An Architecture for a Distributed Lean Innovation Management System

Carl Mai, Dominik Grzelak, Mariam Zia, Diana Lemme and Uwe Aßmann

Abstract— **The current trend for innovation management is going upward, the startup scene is more active than ever and new processes and trends to foster these innovations are developed constantly. Although we can see such an upwards trend, there is not as much development in software architectures supporting innovation management. In this work, a requirements analysis for such a software architecture was done based on various innovation processes. Finally, we propose this architecture as a system of systems together with our current reference implementation. The system is evaluated in various user studies, e.g., teaching, practical use at a university, and innovation competitions.**

Index Terms— **Architecture, design thinking, innovation, lean startup, system of systems.**

I. Introduction

Innovation management is usually a collaborative task done in small groups. To foster innovation, there exist various processes, methodologies and principles. With these, the development of new products is stream-lined causing more and more companies to utilize these techniques. Current approaches are mostly centered around physical interaction, drawing on whiteboards or post-it notes. This falls short, when physical presence is not possible (e.g., in globalized companies or companies supporting remote working) or the created artifacts have to be duplicated and further edited.

Collaborating on a project in innovation management usually requires the physical presence of all stakeholders in customer interview meetings, stand-up meetings, canvas workshops, etc. Alas, current tools do not support lean technologies in a distributed setting well, because they do not support distributed idea management, distributed canvas development, and distributed document creation for business cases.

Our objective is to support innovation in a distribution and collaborative manner for user groups all over the world.

This paper presents LINC (Lean INnovation Center), a web portal with a tool suite to support distributed lean development. LINC is a distributed innovation platform with a holistic approach. Current innovation platforms usually fall short when supporting multiple representation formats or processes, e.g., the innovation platform just supports the ideation process or just the creation of business model canvases. With LINC, we want to propose an integrated innovation platform which supports many existing innovation processes. This work is structured as follows: first we give an overview of the related work. In Chapter III, several influential innovation concepts are presented. From these, requirements for our platform are derived, which gets introduced in Chapter IV together with the implementation details for all components. Chapter V further details our targeted reference architecture and how to model it with advanced concepts from SoS design. The first prototypes of the LINC platform are evaluated in several qualitative case studies in Chapter VI. Finally, we give a conclusion and an outlook.

Manuscript received October 9, 2018; revised December 29, 2018.
All authors are with the Technische Universität Dresden, Germany (e-mail: firstname.lastname@tu-dresden.de).

II. Related Work

The related work, to our approach is quite sparse. To our knowledge, there are no innovation platforms, which integrate as many tools as we do. Especially in academic works, the focus has been always on single applications. Therefore, we will give a short overview of the related work from our core platforms: idea- and canvas- management.

For idea management, there was a survey done by [1] with a comparison of multiple idea management platforms. One of the broader used platforms is Neurovation [2], which is an innovation challenge platform. Businesses can create innovation competitions here and an interested community will create ideas. Each competition has a set of prizes which are distributed among the best ideas. The Neurovation platform was also the basis used by TÜV Austria in InnovaTÜV [1]. From the book [1], no platform was shown, which supported more than just idea management, although InnovaTÜV depicted in their process, that a form of canvas management and project planning is required.

Idea-Mirrors are a research prototype suggested for companies to collaboratively create and edit ideas. The main goal of the idea mirror is to support the idea creation process in its earliest phases, when it is depending on a lot of communication and collaboration. It is meant to be used in conjunction with an existing idea portal which provides an interface for accessing its data [3]. Therefore, it could also be integrated into the LINC-ecosystem.

Digital canvas editing tools were reviewed according to their features in [4, 5, 6]. In [6], all existing tools were compared against a set of 31 features. The most features are implemented by *Realtime Board* with 30. Our proposed solution *Fridolean* reaches 19 features. Since the Fridolean development only started 1 year ago, we are confident to achieve most features soon.

III. Concept

With LINC, our goal is to build an integrated online

platform supporting multiple innovation processes. The main driver behind this platform is the lean innovation process [7] and design thinking methodology [8]. The next chapter will review existing innovation processes and methodologies to derive our requirements from.

A. *Innovation Processes and Methodologies*

In Fig. 1, we depict the process of the **Platform Innovation Kit** [9], which is following 5 consecutive steps. We want to explain each step and inspect it to derive requirements for our system.

1. Environment Scan: The goal is to understand the market, including market & industry forces, key trends and economic forces. To handle this information, documents must be managed and shared, tasks must be created and assigned.

2. Ideation Phase: Here a plethora of ideas are generated based on the previous environment scan. For this a structured way to enter ideas is needed. During the ideation phase it is important to provoke new ideas, e.g., by combining different ideas or other creative processes. At the end of the ideation phase the resulting ideas must be filterable and ranked.

3. Value Proposition: From a generated idea, a value proposition or business model canvas should be filled to understand more on the subject. This step requires a collaborative canvas management tool.

4. Service Design: In this step, a prototype is created or a possible architecture invented. This phase is highly application specific. A platform could support this phase best by improving the communication within the team. As a requirement we derive from this an agile task management tool and a chat platform to share intermediate results and coordinate the development.

5. Strategy: The final phase of the Platform Innovation kit is coming up with a good strategy, involving the investigation into competitors, stakeholders, business case and required resources. This step can be supported with suitable canvases, collaborative documentation and task management.

