INNOCROWD, A CONTRIBUTION TO AN IOT BASED ENGINEERING PRODUCT DEVELOPMENT

Camille Salinesi[1], Clotilde Rohleder[2], Asmaa Achtaich[1, 3], Indra Kusumah[1, 2]

[1]CRI -Paris 1 Sorbonne University, Paris, France
[2]University of Applied Science HTWG Constanz, Germany
[3]Siweb – Université Mohammed 5, Rabat, Maroc

ABSTRACT

System engineering focuses on the realization of complex systems, from design all the way to management. Meanwhile, in the era of Industry 4.0 and Internet of Things, systems are getting more and more complex. This complexity comes from the usage of smart sub systems (e.g. smart objects, new communication protocols, etc.) and new engineering product development processes (e.g. through Open Innovation). These two aspects namely the IoT-related sub system and product development process are our main discussion topics in our research work. The creation of smart objects such as innovative fleets of connected devices is a compelling case. Fleets of devices in smart buildings, smart cars or smart consumer products (e.g. cameras, sensors, etc.) are confronted with complex, dynamic, rapidly changing and resource-constrained environments. In order to align with these context fluctuations, we develop a framework representing the dimensions for building Self-adaptive fleets for IoT applications. The emerging product development process Open Innovation is proven to be three time faster and ten times cheaper than conventional ones. However, it is relatively new to the industry, and therefore, many aspects are not clearly known, starting from the specific product requirements definition, design and engineering process (task assignment), until quality assurance, time and cost. Therefore, acceptance of this new approach in the industry is still limited. Research activities are mainly dealing with high and qualitative levels. Whereas methods that supply more transparent numbers remain unlikely. The project-related risks are therefore unclear, consequently, the Go / noGo decisions become difficult. This paper contributes ideas to handle issues mentioned above by proposing a new integrated method, we call it InnoCrowd. This approach, from the perspective of IoT, can be used as a base for the establishment of a related decision support system.

KEYWORDS

Industry 4.0, Internet of Thing, Crowdsourcing, Neural Network, Decision Support System

1. INTRODUCTION

Industry 4.0 and Internet of Things have a significant impact on system engineering, especially with regard to product development and innovation creation processes. These challenges, along with the following discussions are presented in this paper, from the perspective of a smart car. For the last decade, the definition of a car was restricted to “an automotive vehicle for transportation” [1]. The current state of cars in general, and smart cars in particular, makes such a definition obsolete. This is mainly due to the penetration of miniaturized, powerful, affordable and smart devices to such a market. Today’s cars include various types of sensors, cameras, actuators together with software to manage them. It’s likely to find more code in a car than in the entire information system of a bank, so smart car software can be just as much difficult to design and manage.

Crowdsourcing is an emerging innovation creation process, defined by Howe [2]. Nowadays, thirteen years after the first definition, crowdsourcing has gained popularity in several application areas because of significant cost and time savings [3]. Crowdsourcing covers a large range of applications: execution of simple tasks, such as tagging a picture, medium tasks like for example programming a simple app, and even complex tasks as done by Local Motors during the design and production of a vehicle [4].

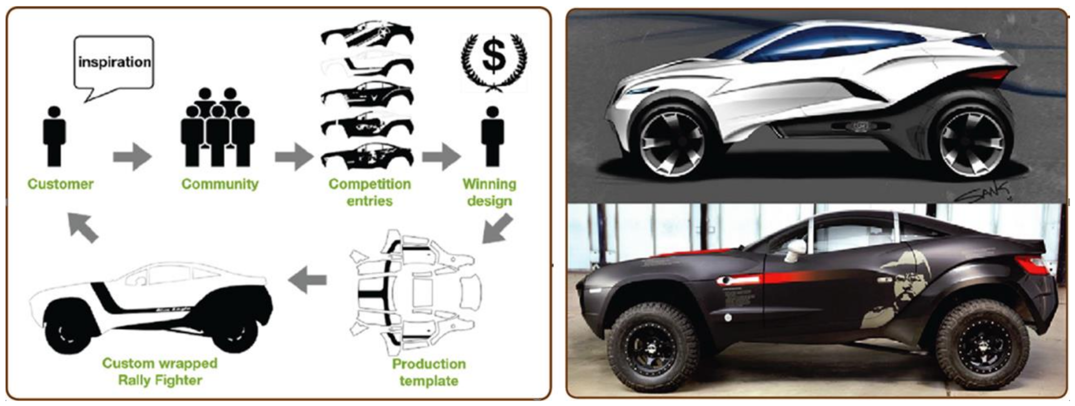


Figure 1. Crowdsourced vehicle development in Local Motors [4]

