Analysing Wiki Quality using Probabilistic Model Checking

Giuseppe De Ruvo, Antonella Santone
Department of Engineering, University of Sannio, Benevento, Italy
e-mail: {gderuvo, santone}@unisannio.it

***Abstract*—Wikis delineate a new work tool in enterprises and they are spreading everywhere. Indeed, they are often used as internal documentation for various in-house systems and applications as well as powerful tools for collaboration and knowledge sharing. As occurs with software, the fundamental growth of a wiki may lead to its degradation. The quality of wikis, especially in enterprise contexts, should not play a trivial role. Software quality is a very discussed topic, but there are not many studies regarding the quality of wikis.**

We propose a probabilistic model to represent wikis and to investigate their quality. Due to the similarity with the World Wide Web it is natural to consider the popular Google PageRank (with minor modifications) to calculate probabilities between pages. Each wiki category, a set of wiki pages, is modelled using the PRISM language in order to verify specific properties in PCTL*. Experiments conducted on a adequate number of (enterprise) wikis assess the validity of our methodology.

***Index Terms*—Wiki, Probabilistic Temporal Logic, PRISM, Quality.**

I. Introduction and Related Work

Wikis have been used as Knowledge Base, Encyclopedia and Support Base, acting as new tools for group work, intra organization and management [1]. They have formed participative environments, allowing anyone to create, share, and modify content in an easy and intuitive way - even users with very limited technical expertise. The first wiki was created in 1995 by Ward Cunningham, but they have deeply changed from the original purpose. There are hundreds of wiki engines available[1] which provide a plethora of features. In this paper, we call *enterprise wikis* special wikis tailored to particular environments and regarding specific topics. We analyse them aiming at understanding what is the quality of their structure. Thus, inspired by software bad smells [2], [3], we identify some clues of good (or bad) quality. We propose a variant of our methodology [4] in order to critically deal with wiki quality. The latter is strictly connected with knowledge organization, management and user interaction [1], e.g. easiness of browsing pages. We use probabilistic model checking which is a formal technique for proving the correctness of a system that exhibit random or probabilistic behaviour. This is accomplished by checking whether a structure representing the probabilistic system (i.e., Discrete Time Markov Chain) satisfies a logic formula describing the expected behavior. In this paper we will show how to use probabilistic model checking to analyse and verify wiki quality transforming them into PRISM [5] modules and then checking whether they satisfy suitably-defined properties with a given probability. Properties are expressed using the PCTL* logic [6]. We exploited the capabilities of (probabilistic) model checking in a rather unexplored area with encouraging results. Our methodology has been tested on a adequate number of *enterprise wikis*.

As far as we know nobody addressed the analysis and discussion of wiki quality using probabilistic model checking. The first attempt can be found in [4], where the authors proposed a novel methodology based on model checking to analyse and verify the architecture of wikis. This approach has been extended in this paper in order to model with Markov chains the navigation among pages, especially because some links are more likely to be used than others. However, there are some works regarding wiki quality and evolution.

Rosenfeld et al. [7] proposed an approach to detect quality problems in semantic wikis inspired, like us, by the bad smell problems in software engineering. The approach mostly focus on annotations in order to incrementally create a structured ontology, while we work on links to identify quality indicators.

Alluvatti et al. [8] also investigated on the quality and evolution of wikis. They proposed to evaluate the quality of wikis basing on the number of edits and contributors per page.

Flekova et al. [9] studied the user-perceived quality of Wikipedia articles based on a novel Wikipedia user feedback dataset. They analyzed ratings of ordinary Wikipedia users along four quality dimensions (complete, well written, trustworthy and objective), using current text classification and language processing tools.

Unlike other works, in our opinion, our approach is valuable also in different contexts developing other types of wiki analyzers [10].

The remainder of this paper is organized as follows. First, in the next Section we give basic definitions of wikis, probabilistic model checking and PRISM and Google PageRank. Section III deals with our methodology to investigate wiki quality. In Section IV we discuss achieved results and how they can be used to summarise wiki quality. Finally, in Section V we provide new inputs for further research.