The TÜV Austria group also developed a process for their global innovation strategy in 2016 called **InnovaTÜV** [1]. The process involves their internal sources (i.e., employees), customers and external sources & trends. In a first step ideas are created, rated and prioritized. The result of this step is an innovation fact sheet. The second step is implementation planning, where a concept is developed together with project planning and budget planning. The results of this step is a business plan. The third step is called innovation project. Within this step the project is carried out by execution and controlling. The result is a completed project. In the last step, broad commercialization is performed and controlled over the span of 6 years.

The **lean startup process** has a 3-phase cycle of build—measure—learn. The main goal is to speed up every aspect of this cycle because a good product has to go through this cycle multiple times. The build phase is the development of the product or prototype. The measure phase is getting customer feedback on the product by interviews or usability tests. Based on this data, the learn phase starts which will incorporate the data in the business model canvas or use this data to refine the idea. [10]

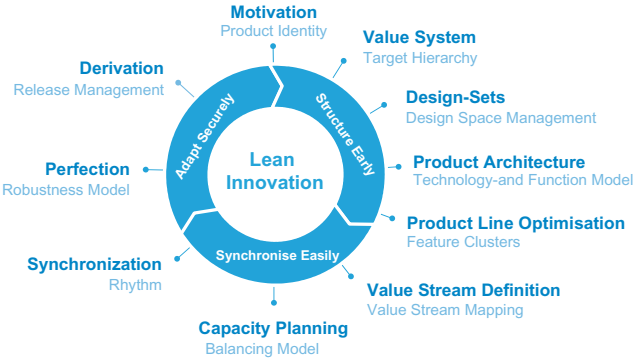

Fig. 2. Lean Innovation Process [7].

Lean Innovation is a set of principles which are shown in Fig. 2. It is based on three core principles: structure early, synchronize easily and adapt securely. These are further refined into 10 aspects. It is difficult to come up with concrete requirements for these principles as they are of abstract nature. Nevertheless, the three core principles are explained here and requirements are deduced. Structure early is the requirement to have as early as possible the goals defined and the basis analyzed. Stakeholder involvement is here already an important task. Synchronize easily is a method to avoid waiting times in the creative process by facilitating the synchronization of intermediate results. This step is usually supported by working in the same office. But nowadays within large companies, this often also means an easy form of document sharing and

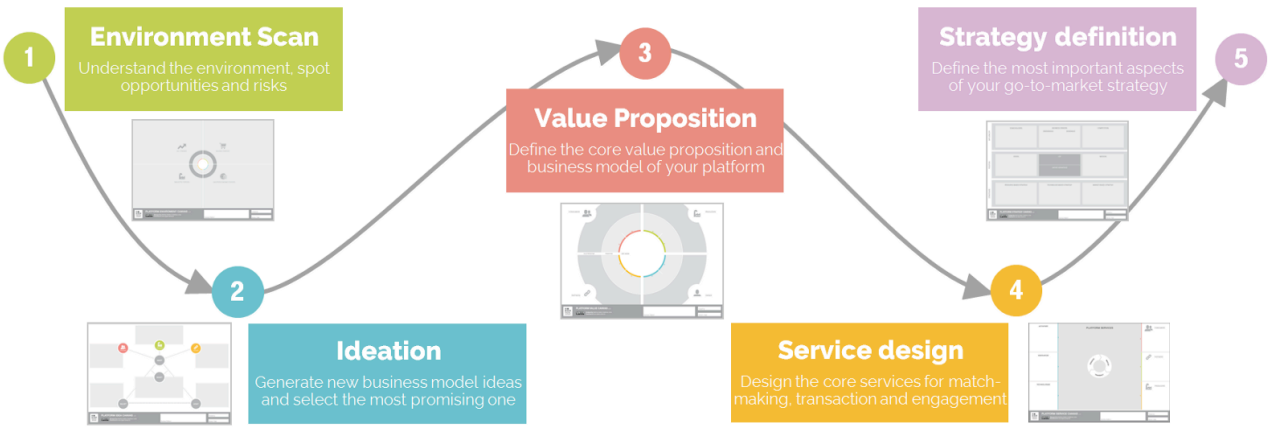

Fig. 1. Process of Platform Innovation Kit [8]

collaboration.

B. Dynamic Workflows

From the previous section it can be seen, that most innovations follow a specified process or workflow. In [7], the conflict on the level of specification of a workflow was detailed. On one hand it is required to give good guidelines. On the other hand, these guidelines should be open enough to support dynamic changes. Products and projects are very individual and could even change within one process [7, 11].

Traditionally, only fixed processes are supported in workflow systems. To implement dynamic behavior there are generally two approaches: modeling all possible alternatives at design time or dynamically modifying the workflow system at run time. Each approach has its advantages and disadvantages. While modeling all possible alternatives has the advantage to previously model check the whole system, the dynamic modifications to the workflow system prevent such checks in advance. On the other hand, the approach which previously models the alternatives cannot express unanticipated processes.

A solution for a dynamic workflow system are the ad-hoc workflows, these kind of workflow systems model a basic process which can be later refined and modified according to the dynamic needs. In [12] such a concept was proposed on the basis of Petri nets. During the execution, the net is extended and shrinked. With each such modification an analyzer is utilized to check global properties of the net. Furthermore, it is possible to store these dynamic nets for later use.