Local Motors is a car manufacturing company, supported by more than 30.000 engineers (the crowd) responsible for the design process of a new car, as illustrated in Fig.1. A traditional automobile manufacturer employs around 1000 until 5000 engineers to execute almost the same task. The great number of people engaged during the engineering process can make the design process more than 3 times faster and the manufacturing cost more than 10 time cheaper. With such unlimited resources and expertise, the options are also endless. Therefore, the smart cars manufactured by Local Motor may even go beyond car-related applications, like smart tracking, smart parking, smart traffic, or smart maintenance. They can also include services from other areas such as smart health, smart surveillance, or smart supply chain (see Fig 2). The first can be implemented to monitor a sick conductor’s vital signs, like blood pressure, stress or sleepiness. The second is concerned with detecting geographic areas where a specific information is required, captures it and reports it. The third one calculates and finds the quickest and cheapest paths to collect or dispatch packages. Other applications can also be added, as the requirements evolve, and the expertise expands. These possible extensions come with rules that guide and facilitate the specification of new cars.

Difficulty of crowdsourcing depends mainly on task complexity. When a task is complex, the company cannot precisely calculate project risk. Thus, it is difficult for the company to decide whether to use crowdsourcing and how to choose or develop appropriate methods and tools (Buhse et al [5] Chatterjee et al. [6] and Schenk et al. [7]).

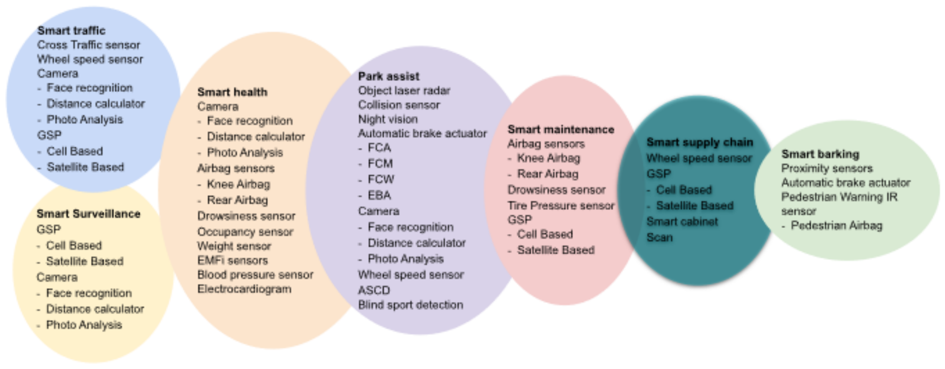


Figure 2. Sample of possible smart car applications provided through crouwdsourcing

In the following, we describe our contribution to a development of methods for simplifying decisions by giving transparency to the project risk. We present our method for handling product and process in IoT based engineering product development. The third chapter reports the verification of our methods. The fourth chapter discusses related works and enhancements to the state of the art through our contribution. The last section summarizes our current work.

2. A FRAMEWORK FOR THE MANAGEMENT OF AN IOT BASED APPLICATIONS

A smart car could be destined for a variety of clients, such as a logistics company, a special needs center, a housing complex or even a city. This variety complicates selection and risk assessment of devices, components, and services. Effective and flexible modelling of a self-adaptive system, in the context of fleets of connected devices, can part of a framework already proposed [8], as shown in Figure 3.

