CORE
Metadata, citation and similar papers at core.ac.uk
Provided by International Journal on Advanced Science, Engineering and Information Technology

International Journal on Advanced Science Engineering Information Technology

Vol.10 (2020) No. 3
ISSN: 2088-5334

The Development and Evaluation of Experience-Based Factory Model for Software Development Process

Mastura Hanafiah[a,1], Rusli Abdullah[a,2], Masrah Azrifah Azmi Murad[a,3], Jamilah Din[a,4]

[a] *Faculty of Computer Science and Information Technology, Universiti Putra Malaysia, 43400 Malaysia*
E-mail: [1]mastura.hanafiah@daimler.com; [2]rusli@upm.edu.my; [3]masrah@upm.edu.my; [4]jamilahd@upm.edu.my

***Abstract*— Knowledge, and experiences in software development have been accumulated over time throughout the project lifecycle. Previous studies have shown that the management of knowledge and experiences in software development has always been an issue. Therefore, the knowledge transfer and information flow are inefficient, misinterpretation, and inconsistencies always occur between individuals or teams, and the organization fails to learn from past projects. It is understood that efficient knowledge and experience management for software development organizations is crucial for the purpose of sharing and future reuse. This paper discusses the prototype development for a proposed model, which is based on the experience factory approach, to manage knowledge and experiences for the software development process. Discussions include the system functionalities and design, infrastructure requirements, and implementation approach. The efficiency and effectiveness of the prototype are evaluated as survey research based on Jennex & Olfman knowledge management success model. Rasch analysis is used for data reliability and validity. Results show positive feedback on the model's efficiency and effectiveness. Additionally, as agreed by most respondents, the top three of the model contributions are: to encourage learning organization, to prevent knowledge loss and to aid in decision making.**

***Keywords*—experience factory; knowledge management; software development process; prototype evaluation.**

I. INTRODUCTION

In software development (SD), there are many stages of events and activities take place typically; some of them are iterative in nature. Throughout the development lifecycle, a lot of methods, techniques, and tools are used. The knowledge and experiences gained during development have become important assets in software organizations. In the previous work, via a systematic literature review [1], it has been identified that there are issues in knowledge management for SD, and it is even more challenging for distributed teams. The main challenges identified are inefficient knowledge transfer and information flow [2], [3], misinterpretation, and inconsistencies [4], [5], and additionally, organizations fail to learn [6]. Due to the importance of knowledge retention and reuse, it is therefore essential to facilitate knowledge management for software process improvements in software organization [2], [7].

Knowledge management (KM) for software development has emerged since the late 1990s, and enormous studies have emerged since then. The interactions between tacit and explicit knowledge allow the conversion of data and information to become useful knowledge and experiences that can be used as a future reference [8].

In general, organizational KM has continuously strived for process improvement. Experience Factory (EF) is a software process improvement framework based on reuse of products, processes, and experiences originating from the system lifecycle [9]. EF focuses on two distinctive organization (Fig. 1):

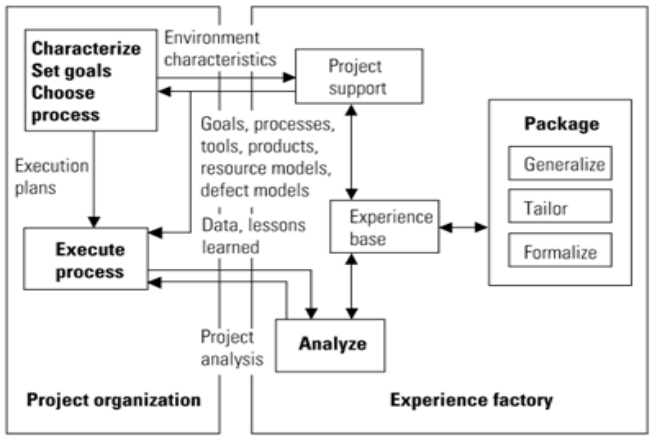

Fig. 1 The Experience Factory [9]

Project Organization provides the experience factory with project data, processes, and models used in the development, and uses packaged experience to deliver software products;

and Experience Factory: transforms those data and information into reusable units and supplies them back to the project organization.

EF employs the concept of Quality Improvement Paradigm (QIP) to improve learning in software quality and process; it includes some aspects as follows:

- Characterize and understand
- Set goals
- Choose process/methods
- Execute
- Analyze
- Package

With a successful implementation of QIP, a growing number of packaged experiences can be created by storing, analyzing, and transforming them into best practices [10].

II. Materials and Method

A. Conceptual Model

Prior to this study, an experience-based factory model for the SD process (EBF-SD) has been proposed [3]. The model consists of two main organizations: Project Organization (PR_ORG) and EF Organization (EF_ORG). PR_ORG consists of the necessary data, information, knowledge, and experiences from the Community of Practice (CoP) and Software Development Process (SDP). EF_ORG consists of the technology and infrastructure needed (TECH) components, along with the processes required within Knowledge Management (KM). EF_ORG is responsible for analyzing and processing data, information, knowledge, and experiences. And transform them into reusable packages, and later send them back to the project organization. Fig. 2 illustrates the proposed model.

