```
Training on single CPU.
Initializing input data normalization.
|=========================================================================================|
|  Epoch  |  Iteration  |  Time Elapsed  |  Mini-batch  |  Mini-batch  |  Base Learning  |
|         |             |   (hh:mm:ss)   |   Accuracy   |     Loss     |      Rate       |
|=========================================================================================|
|       1 |           1 |       00:00:00 |       38.28% |       1.0837 |          0.0100 |
|       9 |          50 |       00:00:19 |      100.00% |       0.0054 |          0.0100 |
|      17 |         100 |       00:00:38 |      100.00% |       0.0027 |          0.0100 |
|      25 |         150 |       00:00:58 |      100.00% |       0.0009 |          0.0100 |
|      30 |         180 |       00:01:09 |      100.00% |       0.0008 |          0.0100 |
|=========================================================================================|
```

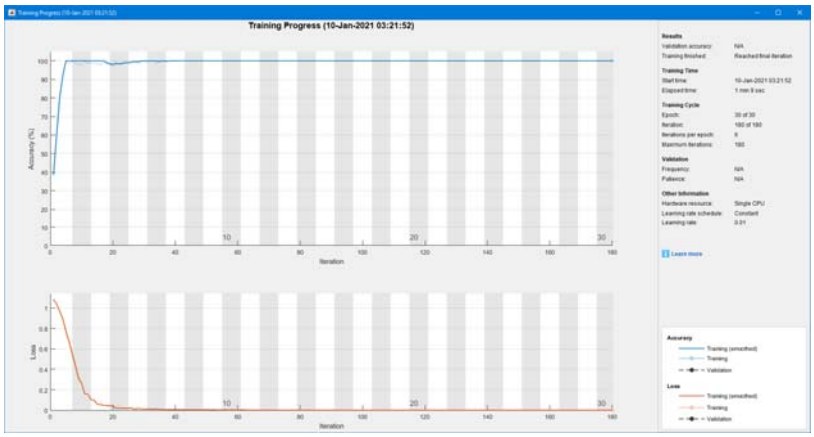

Рис. 3. Результаты применения сверточных нейронных сетей

Применение нейромоделирования в диагностике систем обеспечения энергоснабжения позволит контролировать состояние трансформаторов в режиме реального времени, не выводя трансформатор из работы, что предоставляет дополнительные возможности в обеспечении низкого уровня безаварийности и соблюдения режимов бесперебойного электроснабжения, сопровождающихся, как правило, значительными экономическими и экологическими издержками или реальным ущербом для потребителей. При этом затраты на внедрение данной технологии нейромоделирования относительно невелики (например, применение одноплатных компьютеров), а эффективность от применения будет существенной.

FACILITATING SOFTWARE COMPONENT REUSE IN THE DHIS2 PLATFORM ECOSYSTEM

A. Bengtsson

University of Oslo, Norway

Scientific advisors: P. Nielsen, M. Li

There is an increase in the development of generic software systems that are developed to serve multiple organizations and used for different purposes. Some examples of generic software are the Microsoft Office 365 suite, Adobe Photoshop, and DHIS2 – a generic web-based Health Management Information System (HMIS) platform, which is the focus of my study. The purpose of HMIS is to routinely manage and generate health information data that would serve as a basis for management decisions to foster improvements in health service provision. DHIS2 is currently the world's largest health management information system, and it is in use by 73 low- and middle-income countries [1]. HISP is a global network that develops and supports the DHIS2 platform. The network is comprised of HISP groups – organizations based in developing countries, providing support to DHIS2.

One way of contributing to the DHIS2 is through the development of additional modules or web applications on top of generic software, which are extensions of the user interface and the functionality in the case of the DHIS2. Building these web applications from scratch can be time-consuming. It is also not resource-efficient if different HISP groups are developing similar modules. One way of addressing this problem is by building software from existing components using a component-based software engineering (CBSE) approach. Software reuse is the central focus of this approach, and the main idea is a development of applications by reusing configurable software components. However, there are several barriers to component reuse, and one of them is the poor cataloging and distribution of reusable software components. This has a considerable impact on component discovery and makes the process of component reuse less effective.

