Software component quality prediction using KNN and Fuzzy logic

Lovre Hribar, Denis Duka
Ericsson Nikola Tesla d.d. Croatia, Split
E-mail: lovre.hribar@ericsson.com, denis.duka@ericsson.com,

Abstract: **Prediction of product quality within software engineering, preventive and corrective actions within the various project phases are constantly improved over the past decades. Practitioners and software companies were using various methods, different approaches and best practices in software development projects. Nevertheless, the issue of quality is pushing software companies to constantly invest in efforts to produce enough quality products that will arrive in time, with good enough quality to the customer. However, the quality is not for free, it has a price that is required at the time you notice about her. In this paper fuzzy logic and KNN classification method approaches are presented to predict Weibull distribution parameters shape, slope and the total number of faults in the system based on the software components individual contribution. Since the Weibull distribution is one of the most widely used probability distributions in the reliability engineering, predicting of its characteristics early in the software lifecycle might be useful input for the planning and control of verification activities.**

1. INTRODUCTION

The amount of reported errors in software components and application in general, can be described using different mathematical models as a function of time. It is also possible to observe what happens to the total amount of reported errors over time. It is possible to monitor and change parameters that affect the observed model over time, and improved the component quality through a variety of different applications. For this purpose we use the distribution models based on Weibull, Lognormal, Normal and Exponential distributions. After the amount of reported errors is described as a function of time, mathematical models and their characteristic parameters, it should be concluded in appearance and quantity of errors in the new program components, which is based on existing software components that we have enough historical data.

More precisely, the paper presents a model that is used to predict the Weibull parameters of the whole system based on the Weibull parameters obtained from components forming that same system. The companies are usually using similar history projects for predicting the parameters of the current project. There is a background assumption for using the proposed model which should be stated explicitly.

Firstly, if the model is intended to predict component reliability, then the underlying assumption is that all components forming the system have similar reliability and based on number of measured component reliabilities the most likely reliability for all components is estimated.

Secondly, if the model is intended for predicting system reliability, then the assumption is that the mean of individual probabilities equals the total system probability.

ISO 9126 standard [42] is the software product evaluation standard from the International Organization for Standardization. This international standard defines six characteristics that describe, with minimal overlap, software quality: functionality, reliability, efficiency, usability, maintainability, and portability.

Feedback on software component quality is something which we need very early in the project. It is very hard to predict and determine the quality of the software components at the beginning of the project. In addition, the software quality prediction has a significant role in easing the maintenance of software. In short, the prediction is helpful in software development, testing and maintenance activities.

Predictions can either be made on the basis of historical data collected during implementation of same or similar projects, or it can be made using the design metrics collected during design phase [43].

During the different development stages the quality of the software component must be tracked and visible. There are a number of the applied techniques but most important is the one which are tracking the number of faults for the software component [44].

But the quality is not for free. The quality does cost. So the key is to focus on evaluating the cost of quality and return on quality [45].

This article deals with implementation of the Weibull mathematical model, KNN algorithm and Fuzzy logic in software component quality prediction in the Ericsson Nikola Tesla R&D within TSS (Trunk Signaling Subsystem) part of the AXE. It describes actual application of the discussed principles for the current running projects in Wireline. Some of the applications are new, and some are further development of the existing applications. The project duration is also different, some of the projects were running for 2 years, but some of the projects are only 6 months last. The importance of cost of quality is stressed, together with other factors that influence quality of a software product at the production side.

2. RELATED WORK

During the analysis of over 20,000 electronic products manufactured throughout 1980s and 1990s, [1] Weibull distribution with shape parameter β (beta) values close to 1, or very similar to the exponential distribution shown as the

most appropriate form of distribution for modeling in avionics.

Upadhya & Srinivasan [2] questioned the reliability of modeling with Weibull shape parameter β value of 1.1. Although it was concluded that a more accurate is exponential distribution, probability density function associated with advanced technologies in all areas of software engineering has shown that the Weibull distribution is actually the most representative distribution between the other distribution [3].

Research on the incidence of errors and their causes [4] showed that in fact software system has deterministic behavior and that the errors in the programming unit is function of the deterministic and not a stochastic process (although there are some indications, they are negligible). Weibull, Gamma and exponential distributions are used for modeling systems which have also proved successful.

In fact, common to all models, the reliability of the assumption that the occurrence of errors, and debugging to be a random process that can be presented as a probability density function or a realization of stochastic process [5].

