What does it cost to deploy an XAI system: A case study in legacy systems*

Sviatlana Höhn[1][0000−0003−0646−3738] and Niko Faradouris[2]

[1] University of Luxembourg 2 Av. de'l Université, Esch-sur-Alzette, Luxembourg
sviatlana.hoehn@uni.lu
http://satoss.uni.lu/sviatlana
[2] smartShift Technologies GmbH Augustaanlage 59, Mannheim, Germany
nfaradouris@smartshifttech.com
www.smartshifttech.com

Abstract. Enterprise Resource Planning (ERP) software is used by businesses and extended via customisation. Automated custom code analysis and migration is a critical issue at ERP release upgrade times. Despite research advances, automated code analysis and transformation require a huge amount of manual work related to parser adaptation, rule extension and post-processing. These operations become unmanageable if the frequency of updates increases from yearly to monthly intervals. This article describes how the process of custom code analysis to custom code transformation can be automated in an explainable way. We develop an aggregate taxonomy for explainability and analyse the requirements based on roles. We explain in which steps on the new code migration process machine learning is used. Further, we analyse additional effort needed to make the new way of code migration explainable to different stakeholders.

Keywords: Explainable Automated Source-code Transformation · Multimodal Conversational Interfaces · Explainability Taxonomy.

1 Introduction

Enterprise resource planning (ERP) systems, such as SAP Business One[3] and Oracle E-Business Suite[4], have been implemented by many large and medium size companies [5]. The ERP software market grew globally by 10% and reached $ 35 billion in 2018 [18]. The costs of initial implementation range for medium to large scale businesses from $ 150.000 to over $10 million.

Although ERP system vendors recommend to limit code customisation and adapt the business processes to the system, studies report a significant number of customisations in existing ERP systems. For instance, 74% of the companies studied in [9] have adjusted their systems to a degree between moderate and high.

* Supported by FNR Luxembourg INTER-SLANT grant 13320890
[3] https://www.sap.com/products/business-one.html
[4] https://www.oracle.com/de/applications/ebusiness/

The possibility to adapt an ERP system to the business needs and processes of an organisation (i.e. create custom programs) is one of the key requirements for the choice of a specific ERP software [33].

An ERP upgrade implies major changes caused by an implementation of a new version of an already installed ERP system [5]. ERP upgrades can add up to 25–33% of the initial implementation costs for one upgrade [33], most of them caused by labour costs. In average, the costs can increase by $ 1,5 million (between $37.500 and $3,3 million). According to [36], usually a major ERP system upgrade is needed every three years. Cloud-based SAP solutions, however, move to a quarterly release cycle. Current change in the market leader's platform to the SAP S/4HANA ERP software and HANA database forces SAP customers to spend resources on migration of their custom code base [31]. ABAP (Advanced Business Application Programming), the proprietary programming language of SAP, is normally used for custom programs.

Several companies developed rule-based solutions to support SAP customers in their custom code migration tasks within SAP upgrade projects. However, all state-of-the-art solutions share two major problems:

1. Frequent manual updates of the rule-based solution would be required in order to keep it compatible with quarterly updates of the cloud-based SAP ERP solutions, but they are time-consuming and labour-expensive.
2. The reports generated by the current set of tools usually consist several hundreds of spreadsheets. Usually experts review these reports using standard Business Intelligence software and decide which custom programs must be maintained. This is expensive and exhausting.

This is why, technological innovation that transfers academic advances in Machine Learning (ML) and Robotic Process automation (RPA) to the field of software transformation in legacy ERP systems is urgently needed. To tackle this challenge, we developed a plan for knowledge transfer for this specific problem in a close collaboration with one of the technology leaders in the domain of automated ERP custom-code upgrade, smartShift Technologies[5]. Although SAP itself offers a standard solution for custom code upgrade, it only covers unused code and some database compatibility checks. The smartShift tools also analyse syntax and functional errors in the custom code. This is why we chose this solution as our baseline. The business case discussed here is the service of automated custom code migration offered by smartShift.