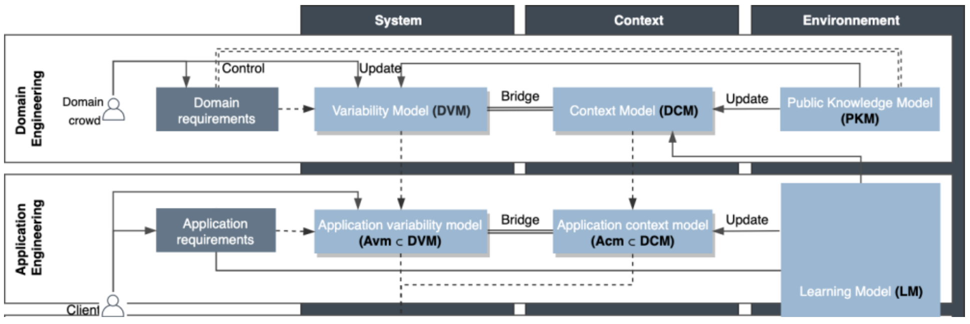


Figure 3. InnoCrowd overview, system set up and system usage

Domain engineering produces the design for reuse [9]. Following the rules of model-driven engineering, the framework separates concerns in order to model distinct aspects of the system, the context, and the environment. Several models can therefore be involved:

Domain Engineering

A Domain Variability Model (DVM) represents possible artifacts of the creation of a system, within the framework of domain engineering. This model specifies all the elements likely to constitute the final system and defines their dependencies.

A Domain Context Model (DCM) defines formally or semi-formally the information that surrounds these systems and is likely to have an impact on them.

A Public Knowledge Model (PKM) encompasses all artifacts that belong to the systems' knowledge domain. Unlike a DVM, which, also organizes this knowledge, a PKM goes beyond the relevant services for the domain in question, as it groups, organizes, and specifies a generic knowledge, which may or may not be injected in the DVM.

Application Engineering

An Application variability model (Avm) is related first to the application requirements, second to the DVM, being a flexible derivation thereof, and finally, to the context that's relevant for the end user.

An Application context model (Acm) is a subset of DCM that describes the elements of the context that affect the choice of the final system elements, according to the client's requirements.

A Learning Model (LM) observes on the one hand the behavior of the user, and on the other hand, the evolution of the context. Implemented both at the application and adaptation level, it empowers the smart car with the necessary artificial intelligence, to understand the actions/reactions it encounters.

3. METHOD FOR THE GENERATION OF CUSTOM SYSTEMS

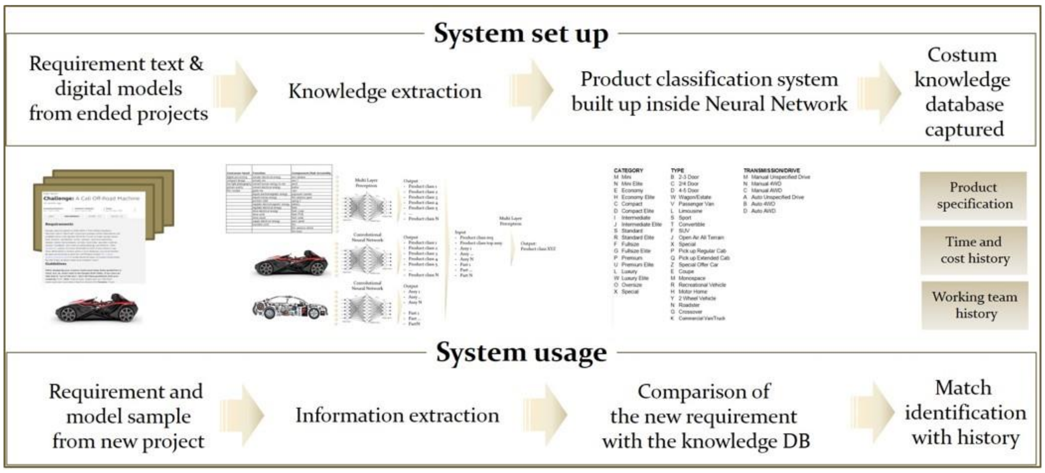

System set up
Requirement text & digital models from ended projects
Knowledge extraction
Product classification system built up inside Neural Network
Costum knowledge database captured
Product specification
Time and cost history
Working team history
System usage
Requirement and model sample from new project
Information extraction
Comparison of the new requirement with the knowledge DB
Match identification with history