II. Preliminaries

In this section, first we introduce wikis with basic definitions - main topic of our research. Second, we quickly recall probabilistic model checking and PRISM [5] which we

[1] http://www.wikimatrix.org

employed to model and verify wikis. Finally, we give a very short introduction regarding Google PageRank [11], since we exploited its calculation to set the probabilities between pages.

A. Wikis

A wiki (from the Hawaiian wiki, to hurry, swift, quick) is a collaborative Web site whose content can be edited by anyone who has access to it.

A wiki is generally divided in categories. Each category may be split into subcategories. Categories contain pages and each page belong to different categories. Pages are divided into sections and each section may be divided into subsections. Each page has links to other pages or categories and have external links (i.e., links to web pages). Since wikis are easy to edit, they have changed how we construct knowledge repositories on the Web. Wikis allow groups to form around specific topics and they are a great way for a group of people to coordinate and create content, even though that group counts thousands of people in different places [12]. Here are some typical things we can do on a wiki: create an article on a specific topic; make changes to other people's articles, without requiring their permission; create links between articles; group similar articles together into convenient categories; view the history of an article to see all the changes, who made them, and when; see interesting statistics about the articles i.e., which ones are most popular and/or need updating.

There are disparate types of wikis such as Wiki Mapia that combines Google Maps with a wiki system supporting over 35 languages; WikiTravel, a travel guide; LyricWiki, a listing of lyrics by album; Flu Wiki, intending to help local public health communities coping with a possible (avian) influenza pandemic, and Ganfyd, an online collaborative medical reference that is edited by medical professionals and invited non-medical experts; Diplopedia, billed as the Encyclopedia of the USA Department of State; IkeWiki [13], a semantic Wiki developed at Salzburg research for collaborative knowledge engineering. While it has been developed primarily as a tool for ontology engineering, it can be used in a variety of application scenarios.

Wikis may also fit in the context of innovation and learning in organizations [14], [15], [16]. Moreover, many FLOSS projects use wikis for technical and user documentation, i.e., Eclipse and its Eclipsepedia.

In Section IV we will analyse an adequate number of *enterprise wikis*.

B. Probabilistic Model Checking and PRISM

Model checking is a technique to establish the correctness of hardware or software systems in an automated fashion. Conventional model checkers input a description of a model, represented as a state transition system, and a specification, typically a formula in some temporal logic, and return yes or no, indicating whether or not the model satisfies the specification. In the case of probabilistic model checking, the models are probabilistic, in the sense that they encode the probability of making a transition between states instead of simply the existence of such a transition, and the analysis normally entails calculation of the actual likelihoods through appropriate numerical or analytical methods [17].

There exist many probabilistic models. In this paper we use discrete-time Markov chains (DTMCs), which specify the probability $\pi(s, s')$ of making a transition from state s to some target state s', where the probabilities of reaching the target states from a given state must sum up to 1, i.e., $\sum_{s'} \pi(s, s') = 1$.

A probabilistic model checker tool automates the correctness proving process. A popular probabilistic model checker is PRISM [5], successfully adopted in many application domains such as communication and multimedia protocols, randomised distributed algorithms, security protocols, biological systems, risk management domain [18], [19]. In PRISM the system model is defined with a probabilistic reactive module and system behaviours are defined in terms of properties written in Probabilistic Computation Tree Logic (PCTL*) [6], a probabilistic extension of the temporal logic CTL. More precisely, a PRISM model is composed of *modules*, whose state is determined by a set of *variables* and whose behaviour is specified by a set of *guarded commands*. A guarded command contains an (optional) action label, a guard variable, and a probabilistic update definition for the modules variables:

```
[action] guard -> prob1 : update1 + ... + probn : update;
```

When a module has a command whose guard is satisfied in the current state, it can update its variables probabilistically, accordingly to the update definition. For action-labelled commands, multiple modules execute updates synchronously, if all their guards are satisfied. Each probabilistic transition in the model is thus associated with either an action label or a single module. System behaviours are defined in terms of properties written in PCTL*. A fundamental feature of PCTL* logic is the probabilistic operator P, which allows one to reason about the probability that executions of the system satisfy some property. For example, the formula: $P =?\ [\ F\ G(\text{“}error\text{”}\ \&\ !\text{“}repair\text{”})\ ]$ returns "the probability of an error occurring that is never repaired".