One of the important questions within this business case is the question of decision explainability. First, the service providers needs to ensure the quality of analysis and migration with a certain level of accuracy. Second, the service provider can be made liable for problems in the work of the ERP system after migration caused by the wrong decisions made by the AI system. Third, errors in the process of code transformation can delay other dependent tasks in the upgrade project and cause additional costs, which should be avoided.

[5] https://smartshifttech.com

The process of code migration is monitored on both sides, the service provider and the customer. This is why the explanations need to be reasonable for multiple stakeholders and multiple roles. With this motivation, we focus on the question, **is additional effort needed in order to make a transformation from an existing rule-based to an ML-based system in an explainable way**.

Because the notion of explainability is a complex phenomenon by itself, we first create an aggregate taxonomy of explainability as explained in Section 2. Section 3 briefly describes the steps in the innovation process. Further in Section 4, we use the aggregate taxonomy to evaluate feasibility and cost of a potential transformation of a rule-based system to an ML-based system used in the ERP domain. Section 5 discusses the results and future research directions.

2 Dimensions of Explainability

Scholar literature on XAI developed several approaches for managing explainability. The works deal with such issues as interpretability [21, 26], fairness [40] and transparency [7]. Each of these terms is recognised as very complex, so that efforts have been made to break them down to a well-defined set of features or parameters [39, 2].

It has been recognised that explainability of ML systems has to target system-related aspects, such as data, models and algorithms [39, 2], and user-related features, such as the user's knowledge of the domain and the interface [37, 12]. Some authors propose to include explainability as a non-functional requirement in the process of software design by focusing on the real users' need for explanations of the system's behaviour [6]. This, in turn, is very difficult given that the notions of *interpretable*, *intelligible*, *explainable* and *understandable* AI are vague and overlapping. For example, [39] discusses multiple definitions of these terms from literature and, finally, uses *interpretability* as a synonym to all termsthat describe the process and the result of human sense-making by using, training and modifying an ML system. In contrast, [13] uses the term *transparency* for "decisions affecting us explained to us in terms, formats, and languages we can understand" [13, p. 29].

What is more, case studies such as [12] show that in industrial applications of XAI, different users have different needs (data scientists want to see the data while ML engineers want to see the trained model).

In order to make the explainability requirement manageable, we created an aggregate of existing taxonomies for XAI [39, 2, 13]. The interface dimension has not been mentioned in the earlier taxonomies, however, it is an important aspect. As pointed out in [2], interactive explanations are rarely studied; a few examples not mentioned in [2] include [16, 1].

The aggregated taxonomy is presented in Table 1. It includes six dimensions: user, object, scope, directness, interaction and interface. The taxonomy can be used as follows: first we need to understand who will be the user of the explanations. Then, based on the description of the role or the persona, we need to determine the object of explanations (can be more than one).

Dimension	Values	Reference
User	E.g. based on roles or personas	[12, 2]
Object	Data: bias, causality, Model: optimiser, training, prediction Evaluation: accuracy, fairness of the decisions, safety and reliability of the system	[39, 2, 13]
Scope	Local vs. global	[2]
Directness	Directly interpretable, post-hoc explanation after training, surrogate approximation model	[2, 13]
Interaction	Static vs interactive	[2]
Interface	Visualisation, voice, text, virtual or augmented reality, multi-modal	Own work

Table 1. Aggregated dimensions of explainability based on [39, 2, 13, 12] and own work