Figure 4. InnoCrowd overview, system set up and system usage

We develop a novel method for the generation of a custom knowledgebase from specific crowdsourcing providers. This knowledgebase is derived from information extracted from the selected crowdsourcing provider to be used in the project. The block "System set up" provides an overview of how we build the DSS (Figure 4). It starts with the collection of input data which consists of text files (product requirements from the customer) and digital models (2D and 3D files of the designed product). These input files are extracted from a selected crowdsourcing provider, so that we have all their project histories as text and images. The second step is the extraction of information by using NLP, Ontology, and a Deep Learning Neural Network (DLNN). On the third step we embed a product classification system in the Neural Network (after NN training and test). The last step establishes the relations between

each product sub class and the project-specific information relevant for decisions: product specification, related time and cost, and the project history of the crowdsourcing provider.
The block "System usage" shows the way we use the DSS for assessment of a new planned project, in order to get a Go or NoGo decision. A new requirement and model sample are taken as input to the system. The second step is the extraction of relevant information from the given dataset (input in the previous step). The third step is the comparison of the dataset to the existing classes in DLNN, in order to find the appropriate position in the embedded product classification system. The final step provides the project-specific information relevant for decisions: product specification, related time and cost, and working team based on the project history. The user can then decide Go or NoGo.

We use DLNN for the automated recognition of the product design. For each step there is a special neural network trained for executing the recognition task. During the first step the visual representation of the car is exported digitally from 3D model into 2D picture taken from six sides: left, right, bottom, up, front and rear side. A deep learning neural network is known as the best analyser / classifier of 2D pictures. It can process the data faster than a 3D model classifier [9]. We use it for recognizing the graphics and classify it into the matching class in our classification system. We develop multiple stages for data analysis during the classification of the input data in form of text and digital models (see Figure 5, graphic for knowledge extraction). The first stage of text and digital model analysis will estimate product class from high level. When there are digital models for assembly and parts then they will be analysed for gaining information regarding the sub class of the product. On the second stage all information gathered from the first stage are taken as input. At the end the final product class will be estimated.

The requirements as a free form text are collected from the source system (crowdsourcing provider platform). Not all information is relevant to be used as input for the establishment of product classification. We need to extract them by using an ontology. We develop the ontology which apply the requirement specification based on the framework of self-adaptive system explained in the chapter 2.1. Each relevant part of text will be mapped to the related domain or application or adaptation layer. The third step will use the extracted information from each layer as input data for executing a supervised training of Neural Network. We provide two Neural Networks, each NN is specializing on handling each layer. At the last step we use output from each NN of third step as input for the last Neural Network. The result is the establishment of product class embedded in the Neural Network.

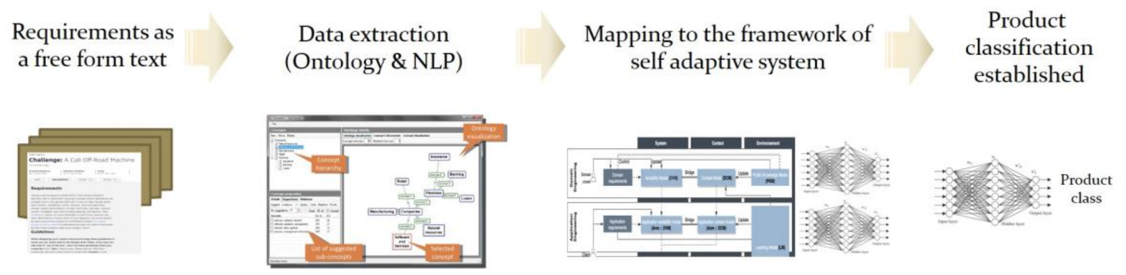
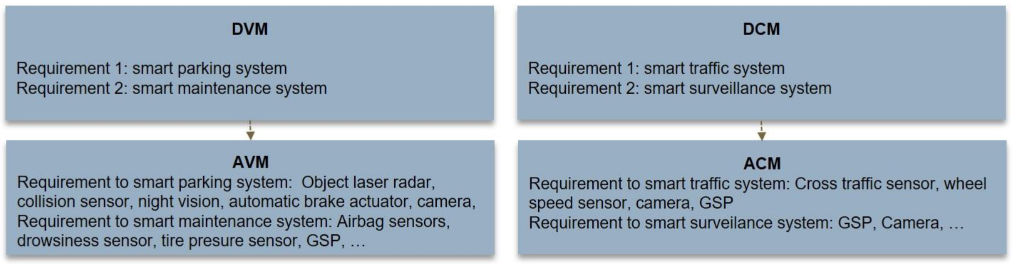