This study aimed at attaining two goals – a practical one and a theoretical one. The practical goal was to conduct engaged research with the HISP community exploring the possibility of creating a component repository that facilitates component reuse in web-based application development. Therefore, a primary focus of the work in this project was the design, implementation, and evaluation of such a repository in collaboration with the DHIS2 core team and members of HISP groups involved in application development work. The theoretical goal of my research was to identify and establish a set of theoretically and empirically grounded design principles for implementing a component repository that facilitates component reuse in a software platform ecosystem. These design principles are a theoretical contribution to the knowledge base on how component repositories can be designed and developed. They are prescriptive in nature and are meant to give value beyond local practice. Given the above, the paper addresses the following research question: *What are the essential design principles for implementing a component repository that facilitates component reuse in a software platform ecosystem?*

Guided by the nature of the research problem, this study was situated within the pragmatic research paradigm. Software reuse is a socio-technical activity, as it clearly has some social aspects in addition to technical aspects. For example, the specification of metadata for a component is highly technical, as it must be exactly specified and machine-readable to ensure proper component cataloging in a component repository. There are, however, also highly social aspects, for example, the developers' attitude towards software reuse, which could be influenced by social factors such as trust and understanding. Given this, I could see a clear application for the pragmatic research paradigm that advocates embracing the approach that gives most utility in the circumstances. I have chosen Design Science Research (DSR) as an overarching methodology to guide the design and development of the component repository. Contrary to other methodologies that have a goal of understanding reality, DSR is a problem-solving approach with the aim of changing situations to a better or more desirable state. DSR offers a cyclical process model that includes such activities as problem identification, definitions of the objectives, design and development of the artifact, demonstration, evaluation of the artifact, and, finally, communication of the conducted research.

To identify the problem, my team and I have conducted focus group discussions with the members of the DHIS2 core team at the University of Oslo. Additionally, we conducted a set of interviews with developers in HISP East Africa. Our goal was to learn about application development practices, motivation for software reuse, current and prospective reuse practices, impediments for reuse, tooling, and collaboration in co-located teams (i. e., within one HISP group) and geographically dispersed teams (i. e., between different HISP groups). Analysis of the gathered data has shown that there is diversity in

technology, tooling, and software reuse practices. One of the practices discerned during the interviews is software reuse through the copying of code, and while it can be seen as code reuse with minimal effort, there is a number of issues pertaining to such a practice. The code might have bugs and security vulnerabilities, and copy-pasting would mean introducing these issues in different applications. Another practice we have encountered during the interviews was CBSE, which involved the creation of reusable components which were stored on Github and as NPM packages in NPM Registry. This has made us question whether there is, in fact, a need for the development of a component repository given that NPM Registry is already in place. We have decided not to develop a completely new component repository but rather cultivate the installed base by reusing and extending the existing infrastructure. The main goal of our solution would be to support and improve the existing CBSE approach by addressing some of the challenges we have encountered with existing technologies, services, and tools.

As a practical contribution to this study, a component repository called the DHIS2 Shared Component Platform (SCP) was developed. The component repository consists of a website (built using React) that aggregates reusable components and two other modules that support the process of component certification: a command-line interface (CLI, written in TypeScript) to provide functionality for local certification, and a GitHub repository with an automated certification workflow using GitHub Actions workflow that invokes the command line interface. During the development phase, SCP was evaluated by the DHIS2 core team members with the intention to improve SCP's functionality and develop a higher quality artifact. SCP aims to increase the productivity of DHIS2 developers and shorten the development life cycle. Component certification improves component trustworthiness and thus, improves the quality and reliability of the developed web applications. The established set of design principles, a theoretical contribution of this study, attempts to address the challenging aspects of the implementation of a component repository that facilitates component reuse in a software platform ecosystem. These principles can serve as guidance for the construction of a similar artifact.