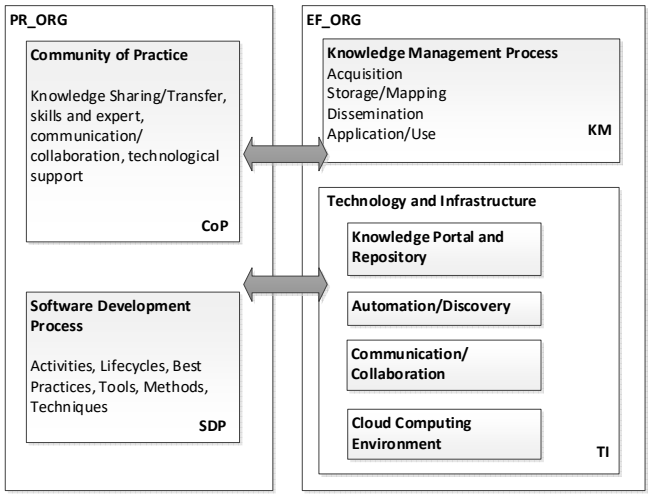

Fig. 2 The EBF-SD Model

Many collaborative KM solutions in SD have been introduced thus far. Some examples include ontology-based solution [3], [4], multi-agent-based [11], [12], and semantic web [13], [14]. It is found that the number of EF based approach for SD process are still insignificant [1]. One of the studies that adopted EF concept was the study by Ivarsson and Gorschek [2] who proposed Practice Selection Framework (PSF) to support utilizing postmortems for organizational improvements. There was a practice repository which consisted of practices and experiences, and a process manager. The practice repository evaluated the state of the practices and decided if improvements were needed. Those practices could then be used and reused in projects. There was a clear separation between the project organization and experience factory organization. During the evaluation, the effort of using PSF was relatively high, and the prescribed practices were not as usable; besides, there were also challenges of using PSF, i.e., the need to accommodate dedicated resources for documentation, trust, legacy, and measurement.

Another study that utilized the EF framework was the Experience Base Model (EBM) by Sharma et al. [15]. In this study, experience about software engineering items or objects (technique/tool/method) is captured and managed up-to-date with identified representation schemas. While the model could benefit software organization in terms of software engineering terminology and concepts by storing them in dedicated repositories, there was no tool proposed on how these schemas can be shared, transferred or reused within the software engineering community.

The study by Ardimento et al. [16] proposed the structure of the Knowledge Experience Base (KEB). The structure of the knowledge is derived from a centralized knowledge content in which it consists of Tool, Evidence, Competence, and Projects. Prometheus supported it, a tool to capture, share, and retain content and ensure automation and management of the content. Users may access any of the component structure and navigate to all the components for their needs. An empirical investigation was done with an experiment to test the model against productivity, which is calculated based on function points and effort. The study showed that productivity for those using the tool was higher compared to those without the tool. Therefore, the reuse grew with the tool.

Generally, these studies do emphasize more on explicit knowledge and storing them in dedicated repositories, but they have less emphasis on tacit knowledge gained during the software development activities. Additionally, they are lack of automation feature for knowledge dissemination appropriate audiences.

Some KM systems that are built as information systems have been evaluated by using DeLone and McLean (D&M) model [17]. D&M is composed of six distinct measures: *System Quality*, *Information Quality*, *Use*, *User Satisfaction*, *Individual Impact*, and *Organizational Impact*. Jennex and Olfman (J&O) KM success model [18] is adapted from D&M model but is tailored towards the KM context. The measures in J&O model are similar to those of D&M, which include *System Quality (SQ)*, *Knowledge Quality (KQ)*, *Service Quality (SVQ)*, *Intent to Use/Perceived Benefit (IN)*, *User Satisfaction (US)* and *Net Benefit (NB)*.

Service Quality deals with how well KM assists the user in capturing, finding, retrieving, manipulating, and using knowledge. The *Knowledge Quality* measures on the usefulness and accuracy of the content and the ability to assist users in their activities The *Service Quality* evaluates the organization's ability to provide the KM system (KMS) and ensure it provides the benefits expected from the knowledge users The *User Satisfaction* measures user satisfaction of using KMS and its knowledge The *Intent To*

Use/Perceived Benefit assists in determining if the KMS is sufficient to ensure that users will use KMS when appropriate and the *Net Benefit* measures the actual benefits derived from using knowledge/KMS, i.e., impacts to business processes, KM strategy, knowledge content, and leadership/management support.

However, J&O's *Net Benefit* is more difficult to measure, by using merely a prototype, without having hands-on experience or the actual user experience in the real world. Therefore, we consider Halawi's KMS Success model [19] as the *Net Benefit* and adopt the questions by Nattapol et al. [20] as the key elements of this construct to reflect the benefits that can be perceived and judged immediately. The proposed perceived benefits items include acquiring new knowledge and innovative ideas, effectively managing and storing needed knowledge, accomplishing tasks more efficiently, improving the decision making, and improving the quality of work-life [19].

B. Development Methodology

The research has undergone several stages in which qualitative and quantitative methods are involved. In the initial stage, the conceptual model has been formulated based on the review of the literature. The model is then reviewed by the relevant experts and followed by survey research to gauge the opinion model from the software development community about components that constitute the model. Publications on these studies can be referred to in [21] and [22]. The research is further continued with prototype development and prototype evaluation. This paper discusses the results of a preliminary evaluation of the prototype. Fig. 3 illustrates the prototyping approach, as introduced in [23], used in model development.

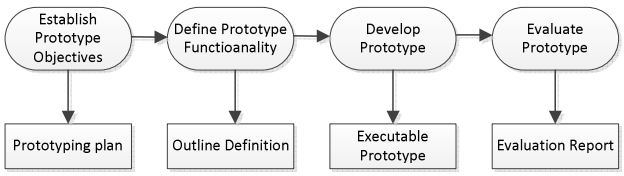

Fig. 3 The Development Methodology

The prototype objective is to outline the main functionalities and to visualize the user interface. The prototyping functionalities are defined by using scenarios and use cases. This will comprehend the whole workflow and the interaction between the system and the users. Use cases and scenarios are effective techniques in requirement elicitation as they can identify the actors and its interaction with the system [23]. Use cases are visualized by using use case diagrams of Unified Modelling Language (UML), a de facto standard for object-oriented modeling. The next stage is to develop the prototype. During the developing prototype stage, the identified software process models are formally structured and defined as ontologies. The end product should also be enhanced with multi-agent systems to assist the necessary automation features.