C. Google PageRank

Underlying the definition of PageRank is the following basic assumption. A link from a page $u \in Web$ to a page $v \in Web$ can be viewed as evidence that v is an "important page". In particular, the amount of importance conferred on v by u is proportional to the importance of u and inversely proportional to the number of pages u points to. The mathematical grounds of ranking of pages are based on the concept of Markov chains and related class of Perron-Frobenius operators naturally appearing in dynamical/bio systems [20]. A concrete implementation of these mathematical concepts to the ranking of WWW pages was started by Brin and Page in 1998 [11].

Thus, let Google Matrix be: $G_{ij} \stackrel{\text{def}}{=} \alpha S_{ij} + (1 - \alpha)/N$, where the matrix S_{ij} is obtained from an adjacency matrix A_{ij} by normalizing all nonzero columns to one and replacing columns with only zero elements by $1/N$ (dangling nodes) with N being the matrix size. The largest eigenvalue of G is $\lambda = 1$ and other eigenvalues have $|\lambda| \leq \alpha$. The right

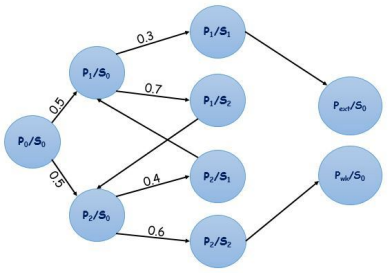

Fig. 1: A DTMC of a small wiki category - 2 pages and 2 sections

```
dtmc
module categoryExample
        p : [0..5] init 0;
        z : [0..4] init 0;
        s : [0..2] init 0;

        [] p=0 -> 0.5:(p'=1) + 0.5:(p'=2);
        [] p=1 -> 0.3:(z'=1)&(s'=1)&(p'=5) + 0.7:(z'=1)&(s'=2)&(p'=5);
        [] p=2 -> 0.4:(z'=2)&(s'=1)&(p'=5) + 0.6:(z'=2)&(s'=2)&(p'=5);
        [] p=3 -> (p'=3); //ext site Pext
        [] p=4 -> (p'=4); //ext category Pwk

        [] z=1 & s=1 -> (p'=3)&(s'=0);
        [] z=1 & s=2 -> (p'=2)&(s'=0);
        [] z=2 & s=1 -> (p'=1)&(s'=0);
        [] z=2 & s=2 -> (p'=4)&(s'=0);
endmodule
```

Fig. 2: Small example of wiki category as PRISM module

eigenvector at $\lambda = 1$, which is called the PageRank, has real nonnegative elements $P(i)$ and gives a probability $P(i)$ to find a random surfer at site i.

III. The Methodology

In this section we present our methodology to analyse and verify the quality of wikis. We extend an approach, based on formal methods [4], exploiting probabilistic model checking that is able to capture the aleatory nature of the problem. More precisely, from the wikis we derive PRISM modules, which are successively used to perform probabilistic model checking, as can be seen in Figure 3. The goal of our methodology is to provide clues to understand the quality of wikis. Four steps are required by our process. The four steps are:

1) PRISM Model Creation; 2) Formal Verification process (probabilistic model checking); 3) Synthesis Generation; 4) Quality Evaluation. In the following subsections the four steps are discussed in detail.

A. PRISM Model Creation

We use as internal representation the PRISM language. Thus, PRISM modules are generated from wikis. For the sake of clarity in this section we only present a simple example.

First of all, when analysing a category C, we have to distinguish between different pages belonging to C. A page can belong to the same category C, to a different category or can be an external web page.