The first design principle, *Principle of installed base cultivation*, advocates the utilization of the existing infrastructure to increase the likelihood of component repository adoption. The process of design and development should not start from scratch; it must consider the existing infrastructure, e. g., attitude towards software reuse, software reuse practices and process, technology, and tooling. Instead of creating a radical change, one should cultivate the installed base towards better practice.

The second design principle, *Principle of component trustworthiness*, advocates the implementation of component certification as an integral part of software reuse in order to increase component trustworthiness and make developers more comfortable reusing software. The review of the previous literature on CBSE has shown that component certification is an important aspect of CBSE, and the DHIS2 core team has also expressed the need for certification functionality to promote components with a certain level of quality. When implementing certification, one must take into consideration the level of human discretion in the certification process. A certification process with a low level of human discretion can be automated and more accurate, while a manual process with a high level of human discretion can be subjective, time-consuming, and less accurate.

The third design principle, *Principle of balanced certification*, emphasizes the importance of governance balance in a software platform ecosystem when choosing individuals for the role of component certifiers. If the DHIS2 core team, as platform owners, takes this responsibility, it might have a significant impact on the autonomy

of third-party developers. If the team of certifiers is entirely comprised of third-party developers, it brings more egalitarianism to the platform ecosystem but reduces the platform owners' control over the platform.

The fourth design principle, *Principle of component granularity,* advocates providing the right level of component granularity in a component repository as it has a high impact on a component's discoverability and usability. NPM packages have an arbitrary level of component granularity, i. e., some packages might contain only one reusable component, while some packages act as component libraries and contain multiple components. This has a negative effect on the component discovery, as NPM registry does not search for components within packages. SCP addresses this challenge by indexing reusable components inside the packages and thus, improves their discoverability.

The fifth design principle, *Principle of orthogonality*, guides the researchers and developers in their work on architecting and implementing a component repository. A component repository is part of the component-management process and must provide support for other processes such as component publishing, component acquisition, and certification. Adopting a modular approach with the aim of building an orthogonal system, i. e., highly cohesive and loosely coupled, can reduce the complexity of the system and increase its maintainability. A high degree of orthogonality has a significant impact on the system's evolution, as each of the modules can evolve in a decentralized way (i. e., the modules can be modified, updated, and removed independently from each other).

References

1. About DHIS2. – 2021. – Access mode: https://dhis2.org/About/. – Accessed: 14.02.2021.

КОМПЬЮТЕРНОЕ МОДЕЛИРОВАНИЕ ФРИКЦИОННОГО УЗЛА

С. Г. Инагамов

Учреждение образования «Белорусский государственный университет транспорта», г. Гомель

Научный руководитель Э. И. Галай

Мощность и эффективность тормозов определяются по трем основным показателям: скорость поезда, его вес и длина тормозного пути. Эти факторы являются доминирующими при выборе тормозных систем [1]. В настоящее время на большинстве вагонов применяются композиционные колодки из материала ТИИР-300, ТИИР-303, ТИИР-308, обладающие высокой износостойкостью – они в 3–3,5 раза долговечнее стандартных чугунных [2]. Чугунные колодки быстро изнашиваются, что требует большого объема работ по замене и регулировке рычажных передач [4], [5]. Коэффициент теплопроводности композиционного материала составляет от 0,7–0,93 до 1–4 Вт/ (м · К). Для сравнения, если твердость тормозной колодки составляет 2400 МПа, то коэффициент теплопроводности чугуна типа *Р* составляет 45 Вт/(м · К) [3], [6]. Поэтому чугунный материал применяется чаще, чем композиционный материал. Коэффициент трения композиционных колодок меньше зависит от скорости. В грузовых вагонах железных дорог СНГ, а также в США применяется одностороннее нажатие тормозных колодок. В Западной Европе на грузовых и пассажирских вагонах используются тормоза с двухсторонним нажатием тормозных колодок на колесо. Колодочный тормоз с односторонним нажатием на колеса обеспечивает меньшую тормозную эффективность по сравнению с двухсторонним нажатием колодок на колесо. К недостаткам одностороннего