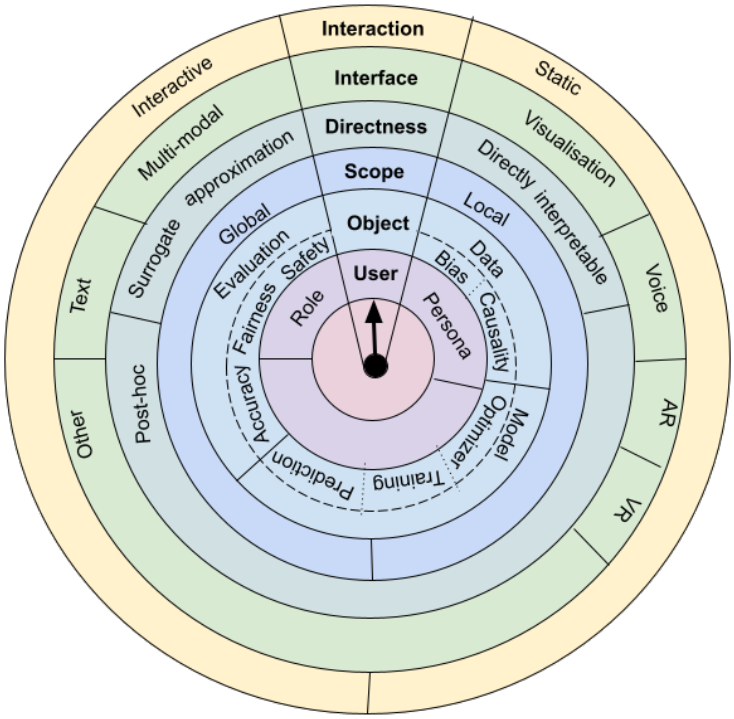

Fig. 1. Cut colour paper circles, write the dimensions on them and assemble in the center, then rotate and bring the needed dimensions together on a line to construct a vector describing your case

For every object, explanations can be of different scope (local or global). The models used for each object can be either directly interpretable (e.g. a small and simple set of rules) or can be explained post-hoc or using a surrogate, approximation model. The explanations for each of those cases can be either static or interactive. For both static and interactive explanations, we can use different interfaces, e.g. conversational, visual and multi-modal. Figure 1 shows how this taxonomy can be visually used to choose the right features.

3 Steps in the Innovation Process

This section explains the changes needed in the custom-code upgrade process in order to go from the rule-based existing system to an ML-based system with a multi-modal conversational interface. In theory it is possible to have multiple interfaces for the same XAI system (e.g. an app, a website and an Alexa skill. However, we chose a multi-modal conversational interface for our business case, as explained below.

3.1 Custom Code Analysis

Custom code analysis includes detection of unused custom code, syntactic, semantic and functional analysis. It addresses two key requirements for a successful ERP upgrade project:

1. System clean-up and
2. Replacement of custom programs whose functionality is already covered by the standard functionality of the target version (obsolete programs).

System clean-up includes identification of duplicates, outdated, or unused functions in the custom code. Between 40% and 70% of the customisations in an average SAP system are not in use [10]. Replacement of obsolete custom programs with standard functionality implies a detailed analysis of the features in the target ERP version [5]. We challenge scholar state of the art by applying natural language processing (NLP) methods, models and tools to legacy programming languages. We analyse the ABAP custom code on syntactic, semantic and functional levels.

In the first step we develop a general method based on NLP state-of-the-art to identify and track custom programs that are obsolete, no longer in use or incompatible with application functionality, data model and interfaces. The automation of the custom code analysis happens in four stages:

1. **Detection of unused custom code**. It includes custom programs that have not been activated in the ERP system for more than N months (in practice, $6 \leq N \leq 18$), including underutilized dependencies (modules that are used only partially and contain blocks of "dead code") [41].

2. **Syntactic analysis**. It includes checks with respect to the target ERP related to the syntactic correctness of the custom programs, the correctness of the data model, and the API interface compatibility with the target platform. This step can be formulated as a machine-translation problem in which the translation happens from code with syntactic errors to correct code [28].
3. **Semantic analysis**. This phase detects code clones. Following [32] we distinguish between structural clones (exact copies of code, copies with renaming and/or modification) and semantic clones (programs that do the same without being structurally similar). Code clones are intentionally introduced by programmers because they help to address changes quickly, save procedure calls and are encouraged by template-based programming. However, code clones increase maintenance costs, cause bug propagation and have negative impact on design and system understanding [32]. Scientists distinguish four types of semantic clones with different grades of syntactic similarities. Type 1 of code clones are syntactically very similar, while Type 4 code clones have no syntactic similarities. The problem can be technically solved using neural models , for example convolutional [42] or rule-based approaches, for example graph-based [35].
4. **Functional analysis**. This step detects obsolete programs. Detection of obsolete programs is one instance of the Type 4 semantic clone detection. Neural models have been shown to be most successful for this task [42]. Custom code is usually unstructured bulk of programs while SAP uses so-called tile architecture in which all programs are classified by business area. Therefore, we can also specify the detection of obsolete custom programs as a text classification task with the classes from SAP core. Deep Learning-based text classification approaches are in this case also the most successful [25].

