsterwalder, "Automating acceptance tests for GUI applications in an extreme programming environment," in *Proceedings of the 2nd International Conference on eXtreme Programming and Flexible Processes in Software Engineering*. Addison-Wesley Boston MA, 2001, pp. 114–117.

[64] S. Park and F. Maurer, "An extended review on story test driven development," University of Calgary, Tech. Rep., 2010.

[65] H. Mili, F. Mili, and A. Mili, "Reusing software: Issues and research directions," *IEEE Transactions on Software Engineering*, Vol. 21, No. 6, 1995, pp. 528–562.

[66] R. Feldt, S. Poulding, D. Clark, and S. Yoo, "Test set diameter: Quantifying the diversity of sets of test cases," in *International Conference on Software Testing, Verification and Validation (ICST)*. IEEE, 2016, pp. 223–233.

[67] W.H. Gomaa, A.A. Fahmy et al., "A survey of text similarity approaches," *International Journal of Computer Applications*, Vol. 68, No. 13, 2013, pp. 13–18.

[68] M. Irshad, K. Petersen, and S. Poulding, "A systematic literature review of software requirements reuse approaches," *Information and Software Technology*, Vol. 93, 2018, pp. 223–245.

[69] M. Irshad, *Source Code for Scripts*, 2021. [Online]. https://zenodo.org/record/4765079

[70] L. Amar and J. Coffey, "Measuring the benefits of software reuse-examining three different approaches to software reuse," *Dr Dobbs Journal*, Vol. 30, No. 6, 2005, pp. 73–76.

[71] K. Petersen and C. Wohlin, "Context in industrial software engineering research," in *3rd International Symposium on Empirical Software Engineering and Measurement*. IEEE, 2009, pp. 401–404.

[72] C. Wohlin, M. Höst, and K. Henningsson, "Empirical research methods in software engineering," in *Empirical methods and studies in software engineering*. Springer, 2003, pp. 7–23.

[73] M. Irshad, "Questionnaire: The reusability of automated acceptance tests," 2021. [Online]. https://zenodo.org/record/4765102

[74] J.S. Molléri, K. Petersen, and E. Mendes, "An empirically evaluated checklist for surveys in software engineering," *Information and Software Technology*, Vol. 119, 2020, p. 106240.

[75] B.G. Glaser, A.L. Strauss, and E. Strutzel, "The discovery of grounded theory; strategies for qualitative research," *Nursing research*, Vol. 17, No. 4, 1968, p. 364.

[76] J. Stoustrup, "Successful industry/academia cooperation: From simple via complex to lucid solutions," *European Journal of Control*, Vol. 19, No. 5, 2013, pp. 358–368.

[77] T.C. Lethbridge, S.E. Sim, and J. Singer, "Studying software engineers: Data collection techniques for software field studies," *Empirical software engineering*, Vol. 10, No. 3, 2005, pp. 311–341.

[78] M. Irshad, "Search and assessment data," 01 2021. [Online]. http://shorturl.at/juIZ6

[79] P.M. Vitányi, F.J. Balbach, R.L. Cilibrasi, and M. Li, "Normalized information distance," in *Information theory and statistical learning*. Springer, 2009, pp. 45–82.

[80] B.Y. Pratama and R. Sarno, "Personality classification based on Twitter text using Naive Bayes, KNN and SVM," in *Proceedings of the International Conference on Data and Software Engineering*, 2015, pp. 170–174.

[81] J.C. Corrales, *Behavioral matchmaking for service retrieval*, Ph.D. dissertation, Université de Versailles-Saint Quentin en Yvelines, 2008.

[82] S.S. Choi, S.H. Cha, and C.C. Tappert, "A survey of binary similarity and distance measures," *Journal of Systemics, Cybernetics and Informatics*, Vol. 8, No. 1, 2010, pp. 43–48.

[83] "Parameterize BDD tests," 2021. [Online]. https://support.smartbear.com/testcomplete/docs/bdd/parameterize.html

[84] G. Gui and P.D. Scott, "Coupling and cohesion measures for evaluation of component reusability," in *Proceedings of the 2006 International Workshop on Mining Software Repositories*, 2006, pp. 18–21.

[85] S. Park and F. Maurer, "Communicating domain knowledge in executable acceptance test driven

development," in *International Conference on Agile Processes and Extreme Programming in Software Engineering*. Springer, 2009, pp. 23–32.

[86] A.A. Kiani, Y. Hafeez, M. Imran, and S. Ali, "A dynamic variability management approach working with agile product line engineering practices for reusing features," *The Journal of Supercomputing*, 2021, pp. 1–42.

[87] I.F. Da Silva, "An agile approach for software product lines scoping," in *Proceedings of the 16th International Software Product Line Conference-Volume 2*, 2012, pp. 225–228.

[88] P. Runeson, M. Host, A. Rainer, and B. Regnell, *Case study research in software engineering: Guidelines and examples*. John Wiley and Sons, 2012.