An alternative based approach are adaptive Petri nets [13]. The alternatives can be expressed with context places, which can prevent the execution of particular sub nets.

For the LINC-system we want to use a hybrid approach, which can utilize as much of the static knowledge as possible within adaptive Petri nets, while still supporting deviations from this model with dynamic modifications to the net — similar to ad-hoc workflows.

A workflow system has several tasks to fulfil: teaching, structuring and documenting. By teaching the user, we expect that the process is not yet known to the user and he or she requires each step laid out in detail. With structuring, we want to support experienced users, who have enough knowledge to modify the process according to their needs but still want to utilize some milestones. Finally, documentation should persist the route a user took to come up with the resulting product. This is mainly relevant for research purposes to discover the best processes and methodologies.

C. System of Systems (SoS)

SoS is defined as an “emergent class of systems that are built from components which are large scale systems in their own right” [14, 15]. Some examples of SoS are integrated air defence systems, traffic management systems and smart grids. Characteristic features of SoS [14, 15] that distinguishes it from monolithic systems are

1. Operational independence: An SoS is composed of constituent systems that are autonomous, independent and are useful in their own right i.e., a constituent system disassembled from an SoS, can continue to fulfil its own valid purpose.

2. Managerial independence: Constituent systems operate independently and are managed to achieve their own purpose. The constituent systems are individually acquired and integrated to SoS while they are continuously managed for their own operational purpose independent from SoS.

3. Emergent behaviour: Behaviour of SoS emerges as a result of interaction among constituent systems and cannot be achieved by any one of the constituent systems alone.

4. Geographic distribution: Constituent systems are geographically distributed that can exchange only information and knowledge from one another and not physical quantities of mass and energy.

5. Evolutionary development: SoS is never complete, rather it evolves continuously over time as requirements change. New functionalities and systems may be added and removed which might change the structure of SoS over time,

Properties 1, 2 and 3 are the most important for an SoS. By properties 1 and 2 it is assured that an SoS is composed of individual systems that are autonomous and independently owned that can perform new functions when placed together. Property 3 is also important because if the SoS does not give new properties or functions that are not possessed by constituent systems, then the system is not considered as a whole. Property 4 and 5 are not absolutes and may or may not exist. Another property identified by [16] is (6) heterogeneity, i.e., an SoS is composed of dissimilar systems but again as discussed in [15] this is also not absolute. Properties 4, 5 and 6 are common but not required and are not distinguishing for an SoS.

IV. Architecture

In this chapter the technology and constituent components of the LINC platform are explained. As discovered in the previous chapter, our required platforms are as follows:

P1 Idea management / Innovation platform
P2 Canvases
P3 Project management
P4 Document management
P5 Communication platform

With following general requirements:

R1 Central authentication
R2 Synchronization between the platforms
R3 Easy installation and administration

The main building blocks here, are the Docker-based containers, supporting R3, and the central authentication system (R1). Furthermore, each platform should have a REST-API, which can be used for synchronization (R2). Besides these three commonalities, the platforms are quite heterogeneous. An overview of all current components from the LINC-system can be seen in Fig. 3.

In this chapter, we will first give an overview of the central authentication system, then all the different services and their implementation are explained. In the end a short sub-section explains how we utilize Docker in the deployment process.

A. [R1] Authentication System: Keycloak

Because we are integrating multiple different platforms, which otherwise have their own authentication mechanisms,

we need a central user management tool. We decided to use the open source software Keycloak, developed by Redhat and based on WildFly. This platform implements many open standards (e.g. SAML, OpenId-Connect) and provides many adapters for different programming languages (e.g. Java, JavaScript, Python). While the integration with new platforms was not always easy, using such a large supported platform was a good choice. Keycloak itself brings a minimal user management system. A user can be part of multiple groups and roles. Attributes can be manually set by the connected services. Which we use right now for a mapping from the Keycloak-user to the service user-id. In the LINC-ecosystem, we allow the creation and participation in groups by the users. Roles are set by administrators only and currently distinguish only between admin and normal user.

B. Platforms

LINC consists of five platforms, which we want to explain here.

[P1] Idea Management: Watch Our Ideas. This platform was developed for Technische Universität Dresden in the context of the Open4Innovation project [17]. The backend is the Java-framework WildFly 13, which in turn is providing a REST-API for a JavaScript frontend. Ideas are always created as part of a board, which contains related ideas. An idea consists of a description and artifacts like images and documents. Access permissions can be given on a group and user level. After an idea is created, it gives several options to synchronize with other LINC-platforms: a new Fridolean project is created if it not exists, synchronizing also all access permissions of the idea. If a Fridolean project exists, all existing canvases are listed on the idea-page. Similarly, the synchronization works with Taiga and CodiMD.

An example, how viewing an idea within Watch our Ideas looks like, can be seen in Fig. 4.

[P2] Canvas Management: Fridolean. Fridolean, was initially developed by students as part of a mandatory software development course at our university. Frontend and backend are based on JavaScript, utilizing the libraries React and Express with a MongoDB database. The platform supports the creation of projects, which can contain multiple canvases. The canvas editing supports multi user real-time collaboration. The changes of one user are reflected on the screen of all other users. This is an important feature, as canvas editing is a highly collaborative task.

