Journal of Imaging Informatics in Medicine (2024) 37:945–951
https://doi.org/10.1007/s10278-024-01034-9

# Streamlining Radiology Workflows Through the Development and Deployment of Automated Microservices

**Anton S. Becker**[1,2] · **Joshua Chaim**[1] · **Hebert Alberto Vargas**[1,2]

Received: 23 October 2023 / Revised: 17 January 2024 / Accepted: 31 January 2024 / Published online: 13 February 2024


## Abstract

Microservices are a software development approach where an application is structured as a collection of loosely coupled, independently deployable services, each focusing on executing a specific purpose. The development of microservices could have a significant impact on radiology workflows, allowing routine tasks to be automated and improving the efficiency and accuracy of radiologic tasks. This technical report describes the development of several microservices that have been successfully deployed in a tertiary cancer center, resulting in substantial time savings for radiologists and other staff involved in radiology workflows. These microservices include the automatic generation of shift emails, notifying administrative staff and faculty about fellows on rotation, notifying referring physicians about outside examinations, and populating report templates with information from PACS and RIS. The report outlines the common thought process behind developing these microservices, including identifying a problem, connecting various APIs, collecting data in a database, writing a prototype and deploying it, gathering feedback and refining the service, putting it in production, and identifying staff who are in charge of maintaining the service. The report concludes by discussing the benefits and challenges of microservices in radiology workflows, highlighting the importance of multidisciplinary collaboration, interoperability, security, and privacy.

**Keywords** Workflow · Efficiency · Safety · Productivity · Software development

## Background

Radiologists are responsible for interpreting medical images and providing diagnoses that guide patient care. The advent of digital imaging has led to significant improvements in the speed and accuracy of radiological diagnoses [1], but it has also increased the complexity of radiology workflows [2]. Radiologists now navigate multiple software applications and communication channels to complete their work, which can be time-consuming and error-prone [3]. In addition, radiologists must master efficient and accurate communication with referring physician teams, administrative staff, and trainees to ensure the smooth functioning of the department. Some of these non-interpretive tasks are challenging due to their repetitive nature and may contribute to mental fatigue and, in turn, increase the odds for errors to occur [4].

McGrath et al. identify informatics solutions as a key factor in increasing productivity and efficiency in radiology departments [5]. Doshi et al. succinctly summarize the problem in the following sentence: "The demands placed on radiologists are continuing to increase such that informatics solutions are no longer a luxury but rather a necessity for a viable practice" [6].

One such innovation, the deployment of automated microservices, has emerged as a promising avenue to address the multifaceted challenges encountered in daily radiology operations. Microservices are a software development approach where an application is structured as a collection of loosely coupled, independently deployable services, each focusing on executing a specific business capability [7]. The primary advantage of microservices lies in their ease of implementation and agility in addressing specific workflow optimizations. Microservices can simplify radiology workflows by automating repetitive tasks, integrating disparate software applications, and improving communication between staff members.

✉ Anton S. Becker
Anton.Becker@nyulangone.org

Joshua Chaim
chaimj@mskcc.org

Hebert Alberto Vargas
Alberto.Vargas@nyulangone.org

[1] Department of Radiology, Body Imaging Service, Memorial Sloan Kettering Cancer Center, 1275 York Avenue, New York, NY 10065, USA

[2] Department of Radiology, Oncologic Imaging Service, NYU Langone, New York, NY, USA

This manuscript outlines the development and deployment of several microservices within a radiology department in a tertiary cancer center, assessing their potential to streamline radiology workflows. Each microservice targets distinct bottlenecks within the department, be it manual dissemination of information, search for clinical data across multiple software platforms, or manual population of report templates. Although a direct measurement of their impact is challenging, direct radiologist feedback and cautious estimates of time and monetary savings provide an insight into the potential benefits of these informatics tools.