Rinsaka and Dohi [6] pay a special attention to the maintenance phase of the product life cycle and quantity of people needed to maintain the same product because it is impossible to detect and remove all errors within the programming of products before delivery to the buyer. They used the Geometric, Binomial and Weibull distribution for modeling systems.

Zhou and Davis have been empirically confirmed the reliability of models for open source software [7]. Among other things, they confirmed that the occurrence of errors follows the Weibull distribution, and that does not follow the usual Rayleigh distribution, accepted in a non-FOSS software industry. Also, the authors suggested that each project has its own distinctive shape parameter β and the parameters can change over time and projects.

Yamada and Tamura confirmed that the interaction between application components should also consider in the modeling of open source software [8].

Billinton and Allan, 1983 confirmed that the Weibull distribution can be used for modeling the problems associated with aging, wear and deterioration of mechanical components [9]. In a series of surveys conducted in the area of rail transport industry and the accompanying industry, analysis of errors are made in order to find the best distribution model [10], [11], [12], [13]. The vast majority of observed cases, the Weibull distribution proved to be most successful [14].

Among the other things because the Weibull distribution has an important feature and that is that one does not advance a particular, typical form and that depending on the values of parameters can be closer to other distributions (e.g. Lognormal or exponential). For example, the shape parameter β has an impact on the frequency of errors observed components. Value for β less than 1 represents a phase of "infant mortality", while a value of 1 observed a consistent component of the error rate, and follows an exponential distribution. This phase is actually a normal phase of the operation components or components useful life. If the value is greater than 1 is the phase of wear components. The parameter η measures common characteristic component of life is defined as the moment at which the components have a 63.2% error [15].

Classical reliability models are divided into several classes and include Exponential class of models (which include the Jelinski-Morandi De-eutrophication model, Nonhomogeneous Poisson Process (NHPP) model, Schneidewind's Model, Musa's Basic Execution Time Model, Model Hyperexponential) then Weibull and Gamma Class models (which include the Weibull model, S-Shaped Reliability Growth Model), and Infinite Failure class model (which includes Duane's Model, Geometric Model), and finally Bayesian class of models with Musa-Okumoto logarithmic Poisson and Littlewood-Verrall Reliability Growth model are summarized in [16].

All of the above models imply that each error and defect is equally important.

Error prediction based on the complexity and size of software components written in the Java programming language has shown that in fact the size of monitored programming code are following lognormal distribution, distribution of errors in software components sorted by the number of lines of code follows Weibull distribution and the size of software components is important fact in understanding the magnitude and complexity of programming applications [17].

Benčić and Šestan 2002 have been successfully applied Weibull distribution with shape parameter $\beta = 1$ (special form Weibull distribution which represents exponential distribution) to model the ship's energy system with uniform frequency of errors [18]. Weibull distribution was used in the prediction error, ie the modeling of two IBM systems during phases of commercial use of the same [19].

System reliability in telecommunication industry is directly related to the amount of errors found during the system verification and reported errors with the specification for the system in operation. Number of reported errors is the most common way of measuring the quality of software components [20].

3. SOFTWARE COMPONENT

The presented model is dealing with the software components and measuring trouble report inflow in period of time. But this would be valid if it is observed on any other entity, such as just production/software unit. So the model is also applicable to the other parts of the system (group of software components, modules, subsystems etc.). Components are abstract, self-contained packages of

functionality performing a specific business function within a technology framework. These business components are reusable with well-defined interfaces. A business component is the software implementation of an autonomous business concept or business process. It consists of all the software artifacts necessary to represent, implement, and deploy a given business concept as an autonomous, reusable element of a larger distributed information system [26].

A software component is a unit of composition with contractually specified interfaces and explicit context dependencies only. A software component can be deployed independently and is subject to composition by third parties [27], [29].

A component represents a modular part of a system that encapsulates its contents and whose manifestation is replaceable within its environment. [...] A component specifies a formal contract of the services that it provides to its clients and those that it requires from other components or services in the system in terms of its provided and required interfaces [28].

A Software Component is a software element that conforms to a component model and can be independently deployed and composed without modification according to a composition standard [30]

A software component is a unit of composition with explicitly specified provided, required and configuration interfaces, plus quality attributes [31].

Software component is a logically, cohesive, loosely coupled module that denotes a single abstraction [32].

Reusable components are self-contained, clearly identifiable artifacts that describe or implement a specific function and that have clear interfaces in conformity with a given software architectural model, an appropriate documentation, and a defined degree of reuse [33].