For the prototype evaluation, the system was demonstrated to software practitioners randomly selected from identified software companies, and they were then asked to answer the survey questions related to KM success model discussed earlier. Survey questionnaires were provided as a paper-based and online questionnaire (surveymonkey.com and docs.google.com). Threats to validity may occur throughout the research process and can cause an error in measurement due to several factors such as instability of measurement instrument or response bias. In this research, we use the Rasch measurement model [24] as a method to evaluate construct validity. Construct validation is an evaluation of a measurement instrument for its validity and reliability [25]. Validity refers to how well a construct measures what it is supposed to measure. The reliability refers to the consistency of the scores when similar tests is performed with a more extensive set. Validity and reliability of the collected data are further examined for summary statistics, dimensionality, person, and item fit criteria with Rasch fit indicators. Rasch fit statistics can be analyzed by the acceptable range values as in Table I.

TABLE I
THE ACCEPTABLE RANGE OF RASCH MEASUREMENT MODEL

Person/Item	**Acceptable range**
Point measure correlation (PTMEA CORR)	$0.4 < x < 0.8$ [26]
Infit/Outfit means square (MNSQ)	0.6 - 1.4 [27]
Infit/Outfit Z-Standardized value	$-2 < x < 2$ [26]

The data is further analyzed for the KM success constructs on KQ, SQ, SVQ, US, IN, and NB for their mean and standard deviation values. The perceived contribution of the model is also analyzed. At the end of the survey session, participants were given five options on the possible model contribution, i.e., encourage learning organization, reduce development cost, improve software quality, prevent knowledge loss, and aid in decision making. They were asked to select three options that they thought the model would contribute the most.

III. RESULT AND DISCUSSION

A. The Prototype Development

The development of the prototype begins with the analysis of the functionalities, use case identification, and workflow design. KM process includes knowledge acquisition, knowledge storage, knowledge dissemination, and knowledge application, and these are the main functionalities of the prototype that will be implemented.

Fig. 4 below depicts the use case diagram of the system. Identified actors are as follows:

- Project Owner
- Project Contributor
- Public User
- Notification Agent
- Recommendation Agent

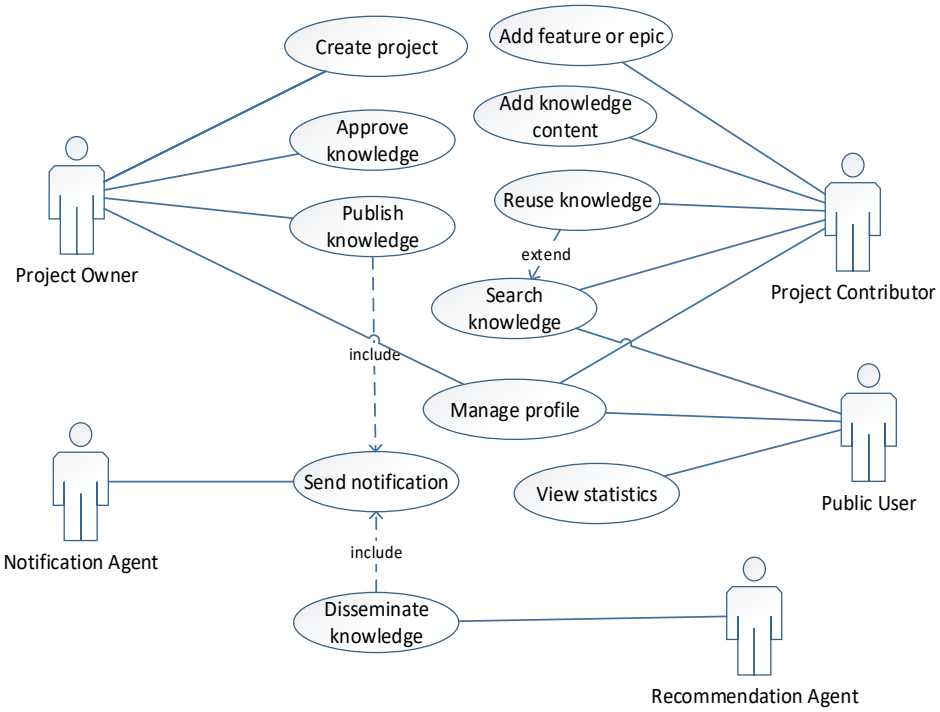

Fig. 4 The Use Case Diagram

The project owner is responsible for creating the project, approve the knowledge entries, and publish the knowledge. The Project Contributor can add project features or epic, add knowledge entries, search and reuse knowledge; and the Public user may also search and reuse knowledge where applicable. All the reused knowledge would be cloned to the users' project, and this reused knowledge require approvals from the project owner. Notification Agent is responsible for sending relevant notifications whenever there are new knowledge entries or when a project has adequate knowledge entries for publishing. Meanwhile, Recommendation agent functions to find proper knowledge and disseminate the knowledge according to user profiles. Scrum, an agile approach, and the waterfall model, a traditional approach, are used in the prototype to demonstrate the software process model classification and structure. Such a structure is defined and classified by using an ontology, based on the ontology for software engineering [28].