We aim to contribute to the growing body of knowledge in radiological informatics via practical implementation, offering a perspective on how automated microservices can enhance operational efficiency and improve communication, aligning with the broader goal of delivering enhanced patient care.

## Methods

The development of microservices for radiology workflows requires a multidisciplinary approach, combining expertise in both radiology and informatics. Our team included radiologists with direct end user experience as well as dedicated informaticists. Moreover, close collaboration with the respective application specialists of the radiology IT department was indispensable. The following steps outline a prototyping approach [8] that can be used to develop microservices for radiology departments, as also illustrated in Fig. 1. At the end of each step, a specific example from the development of one of the microservices (RIS – PACS – content delivery) is given in *italics*:

1. Identify a problem: The first step in developing a microservice is to identify a problem that can be solved using automation. This can be done through discussions with trainees, radiologists, administrators, and other staff involved in radiology workflows.

*Example: The verbatim dictation of information easily available elsewhere in electronic form is a long-standing pain point of the radiological workflow. For RIS-PACS content delivery, this problem was well known and its solvability depends mainly on available APIs.*

2. Identify Application Programming Interfaces (APIs): Once a problem has been identified, the next step is to identify the APIs that can be leveraged to automate the task. APIs provide a way to connect different systems and services, allowing data to be transferred between them.

   *Example: RIS database access was readily available. PACS was browser-based, hence, HTTP offered a programmable interface to various PACS functions that were reverse engineered* via *the browser developer tab. Lastly, the dictation software offered a dedicated HTTP API with a comprehensive manual by the vendor.*

3. Collect/Mirror data in a SQL database. The data collected from various sources are stored in a SQL database. For instance, if the problem pertains to timely reporting, data from the examination (completion timestamp, etc.) is pulled from RIS, and the radiology roster from the workforce management software. Though this step is optional, data amalgamation and centralized storage is highly recommended. It not only allows data to be easily accessed and manipulated and for the same data to easily be recycled for different microservices. Lastly, this also shifts some load from APIs to the database, which is well-suited to deal with a large volume of queries.

*Example: The RIS database had been mirrored on a dedicated database server for operational purposes and to reduce load on RIS. On the same server, a table was added to log all cases, where content had already been transferred from RIS or PACS to the dictation software.*

4. Write a prototype and deploy it: After the data has been collected, a prototype of the microservice is developed. This is ideally done using high-level programming languages such as Julia, Python, or R, but this choice depends on the available expertise. The code is deployed on an internal server ensuring data integrity and safety.

*Example: In the beginning, it needs to be ascertained that the microservice is able to establish a connection and log into all the systems. Then, typically a single case is taken as an example, and the authenticity of the data is verified in parallel in the GUI of the respective clinical application. Once it works for a single case, it can gradually be expanded to all eligible cases.*

5. Gather feedback and refine the service: Once the prototype is deployed, feedback is collected from a small group of pilot users (3–5 users, i.e., ~ 5% of total end users) [9] to identify any issues or areas for improvement. This feedback is used to refine the service and make it more efficient and user-friendly. This step may be repeated in an iterative fashion, until the desired result is achieved [8].

*Example: First, two users were given access to a macro that contained the automatically populated information from the script. These users then tested the new function at their convenience. In a next step, the newly autopopulated fields were inserted into a small number of our templates, affecting only one different subsection of our department.*

6. Deploy it in production: After refining the miniservice, it is transitioned into a production environment. This step involves deploying the miniservice on an expanded scale, ensuring it can handle increased load, maintain consistent performance (scalability), and operate without failure under given conditions (reliability). Techniques to assess reliability and scalability include