These four stages of the custom code analysis deliver facts about the actual state of the software code in a particular ERP system with respect to a particular target release version. Thus, these analysis steps need to be repeated for each source-target pair each time.

Usually, the results of such an analysis are presented to a business analyst in the form of hundreds of spreadsheets (the number depends on the system's size and the number of identified issues in the custom code). The business analyst makes decisions related to the code transformation using standard business intelligence software. We decided to automate this process using multi-modal conversational interfaces. We store the facts learned about the current state of the system in a knowledge base. The next section explains the implementation of the multi-model conversational interface.

3.2 Multi-modal Conversational Interface

Conversational interfaces, also called chatbots, support explainability of complex technical systems [16] and facilitate complex problem solving [11]. Multi-modal conversational interfaces have been explored in the bio-informatics domain [8] and fashion retail applications [20]. The authors in [8] formulate seven design

principles for multi-modal conversational interfaces which we use for the design of our system. Figure 2 shows the current modalities.

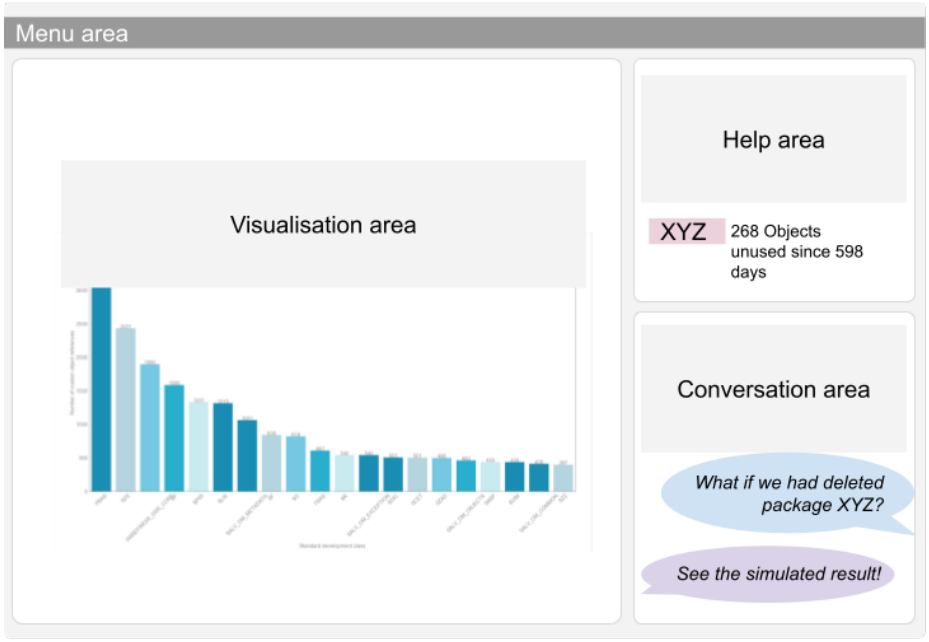

Fig. 2. Preliminary design of the multi-modal conversational interface

The visualisation area presents the contextual view on the data from the knowledge base as requested by the user via the conversation area. The conversation area is designed and developed as a standard chatbot design pipeline consisting of skills, natural language understanding (NLU) module and natural language generation (NLG) module

1. Skills are defined manually using the SAP CAI interface[6]. Each skill contains the domain knowledge of the analysed custom code, e.g. obsolete code, unused code etc. Therefore, the skills are the same for each system to be analysed and explored with the help of the new interface.
2. Because Intent-based language understanding is commonly used in NLU platforms for task-based interactions. Intents represent the identified meaning of a user's utterance. NLU models are usually trained on a large number of examples of user inputs. Because no conversational training data for the custom code upgrade domain are available, we use other types of documents such as analysis reports and written customer communication.

[6] https://cai.tools.sap, former recast.ai

3. For the NLG module, we define the chatbot's interaction profile that is consistent with the corporate environment. We use the persona methodology to define the chatbot's personality aligned with the company's brand [23].