A small DTMC regarding two pages composed of two sections each is depicted in Figure 1, where P_{wk} and P_{ext} indicate respectively an external page of category and an

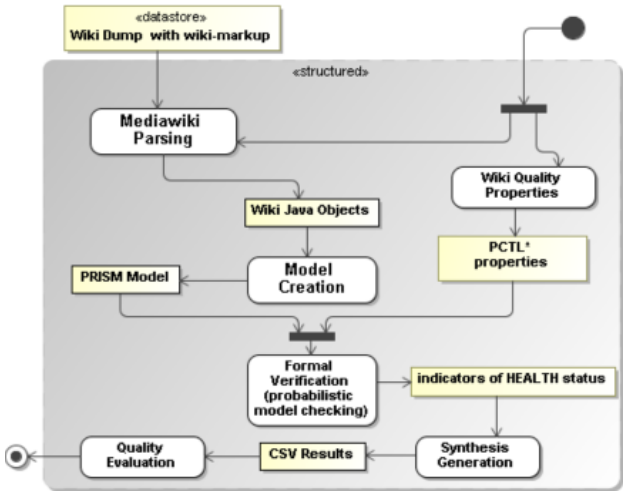

Fig. 3: An UML activity diagram explaining our methodology

external web page. Each node of the DTMC in Figure 1 represents a section of a page, i.e., P_1/S_2 indicates Section 2 of Page 1. Every category C has been translated into a PRISM module. The module that translates the wiki category in Figure 1 is shown in Figure 2. Thus, in our model we divide pages into:

- the internal page P belongs to the analysed category C ($p = 1$ or $p = 2$ in Figure 2);
- the internal page P belongs to a different category ($p = 4$ in Figure 2);
- the page P is an external web page ($p = 3$ in Figure 2).

The variable p in Figure 2 represents a wiki page. It is initially set to 0 meaning that we are choosing a page to navigate. Instead, states of s indicate that we are choosing a section. Therefore, for example, if $p = 0$ we are in the initial state and we will move to state $p = 1$ with probability 0.5 or to state $p = 2$ with probability 0.5, with s still equal to 0, meaning that no section has yet been chosen.

Looking at Figure 2 we can also notice a state $p = 5$ for the variable p and another integer variable z. Both are needed to represent the user behaviour. In other words, every update in the module is a link of the wiki category and z is a necessary duplication to correctly represent the behaviour of wiki readers, i.e. to trace from which page the user is coming.

Google PageRank [11] yields a score for each page. We exploited the matrix of probabilities employed by Power Method [21] to obtain the weights of the links. We also have to consider the *damping* factor i.e., α (see Section II-C). Such a factor exists to allow users to move from one page to another even though the pages are not connected i.e., the ones which we got writing the exact URL in the navigation bar. Notwithstanding, we do not need neither α nor the entire $(1-\alpha)$ term, since we want to examine only the pages which can be reached by links. For this aim, we choose an α very next to 1 whilst retaining constrain $0 < \alpha < 1$ to have a primitive matrix as in the original algorithm.

B. Formal Verification process (probabilistic model checking)

In our approach, we use probabilistic model checking to verify wikis. Once we have the PRISM modules of wikis, we can use PCTL* logic to specify desired properties. Inspired by software engineering refactoring, we reformulate code smells [2] in wiki content, that can be viewed as a drawback of a wiki page (and category) and generally they indicate the quality of the structure of a wiki. We consider the following properties representing wiki smells, inspired by greek classics.

Temple Property

$$P = ?\ [F\ ((p = j)\ \&\ F\ ((p = i))) \mid (\ F\ (p = i) \& ((XX\ p! = extC)\ \&\ (XX\ p! = extS)))]$$

where $extS$ is an external web page, $extC$ is a page that does not belong to the inspected category and j indicates a page belonging to the inspected category. Thus we need to verify this property for every j.
If $P = 0$ page i is a temple, i.e., page i cannot be reached starting from any page of the analysed category and no page can be reached from the page i apart from $extS$ and $extC$ pages.

Closed Chain Property

$$P = ?\ [\ F\ p = i\ \&\ (X\ F\ (p = i))\ ]$$

If $P > 0$ page i belongs to a closed chain, i.e., page i belongs to an infinite path starting from the initial state.

God Property

$$P = ?\ [\ F\ p = j\ \&\ (X\ X\ (p = i))\ ]$$

If $P > 0$ page i is a god (candidate) page for page j, i.e., page i can immediately be reached from page j.