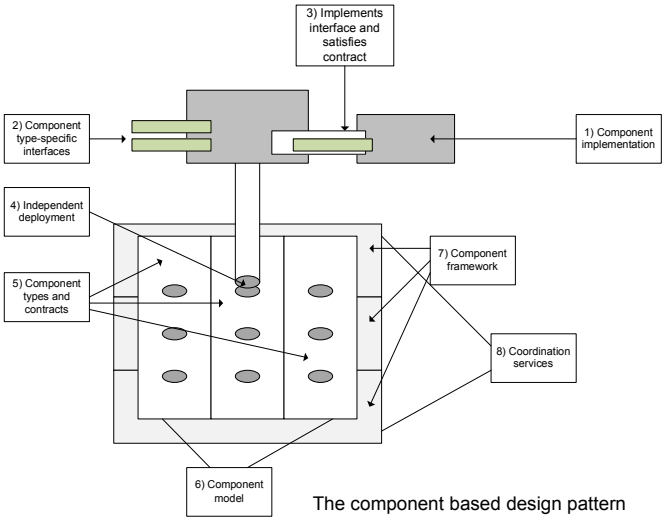

Figure 1. The Component-Based design pattern

Software component is defined as "a physical packaging of executable software with a well-defined and published interface [34].

Software component is a language neutral, independently implemented package of software services, delivered in an encapsulated and replaceable container accessed via one or more published interfaces. While a component may have the ability to modify a database, it cannot be expected to maintain state information. A component is not platform constrained nor is it application bound [35].

Note that the numbers in the brackets and following explanations are related to the figure 1:

A component (1) is a software implementation that can be executed on a physical or logical device. A component implements one or more interfaces that are imposed upon it (2). This reflects that the component satisfies certain obligations, which we will later describe as a con-tract (3). These contractual obligations ensure that independently developed components obey certain rules so that components interact (or can not interact) in predictable ways, and can be deployed into standard build-time and run-time environments (4). A component-based system is based upon a small number of distinct component types, each of which plays a specialized role in a system (5) and is described by an interface (2). A component model (6) is the set of component types, their interfaces, and, additionally, a specification of the allowable patterns of interaction among component types. A component framework (7) pro-vides a variety of runtime services (8) to support and enforce the component model. In many respects component frameworks are like special-purpose operating systems, although they operate at much higher levels of abstraction [36].

4. WEIBULL DISTRIBUTION

The Weibull distribution is by far the world's most popular statistical model for life data. Weibull distribution is also used in many other applications, such as weather forecasting and fitting data of all kinds.

Among all statistical techniques it may be employed for engineering analysis with smaller sample sizes than any other method. The Weibull distribution was first published in 1939, over 60 years ago and has proven to be invaluable for life data analysis in aerospace, automotive, electric power, nuclear power, medical, dental, electronics, and every industry.

In this paper 3-parameters Weibull distribution is used. Weibull distribution parameters related to this paper are Weibull parameter η = scale parameter, β = shape parameter (or slope) and γ = location parameter. Detailed explanation about Weibull distribution and parameters can be found in [46]. Correlation coefficient (cc) is used for evaluation of proposed model [47].

5. KNN CLASSIFICATION AND FUZZY LOGIC

K-nearest-neighbor (KNN) classification is one of the most fundamental and simple classification methods and should be one of the first choices for a classification study when there is little or no prior knowledge about the distribution of the data [37].

KNN classification was developed from the need to perform discriminant analysis when reliable parametric estimates of probability densities are unknown or difficult to determine. Fix and Hodges, 1951 introduced a non-parametric method for pattern classification that has since become known the KNN rule [38].

Later in 1967, some of the formal properties of the KNN rule were worked out by Cover & Hart [22], [23], Teknomo research covers different aspects in prediction techniques and KNN implementation, [21], [24], [25].

KNN algorithm is a controlled study where the results of the query are classified based on the closest results for each category. The purpose of the algorithm is similar to the neural-fuzzy network based prediction on the teaching and learning parameters needed for predicting response to each new issue or set value. KNN algorithm as such, does not use a separate model, but the results based on previous "Learning." KNN classification algorithm uses the neighborhood as a result of a query for the new desired value.

The concept of fuzzy set was published in 1965 by Lotfi A. Zadeh [39]. Since that seminal publication, the fuzzy set theory is widely studied and extended. Its application to the control theory became successful and revolutionary especially in seventies, and eighties, the applications to data analysis, artificial intelligence, and computational intelligence are intensively developed, especially, since nineties. The theory is also extended and generalized by means of the theories of triangular norms and conorms, and aggregation operator's [40].