The advantage of the multi-modal interface is that the users can simulate software code transformation results before they trigger the actual transformation in the development system. The so-called "what-if"-analysis can be easily initiated in the conversational area of the interface, and visualised in the visualisation area. Because bad user experience also may negatively influence the explainability of the system [34], we apply recent conversational user experience (CUX) findings when designing the interaction with the multimodal interface, see for instance [14].

3.3 Source-code Transformation

An automated source-code transformation can be then triggered via the conversational interface and includes:

1. Archiving of unused custom programs;
2. Removal of obsolete programs;
3. Code transformation for custom programs that are currently in use and whose function is not covered by the core programs. Modification of functions incompatible with new interfaces (i.e. cloud interfaces).

The latter item, automated source-code transformation, is also known as source code translation or language migration. Academic literature reports various types of source-code translation: translation from a programming language to pseudo-code and back [30]; from one programming language to a different programming language [17]; from code with errors to code without errors (code repair) [24]; from proprietary languages to non-proprietary languages (e.g. ABAP to Java) [27]; from source code to natural language (source code summarisation) [15]; and from a natural language to API code templates [29]. In this project we perform transformations from ABAP to ABAP (proprietary language) while correcting errors in the code.

4 Explainability Requirements and Costs

As explained in Section 3, ML is used in several phases of the analysis and transformation process, and also for the conversational part of the multi-modal interface. In this section we analyse, what kind of explainability we need at which step, and what are their costs. In order to obtain the requirements for explainability, we use the aggregate taxonomy presented in Sec. 2. We summarise in Table 2 how the explainability requirements change if the proposed innovation replaces the existing rule-based system.

Dimension	Before	After
User	ABAP developer, project manager, business analyst	ABAP developer, project manager, business analyst, **ML/NLP engineer**
Object	Data: causality	Data: **bias**, causality
	Model: classification	Model: **optimiser**, **training**, prediction
	Evaluation: accuracy, safety	Evaluation: accuracy, safety
Scope	Local: single rules	Local: **single prediction of an error** **Global: behaviour of the code analyser and its parts, behaviour of the NLU**
Directness	Directly interpretable	**Post-hoc**, **surrogate**
Interaction	Static	**Interactive**
Interface	Spreadsheets	**Multi-modal, conversational**

Table 2. Changes in explainability requirements caused by transformation from the rule-based solution to the ML-based solution described in Sec.3, differences marked in bold

4.1 Explainability Requirements

As suggested in Sec. 3.1, the code transformation task can be defined as a machine-translation problem. NLP methods have been successfully used for similar tasks for software code (see Sec. 3.3 for references). Neural machine-learning models deliver the most accurate results on tasks similar to machine translation, including software code migration. However, these models are the least interpretable. Although attempts are made to make neural models explainable [38, 3], the explanations are mostly limited to visualisation of functions and variables. Such explanations are only accessible to ML experts. For instance, if visualisation of a function in the model training process has a particular curve, what does it mean for the expert? How would an expert explain this curve to a non-expert (simplified explanations)? This question must find an answer in the design of the multi-modal interface: the knowledge stored in the knowledge base needs to be formulated in natural-language utterances and/or visual elements in a way accessible to the target user group.

Because different roles may need different explanations, we analyse here the explainability requirements by role:

1. ML/NLP engineer: is interested in validating the models and the data for a particular release version and a particular set of custom programs. This role would need static views on data (both bias and causality) and models (optimizer and training) as well as evaluation of the accuracy. Static visualisations are an appropriate way used in most ML tools to generate surrogate-based and post-hoc explanations. This role would need explanations on the global level, but may need to have access to single predictions for evaluation.

2. ABAP developer: wants to know why a particular program has been classified as incorrect or obsolete. The explanation needs to be static, local and post-hoc at the level of model.
3. Project manager: needs to ensure a smooth execution of the project and is interested in the global picture of the code analysis in order to plan human resources for post-processing and quality assessment. Explanations for this role need to be interactive, global, post-hoc, the object of the explanations will be mostly model and evaluation.
4. Business analyst: is interested in the global picture and in simulations. This role would need interactive, global explanations of the surrogate model.