Minor God Property

$$P = ?\ [\ F\ p = j\ \&\ (F\ (p = i))\ ]$$

If $P > 0$ page i is a minor god (candidate) page for page j, i.e., page i can be reached from page j after having crossed other pages. In other words, page i can be indirectly reached from page j.

Other properties
Unlike other bespoke wiki analysis techniques, methods and tools, we can easily add new properties and proceed with new analyses thanks to the power of (probabilistic) model checking. In fact, it is sufficient to express the new meaningful properties of interest in PCTL* logic.

C. Synthesis Generation

PRISM is the state of the art of probabilistic model checking. It can even export the results of verification in various formats i.e., CSV. Such results are further employed by a Java prototype to check how many and which pages satisfy the properties and proceed with quality evaluation.

D. Quality Evaluation

The final activity deals with quality evaluation. After model creation, automatic (probabilistic) model and synthesis generation, we have all the necessary products to discuss regarding wiki quality. This is a manual task.

We employ the results of probabilistic model checking to figure out their meaning and possible solutions. Our goal is to provide "indicators of health status" of *enterprise wikis*. The success factors and main features of the *enterprise wikis* are knowledge sharing, organization, management and better users interaction [1]. A page identified as "temple" does not communicate with any other page: it is an isolated page. The latter does not contribute to a good organization and sharing of knowledge. If we are dealing with a whole wiki, this may be a bad sign of quality, with content that may not fit in that context - i.e., a wrong page in a project. Maybe something was planned, but we forgot to deal with that page or to link it to other pages in order to be reachable and readable by users or we forgot to create a category to group a certain amount of pages. An *enterprise wiki* with many temples is more an obstacle than an aid.

A "Closed Chain" (CC) indicates that the page is involved in a closed chain. This is a clue that the pages composing such a chain behave as concepts that span across multiple pages. Without considering the other indicators, the presence of CCs is a sign of medium quality. It is like the wiki is migrating towards an improved structure. In fact, such chains may be employed to create categories regarding the pages involved.

When a page is referred by another page, it is a candidate to be elected as a "God Page". On the one hand, a "Minor God" page may arise when a page is indirectly referred by another page. Moreover, a "God" page means that such a page is directly linked to the one whose property is verified with a probability greater than a chosen threshold. God pages operate as little concentrations of knowledge and are a good source for learning how to use the wiki for new users. In fact, they can understand how to link and cope with old and new pages. Conversely, minor gods are another good origin towards a better organization. Indeed, if we follow the indirect links, the involved pages act as "cut chains" or part of chains.

The properties we defined are flexible and may be adapted to various domains. For example, one may change aforementioned probabilities or even add new domain oriented quality attributes and properties.

IV. Experimental results

MediaWiki[2] is a popular free web-based wiki software application, developed by Wikimedia Foundation. It is the main engine of most wikis all over the world. Indeed, all the wiki pages are stored in a MySQL database. The Wiki Markup Language is adopted for both pages and relationships (i.e., links) [4]. The database dumps of wikis are usually freely available. We downloaded some ones dumped by WikiTeam [3]. We employed a MediaWiki parser - capable of locating categories, pages and sections from Wiki Markup Language

[2]http://www.mediawiki.org/wiki/MediaWiki
[3]https://archive.org/details/wikiteam

TABLE I: Results of eight wikis considering a probability greater than zero (P $>$ 0) and greater than a given α (P $>$ α)

Wiki	Temples		CC		Gods		mGods	
	P $>$ 0	P $>$ α	P $>$ 0	P $>$ α	P $>$ 0	P $>$ α	P $>$ 0	P $>$ α
sea	0%	-	20,75%	3,77%	50%	33%	30%	18%
gt4	70,37%	-	0%	0%	0%	0%	0%	0%
start	60%	-	16,2%	4,32%	6,2%	6,2%	3%	3%
cond	43%	-	9%	5%	10%	9%	24%	14%
bio	70,1%	-	11%	7%	6,1%	2,5%	8%	4%
vigil	6,22%	-	1%	1%	10,36%	4%	2,6%	1,2%
emb	70%	-	0%	0%	5%	4%	6%	2%
prod	50%	-	6%	2%	5%	3%	18%	10%

and translated wikis in Java Objects, in order to automatically create a PRISM model (see Figure 3).