[P3] Task Management: Taiga. For the task management, we utilize the open source software Taiga. The backend is Python with the web-framework Django and a frontend written in CoffeeScript. The decision for this platform was mainly its support for all the different task management methodologies. It supports SCRUM and Kanban [18] and would even allow a waterfall model of issue management. All these methodologies can be used and combined in a single project.

[P4] Document Management: CodiMD. CodiMD is an open source collaborative document editor. Each participant gets its own cursor inside the document and each edit is synchronized to all users. This platform is JavaScript based in frontend and backend.

[P5] Chat Platform: Rocket.Chat. As an open source chat platform Rocket.Chat is used. Besides giving users the possibility to chat with each other, it is also our portal for important announcements and collection of error logs.

[R3] Health Monitor: Checkup. Important for the system administration is the health monitor. This monitor

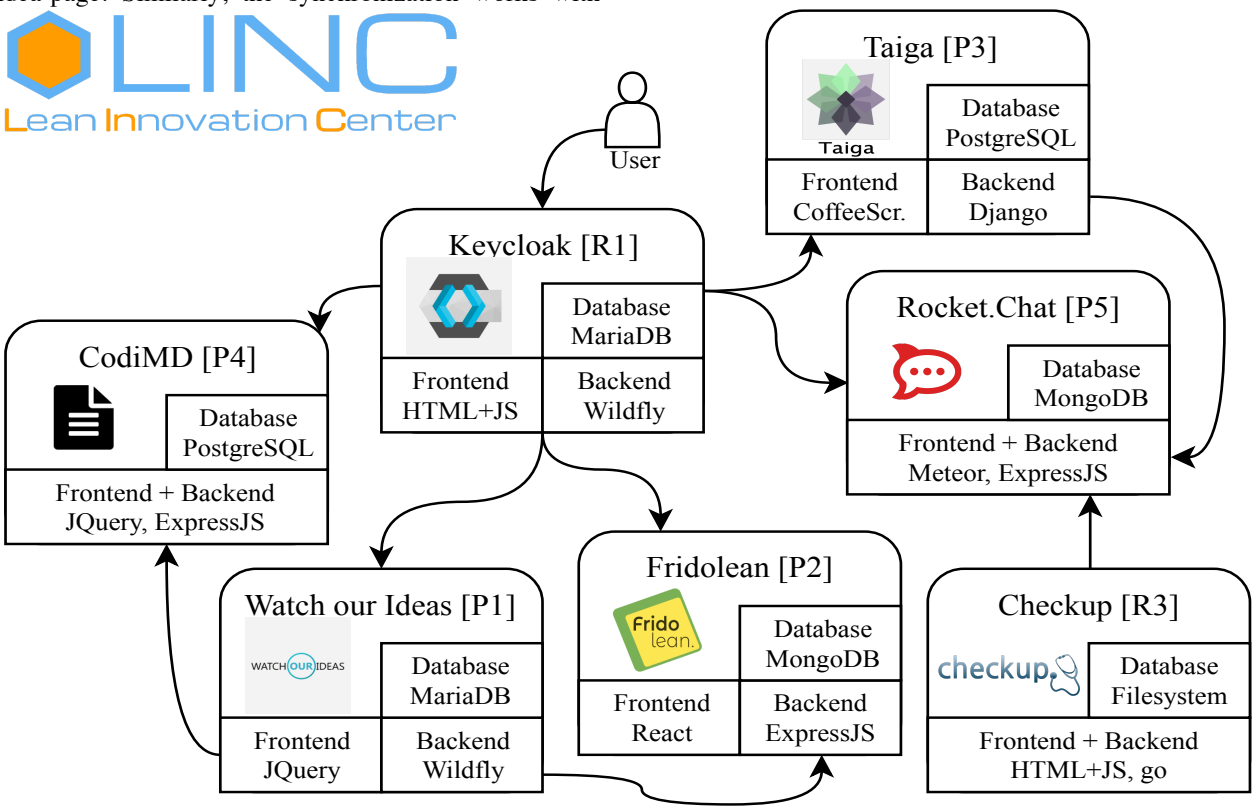

Fig. 3. Overview of all components within the LINC system. The arrows denote the information flow.

will poll in a specified period each deployed resource and warn the administrator when a platform is not available. As solution for this, we decided for Checkup. This open source tool allows direct integration with our chat platform with webhooks and therefore can give the administrator a timely warning, when one of the system fails.

C. [R2] Synchronization

Integration of all systems is done with two key technologies: single sign on with Keycloak and by utilizing the REST API of all platforms. The best integrated platforms are innovation (P1) and communication (P5). After creating an idea, it is possible to generate a Fridolean and Taiga project, as well as a document. This synchronization is especially helpful for initializing the tools with a description and to synchronize the access control across the platforms (i.e., all collaborators of an idea are also added to the project or document). The communication platform is also well integrated, as some of the used open source tools already support the generic webhook API.

D. [R3] Deployment

Deployment was of special interest because the integration of so many different platforms make an easy deployment difficult. Each platform requires different system components which also sometimes contradict each other (e.g., conflicting database versions). Furthermore, innovation is one of the most important goods a company possesses. These companies will not follow the current trend to put these into the cloud, so that a security breach would expose all their future plans.