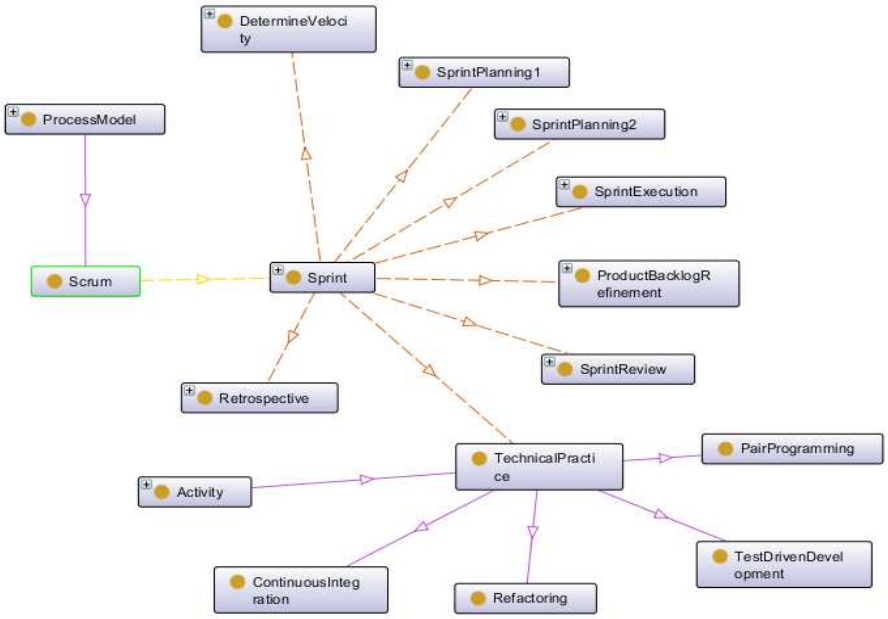

Fig. 5 Part of Scrum Ontology

Fig. 5 shows part of the ontology definition for Scrum. Scrum is planned by sprints, which is a time-boxed (normally 2 to 4 weeks) for one iteration [29]. A sprint contains several stages, i.e., Sprint Planning 1 and 2, Sprint Review, and Retrospective. Several other topics relevant in agile development are also included, such as technical practices, agile analysis and design, and such. The ontology design serves as a well-defined structure for the Scrum process model and is represented as a tree-like structure in the user interface. In each leaf of the tree-view structure, the user may enter the relevant knowledge entries whereby the approval from the project owner is required before it can be published to the community.

For the implementation of multi-agent systems, we simulate them as time-based event programs running in the background (cronjobs) at a fixed time. Fig. 6 shows the system overview diagram of the multi-agent systems.

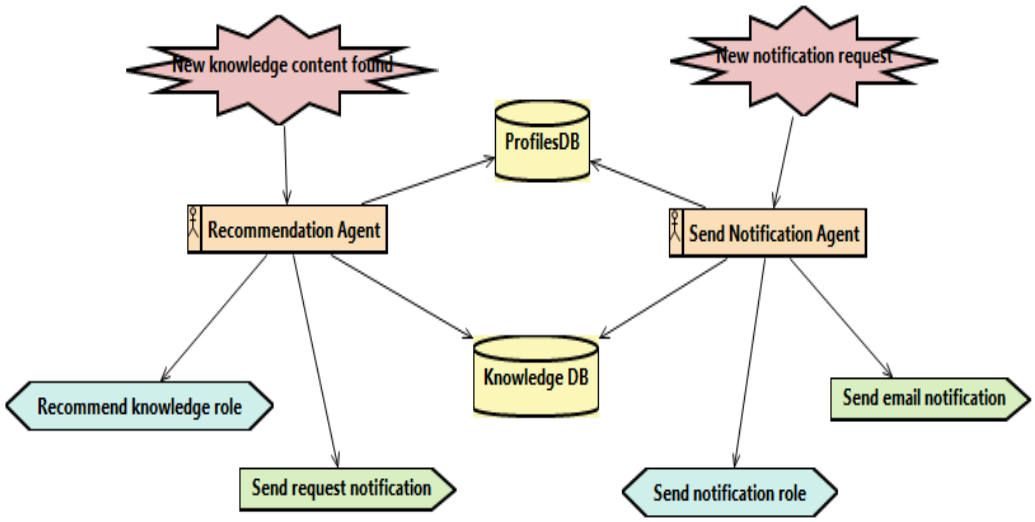

Fig. 6 MAS System Overview Diagram

The *Recommendation Agent* reads from the database about the available knowledge and the matched profiles, and then it sends notification requests to the notification agent. *Notification Agent* receives the request and sends alerts to the respective users. Notifications for this prototype will be shown in the user interface with the notification icon at the top right bar of the user interface. In future enhancement, email notifications will be used as well to increase awareness of the requests and available knowledge.

To implement the system, we have defined the necessary infrastructure requirements, which include the technical requirements on availability and reliability, storage requirement, automation of KM processes, security, and network and performance [30]. Cloud security features proposed by [31] could be a value-added feature as it offers data confidentiality, correctness, availability, and integrity of cloud data storage. For the prototype, we have chosen Amazon Web Services (AWS) Cloud [32] as the provider of the computing needs. Amazon Elastic Compute Cloud (EC2) provides the computing needs and allows the flexibility for server configuration and has the auto-scaling features to ensure high availability.

For the storage requirement, the model requires storage that should be flexible where the structure can be changed from time to time and able to support a massive amount of data. In this case, we choose MongoDB [33] as our 'nosql' data storage. Automation of the KM process is supported with customized workflow and agent-based programs. Security would be supported with authentication and authorization mechanisms, as well as leveraging the security features offered from AWS. In regards to network and performance, we can also rely on AWS elastic network adaptor in which it can run across multiple EC2 instances and ensure that all available network bandwidth are fully utilized. Table II summarized the framework and technologies used in the prototype.

Note that for the front-end and back-end implementation, Javascript-based programming languages are used in which codes are executed at the users' processors, thus saving bandwidth and strain at the webserver.

TABLE II
PROTOTYPE IMPLEMENTATION APPROACHES

Component	Framework	Reference
Front-end	Materialize	http://materializecss.com/
Back-end	Meteor	https://www.meteor.com/
Agent classes	Meteor	
Database	Mongo-DB	https://www.mongodb.com/
Application hosting	Amazon Elastic Compute Cloud	https://aws.amazon.com/ec2/?nc2=h_m1

Materialize components are used for a more customizable and responsive user interface, and they are more adaptable for mobile interface. Fig. 7 shows a sample screen for the user interface. The design is kept simple and consistent. Menu navigation is on the left, project lifecycle model structure, and the knowledge entries area are on the main page.