Requirements as a free form text
Data extraction (Ontology & NLP)
Mapping to the framework of self adaptive system
Product classification established
Challenge: A Cab Off-Road Machine
Requirements
Ontology visualization
Concept hierarchy
List of suggested sub-concepts
Selected concept
System
Context
Environment
Product class

Figure 5. Detail of requirement analysis by applying the framework of self adaptive system

4. APPLYING THE METHODS

As shown in Figure 3, there is a relation (line) between Domain Engineering and Application Engineering. The DVM is a domain high level requirement that to be fulfilled by AVM in application level. The DCM is a context related requirement that to be fulfilled by ACM in application level. PKM and LM represent requirements beyond the standard context, which are not to be applied in the current work, but in the near future. Figure 6 below gives a mapping example on the context of a smart car.



DVM

Requirement 1: smart parking system
Requirement 2: smart maintenance system

DCM

Requirement 1: smart traffic system
Requirement 2: smart surveillance system

AVM

Requirement to smart parking system: Object laser radar, collision sensor, night vision, automatic brake actuator, camera,
Requirement to smart maintenance system: Airbag sensors, drowsiness sensor, tire presure sensor, GSP, …

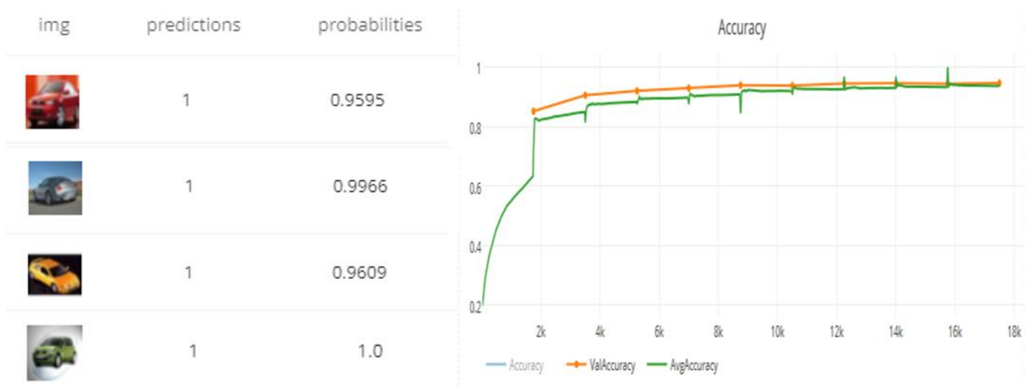
ACM

Requirement to smart traffic system: Cross traffic sensor, wheel speed sensor, camera, GSP
Requirement to smart surveilance system: GSP, Camera, …

Figure 6. Domain and Application Engineering mapping example

Each user requirement can be formulated freely in each block (DVM, DCM, AVM, or ACM). It is the task of NLP and ontology developed through this concept to identify automatically each case. We use this level of detail for classifying each new requirement given in a new crowdsourcing project. The DLNN was trained by using input data (requirement text) collected from a leading crowdsourcing provider. After being trained the DLNN can classify the new requirement with accuracy level 88% and 91%

For the classification of digital models, we collect images from successful projects. We build and train DLNN by using this input data. Then we develop digital models for the panned new project. The x axis of the graph below represents the elapsed training time. The y axis represents the accuracy level. As a result, DLNN can classify this digital model with the accuracy level between 95% and 98%.



img	predictions	probabilities
	1	0.9595
	1	0.9966
	1	0.9609
	1	1.0

Figure 7. Result of the image classification with deep learning Neural Network

The accuracy level of the new requirement classification is combined with the accuracy level of digital model classification. It gives the average accuracy level of 91%. This is acceptable for general estimation of the crowdsourcing effort. At the end the user gets the information of the past similar crowdsourcing task with the accuracy level or matching level of minimally 91%. This information consists of the product specification as well as time, cost and working team needed for finishing the task.