The solution must be some form of virtualization to provide an isolated environment for each platform. A lightweight virtualization technique are Linux containers with the most popular implementation being Docker, which is explained in the next section.

E. Docker

Docker is a popular container implementation for the Linux kernel. Containers are a virtualization mechanism with very small overhead. Besides sharing the kernel of the host system, containers are isolated in most aspects from the host: the processes run in a different namespace and filesystem access is only granted at predefined points. Docker offers to define containers based on a sequence of shell commands for the installation. One fetches a

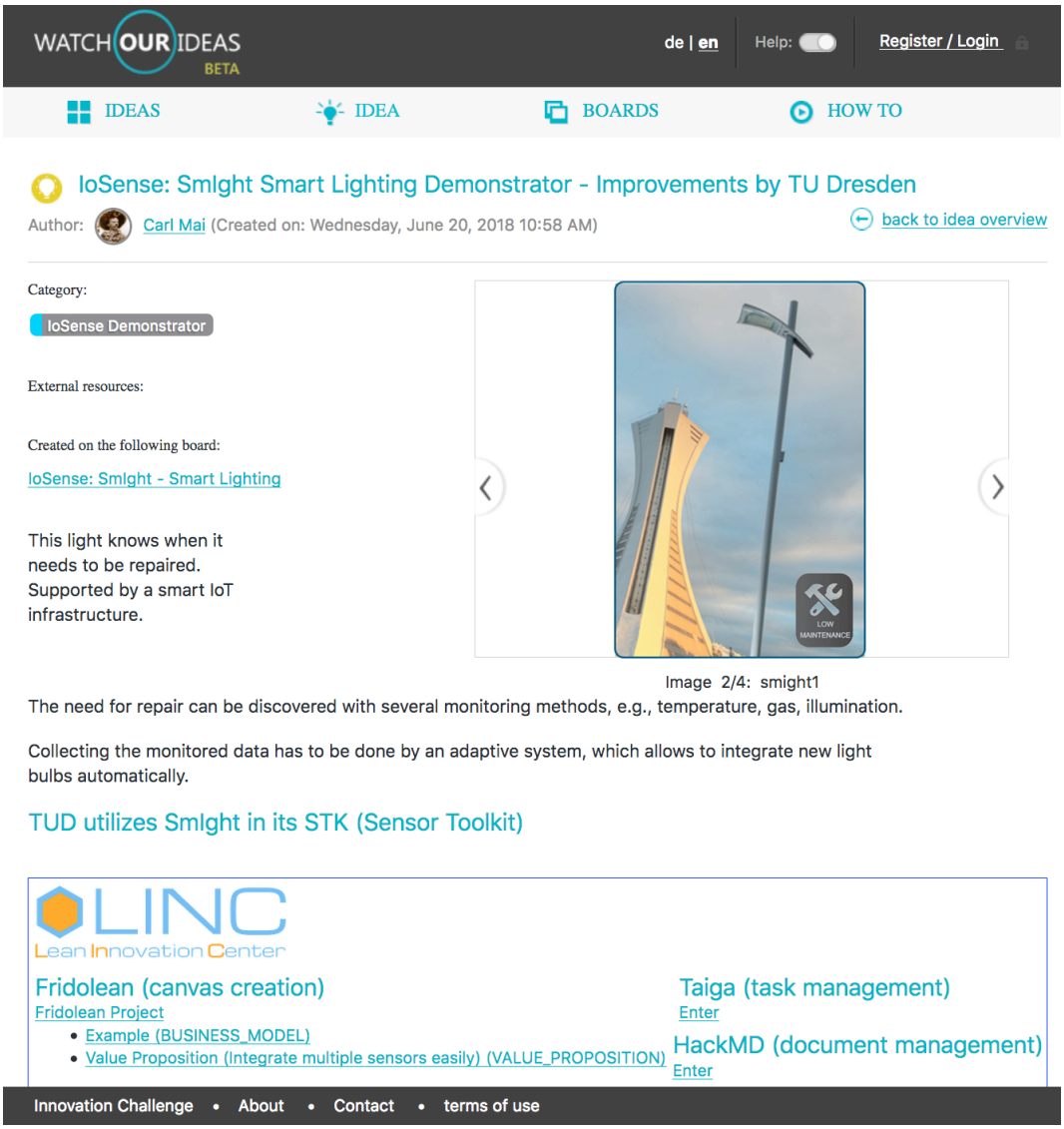

Fig. 4. Watch Our Ideas (P1): An idea management software. The image shows an idea with title, author, description and images. At the bottom the connection to other LINC-components, such as Fridolean, can be seen.

base-image, on which additional dependencies can be installed. This allows to customize the desired image, so that it fits with all individual applications.

Each Dockerfile should contain just one application. When a service requires multiple applications (e.g. the service and a database), one can use Docker-compose to structure these. A Docker-compose file defines which container work together and sets their environment variables, on which location of the filesystem they can write and which ports to expose.

Deployment: The deployment of the LINC platform is described in a single archive containing two Docker-compose files for each service: one file for a default configuration and one for the individualized configuration. Furthermore a single .env-file is used to configure variables, which are expanded within the Docker-compose files (e.g. ports, usernames, passwords or other configurations).