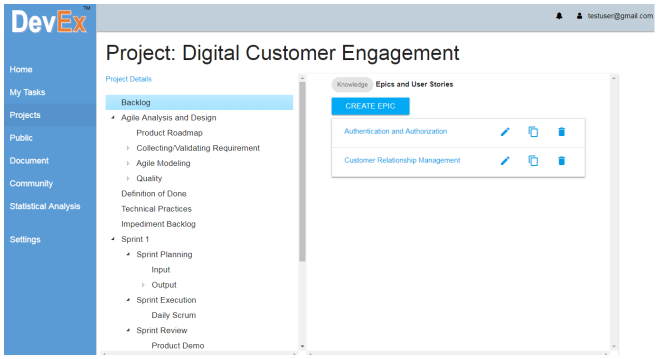

Fig. 7 Sample Screen

The prototype is called DevEx, a short form of the words Development and Experiences. The user interface should be designed in such a way that it can be easily adapted in mobile applications in the future. This prototype development, however, focuses more on web applications.

B. Prototype Evaluation

The evaluation of the model was conducted via a 20 minutes prototype demonstration followed by a survey with 36 questionnaire items. Likert-scale items were used with 1 representing 'strongly disagree' and 4 represented 'strongly agree.' The purpose of the survey was to evaluate the model prototype based on J&O model discussed earlier. The following constructs were evaluated: Knowledge Quality (KQ), System Quality (SQ), Service Quality (SVQ), User Satisfaction (US), Intent to Use/Perceived Benefit (IN), and Net Benefit (NB). The questionnaire items for SQ, SVQ, KQ, US, and IN were self-developed based on J&O model description, while for NB, the questionnaire items were adopted from Nattapol at el. [19]. The survey had collected 48 respondents from three different software companies in which the participants came with various roles, i.e., software engineers (21), project managers (6), system analysts (6), consultants (4), testers (2), architect (1), IT managers (3), others (5). The working experience of the respondents in software development ranges from 2 to 20 years.

For Rasch analysis, the Analysis tool WINSTEPS Version 3.68.2 is used. Rasch analysis can be carried out several times until a satisfactory result is achieved. In this case, we had run the Rasch analysis twice before a satisfactory result could be obtained. Table III shows the Rasch analysis on summary statistics, dimensionality and misfit items and persons for both runs.

During the 1st run, the raw data was used: 48 persons and 36 items. Results show that Cronbach's alpha, item reliability, and person, respectively. For unidimensionality testing, a measured variance of over 40%, the eigenvalue of less than 2.0, and the variance for the first contrast less than 5%. These are the criteria for unidimensionality [34]. Reliability is excellent with values 0.95, 0.85 and 0.95 In the 1st run, the measured variance is 43.6%, and the first contrast of unexplained variance is 9.6%, which is fair, but eigenvalue shows the strength of 6 items. A value higher than 2 could indicate the existence of a secondary dimension.

TABLE III
SUMMARY STATISTICS

		1st run	2nd run
Summary statistics	Cronbach's α	0.95	0.95
	Item reliability	0.85	0.87
	Item separation	2.34	2.56
	Person reliability	0.95	0.95
	Person separation	4.26	4.21
	Item measure	0.00	0.00
	Person measure	0.77	0.66
	Standard error item	0.14	0.17
	Standard error person	0.24	0.27
Dimensionality	Measured variance, %	43.60	45.40
	1st contrast unexplained variance, %	9.60	7.10
	Eigenvalue	6.10	4.30
Misfit	Items	Yes	No
	Person	Yes	No

Further investigation of misfitting persons and items reveals that some persons and items are outside the fit indicators. Negative or 'nearly zero' point measure correlations could indicate problematic items or persons [26]. For misfitting persons, it is found that seven persons have negative correlations, i.e., P13, P6, P33, P35, P40, P44 and P21 (Fig. 8), while misfitting items indicate three items with out-of-range Z-standardized, i.e., SQ4, KQ7, and KQ3 (Fig. 9). These misfitting persons and items are removed, and we are left with 41 persons and 33 items.

Rasch unidimensionality test is run again with this reduced dataset, and the summary statistics and dimensionality testing are as shown in Table III, at the column 2nd run. Person and items reliability remain excellent, while the measured variance, eigenvalue and unexplained variance of first contrast have improved with values 45.4%, 4.3 and 7.1% respectively. The plot of the standardized residual contrast however shows that the plots for items A-E are quite random vertically and are close with other items (Fig. 10); therefore, it can be concluded that there is no potentially secondary dimension.