A set of explainability tools similar to AIX360 described in [2] can be built and offered to the users via multi-modal interface.

Because the user of the analysis results will mainly interact with the new multi-modal interface, static explanations can be generated in the process of interaction with the interface. The explanations can have the form of natural-language utterances generated by the chatbot, or visual shown in the visualisation area, or a synchronised version of both areas where a visualisation is generated and an utterance explains what the visualisation is supposed to show. The conversational area of the interface does not have to be restricted on text input. As CUX research suggests, a wise combination of textual and visual elements improves the CUX [14], and consequently, it improves transparency of the system to the target user group. Therefore, we use the 12 heuristics suggested in [14] for the interface and interaction design.

In addition, humans usually use less precise formulations in their requests that computer systems store in the models. As Hagras (2018) [13] and Alonso et. al ([1]) point out, humans are able to communicate effectively with imprecisely defined labels such as *slow*, *slightly* and *infrequent*, while computer systems need a specific numerical value for such labels. The numerical values for these labels would be different for different people, but the conversational interface needs to map them to precise numbers. Both works suggest using fuzzy logic to deal with such issues. One possible consequence of this could be additional effort in the system's explanations of the sort "Did you mean...?"

4.2 Explainability Costs

The number of tools that support explainability tasks is growing and the range of the problems tackled becomes wider and wider. We can see the explainability requirements for which no tools or methods currently exist as infinitely large. For those requirements that are currently supported by tools, we set the cost at 1. For directly interpretable parts of the systems, we set the costs as 0.

As shown in Table 2, the new system needs to be made explainable for different roles. Each role may need different parts of the system to be explained in different ways. Therefore, costs of explainability can be different for different roles. In Table 3 we specify the explainability costs for each of the parts of the new technology.

Step	Explainability type	Cost
Unused code detection	Directly interpretable	0
Syntactic analysis	Post-hoc or surrogate	1
Semantic analysis		
- Structural clones	Directly interpretable	0
- Semantic clones	Post-hoc or surrogate	1
Funcional analysis	Post-hoc or surrogate	1
NLU	Post-hoc	1
NLG	Directly interpretable	0
Simulation	Surrogate	1

Table 3. Explainability costs for each step in the innovation process

As we can see from Table 3, it is possible to find an explainable model for each part so that costs are never infinitely large. Specific methods for explainable neural models have been developed, for example [38, 19, 4, 22]. However, they may be insufficient for users who are not ML engineers. The multi-modal conversational interface can help to solve this issue in the form of feature generalisation and zoom-in/zoom-out visualisations. Also, simplified explanations as described above would be required.

5 Conclusions

This research was motivated by the question, whether an XAI system would cause addition costs as compared to non-XAI version. We analysed a use case from the SAP custom code transformation domain. To evaluate the costs we proposed an aggregated taxonomy for XAI based on six dimensions: user, object, scope, directness, interaction and interface.

We found that the new approach will cause additional effort at all six dimensions. However, due to recent progress in the implementation of explainable neural models such as [38, 19, 4, 22], the costs are not infinitely large. Nevertheless, the creation of a multi-modal interactive interface for explanations would require integration of the explainable models in the process from the beginning.

The proposed approach brings the risk that the explainable methods described in the academic literature are not suitable for deployment in a deployed system. However, tool sets for XAI such as AIX360 show that it is technically possible at least in some cases.

References

1. Alonso, J.M., Barro, S., Bugarin, A., van Deemter, K., Gardent, C., Gatt, A., Reiter, E., Sierra, C., Theune, M., Tintarev, N., Yano, H., Budzynska, K.: Interactive natural language technology for explainable artificial intelligence. In: et al., F.H.