Most services expect to be in the root of a domain: therefore, a subdomain for each service is recommended. A reverse-proxy, like Nginx or Apache2, is utilized. Some platforms require some manual configuration on their webpages: e.g., copying secrets from Keycloak or setting some configurations. In a future version this will be automated. The current setup time of a new server is ca. 30 minutes, but our goal is to minimize these actions further, so that a reverse-proxy becomes self-configuring and that the configurations can be extracted automatically. Furthermore, the deployment should later support Kubernetes or Docker-swarm, so that the services can be easily distributed across a network of servers.

V. Reference Architecture: LINC as System of Systems (SoS)

In this chapter we want to take a closer look on the LINC architecture in regards to SoS modelling. As mentioned in Chapter IV, all 5 platforms of LINC were developed and managed independently for their own valid purpose and therefore holds property 1 and 2 of SoS, which are described in Section III-C. These independent systems are brought together to create a LINC SoS that enable us to perform lean innovation management, which cannot be achieved by any of the individual systems alone. This lean innovation is the emergent behaviour achieved as a result of interaction of constituent system of LINC and thus it adheres to the property 3 of an SoS. In addition to the three critical properties of SoS, LINC also fulfils the evolutionary development and heterogeneity. Regarding the property 4 of geographic distribution LINC may or may not be geographically distributed. Property 5 describing an SoS in constant evolution is true for our current implementation. Property 6, the SoS consists of heterogeneous components is definitely true for LINC, which consist of a many different programming languages and frameworks.

A. *Emergent behavior, ensemble modelling and dynamic workflows*

Constituent systems in SoS can perform various tasks since they are autonomous systems that can be used for various purposes. SoS creates a dynamic context in which constituent systems perform SoS specific tasks and collaborate to exhibit desired emergent behaviour. An SoS runs in a dynamic environment where the context and requirements might change over time. In response to these changes, the emergent behaviour must also adapt (**adaptive**) to ensure continued valid operation. Emergent behaviour might appear (or is created) in response to some environmental change (**transient**) then it might grow i.e., more systems are added to SoS or shrink over time (**elastic**) and is finally dissolved. Thus, emergent behaviour is a dynamic context for constituent systems that has adaptable, transient and elastic properties.

Ensembles [19], [20] are defined as group of components that interact to achieve a certain goal. Notion of ensemble used by [19] can be used to present the emergent behaviour with **elastic** properties. Ensemble defines a membership predicate that evaluates whether the system qualifies to be a part of SoS or not. Therefore, systems can add or leave the SoS. Another notion of dynamic context was defined in [21] as emergent gummy modules that defines **transient** nature by defining construction and destruction predicates. A dynamic context is created in response to a construction predicate qualifying to true and then it is removed when destruction predicate is satisfied. To add the concept of dynamic **adaptation**, architecture of ensemble can be represented using adaptive Petri nets [13] that can be adapted in response to changes.

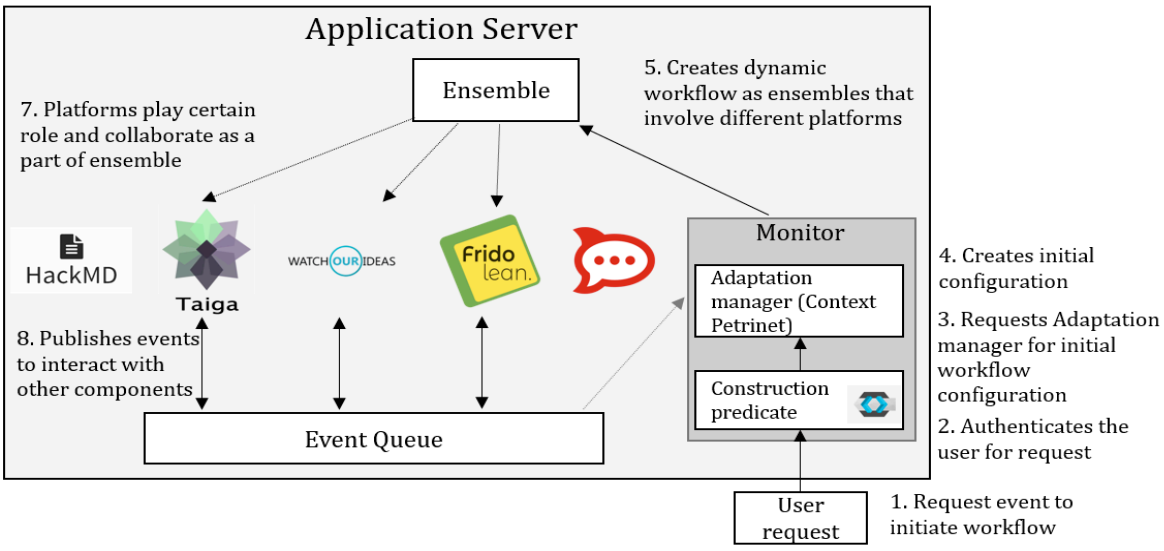

Fig. 6. Reference architecture for an ensemble based LINC SoS

Currently, no version of ensembles (dynamic contexts) fulfil above mentioned properties. In our future work we plan to develop a notion of ensembles that uses a construction / destruction predicate to account for the transient nature, a membership predicate to account for the elastic nature, an adaptive petri net [13] for the adaptive nature

Since constituent systems are autonomous and perform a variety of tasks. The tasks that constituent systems must perform in the context of current SoS interaction is defined by an ensemble in the form of role assignment [22]. Ensembles also define the interaction between the roles necessary to carry out the task. Interaction between the systems is event based to ensure loose coupling, distributed nature and heterogeneity of an SoS. When architectural configuration changes the role assignment and interaction among the constituent systems change as well.