Humans have a remarkable capability to reason and make decisions in an environment of uncertainty, imprecision, incompleteness of information, and partiality of knowledge, truth and class membership. The principal objective of fuzzy logic is formalization/mechanization of this capability [41].

In this paper for prediction of characteristic Weibull distribution parameters (shape parameter β, scale parameter η) fuzzy logic is used. For the TR prediction KNN classification algorithm is used.

6. CASE STUDY

Wln3.1 application and SIP-I protocol complex which includes 31 monitored software components were observed.

Software components can be measured and compared using the number of faults (Trouble Reports) received on them. Large data base is needed to collect and store different kind of faults received in different projects, applications or live exchanges all over the world for Ericsson projects or exchanges. Different types of data are collected for every single applications and every single software component.

MHWeb is an integrated system on the web with components which are used in Design, Design Maintenance (DM) and Support organizations for Maintenance and Customer Support purposes [48].

A trouble reports occurrence as a reported problem that requires developer intervention to correct is defined. This is the observable event of interest for both maintenance and insurance purposes. The operational definition of a trouble reports occurrence varies across organizations. In this paper, we use the approach to analyze fault occurrences in Ericsson Nikola Tesla R&D organizations using the term TR and the TR inflow over the weeks. The TRs are reports either from the customer, either from the internal verification team, either from ongoing development projects. It must be noted that Weibull model presented in this paper used total number of reported faults in time frame without taking care about TR answer code distribution.

Typical parameters for Weibull distribution are presented and evaluated for every single software component and application in general. It should be noted that among 31 observed software components are actually 9 different components and their enhanced versions, working in several applications over an 8 years time period. For example one software component are developed and deployed for several applications. On the other hand there are software component consists of only 2 applications within the monitored period. Of course, one of the key factors is the complexity of the software component based on the number of interactions with other software components and the functionality within the entire SIP-I complex through several applications and architecture. After comparing all 31 software components, individually and collectively, typical data for Wln3.1 application have obtained in Figure 2.

These are shape parameter β, scale parameter η, cc and TR number (in percentage due to sensitive information).

Individual contributions of each of the 31 program components that make up the overall distribution are presented on Figure 3.

Overall pdf presented in pink color is actually broken line because the probability of occurrence of errors is obtained by summing the individual probability density function and divided by the total number of software components in the application Wln3.1. It is combined distribution which combines these "individual" distributions into one final distribution that incorporates all the information:

$$Pdf(Wln3.1) = (Pdf(psk1) + Pdf(psk2) + ...Pdf(psk31))/31.$$

On the other hand, the group contribution of all software components on the overall distribution of probability density function is presented on Figure 4. It should be noted that the

areas below both curves on figures 3&4 are equal to 1 (it match probability density function for continuous variable and probability mass function for discrete variable).

The input data are Weibull parameters η and β for the software component in previous application (η(t-1) and β(t-1)), and Weibull parameters η and β for the software component for current application (η(t) and β(t)).

	β	η	cc	TR
SIPHIT_1	2,359302	18,87314602	0,971230474	7,9832%
SIPHIT_2	1,860543	14,05880833	0,989198576	5,1354%
SIPHIT_3	1,008976	16,4373531	0,96490406	5,6022%
SIPHIT_4	1,185758	52,48521778	0,985916073	6,6760%
SIPHIT_5	1,320833	64,52315206	0,964830919	2,5677%
SIPHOT_1	2,725801	27,50843545	0,922002357	7,3763%
SIPHOT_2	1,325121	12,497813	0,949440684	6,4426%
SIPHOT_3	1,245959	18,32599336	0,975528822	7,7498%
SIPHOT_4	1,088161	54,79901931	0,986913773	9,6172%
SIPHOT_5	1,094312	69,52712966	0,947116153	2,6144%
SIPD_1	1,46304	22,68728244	0,974433736	5,1354%
SIPD_2	1,388908	14,3379966	0,96025723	1,2605%
SIPD_3	0,658683	36,89459159	0,93387596	1,1671%
SIPD_4	1,077781	71,42383883	0,981857771	2,7077%
SIPME_1	1,275053	20,06192307	0,886037223	0,7003%
SIPME_2	1,356908	20,44628155	0,868470607	0,9804%
SIPME_3	1,088161	54,79901931	0,986913773	9,6172%
SIPMD_1	0,79958	20,32294658	0,940739128	0,2801%
SIPMD_2	1,034482	27,19744748	0,977205325	0,7003%
SIPMD_3	0,657952	27,95098697	0,955567917	0,6536%
SIPIS_1	1,032757	10,07876461	0,957054235	0,3268%
SIPIS_2	1,13237	7,648615679	0,841446452	0,2801%
TPCOM_1	1,23961	17,78468755	0,887592694	0,4669%
TPCOM_2	0,918585	3,186861654	1	0,0934%
TPCC_1	1,497001	22,79559338	0,97671747	5,1821%
TPCC_2	1,692704	18,51353066	0,949079898	1,4472%
TPCC_3	1,155621	13,80206028	0,978492528	1,6340%
TPCC_4	0,719149	50,86989538	0,959003758	2,3343%
TPCCR_1	1,52287	21,01525497	0,839086312	1,0738%
TPCCR_2	1,510709	16,30247739	0,98256086	1,6807%
TPCCR_3	1,158269	14,48895766	0,942181067	0,5135%