Experiments were conducted on eight *enterprise wikis*:

- "Marine Science wiki of the University of OTAGO, NZ" (sea, 53 pages);
- "Celica GT4 Drivers Club wiki" (gt4, 54 pages);
- "Startup wiki" (start, 162 pages);
- "Cookipedia UK - Condiments category" (cond, 172 pages);
- "Vegetable oil and Biodiesel wiki" (bio, 181 pages);
- "Vigilmetrics wiki" (vigil, 193 pages);
- "UNSW Embryology - 2014 category" (emb, 199 pages);
- "YSTV History wiki - Productions category" (prod, 230 pages).

We have randomly chosen wikis (or categories) different for size (increasing) and topic.

The execution time was maximum 15 minutes. We found that it is mostly due to the overhead needed to invoke the external PRISM probabilistic model checker from the Java prototype. Nevertheless, the objective of our work is to focus on the methodology rather that on the efficiency of the tool. In fact, the prototype has been shown to yield good results, even though it is not fast.

Table I shows the percentage of temples, CC, God pages (Gods) and Minor God pages (mGods) candidates per wiki (or category) when probabilities are greater than zero (P $>$ 0).

"Sea" regards the Department of Marine Science of the university of Otago in New Zealand, a multidisciplinary department with research strengths in both biological and physical marine sciences, focused on pure and applied research in oceanography and aquaculture. As can be seen in Table I, there is no isolated knowledge in the wiki, i.e., 0 temples, 20% of CC and more than 50% and 30% of god and minor god pages. This is indeed, a good quality wiki with distributed knowledge and various sources for future improvements, even though "sea" is a rather small one.

Very different are the cases of "start", a guide to city's startup and entrepreneurship community, and "bio", an unbiased and independent knowledge base for home biodiesel production and vegetable oil motoring. They contain a lot of isolated knowledge (more than 60%) and few hints to migrate towards a better organization (at least than 11% of CC and 6% of god pages), although in "bio" there exists 8% of minor god pages which may or may not be sufficient to group temples into convenient categories.

"Gt4" concerns a community of Celica GT4 owners. There are no closed chains, neither god or minor god pages and many temples (70%). "Vigil" deals with Vigilmetrics product suite, it is a good quality wiki with few temples (6%) and an adequate number of minor gods (2, 6%) which can be used as a starting point to categorize the isolated knowledge. Moreover, 10% of god pages constitutes knowledge concentration to help new users. We also analysed categories "condiments" from an UK cooking wiki, "2014" from a medical wiki regarding "Embriology" and "productions", the biggest category of a private TV wiki. As it can be seen, we can use the same quality indicators even for single categories, especially if we have an adequate number of pages.

We used probabilities (and probabilistic model checking) because we want to model the behavior of users. For this aim, we show what occurs with a probability threshold greater than zero. In fact, in Table I we also chose another threshold (P $>$ α), i.e., the average of probabilities among pages regarding each indicator for each wiki. Temples are not affected, since they do not depend on thresholds. Closed chains, gods and minor gods are strongly influenced instead. This is a very important point when we want evaluate the quality of wikis with respect to the hypothetical user behavior i.e., depending on how users actually navigate the pages we are analysing. Thus, choosing only the average probability, the number of CC, gods and minor gods decreased, since we employed Google Page Rank and probabilistic model checking. Thus, we excluded less important pages which may be very inaccurate to create categories and new links among pages. For example, in "vigil" only 4% of gods are considered, with a reduction of more than 50% with respect to the previous result. In other words, without considering a threshold greater than zero, we may have misleading clues.

Probabilities thresholds have to be decided by a human. Notwithstanding, empirically speaking, choosing the average is a first step in the direction of full automation. In our experiments we employed a common probabilities threshold like zero or average, but one may change them according to a particular domain or need.