Dynamic workflows in LINC can be represented using the concept of ensembles where a dynamic workflow can be created on demand (user request), adapted to a certain context and can eventually involve more and less platforms in different roles depending on the context. Configuration of the workflow may change as context changes.

B. Keycloak in SoS

Keycloak is responsible for authentication. With the proposed architecture for LINC with ensembles, Keycloak will be a part of construction predicate. Construction predicates are evaluated before a dynamic workflow is initiated. It checks whether all requirements are fulfilled or not for starting the dynamic workflow. Before a workflow is established the construction predicate with Keycloak can check if the user who requested the workflow has the access and has the right role to initiate the desired workflow.

C. Proposed Reference Architecture

We propose an event based architecture for ensemble modelling for the LINC-SoS. Monitor component runs the construction predicate with Keycloak. Adaptation manager runs the context Petri nets [13] that generate the initial configuration for ensemble. This configuration will be updated as monitor receives further update events from the queue indicating a change in the context. Each ensemble (workflow) might not involve all the constituent systems, for instance in Fig. 6, it involves only 3 components. Participants of an ensemble communicate through a global event queue.

VI. Evaluation

For developing the platform, we used the lean startup methodology with the build—measure—learn cycle. Within each iteration, we let users test our platform and incorporated their feedback. Because of our small user-base, we mostly evaluated the results on an individual basis. To attract enough users, we conducted innovation competitions, utilized the platforms ourselves and used it in teaching.

A. Innovation Competition

We performed two innovation competitions within the IoSense Project (an EU ECSEL Project). The first competition was used to evaluate our platform and invite stakeholders of the project to generate new ideas in the area of IoT. Central for this competition was the idea platform and partially the canvas system. To attract a higher audience, the best ideas could win prizes. As a result, more than 20 ideas were entered within 2 months and we got 30 new registrations. For some ideas a BMC (business model canvas) was created in the canvas platform.

B. Project Management

Within our group at the university, we employ some parts of the LINC-platform to support PhD students with organizing the research projects and progress with their dissertation and paper writing. After 8 month of use, the platforms for task management, document management and chat are utilized a lot. Every research project is now documented inside the task management system, which improved the quality of the research a lot. Publications are often structured within the collaborative document editor. Protocols for presentations are also often collaboratively written within this tool. The chat platform unified other (commercial) chat platform and most emails. It helped our group to come closer together by an improved communication.

C. Teaching

As of this writing, the platform is used in teaching the Software as a Business course at Technische Universität Dresden (Germany). Around 15 students are learning in this course how to create a business from a software project. The course is based on design thinking and Lean Innovation principles. The course is divided in 50% lecture and 50% practical work. During the lecture the students learn agile project management and how to derive an MVP (minimum viable product). Which then is practically used with the tasks and canvas platform in LINC. Ideas, documents and chat are also utilized for collaborative innovation. In the end of this course, we let the students evaluate LINC with a questionnaire.

VII. Conclusion and Outlook

We presented in this work a collaborative platform for lean innovation. It is a generic solution, which supports various innovation processes. It is targeted for innovation in small, medium and large businesses as well as research and technology organizations. Furthermore, this platform was designed with the intent to teach students on the steps in lean innovation. To reach the current state of the platform, we reviewed existing innovation processes and analyzed them, which software platforms and general requirements are needed. Based on this, 5 platforms and 3 general requirements were identified. The platforms were instantiated with a system of systems architecture. Finally, the suitability was evaluated with several user studies. While each user study created new feature requests, the general result was that this kind of system is highly required in teaching as well as in industry and research. Future research will go in two directions: improve the platforms teaching capabilities with a better user-guidance concept based on dynamic workflows; further improve the architecture to allow better scaling to more easily integrate new platforms.

ACKNOWLEDGMENT

We gratefully acknowledge support from the German Excellence Initiative via the Cluster of Excellence "Center for advancing Electronics Dresden" (cfAED).

This project has received funding from the Electronic Component Systems for European Leadership Joint Undertaking under grant agreement No 692480. This Joint Undertaking receives support from the European Union's Horizon 2020 research and innovation programme and Germany, Netherlands, Spain, Austria, Belgium, Slovakia."

REFERENCES

[1] C. Seja and J. Narten, *Creative Communities / Ein Erfolgsinstrument für Innovationen und Kundenbindung* Springer Gabler, 2017.

[2] A. Stocker, G. Granitzer, P. Hoefler, V. Pammer, R. Willfort, A. M. Koeck, K. Tochtermann, "Towards a framework for social web platforms: The neurovation case," *Third International Conference on Internet and Web Applications and Services*, Athens, Greece, 2008, pp. 227-232.