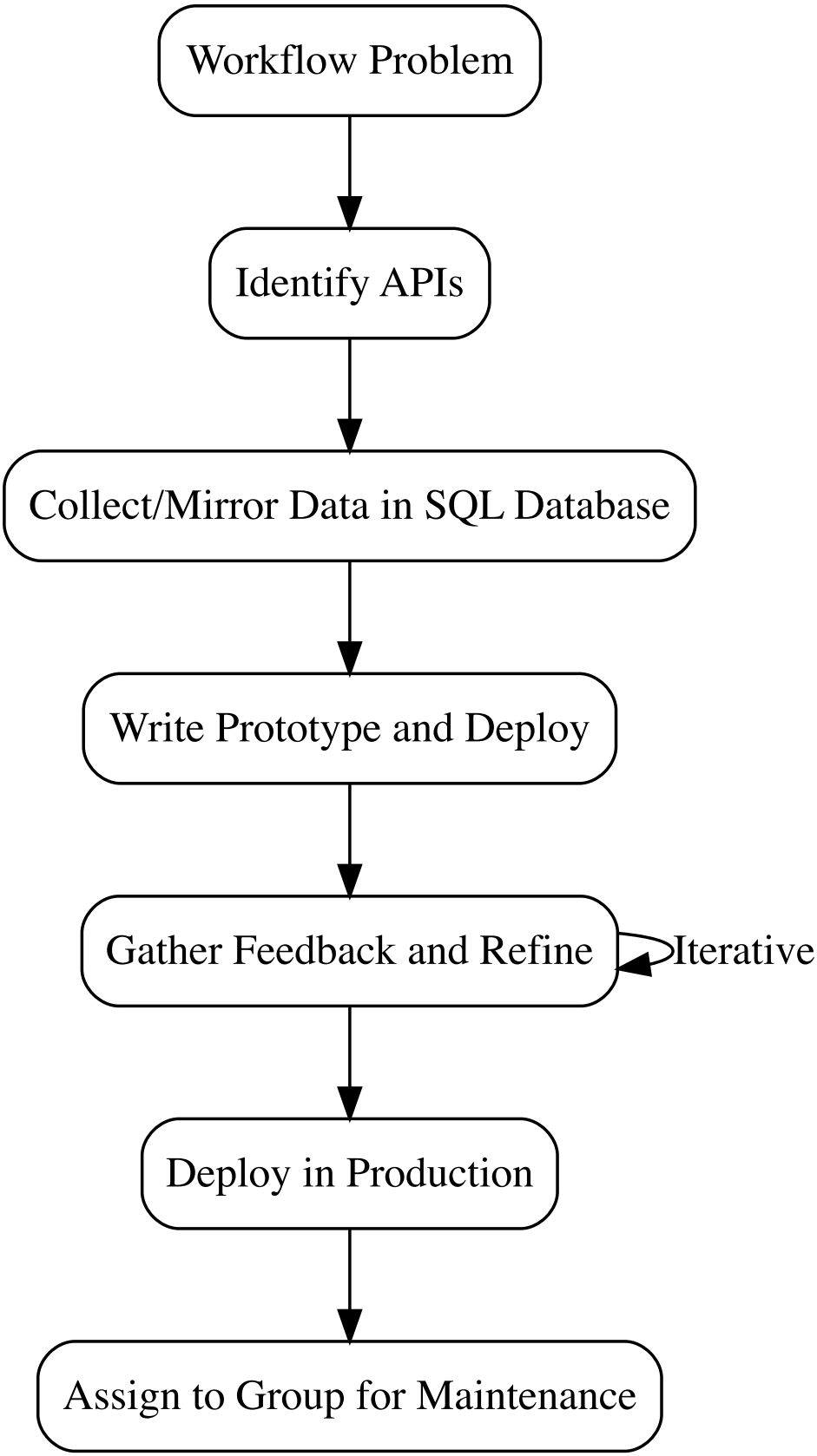


**Fig. 1** Flowchart depicting process of the design and deployment process of radiological microservices

performance testing, stress testing, and load testing. For reliability and fault tolerance, failover and recovery mechanisms should be in place [10]. If a microservice performs a critical task, it may be advisable to deploy multiple instances in a redundant fashion. The deployment step may include bundling certain functions into a library and/or containerization of the application, which eases distribution and use in future microservices [11].