After performing probabilistic model checking, we figured out that there were pages completely excluded by the indicators, i.e., pages that are neither temples or closed chains or gods. We called such pages *Unstructured Temples (UT)* and we obtained them through manual inspection. They are completely unorganized pages since they do not contain sections or every kind of link (even external links). They are very far from the idea of wiki: they are just text, more near to a file than a wiki page. Therefore, we consider the presence of UTs a very bad sign of quality. Figure 4 shows the percentage of temples and

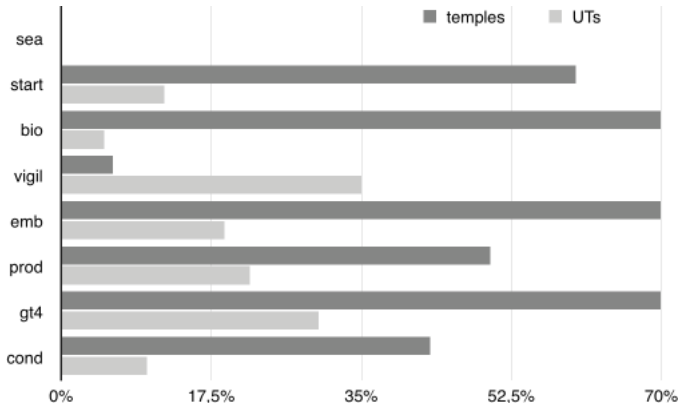

Fig. 4: Percentage of temples and UTs per category

UTs, the worst clues of bad quality we found in a wiki. For example, even "vigil", that looked a good quality wiki, has 35% of UTs. Single categories like "emb", "cond" and "prod" have few or more UTs, proving that we can obtain hints of good or bad quality wikis analysing only one category.

We work offline and avoid parsing wikis on the fly or do web-crawling. Our methodology can be easily integrated as a part of existing processes because of its simplicity.

V. Conclusion and Future Work

In this paper we have investigated a methodology based on probabilistic model checking for analyse the quality of wikis. Starting from a wiki dump, after parsing wiki markup using a MediaWiki parser, we generate a process model for each wiki (or category) employing the PRISM language and a set of specific properties expressed using the PCTL* logic. We verified these properties against the model in order to gather indicators of wiki health status. We exploited the calculation of Google PageRank to set the probabilities between pages in the probabilistic model.

We think we made a first step towards the analysis of wiki quality. Future work will focus on improving and extending the methods outlined in this article. We want to consider other kind of wikis and contact enterprises which use wikis as internal tool for their competitive advantage. The indicators we employed may be used to create new metrics to understand wiki trends, e.g. during weeks, months, years. The achieved results may be supported and compared by the types and number of contributors in a wiki, in order to understand how a team is working in that context. More data and indicators can be used to gain a full automation without human intervention [22], [23].

Finally, from an organizational point of view, our methodology may be adapted to other social contexts, like communities in twitter, facebook or linkedin.

Acknowledgements

The authors want to thank Federico Leva maintainer of Wikipedia Research Newsletter and part of WikiTeam for having provided spontaneous feedbacks and hints.

References

[1] A. Richter *et al.*, "Malleable end-user software," *Business & Information Systems Engineering*, vol. 5, no. 3, pp. 195–197, 2013.

[2] M. Fowler and K. Beck, "Refactoring: improving the design of existing code," *Addison-Wesley Professional*, 1999.

[3] F. Palomba, G. Bavota, M. D. Penta, R. Oliveto, and A. D. Lucia, "Do they really smell bad? A study on developers' perception of bad code smells," in *30th IEEE ICSME, Victoria, BC, Canada, September 29 - October 3, 2014*, 2014, pp. 101–110.

[4] G. De Ruvo and A. Santone, "A novel methodology based on formal methods for analysis and verification of wikis," in *2014 IEEE 23rd International WETICE Conference, WETICE 2014, Parma, Italy, 23-25 June, 2014*, 2014, pp. 411–416.

[5] M. Kwiatkowska, G. Norman, and D. Parker, "PRISM 4.0: Verification of probabilistic real-time systems," in *Proc. CAV'11*, ser. LNCS, G. Gopalakrishnan and S. Qadeer, Eds., vol. 6806. Springer, 2011, pp. 585–591.