5. Related Work

Adapting new products in various contexts can be built around the goals and soft goals required or appreciated from the new system [10], using specific design patterns like reflection [11], or by the means of logical reasoning approaches such as constraint programming [12]. These approaches present limitations, such as rigidity and poorness of expression in the representation of a wide spectrum of requirements, and in the representation of dynamic evolving systems, specifically in terms of states [13]. Several authors have tackled the problem from a requirements and knowledge reuse perspective [14], however, ad-hoc and manual practices are persisting [15].

Crowdsourcing have proved efficient for building such a knowledge. Thuan et al. [16] built an enterprise ontology of business process crowdsourcing. It includes the main business processes, data entities, data attributes, and their hierarchical relationships, which were structured into a lightweight ontology. He added decision-making relationships and business rules that turn a lightweight ontology into a heavyweight. The method developed by Thuan has general application to estimating the ability to crowdsource. It is universal but it gives only a qualitative estimation. It does not explicitly estimate business-relevant aspects such as time and project cost.

This paper introduces InnoCrowd, which provides tools for the systematic reuse of requirements, from project histories. This minimizes risks and reduces resources when building new projects. The decision-making process is based on a domain ontology, which describes system components and the metrics that guide their selection, according to user requirements. Furthermore, the framework provides a blueprint for the implementation of such systems, while managing the high level requirements and dependencies.

6. DISCUSSION

During the system usage (see Chapter 3) we are assuming that the user provides 2D or 3D sample models of the target product. In practice, these sample models are not always provided by the system user. In that case, 50% of input information is missing. We don’t handle this case yet. The knowledge database currently available is based on a full existence of input files (text and sample models).

During the preparation of the system set up we need to collect dataset of 2D or 3D models. Digital models of some products are available in a great amount and free to be used. But some other specific products are not easy to be found. For example, we can find many digital models of various cars in the world wide web. But digital models of various helicopters are much more difficult to be found than cars. Lack of model samples can lead to system performance instability. Therefore, in this system prototype we focus on area where digital models are easy to be found.

During the data extraction scenario (see Figure 5) we need to have a powerful ontology which can be used for extracting the input text file efficiently. The creation of the ontology is partly automated [17] and partly manual. The dataset volume is big and the ontology to be created for handling this big data is very complex. The ontology creation is very time consuming because of this complexity. We need to find a more automated approach in order to reduce the effort needed.

7. FUTURE WORK, LIMITATIONS AND CONCLUSION

Three aspects are important in developing enterprise systems: product, process and people. Our research focused on the first two aspects: product and process. Our future work will address the third aspect, people. In figure 4 we can see there “Working team history” mentioned as the captured custom knowledge database. The team consist of a number of workers (people) but the information captured is limited only to the amount of the workers. This information is relevant for the scope of the current research work namely for the estimation of effort needed during the evaluation of new crowdsourcing tasks. In the near future we will manage and process information of people in order to optimize their occupation.

IoT-based engineering for product development and the related innovation creation processes are getting significantly more complex through new features and extended workflows (sub processes). We develop methods for managing the complexity and provide intelligent features for extracting information relevant to decision-making. Methods for handling complex products are developed and embedded in the system for handling innovation creation process that we call InnoCrowd. We verify it by using a smart car as an example. We defined product requirements in text files. Requirements analysis classified them with accuracy between 88% and 91%. Digital model analysis gave an even a higher accuracy, between 95% and 98%. Results confirmed the value of InnoCrowd as a decision support system. Further research could enable the system to inform the user how a new project relates to a previous project.