*Example: In this step we determined that the dictation system would not handle API requests affecting more than 2000 orders. Hence, we built a hard stop into the code if this case should occur. Another possibility would have been splitting up the cases into chunks, but we observed in production that there should never be more than 50–100 cases processed at a time, so it was considered reasonable to trigger an investigation at this high limit. Also, many of the functions originally contained in the script itself were integrated into the departmental R package for clarity and reusability. Lastly, all eligible report templates were modified to include the autopopulated fields, after information of the radiologists at a faculty meeting.*

7. Identify staff who are in charge of maintaining the service: Finally, staff who are responsible for maintaining the microservices are identified. This includes both technical staff who are responsible for the server and software maintenance, as well as clinical staff who can give feedback if any changes to the workflow occur.

Using this approach, several microservices were developed to automate routine tasks in radiology workflows at a tertiary cancer center. We mainly relied on the programming language R and the proprietary platform Posit Connect© for rapid deployment of the microservices from the integrated development platform RStudio© or Posit Workbench© (all products of Posit Inc., Boston, MA, USA). When configured correctly (i.e., with appropriate access restrictions in place) and hosted on an internal server, these products are suitable to house sensitive personal health information.

It is advisable to create a detailed plan including affected processes and impact on workflow efficiency after steps 1 or 2 and obtain executive approval. This ensures alignment with the broader organizational goals and secures necessary support for ongoing maintenance.

## Results

In this section, several exemplary microservices that were successfully developed and deployed at our department are listed. While their impact has not been directly measured, feedback has been overwhelmingly positive, and cautious estimates of time and monetary savings for some of them are provided. The type of API protocol is given in square brackets, if applicable.

### Email Notifications

Before the microservices were implemented, radiologists had to spend time searching for information and manually sending emails. For example, the "Doc of the Day" (DoD) radiologist (responsible for ad-hoc protocols, contrast reactions, etc.) had to search the clinical roster software QGenda (QGenda LLC, Atlanta, GA, USA) to find out who else was working that day (afternoon and late shift), then search old emails for the distribution list of the site they were working at, and manually send an email to notify their colleagues and local staff. Similarly, one of the radiology trainees on rotation had to manually look up who else was on that rotation that day and then send an email to the central scheduling team. These tasks were time-consuming and sometimes forgotten, leading to delays in communication thus representing latent errors which may potentially lead to patient harm.

The deployed microservice automated these tasks by querying QGenda API [HTTP] and automatically sending a formatted email [IMAP] to the relevant staff members at the start of each shift. This intuitively reduced the time and effort required to send emails, improved communication among staff, and reduced the likelihood of tasks being forgotten. Furthermore, the timesaving for the DoD microservice can be cautiously estimated with the following "back-of-the-envelope-calculation": The manual process took around 5–10 min, for five radiologists at different sites ($5 \times 5 = 25$ min) every morning (260 workdays per year $\times$ 25 min $= 6500$ min), meaning $\approx$ 108 h of radiologist time saved each year. Similar calculations could be made for other departments adapted to their specific operations and accounting for the number of radiologists and sites involved in these processes.

In addition, a microservice that scans the RIS [SQL] for conversions of outside examinations and automatically sends emails to the ordering provider to notify them of the status was developed. This reduced the amount of time the film librarian needs for these conversions by eliminating the need to spend notifying referring physicians' offices and reduced the likelihood of notifications being forgotten or delayed.

### RIS—PACS—Dictation Software Content Delivery

One of the "pain points" of the radiological workflow is the manual search for relevant technical and clinical information in RIS and PACS systems and manually dictating it into report templates. This process is time-consuming and prone to errors [3].

To address this issue, microservices that automatically extract relevant information from the RIS [SQL] and PACS [HTTP] systems and populate it into report templates in the dictation software (PowerScribe, Nuance Inc.) were developed. For example, a microservice that automatically extracts oral contrast information from the technologist's note in PACS and populates it into the report template eliminates the need for radiologists to manually search for this information.