Figure 2. Typical parameters for Weibull distribution

Our task is to set the algorithm to automatically determine the Weibull parameters η and β for the software component that will be developed (η(t+1) and β(t+1)) based on linguistic descriptions set. As a result we will express the Weibull model for next software component and will predict the number of faults.

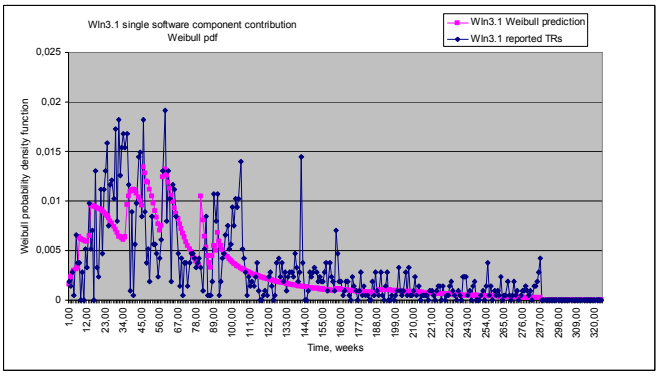

Figure 3. Individual software component contribution in application

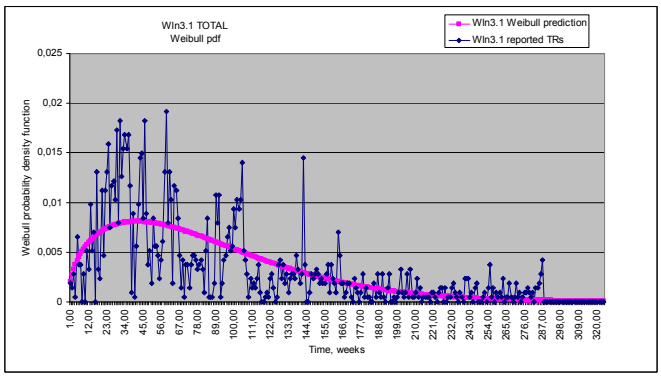

Figure 4. Software components contribution in total for application

The causal part of the conditional sentence has two variables that most often are called the variable situations. In our example this is the value of Weibull parameters η and β in time t-1 and the value of Weibull parameters η and β in t time. Variable in the subsequent part of the rules is called a variable action, and it is in our case the value of Weibull parameters η and β in t+1 time.

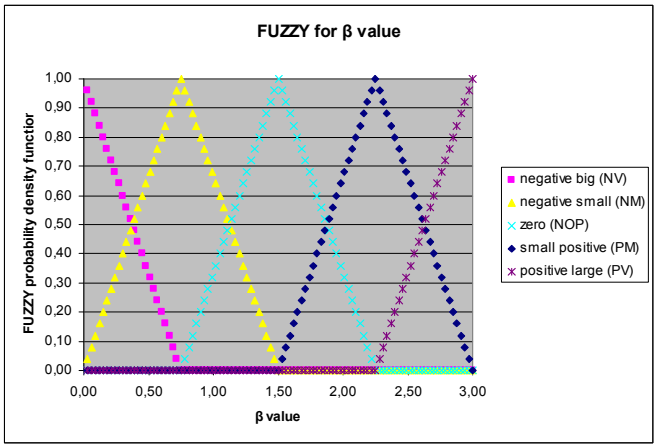

Figure 5. Fuzzy sets for β value

In parallel with setting the rules of conduct is necessary to define the underlying fuzzy sets that give the meaning of linguistic values of variables of the situation and actions (elements of the set of basic terms). In our example, we restrict the set of basic terms of the size of the 5 primary dates, as follows: positive large (PV), small positive (PM), zero (NOP), negative small (NM), negative big (NV).