PERSON STATISTICS: MISFIT ORDER

ENTRY NUMBER	TOTAL SCORE	COUNT	MEASURE	MODEL S.E.	INFIT MNSQ	INFIT ZSTD	OUTFIT MNSQ	OUTFIT ZSTD	PT-MEASURE CORR.	PT-MEASURE EXP.	EXACT MATCH OBS%	EXACT MATCH EXP%	PERSON
13	111	36	.43	.39	3.16	4.7	3.41	4.8	A .27	.38	30.6	79.6	P13
37	108	36	-.03	.40	1.90	2.3	2.03	2.4	B .31	.41	66.7	81.1	P37
20	137	36	3.80	.43	1.28	1.1	1.91	2.3	C .19	.18	83.3	80.5	P20
30	129	36	2.65	.35	1.73	4.4	1.76	4.2	D .26	.25	61.1	63.8	P30
34	107	36	-.19	.40	1.62	1.7	1.48	1.3	E .44	.42	69.4	81.0	P34
29	124	36	2.08	.34	1.56	3.5	1.52	3.1	F .17	.28	58.3	61.3	P29
46	108	36	-.03	.40	1.53	1.5	1.48	1.3	G .06	.41	77.8	81.1	P46
44	114	36	.86	.37	1.29	1.1	1.51	1.6	H .00	.35	77.8	75.6	P44
21	132	36	3.03	.36	1.27	1.0	1.44	2.1	I -.11	.23	75.0	68.7	P21
2	129	36	2.65	.35	1.41	2.7	1.40	2.3	J .31	.25	38.9	63.8	P2
31	128	36	2.54	.34	.86	-1.0	.84	-1.1	o .44	.26	63.9	62.4	P31
5	100	36	-1.25	.38	.81	-.7	.85	-.3	n .44	.45	80.6	74.4	P5
32	123	36	1.96	.34	.85	-1.0	.82	-1.2	m-.01	.29	66.7	61.8	P32
23	135	36	3.46	.40	.83	-.8	.77	-.9	l .32	.20	80.6	75.3	P23
41	107	36	-.19	.40	.80	-.5	.82	-.4	k .39	.42	80.6	81.0	P41
4	110	36	.28	.39	.28	-3.2	.27	-3.0	e .15	.39	94.4	80.4	P4
6	108	36	-.03	.40	.08	-5.1	.07	-4.8	d .61	.41	100.0	81.1	P6
33	108	36	-.03	.40	.08	-5.1	.07	-4.8	c .61	.41	100.0	81.1	P33
35	108	36	-.03	.40	.08	-5.1	.07	-4.8	b .61	.41	100.0	81.1	P35
40	108	36	-.03	.40	.08	-5.1	.07	-4.8	a .61	.41	100.0	81.1	P40
MEAN	114.1	35.9	.77	.38	.97	-.3	.98	-.2			76.6	74.2	
S.D.	12.6	.3	1.66	.02	.52	2.1	.57	2.1			13.4	6.7	

Fig. 8 Person Misfit

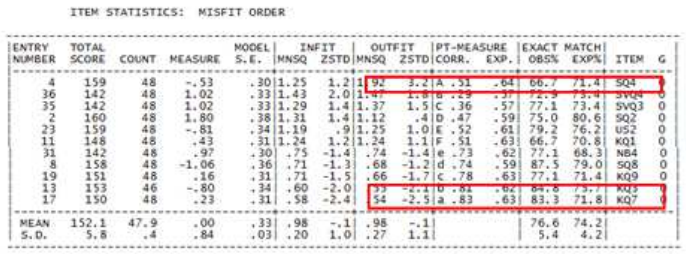

Fig. 9 Item Misfit

The analysis continues with the respondents' feedback on the KM success constructs. Cronbach's alpha is re-analyzed with IBM SPSS 22 to re-confirm the internal consistency. Cronbach's alpha indicates consistent value with that of the Rasch reliability test, with a value of 0.95, as shown in Table IV.

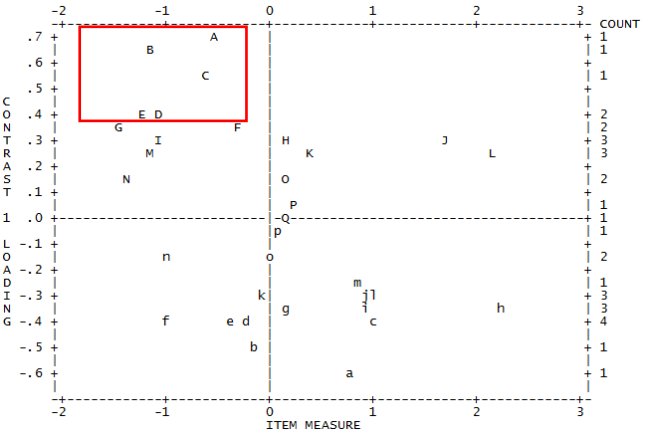

Fig. 10 Standardized residual contrast 1 plot

TABLE IV
CRONBACH'S ALPHA BASED ON STANDARDIZED ITEMS

Cronbach's alpha	N of Items
0.95	33

The categorized mean values and standard deviation for each construct are shown in Table V. Overall, the mean is between 2.9 and 3.3, indicating positive feedback on all items. Standard deviation is also very small between the values of 0.4 and 0.8 indicating that the data points tend to be close to the mean. Except for SVQ and NB, other constructs are mostly agreeable by most respondents.

TABLE V
DESCRIPTIVE STATISTICS FOR KM SUCCESS CONSTRUCTS

	N	Mean	Std. Deviation
SQ	41	3.2964	.43448
KQ	41	3.1626	.41803
SVQ	41	2.9878	.44353
US	41	3.2520	.42019
IN	41	3.1789	.42881
NB	41	3.0780	.46450
Valid N (listwise)	41		

Table 6 shows the mean and standard deviation values for each item. Overall, most of the respondents agree with the constructs for SQ, KQ, US, and IN (mean scores > 3). For NB, the scores are somewhat lower (< 3) for the items NB2, NB3, and NB4. Perhaps, for NB2 (*Accomplish tasks more efficiently*), NB3 (*Improve the quality of my work life*), and NB4 (*Improve in decision making*) are somehow difficult to endorse because the system needs longer time of usage before the perception on these items can be evaluated as compared to item NB1 (*Effectively manage and store required knowledge.)* which is easier to agree with as it can be measured promptly for agreeableness.

It is also observed that the items for Service Quality (SVQ) focus more on management support, KM strategy, and KM governance. These items are somehow hard to measure by employees as they might not able to predict what the management can do in terms of service quality. Therefore, these items result in reliably low scores.