Another microservice was developed that applies a regular expression (regex) engine to format the order history in reports according to departmental standardized reporting guidelines. Previously, all content in RIS orders was directly populated in the "clinical indication" field of the report, including a lot of non-relevant content, making them difficult to read and potentially leading to errors. With the new microservice, ~ 250 handcrafted regex rules were applied to the order history to remove "electronic garbage," expand abbreviations, and comply with standardized reporting guidelines, resulting in reports that are easier to read from the first instance of opening the report. Here is a simple example of such an automatic order alteration:

> Old format: [none indicated;Ordered by Doe/John, MD] [Prostate CA; (CS: CAT NA). EOD].
>
> New format: [Prostate cancer. Evaluate extent of disease.] […].

In this case, since no useful information was found in the first field, the information was discarded. From the old 2nd field, the abbreviations CA and EOD were expanded to "cancer" and "'Evaluate extent of disease," respectively. Note the new 2nd field is left empty for the radiologist to dictate additional information, e.g., obtained through review of patient chart and/or prior examination.

### Displaying Information from Multiple Sources

One example of this are microservices underlying an interactive online app that query information from the order system [DB2], RIS [SQL], PACS [HTTP], and automatic exam assignment system [SQL], merge them, and display them in one coherent audit trail. This app and associated microservices eliminates the need for radiologists and IT/administrative staff to manually search for information across multiple systems, improving the efficiency of the process.

Another example is a web page that displays which faculty are on a clinical rotation on a given day and what their subspecialties are. This webpage is essential in a subspecialized workflow, where different classes of examinations need to be interpreted by different groups of radiologists. Before the implementation of this microservice, radiologists and trainees staff had to manually gather this information from the day's QGenda plan and tables on the intranet in Word/PDF format which was time-consuming due to visual clutter and complexity of both of these systems and prone to errors (e.g., wrong date selected in the shift plan or outdated information in the PDF). With the new microservice, this information was pulled from two distinct data sources [HTTP API] into a database and displayed in a clear and concise fashion, as illustrated in Fig. 2. The webpage can be launched directly from PACS and is updated hourly, ensuring that short-term changes (e.g., someone calling out sick or covering another shift) are accurately reflected.

The source code for all the described microservices is available at https://github.com/ASBecker/radiology_workflow_optimization.

## Discussion

The development of microservices for radiology workflows has a positive impact on the efficiency and accuracy of radiology operations. The microservices developed in this technical report have been successfully deployed at a tertiary cancer center and were very well received by radiologists and other staff involved in radiology workflows.

Automating routine tasks that do not require a radiologists' expertise, such as writing emails and verbatim dictation of technologists' PACS notes, allows radiologists to focus on the actual diagnostic task at hand and reduces "friction" in the workflow.

Furthermore, the development of these microservices highlighted the importance of multidisciplinary collaboration between radiologists and informaticists. Combining expertise in both fields allowed for the identification of problems that can be solved through automation and the development of solutions that are both technically sound and practical for use in a clinical setting.

It should be noted that APIs are a *sine qua non* for the development of such microservices. When purchasing radiology software, evaluating its API regarding richness of features and ease of access of APIs should be a major component of the decision-making process.

Improving the radiological workflow and reducing errors [12] through means of tailored software is not a novel approach. For example, Doshi et al. describe several tasks, some of which are similar to the ones described in the present paper, for example, auto-population of results in reporting templates, which they solved via means of custom software [6]. The main difference of the approach presented is that the software is developed in-house and all hosted on a single coherent platform, which makes it vendor-agnostic and thus less affected by, e.g., changes in contracts or institutional digital strategy. This approach may become more commonplace as Imaging Informatics Fellowships are continuously improving and on the rise [13].