Underlying fuzzy sets that give the meaning of these terms is defined Figure 5.

Control algorithm, (or rules of conduct) is a set of 25 rules presented in Table 1.

If the β(t-1) is PV and if the β(t) is PV, then the β(t+1) is PV also. If the β(t-1) is NV and if the β(t) is NM, then the β(t+1) is NM also.

Since the value for parameter β is never above 4, two fuzzy controllers were used.

		t=-1	t=-1	t=-1	t=-1	t=-1
		NV	**NM**	**N0P**	**PM**	**PV**
t=0	**NV**	NV	NV	NV	NV	NV
t=0	**NM**	NM	NM	NM	NV	NV
t=0	**N0P**	N0P	N0P	N0P	N0P	N0P
t=0	**PM**	PV	PV	PM	PM	PM
t=0	**PV**	PV	PV	PV	PV	PV

Table 1. CONTROL ALGORITHM

In first one from 0 up to 3 and in second one from 0 up to 4. Results are presented on Figure 6.

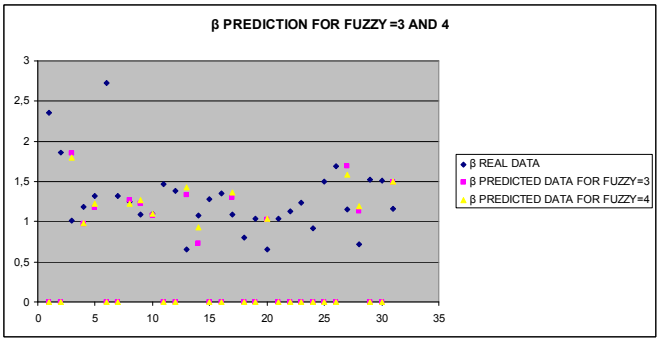

Figure 6. Beta prediction for FUZZY 3&4

Since the value for parameter η is never above 100, two fuzzy controllers were used. In first one from 0 up to 80 and in second one from 0 up to 100. Results are presented on Figure 7.

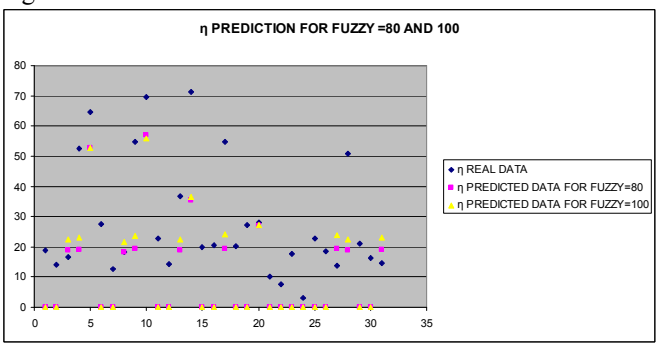

Figure 7. Beta prediction for FUZZY 80&100

The results presented on figure 6 shows that the better accuracy is gained for fuzzy logic controller 3 then 4. Also, the results from the figure 7 shows that the better accuracy is gained for fuzzy logic controller 80. Better results are obtained with parameter K= 2. The results can be used further in prediction and estimation software component trouble reports in any lifecycle stages, including design maintenance phase.

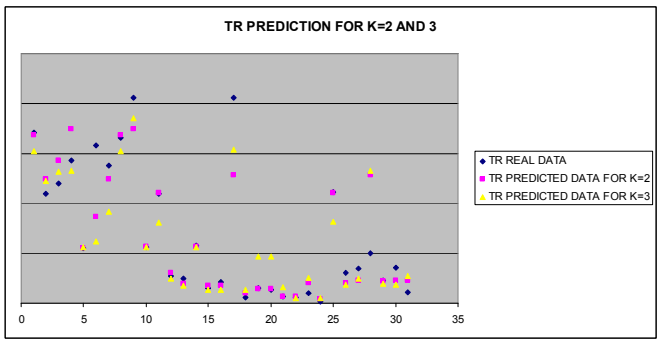

Figure 8. KNN prediction for TR number for 31 software components

Monitored software components developed in the same environment, the same project processes and methods, but each has a different number of interactions with other software components and the functionality within the entire SIP-I complex were used for the learning. Results are presented on Figure 8.

This data are obtained on software components from the same system and are compared with the real data for Weibull shape parameter β, scale parameter η and TR number.