(ed.) 1st Workshop on Foundations of Trustworthy AI integrating Learning, Optimisation and Reasoning, European Conference on Artificial Intelligence. Springer (2021)
2. Arya, V., Bellamy, R.K., Chen, P.Y., Dhurandhar, A., Hind, M., Hoffman, S.C., Houde, S., Liao, Q.V., Luss, R., Mojsilović, A., et al.: One explanation does not fit all: A toolkit and taxonomy of AI explainability techniques. arXiv preprint arXiv:1909.03012 (2019)
3. Bach, S., Binder, A., Montavon, G., Klauschen, F., Müller, K.R., Samek, W.: On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. PloS one **10**(7), e0130140 (2015)
4. Baldassarre, F., Azizpour, H.: Explainability techniques for graph convolutional networks. arXiv preprint arXiv:1905.13686 (2019)
5. Barth, C., Koch, S.: Critical success factors in erp upgrade projects. Industrial Management & Data Systems (2019)
6. Chazette, L., Schneider, K.: Explainability as a non-functional requirement: challenges and recommendations. Requirements Engineering **25**(4), 493–514 (2020)
7. Chiticariu, L., Li, Y., Reiss, F.: Transparent machine learning for information extraction: state-of-the-art and the future. EMNLP (tutorial) (2015)
8. Crovari, P., Pidó, S., Garzotto, F., Ceri, S.: Show, don't tell. reflections on the design of multi-modal conversational interfaces. In: International Workshop on Chatbot Research and Design. pp. 64–77. Springer (2020)
9. Davenport, T.H., Harris, J.G., Cantrell, S.: Enterprise systems and ongoing process change. Business process management journal (2004)
10. Eder, S., Junker, M., Jürgens, E., Hauptmann, B., Vaas, R., Prommer, K.H.: How much does unused code matter for maintenance? In: 2012 34th International Conference on Software Engineering (ICSE). pp. 1102–1111. IEEE (2012)
11. Fast, E., Chen, B., Mendelsohn, J., Bassen, J., Bernstein, M.S.: Iris: A conversational agent for complex tasks. In: Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems. pp. 1–12 (2018)
12. Ferreira, J.J., Monteiro, M.d.S.: Do ML experts discuss explainability for AI systems? a discussion case in the industry for a domain-specific solution. arXiv preprint arXiv:2002.12450 (2020)
13. Hagras, H.: Toward human-understandable, explainable AI. Computer **51**(9), 28–36 (2018)
14. Hoehn, S., Bongard, K.: Heuristic evaluation of covid-19 chatbots. Proceedings of CONVERSATIONS 2020 (2020)
15. Iyer, S., Konstas, I., Cheung, A., Zettlemoyer, L.: Summarizing source code using a neural attention model. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 2073–2083 (2016)
16. Jentzsch, S.F., Höhn, S., Hochgeschwender, N.: Conversational interfaces for explainable AI: a human-centred approach. In: International Workshop on Explainable, Transparent Autonomous Agents and Multi-Agent Systems. pp. 77–92. Springer (2019)
17. Karaivanov, S., Raychev, V., Vechev, M.: Phrase-based statistical translation of programming languages. In: Proceedings of the 2014 ACM International Symposium on New Ideas, New Paradigms, and Reflections on Programming & Software. pp. 173–184 (2014)
18. Kostoulas, J., Anderson, R., Pang, C.: Market share analysis: Erp software, worldwide, 2018. Gartner (2019)