TABLE VI
DESCRIPTIVE STATISTICS FOR ITEMS

	Items	N	Mean	Std. Dev.
SQ1	Easy to use	41	3.317	.6496
SQ2	Acceptable response time	41	3.341	.4801
SQ3	Accessible anytime, anywhere	41	3.415	.5466
SQ5	Appropriate interface	41	3.341	.5296
SQ6	Adequate structure organization	41	3.293	.6420
SQ7	Integrated KM process	41	3.171	.7383
SQ8	Search function	41	3.293	.5120
SQ9	Navigation tree	40	3.275	.4522
SQ10	Sorting feature	41	3.220	.6129
KQ1	Knowledge classification understandable	41	3.073	.6477
KQ2	Capture and store knowledge adequate/reliable	41	3.098	.6247
KQ4	Appropriate knowledge content	41	3.195	.6008
KQ5	Useful knowledge content	41	3.171	.5875
KQ6	Accurate knowledge content	41	3.244	.6237
KQ8	Comprehensive knowledge content	41	3.122	.5998
KQ9	Easy to find knowledge sources	41	3.146	.6543
KQ10	Easy to find knowledge expertise	41	3.098	.5387
KQ11	Relevant knowledge structure	41	3.317	.5215
US1	Satisfy with KM process features (effectiveness)	41	3.268	.5012
US2	Satisfy with system performance (efficiency)	41	3.390	.5421
US3	Satisfy with knowledge or information processing needs	41	3.098	.5387
IN1	Support for knowledge /experience recording	41	3.098	.5387
IN2	Support for knowledge /experience sharing	41	3.268	.5012
IN3	Support for knowledge /experience reuse	41	3.171	.6286

	Items	N	Mean	Std. Dev.
NB1	Effectively manage and store the required knowledge.	41	3.268	.4486
NB2	Accomplish tasks more efficiently	41	2.976	.7241
NB3	Improve the quality of my work life	41	2.902	.7002
NB4	Improve in decision making	41	2.951	.6305
NB5	Acquire new knowledge and innovative ideas	41	3.293	.5120
SVQ 1	Management able to provide KM direction	41	2.927	.6079
SVQ 2	Management able to encourage / develop knowledge sharing culture	41	3.122	.5097
SVQ 3	Management able to ensure KM strategies are realized	41	2.951	.5455
SVQ 4	Management able to ensure risk is monitored and controlled	41	2.951	.5455
Valid N (listwise)		40		

At the end of the survey, we also asked the respondents about their perception of the model's contribution to software development. Five choices were given, and they were encouraged to select three contributions that the model could offer. Fig. 11 illustrates the result.

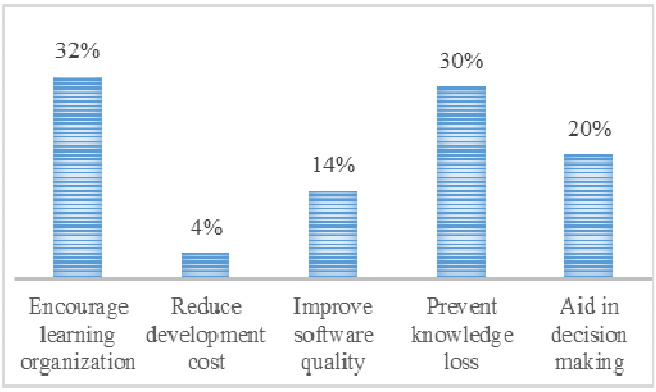

Fig. 11 Perception of EBF-SD contribution

The result shows that the three main contributions of the model are: encouraging learning organization (32%), preventing knowledge loss (30%), and aid in decision making (20%) (Fig. 10). This indicates that the participants do agree that the model could bring benefits to the organizations.

IV. Conclusion

This research has presented the stages of how the model of EBF-SD is developed and implemented as a prototype. And, how it is used as the instrument to evaluate the model together with a survey questionnaire. The prototype has demonstrated the provisioning of a cloud environment for its computing needs, the usage of 'nosql' database for flexible data storage, the addition of ontology to define the software lifecycles, and the implementation of software agents to assist automation on knowledge dissemination. The prototype has gone through the necessary evaluation for its efficiency and effectiveness based on defined KM success factors. Results show positive feedback on system quality, knowledge quality, user satisfaction, and intent to use/perceived benefits. Some improvements, however, are needed for net benefits and service quality measures. Perhaps, the questionnaire should be tailored to reflect more towards the respondents and towards the aspects that can be measured immediately.

The perceptions on the model's contribution are also assessed in which the result shows that the model can contribute more towards encouraging learning organization, preventing knowledge loss, and providing aid in decision making. In the future, the overall structural J&O model will be analyzed using structured equation modeling (SEM) to assess the measurement and structural model individually. To further benefit software development industries, this model can be deployed in a real-world application, and further rigorous evaluations can be assessed. This, however, will require a bigger sample size and a longer period of evaluation time.

Acknowledgment

This study is part of doctorate research, which is supported by GERAN PUTRA GP-IPS/2017/9518400.

References

[1] M. Hanafiah, R. Abdullah, M. Azrifah, A. Murad, and J. Din, "Towards Developing Collaborative Experience-Based Factory Model for Software Development Process in Cloud Computing Environment," *Int. Review of Computers and Software*, vol. 10, pp. 340–350, 2015.

[2] M. Ivarsson, and T. Gorschek, "Tool Support for Disseminating and Improving Development Practices," *Software Quality Journal*, vol. 20 no. 1, pp. 173-199, 2011.

[3] R. G. C. Rocha, R. Azevedo, and S. Meira, "A Proposal of an Ontology-Based System for Distributed Teams," in *40th EUROMICRO Conf. Softw. Eng. Adv. Appl.*, pp. 398–401, 2014.

[4] F. Salger and G. Engels, "Knowledge transfer in global software development: leveraging acceptance test case specifications," in *ACM/IEEE 32nd Int. Conf. Softw. Eng.*, vol. 2, pp. 211–214, 2010.

[5] P. Wongthongtham, N. and Kasisopha, N., "An Ontology-Based Method for Measurement of Transferability and Complexity of Knowledge in Multi-site Software Development Environment," *Lecture Notes in Computer Science*, vol. 6746, pp. 238-252, 2011.