7. CONCLUSION AND FUTURE WORK

Usage of KNN algorithm and fuzzy logic, together with Weibull model represent a good foundation for the future work in software quality. Weibull parameters and faults prediction accuracy in the software components are above the expectation. Future studies should expand research on more software components developed in the same or similar architectural environment, same project processes and methods. It should be checked too how these software components are evolved in the next releases/projects and prediction accuracy should be checked too. The benefits of application of this model should be also evaluated over other existing models (time, cost and number of the faults in the system).

Also, the future research should cover other mathematical models such as Lognormal, Normal and Exponential which are proven to be good for the modeling of software faults. Fuzzy logic controller and KNN algorithm should be checked too if there are spaces for more improvements in prediction.

8. REFERENCES

[1] J. Qin, B. Huang, J. Walter, J. Bernstein, and M. Talmor, "Reliability analysis of avionics in the commercial aerospace industry," *Journal of the Reliability Analysis Center*, First Quarter 2005.

[2] K. S. Upadhya and N. K. Srinivasan, "Availability of weapon systems with multiple failures and logistic delays," International Journal of Quality & Reliability Management, vol. 20, no. 7, pp. 836-846, 2003.

[3] L. Condra, "Integrated aerospace parts acquisition strategy," Technical committee GEL/107. Process management for Avionics, BSI Chiswick; October 7, 2002.

[4] Scott Dick, Cindy Bethel, Abraham Kandel, „Are Software Failures Chaotic?" 0-7803-7461-41021$17.00 0 IEEE, 2002

[5] A. L. Goel, "Software reliability models: assumptions, limitations, and applicability," IEEE Transactions on Software Engineering, vol. 11 no. 12, Dec. 1985, pp. 141 1- 1423

[6] Rinsaka, K. and Dohi, T. (2004), Optimal Testing/Maintenance Design in a Software Development Project. *CD-ROM Proceedings of Fourth International Conference on Mathematical Methods in Reliability - Methodology and Practice*, Santa Fe, New Mexico, USA.

[7] Zhou, Y. and Davis, J., Open source software reliability model: an empirical approach. Proceedings of the fifth workshop on open source software engineering, New York, NY, USA, 2005.

[8] Tamura, Y. and Yamada, S., Comparison of software reliability assessment methods for open source software. Proceedings of the 11th international conference on parallel and distributed systems – workshops, Washington, DC, USA, 2005.

[9] Billinton, R. and Allan, R.N. (1983). Reliability Evaluation of Engineering Systems: Concepts and Techniques. Pitman Books Limited, Boston.

[10] Kumar, S. (2006). *A Study of the Rail Degradation Process to Predict Rail Breaks.* Licentiate Thesis. Division of Operation and Maintenance Engineering, Luleĺ University of Technology, Luleĺ, Sweden, ISSN 1402-1757; 2006:73.

[11] Kumar, S., Espling, U. and Kumar, U. (2007). A holistic procedure for rail maintenance in Sweden. Accepted for publication in Journal of Rail and Rapid Transit: Proceedings of the Institution of Mechanical *Engineers, Part F.*

[12] Chattopadhyay, G. and Kumar, S. (2008). Parameter estimation for rail degradation model. *Accepted for publication in International Journal of Performability Engineering.*

[13] Kumar, S., Gupta, S. and Ghodrati, B. (2007). Rail defect prioritization and risk assessment using a hybrid approach. *Submitted to an International Journal.*

[14] Saurabh Kumar, Doctoral Thesis, Luleå University of Technology, Division of Operation and Maintenance Engineering 2008, „Reliability Analysis and Cost Modeling of Degrading Systems"

[15] Abernethy, R.B., *The New Weibull Handbook,* 4th edition, 2003ISBN 0- 9653062-1-6.

[16] Lyu, Jean-Claude Laprie and Karama Kanoun, in Michael Lyu, ed., Handbook of Software Reliability Engineering, IEEE Computer Society Press, McGraw Hill, 1996, 27-69.

[17] H. Zhang, X. Zhang, M. Gu, Predicting Defective Software Components from Code Complexity Measures, *Proc. 13th IEEE Pacific Rim Symp. on Dependable Computing (PRDC 2007)*, Australia, 2007.

[18] R. Benčić, A. Šestan, Calculation Methodology and Models for the Computer Determination of Marine Power Plants RAM Characteristics, UDC 629.5.03-192:004.332.3, 2002

[19] Garrison Kenny. Estimating Defects in Commercial Software during Operational Use. *IEEE Tr. on Reliability,* vol. 42 no. 1, Mar 1993, pp. 107-115.