19. Kumar, P., Singh, A., Kumar, P., Kumar, C.: An explainable machine learning approach for definition extraction. In: International Conference on Machine Learning, Image Processing, Network Security and Data Sciences. pp. 145–155. Springer (2020)
20. Liao, L., Zhou, Y., Ma, Y., Hong, R., Chua, T.s.: Knowledge-aware multimodal fashion chatbot. In: Proceedings of the 26th ACM international conference on Multimedia. pp. 1265–1266 (2018)
21. Lipton, Z.C.: The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery. Queue **16**(3), 31–57 (2018)
22. Mahoney, C.J., Zhang, J., Huber-Fliflet, N., Gronvall, P., Zhao, H.: A framework for explainable text classification in legal document review. In: 2019 IEEE International Conference on Big Data (Big Data). pp. 1858–1867. IEEE (2019)
23. Mano Ferreira, C., Hoehn, S.: Crafting conversational agents' personality in a user-centric context. In: Proceedings of the 31st Benelux Conference on Artificial Intelligence (BNAIC 2019) and the 28th Belgian Dutch Conference on Machine Learning (Benelearn 2019). pp. 1–2 (2019)
24. Mesbah, A., Rice, A., Johnston, E., Glorioso, N., Aftandilian, E.: Deepdelta: learning to repair compilation errors. In: Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. pp. 925–936 (2019)
25. Minaee, S., Kalchbrenner, N., Cambria, E., Nikzad, N., Chenaghlu, M., Gao, J.: Deep learning based text classification: A comprehensive review. arXiv preprint arXiv:2004.03705 (2020)
26. Murdoch, W.J., Singh, C., Kumbier, K., Abbasi-Asl, R., Yu, B.: Definitions, methods, and applications in interpretable machine learning. Proceedings of the National Academy of Sciences **116**(44), 22071–22080 (2019)
27. Nandivada, V.K., Nanda, M.G., Dhoolia, P., Saha, D., Nandy, A., Ghosh, A.: A framework for analyzing programs written in proprietary languages. In: Proceedings of the ACM international conference companion on Object oriented programming systems languages and applications companion. pp. 289–300 (2011)
28. Nguyen, A.T., Nguyen, T.T., Nguyen, T.N.: Lexical statistical machine translation for language migration. In: Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering. pp. 651–654 (2013)
29. Nguyen, A.T., Rigby, P.C., Nguyen, T., Palani, D., Karanfil, M., Nguyen, T.N.: Statistical translation of english texts to api code templates. In: 2018 IEEE International Conference on Software Maintenance and Evolution (ICSME). pp. 194–205. IEEE (2018)
30. Oda, Y., Fudaba, H., Neubig, G., Hata, H., Sakti, S., Toda, T., Nakamura, S.: Learning to generate pseudo-code from source code using statistical machine translation (t). In: 2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE). pp. 574–584. IEEE (2015)
31. Olson, D.L., Staley, J.: Case study of open-source enterprise resource planning implementation in a small business. Enterprise Information Systems **6**(1), 79–94 (2012)
32. Rattan, D., Bhatia, R., Singh, M.: Software clone detection: A systematic review. Information and Software Technology **55**(7), 1165–1199 (2013)
33. Rothenberger, M.A., Srite, M.: An investigation of customization in erp system implementations. IEEE Transactions on Engineering Management **56**(4), 663–676 (2009)
34. Shariat, J., Saucier, C.S.: Tragic Design: The Impact of Bad Product Design and How to Fix It. " O'Reilly Media, Inc." (2017)

35. Svacina, J., Simmons, J., Cerny, T.: Semantic code clone detection for enterprise applications. In: Proceedings of the 35th Annual ACM Symposium on Applied Computing. pp. 129–131 (2020)
36. Technologies, S.: Removing unused code matters (2017), https://smartshifttech.com/removing-unused-code-matters/
37. Tomsett, R., Braines, D., Harborne, D., Preece, A., Chakraborty, S.: Interpretable to whom? a role-based model for analyzing interpretable machine learning systems. arXiv preprint arXiv:1806.07552 (2018)
38. Vaughan, J., Sudjianto, A., Brahimi, E., Chen, J., Nair, V.N.: Explainable neural networks based on additive index models. arXiv preprint arXiv:1806.01933 (2018)
39. Ventocilla, E., Helldin, T., Riveiro, M., Bae, J., Boeva, V., Falkman, G., Lavesson, N.: Towards a taxonomy for interpretable and interactive machine learning. In: XAI Workshop on Explainable Artificial Intelligence. pp. 151–157 (2018)
40. Verma, S., Rubin, J.: Fairness definitions explained. In: 2018 ieee/acm international workshop on software fairness (fairware). pp. 1–7. IEEE (2018)
41. Wang, P., Yang, J., Tan, L., Kroeger, R., Morgenthaler, J.D.: Generating precise dependencies for large software. In: 2013 4th International Workshop on Managing Technical Debt (MTD). pp. 47–50. IEEE (2013)
42. Yu, H., Lam, W., Chen, L., Li, G., Xie, T., Wang, Q.: Neural detection of semantic code clones via tree-based convolution. In: 2019 IEEE/ACM 27th International Conference on Program Comprehension (ICPC). pp. 70–80. IEEE (2019)