[6] M. H. Bazerman, and M. D. Watkins, M. D., *Predictable surprises: The disasters you should have seen coming and how to prevent them.* Boston: Harvard Business School Press, 2004.

[7] R. Abdullah, and A. Talib, "Knowledge management system model in enhancing knowledge facilitation of software process improvement for software house organization," in *Information Retrieval Knowledge Management (CAMP)*, pp. 60–63, 2012.

[8] I. Nonaka, and H. Takeouchi, *The Knowledge-Creating Company.* NY: Oxford University Press, 1995.

[9] V. R. Basili, G. Caldiera, and H. D. Rombach, "The experience factory," *Encylopedia of Software Engineering*, pp. 470-476, 1994.

[10] K. Schneider, *Experience and Knowledge Management in Software Engineering*, Berlin: Springer-Verlag, 2009.

[11] H. H. L. C. Monte-Alto, A. B. Biasão, L. O. Teixeira, and E. H. M. Huzita, "Multi-agent applications in a context-aware global software development environment," in *Adv. Intell. Soft Comput.*, vol. 151, pp. 265–272, 2012.

[12] M. Z M. Nor, R. Abdullah, M. H. Selamat, and M.A.A Murad, "An Agent-Based Knowledge Management System for Collaborative Software Maintenance Environment," in *Int. Conf. on Design and Eval. Information Retrieval and Knowledge Management*, pp. 115–120, 2012.

[13] D. T. Tuan, and D. C. Tuan, "Enhance Java Software Development with Knowledge Acquisition and Management Tools," in *3rd*

International Conference on Knowledge and Systems Engineering, pp. 70, 2010.
[14] Y. F. Li, and H. Zhang, "Integrating software engineering data using semantic web technologies," in *Proceedings of the 8th Working Conference on Mining Software Repositories*, pp. 211, 2011.
[15] N. Sharma, K. Singh, D.P. Goyal, "Experience Base Approach to Software Process Improvement: Comparative Analysis and Design of Improved Model Advanced," in *2nd International Conference on Computing and Communication Technologies (ACCT)*, pp. 30, 2012.
[16] P. Ardimento, M. Cimitile, and G. Visaggio, "Distributed Software Development with Knowledge Experience Packages," in *Packages, Lect. Notes in Computer Science*, vol. 8186, pp. 263–273, 2013.
[17] W. H. DeLone, and E. R McLean, "Information Systems Success Measurement. Foundations and Trends," in *Information Systems*, vol. 2, no. 1, pp. 1–116, 2016.
[18] M. E. Jennex, "Re-examining the Jennex Olfman Knowledge Success model." in *Proceedings of the 50th Hawaii International Conference on System Sciences*, pp. 4375-4384, 2017.
[19] A. L. Halawi, R.V. McCarthy, J. E. Aronson, "An empirical investigation of knowledge management systems success," *The Journal of Computer Information Systems*, vol. 48, no. 2, 121, 2008.
[20] N. Nattapol, R. Peter, R., and K. Laddawan, "An Investigation of the Determinants of Knowledge Management Systems Success in Banking Industry," *International Journal of Social, Behavioral, Educational, Economic, Business and Industrial Engineering*, vol. 4, no. 11, 2010.
[21] M. Hanafiah, R. Abdullah, M. Azrifah, A. Murad, J. Din, M. Z. M. Nor, "Experience Based Factory Model for Software Development Process: Item Construct Validation on Questionnaire Design", *Journal of Theoretical and Applied Information Technology*, vol. 95, no. 1, pp. 177-195, 2017.
[22] M. Hanafiah, R. Abdullah, M. Azrifah, A. Murad, and J. Din, "Regression Analysis on Experience Based Factory Model for Software Development Process", *Journal of Telecommunication, Electronic and Computer Engineering*, vol. 9, no. 3, pp. 19-26, 2017.
[23] I. Sommerville, *Software Engineering*, 9th Edition. Pearson, 2011.
[24] G. Rasch, *Probabilitics models for some intelligence and attainment tests*, Copenhagen: DanmarksPpaedagogoske Institut, 1960.
[25] M. Tavakol, R. Dennick, "Making sense of Cronbach's alpha," *Int J Med Edu*, vol. 2, pp. 53-5, 2011
[26] W. P. Fisher, "Rating Scale Instrument Quality Criteria,"*Rasch Measurement Transactions*, vol. 21, no. 1, pp.1095, 2007.
[27] Wright B. D. and Linacre J. M., "Reasonable mean-square fit values," *Rasch Measurement Transactions*, vol. 8, no. 3, pp.370, 1994.
[28] R. Abdullah, Z. D. Eri, and A. M. Talib, "A model of knowledge management system for facilitating knowledge as a service (KaaS) in cloud computing environment," in *Proc. International Conference on Research and Innovation in Information Systems*, pp. 1-4, 2011
[29] K. S. Rubin, *Essential Scrum.* Addison-Wesley, 2013.
[30] M. Hanafiah, R. Abdullah, M. Azrifah, A. Murad, and J. Din, "Infrastructure Requirements For Experience Based Factory Model in Software Development Process in a Collaborative Environment", *Journal of Acta Informatica Malaysia (AIM)*, vol. 1, no. 2, pp. 9-10, 2017.
[31] A. M. Talib, R. Atan, R. Abdullah, and M. A. A. Murad, " Multi Agent System Architecture Oriented Prometheus Methodology Design to Facilitate Security of Cloud Data Storage," *Journal of Software Engineering*, vol. 5, no. 3, pp. 78-90, 2011.
[32] AWS, https://aws.amazon.com/, retrieved on 2nd April, 2018.
[33] MongoDB, https://www.mongodb.com/ retrieved on May 2nd, 2018.
[34] J. M. Linacre, "A User's Guide to Winsteps: Rasch-Model Computer Programs," retrieved at http://www.winsteps.com/winman/index.htm, 2016.