[20] Xu, Z. and M. Khoshgoftaar (2001). "Software quality prediction for high-assurance network telecommunications systems." *The Computer Journal [H.W. Wilson - AST]* 44(6): 557.

[21] K. Teknomo, „KNN for Smoothing and Prediction", http://people.revoledu.com/kardi/tutorial/KNN/KNN_Numerical-example.html

[22] T.M. Cover and P.E. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, 1967.

[23] T. M. Cover and P. E. Hart, "Nearest neighbor pattern classification," *IEEE Trans. Inform. Theory*, vol. IT-13, pp. 21–27, Jan 1968.

[24] Teknomo, K. and Millonig, A., "A Navigation Algorithm for Pedestrian Simulation in Dynamic Environments", Proceeding of the 11th World Conference on Transport Research (WCTR), University of California, Berkeley, USA, June 24-28, 2007

[25] Teknomo, Kardi, "Application of microscopic pedestrian simulation model", Transportation Research Part F: Psychology and Behaviour Vol 9 issues 1, January 2006, p. 15-27.

[26] Herzum, P., Sims, O.: Business Component Factory: A Comprehensive Overview of Component-Based Development for the Enterprise. John Wiley & Sons, New York (2000)

[27] Szyperski, C., Gruntz, D., Murer, S.: Component Software. Beyond Object-Oriented Programming, 2nd edn. Addison-Wesley, Harlow (2002)

[28] Object Management Group (OMG): Unified Modeling Language Specification: Version 2, Revised Final Adopted Specification, ptc/05-07-04 (2005)

[29] Broy,M.: Towards a Mathematical Concept of a Component and its Use. Software – Concepts and Tools 18, 137–159 (1997)

[30] Heineman, G., Council, W.: Component-Based Software Engineering: Putting the pieces together. Addison-Wesley, Reading (2001)

[31] Component+. Built-in testing for component-based development. Technical report, Component+ Project, http://www.component-plus.org, 2001.

[32] G. Booch. Software Components with Ada: Structures, Tools and Subsystems. 1987.

[33] Villela, R. M. B.: Components searching and retrieving in software reuse environments, COPPE/Federal University of Rio de Janeiro. Doctoral Thesis, Rio de Janeiro, Brazil. (2000)

[34] Hopkins, J.: Component primer. Comm. of the ACM 43 (2000) 27–30

[35] Sparling M.: "Lessons learned through six years of Component Based Development", Communications of the ACM, 2002.

[36] F. Bachmann, L. Bass, Ch. Buhman, S. Comella-Dorda, F. Long, J. Robert, R. Seacord y K. Wallnau. Volume II: Technical concepts of component-based software engineering, 2nd edition. Technical report, Software Engineering Institute, Carnegie Mellon University, May 2000.

[37] Leif E. Peterson: "K-nearest neighbor", Scholarpedia, 4(2):1883.(2009), http://www.scholarpedia.org/article/K-nearest_neighbor

[38] Fix, E., Hodges, J.L. Discriminatory analysis, nonparametric discrimination: Consistency properties. Technical Report 4, USAF School of Aviation Medicine, Randolph Field, Texas, 1951.

[39] L.A. Zadeh: Fuzzy sets. Information and Control 8 (3) 338--353. (1965)

[40] M. Mares: "Fuzzy sets", Scholarpedia, 1(10):2031., (2006)

[41] Lotfi A. Zadeh:"Fuzzy logic", Scholarpedia, 3(3):1766. (2008)

[42] ISO/IEC 9126-1:2001 --- Software engineering -- Product quality -- Part 1: Quality model, 2001.

[43] Lovre Hribar, "Usage of Weibull and other models for software faults prediction in AXE", 2008. International Conference on Software, Telecommunications & Computer Networks, SoftCOM 2008

[44] Lovre Hribar, Sanja Bogovac, Zdenko Marinčić, "Implementation of Fault Slip Through in Design Phase of the Project", 31. International Conference, MIPRO 2008.

[45] Snadra A., Slaughter Donald E., Harter and Mayuram S. Krishnan, "Evaluating the cost of Software Quality", Communication of ACM, August 1998/Vol. 41, No.8.

[46] Lovre Hribar, "Software Component Quality Prediction in the Legacy Product Development Environment Using Weibull and Other Mathematical Distributions", 2009. International Conference on Software, Telecommunications & Computer Networks, SoftCOM 2009

[47] Edgeman, R.L., "Correlation Coefficient Assessment of NPPs", Reliability Review, Vol. 10, (14-16), 1990.

[48] http://mhweb.ericsson.se/index2.html