[6] A. Bianco and L. De Alfaro, "Model checking of probabilistic and nondeterministic systems," in *Foundations of Software Technology and Theoretical Computer Science*. Springer, 1995, pp. 499–513.

[7] M. Rosenfeld, A. Fernández, and A. Díaz, "Semantic wiki refactoring. a strategy to assist semantic wiki evolution," in *Proceedings of the Fifth Workshop on Semantic Wikis (SemWiki 2010), co-located with 7th European Semantic Web Conference, ESWC*, 2010.

[8] G. M. Alluvatti, A. Capiluppi, G. De Ruvo, and M. Molfetta, "User generated (web) content: Trash or treasure," in *Proceedings of the 12th International Workshop on Principles of Software Evolution and the 7th Annual ERCIM Workshop on Software Evolution*, ser. IWPSE-EVOL'11. New York, NY, USA: ACM, 2011, pp. 81–90.

[9] L. Flekova, O. Ferschke, and I. Gurevych, "What makes a good biography?: multidimensional quality analysis based on wikipedia article feedback data," in *Proceedings of the 23rd international conference on World wide web*. International World Wide Web Conferences Steering Committee, 2014, pp. 855–866.

[10] G. De Ruvo and A. Santone, "Equivalence-based selection of best-fit models to support wiki design," in *2015 IEEE 24th International WETICE Conference, WETICE 2015, Larnaca, Italy, 15-17 June, 2015*.

[11] S. Brin and L. Page, "Reprint of: The anatomy of a large-scale hypertextual web search engine," *Computer Networks*, vol. 56, no. 18, pp. 3825–3833, 2012.

[12] R. Chebil, W. Chaari, S. Cerri, and K. Ghedira, "A causal graph based method to evaluate e-collaboration scenarios," in *Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), 2013 IEEE 22nd International Workshop on*, June 2013, pp. 225–230.

[13] S. Schaffert, "Ikewiki: A semantic wiki for collaborative knowledge management," in *WETICE '06*, June 2006, pp. 388–396.

[14] C. Pruski, R. Bonacin, and M. D. Silveira, "Towards the formalization of guidelines care actions using patterns and semantic web technologies," in *AIME 2011, Bled, Slovenia, July 2-6*, 2011, pp. 302–306.

[15] N. Dessì and G. Garau, "A case study for a collaborative business environment in real estate," in *ADBIS, 2013, Genoa, Italy, September 1-4, 2013. Proceedings II*, 2013, pp. 351–360.

[16] O. Nabuco, "Delivering deep health information using clinical eye," *IJWP*, vol. 5, no. 1, pp. 28–40, 2013.

[17] E. Ábrahám, B. Becker, C. Dehnert, N. Jansen, J.-P. Katoen, and R. Wimmer, "Counterexample generation for discrete-time markov models: An introductory survey," in *Formal Methods for Executable Software Models*. Springer, 2014, pp. 65–121.

[18] M. Fugini, G. C. Hadjichristofi, and M. Teimourikia, "Dynamic security modeling in risk management using environmental knowledge," in *2014 IEEE 23rd International WETICE Conference, WETICE 2014, Parma, Italy, 23-25 June, 2014*, 2014, pp. 429–434.

[19] M. Ceccarelli, L. Cerulo, G. De Ruvo, V. Nardone, and A. Santone, "Infer gene regulatory networks from time series data with probabilistic model checking," *FormaliSE 2015*.

[20] E. Demidenko, "Microarray enriched gene rank," *BioData Mining*, vol. 8, no. 1, pp. 1–18, 2015.

[21] A. D. Sarma, A. R. Molla, G. Pandurangan, and E. Upfal, "Fast distributed pagerank computation," *Theoretical Computer Science*, vol. 561, pp. 113–121, 2015.

[22] M. L. Bernardi, M. Cimitile, G. De Ruvo, G. A. Di Lucca, and A. Santone, "Improving design patterns finder precision using a model checking approach," *CAiSE forum*.

[23] G. De Ruvo and A. Santone, "An eclipse-based editor to support lotos newcomers," in *2014 IEEE 23rd International WETICE Conference, WETICE 2014, Parma, Italy, 23-25 June, 2014*, June 2014.