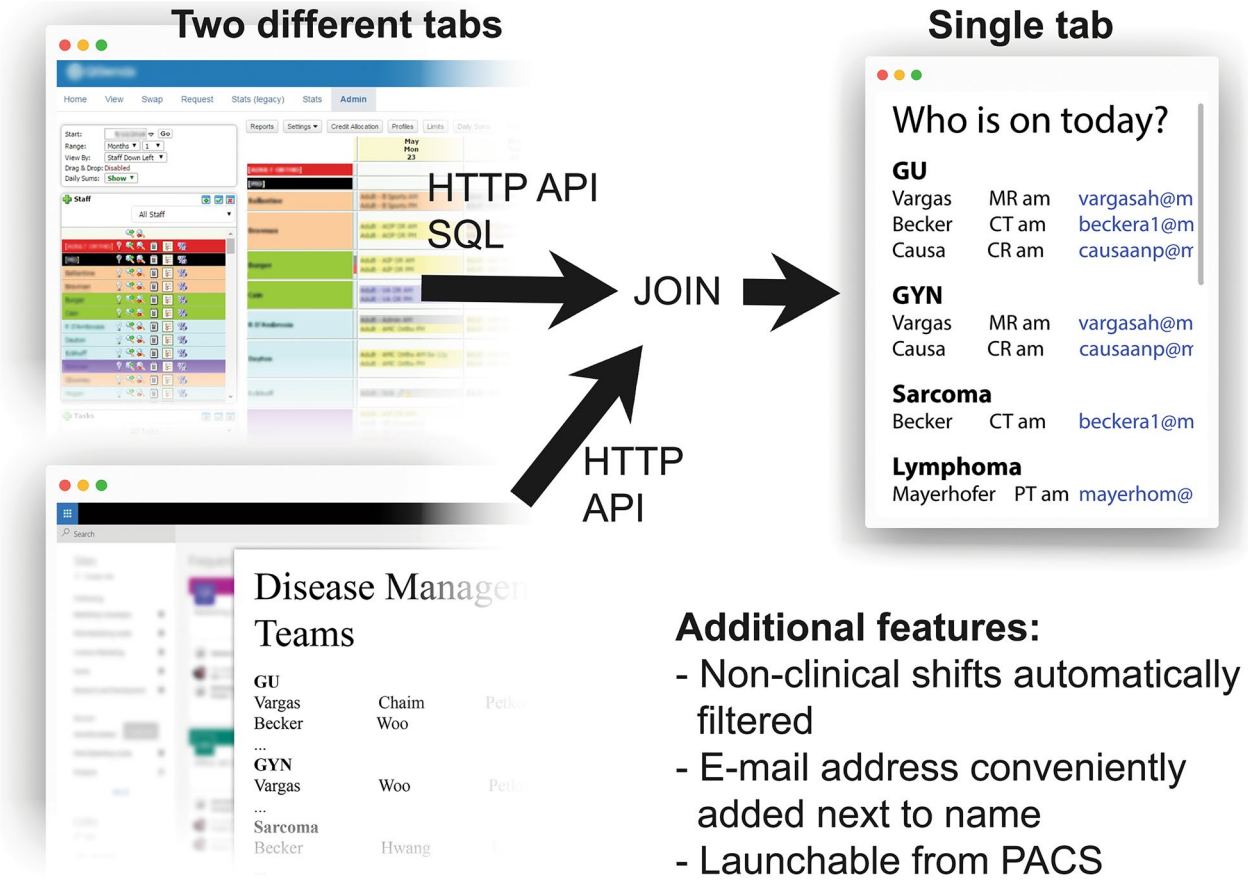


**Fig. 2** Illustration of a microservice for facilitated lookup of subspecialized radiologists. Information is needed from the online clinical roster planning software and from a document on the intranet. For this relatively simple task, both tools are too complex and have ample visual clutter. The microservice solves this by unifying the information into a database and displaying it on a dedicated, up-to-date internal webpage

Despite the clear benefits of microservices in radiology workflows, there are still challenges that need to be addressed. One is the need for interoperability between different systems and services. As radiology departments continue to adopt new technologies, it will be important to ensure that these technologies offer rich APIs so they can communicate with each other, allowing data to be shared seamlessly. Radiology informatics plays a key role to ensure standards and interoperability in this regard [14].