REFERENCES

[1] R. E. Allen, F. G. Fowler, H. W. Fowler, and eds, (1976) “The pocket Oxford dictionary of current English” Oxford University Press.

[2] J. Howe, (2009) “Crowdsourcing: How the power of the crowd is driving the future of business” New York, NY: Random House.

[3] L. Wang, Z.H. Zhou, (2016) “Cost-Saving Effect of Crowdsourcing Learning” Proceedings of the Twety-Fifth International Joint Conference on Artificial Intelligence (IJCAI-16).

[4] A. Brem, V. Bilgram, (2015) “The search for innovative partners in co-creation: Identifying lead users in oscial media through netnography and crowdsourcing” Journal of Engineering and Technology Management, Elsevier.

[5] W. Buhse, L. Reppesgard, and U. Lessmann, (2011) “Der Case Local Motors: Co-Creation and Collaboration in der Automotive-Industrie”.

[6] S. Chatterjee, P. Khandekar, and B. Kumar, (2014) “Reimagining enterprise innovation through crowdsourcing” WhitePaper, TCS, http://www.tcs.com/SiteCollectionDocuments/White%20Papers/Reimagining-enterprise-innovation-crowdsourcing-1114-1.pdf [access: 5.11.2016]

[7] E. Schenk and C. Guittard, (2009) “Crowdsourcing: What can be Outsourced to the Crowd, and Why?” In Sciences de l'Homme et Societe, Strasbourg, France.

[8] A. Achtaich, N. Souissi, R. Mazo, O. Roudies, and C. Salinesi, (2018) “A DSPL Design Framework for SASs: A Smart Building Example” EAI Endorsed Trans. Smart Cities, vol. 2, no. 8, p. 154829.

[9] K. Pohl, G. Böckle, and F. Van Der Linden, (2005), Software Product Line Engineering. Foundations, Principles, and Techniques, vol. 49, no. 12.

[10] J. Whittle, P. Sawyer, N. Bencomo, B. H. C. Cheng, and J.-M. Bruel,(2010) “RELAX: a language to address uncertainty in self-adaptive systems requirement” Requir. Eng., vol. 15, no. 2, pp. 177–196.

[11] M. Mongiello, G. Boggia, and E. Di Sciascio, (2016) “ReIOS: Reflective Architecting in the Internet of Objects” Proc. 4th Int. Conf. Model. Eng. Softw. Dev., no. February, pp. 384–389.

[12] P. Sawyer, R. Mazo, D. Diaz, C. Salinesi, and D. Hughes, (2012) "Constraint Programming as a Means to Manage Configurations in Self-Adaptive Systems" Spec. Issue IEEE Comput. J. "Dynamic Softw. Prod. Lines," vol. 45, no. October 2015, pp. 56–63.

[13] I. Reinhartz-Berger, A. Sturm, and Y. Wand, (2011) "External Variability of Software: Classification and Ontological Foundations" Springer, Berlin, Heidelberg, pp. 275–289.

[14] N. Niu, J. Savolainen, Z. Niu, M. Jin, and J.-R. C. Cheng, (2014) "A Systems Approach to Product Line Requirements Reuse" IEEE Syst. J., vol. 8, no. 3, p. 827.

[15] C. Palomares, X. Franch, and C. Quer, (2014) "Requirements Reuse and Patterns: A Survey" in nternational Working Conference on Requirements Engineering: Foundation for Software Quality.

[16] N. Hoang Thuan, P. Antunes, D. Johnstone, H. Xuan Son, N. Hoang, and H. Xuan, (2015) "Building an Enterprise Ontology of Business Process Crowdsourcing: A Design Science Approach" in PACIS 2015 Proceedings, vol. 112.

[17] J.G. Polhill, (2015), "Extracting owl ontologies from agent-based-models: A netlogo extension", Journal of Artificial Societies and Social Simulation 18.

AUTHOR

Indra Kusumah has studied production technology in the Technical University Berlin. He has worked in several research institutions and in industries for working on topics Artificial Intelligent, Product Development, Manufacturing, Product Lifecycle Management and Open Innovation.