[3] M. Koch, F. Ott, "Idea Mirrors – Einsatz großer Wandbildschirme zur Förderung diskontinuierlicher Innovation in der Softwarebranche," *Workshop Virtuelle Organisation und Neue Medien*, Dresden, Germany, 2008, pp 241-252.

[4] T. Schoormann, D. Behrens, R. Knackstedt, "Softwaregestützte Modellierung von Geschäftsmodellen – Vergleich und Weiterentwicklungsperspektiven am Beispiel der Business Model Canvas," *Informatik 2016*, Bonn, Germany, 2016, pp. 1333-1347.

[5] M. Oddoy, "Entwicklung eines Frameworks für Kollaboratives Systemdesign mit Interaktiven, Digitalen Canvases," Master thesis, Dept. Software Engineering, Technische Universität Dresden, Dresden, Germany, 2014.

[6] T. D. Pham, "Anforderungsanalyse und Konzeption eines Allgemeinen Modells für Canvases," Bachelor thesis, Dept. Software Engineering, Technische Universität Dresden, Dresden, Germany, 2018.

[7] G. Schuh, M. Lenders, S. Hieber, "Lean innovation–introducing value systems to product development," *International Journal of Innovation and Technology Management* 8.01, pp. 41-54, 2011.

[8] I. Rauth, B. Jobst, E. Köppen, C. Meinel, "Design thinking: An educational model," *Proceedings of the 1st international conference on design creativity*, 2010.

[9] M. Walter, M. Lohse, and S. Guzman. "Platform Innovation Kit, 5 steps to a new platform business model," *Retrieved* Oct. 2018 https://medium.com/platform-innovation-kit/in-5-steps-to-a-new-platform-business-model-7660391cafdd.

[10] R. Kaiser, G. Püschel, S. Götz, K. Kahle, U. Aßmann, "Von der software-dissertation zum lean startup," *Software-engineering and management*, Dresden, Germany, 2015.

[11] B. Vandenbosch, A. Saatcioglu, S. Fay, "Idea management: A systemic view," *Journal of Management Studies* 43.2, pp. 259-288, 2006.

[12] M. Voorhoeve, W. V. d. Aalst, "Ad-hoc workflow: problems and solutions," *Database and Expert Systems Applications. 8th International Conference,* pp. 36–40, 1997.

[13] C. Mai, R. Schöne, J. Mey, T. Kühn, U. Aßmann, "Adaptive petri nets – a petri net extension for reconfigurable structures," *The Tenth International Conference on Adaptive and Self-Adaptive Systems and Applications*, Barcelona Spain, pp. 15–23, 2018.

[14] M. W. Maier, "The Role of Modeling and Simulation in System of Systems Development," *Modeling and Simulation Support for System of Systems Engineering Applications*, 2015.

[15] M. W. Maier, "Architecting principles for systems-of-systems," *Syst. Engineering, 1*, pp. 267-284, 1998.

[16] D. DeLaurentis,"Understanding Transportation as System-of-Systems Design Problem," *43rd AIAA Aerospace Sciences Meeting and Exhibit*, 2008.

[17] A. Graning, S. Rottger, "Innovationsforum Open4Innovation 2012 regional kooperativ-global innovativ", *Technical Reports Technische Universität Dresden*, 2012.

[18] K. Schwaber, and M. Beedle "Agile software development with Scrum" *Vol. 1. Upper Saddle River: Prentice Hall*, 2002.

[19] J. Keznikl, T. Bures, F. Plasil, and M. Kit, "Towards Dependable Emergent Ensembles of Components: The DEECo Component Model. *2012 Joint Working IEEE/IFIP Conference on Software Architecture and European Conference on Software Architecture*, Helsinki, pp. 249-252, 2012.

[20] R. Hennicker and A. Klarl, "Foundations for Ensemble Modelling-The Helena Approach," *Lecture Notes in Computer Science, vol 8373*, Springer Berlin Heidelberg, pp. 359-381, 2014.

[21] S. Malakuti, "Programming with Emergent Gummy Modules," *Trans. Modularity and Composition. Vol 1*, pp. 80-119, 2016.

[22] T. Kühn, M. Leuthäuser, S. Götz, C. Seidl, and U. Aßmann, "A metamodel family for role-based modeling and programming languages," *International Conference on Software Language Engineering*, Springer, Cham., pp. 141-160, 2014.

Carl Mai is a PhD student and research assistant at the Technical University of Dresden. His research focuses on adaptive petri nets and model-driven software development. Mai received his Dipl-Inf. from Technische Universität Dresden.

Dominik Grzelak is a PhD student and research assistant at the Technische Universität Dresden. His research interests include distributed computing applications and bigraphs. Grzelak received a M.Sc. in computer science from BTU Cottbus-Senftenberg.

Mariam Zia is a PhD student and research assistant at the Technische Universität Dresden. Her research focuses on SoS and role oriented programming. Zia received a M.Sc. in computer science from Technische Universität Dresden.

Diana Lemme is a PhD student and research assistant at the Technical University of Dresden. Her interests are on software ecosystems and innovation processes. She received her Dipl-Inf. from Technische Universität Dresden.

Uwe Aßmann is Professor and Dean of the faculty of Computer Science at the Technische Universität Dresden. He leads the Software Technology group. His research interests lie in the area of software engineering, with emphasis on model-driven development, software composition and component-based software.