Another challenge is the need to ensure that the microservices developed are secure and comply with all applicable privacy regulations. As radiology departments continue to handle large amounts of sensitive patient data, it will be essential to ensure that microservices are designed with security and privacy in mind. For example, all of the microservices described in this paper were deployed on a single local server with enterprise authentication and user-based access restrictions in place. However, if microservices were deployed on different servers, secure communication must be ensured.

In addition, the microservices presented in this report largely follow a "single responsibility principle," where each service is tailored for one single task at a specific institution. This, by design, prevents the software from being more broadly applicable to distribute to other institutions without major modifications.

Lastly, the integration of machine learning or artificial intelligence (AI) into microservices will pose its own set of challenges, a thorough discussion of which is beyond the scope of this effort. In the literature, examples of natural language processing pipelines for automation of protocoling [15] or computer vision models to prepopulate reports for simple conventional nuclear medicine examinations (lymphoscintigraphy) [16] have already been proposed for AI-assisted workflow optimization.

## Conclusion

The development of microservices allowed for small but repetitive and time-consuming routine tasks to be automated. A conservative extrapolation suggests non-negligible time and thus cost savings for the institution. By combining expertise in radiology and informatics, multidisciplinary teams can identify problems that can be solved through automation and develop solutions that are both technically sound and practical for use in a clinical setting.

**Acknowledgements** The authors thank the Radiology IT staff and informatics team for their commitment and support.

**Author Contribution** All authors contributed to the study conception and design. All authors were involved in designing and refining the miniservices presented in this paper. The first draft of the manuscript was written by ASB, and all authors commented on previous versions of the manuscript.

**Funding** The research of this department is in part funded by the NIH/NCI Cancer Center Support Grant P30 CA008748.

**Data Availability** All relevant data are contained within the manuscript.

## Declarations

**Ethics Approval** Not applicable.

**Consent to Participate** Not applicable.

**Consent for Publication** All authors read and approved the final manuscript. On behalf of all authors, the corresponding author states that there is no conflict of interest.

**Competing Interests** The authors declare no competing interests.

## References

1. Lepanto L, Paré G, Aubry D, Robillard P, Lesage J. Impact of PACS on dictation turnaround time and productivity. J Digit Imaging. 2006 Mar;19(1):92–7.
2. Reiner BI, Siegel EL, Siddiqui K. Evolution of the digital revolution: a radiologist perspective. J Digit Imaging. 2003 Dec;16(4):324–30.
3. Chang CA, Strahan R, Jolley D. Non-clinical errors using voice recognition dictation software for radiology reports: a retrospective audit. J Digit Imaging. 2011 Aug;24(4):724–8.
4. Degnan AJ, Ghobadi EH, Hardy P, Krupinski E, Scali EP, Stratchko L, et al. Perceptual and Interpretive Error in Diagnostic Radiology-Causes and Potential Solutions. Acad Radiol. 2019 Jun;26(6):833–45.
5. McGrath AL, Dodelzon K, Awan OA, Said N, Bhargava P. Optimizing radiologist productivity and efficiency: Work smarter, not harder. Eur J Radiol. 2022 Oct;155:110131.
6. Doshi AM, Moore WH, Kim DC, Rosenkrantz AB, Fefferman NR, Ostrow DL, et al. Informatics solutions for driving an effective and efficient radiology practice. Radiographics. 2018 Oct;38(6):1810–22.
7. Dragoni N, Giallorenzo S, Lafuente AL, Mazzara M, Montesi F, Mustafin R, et al. Microservices: yesterday, today, and tomorrow. In: Mazzara M, Meyer B, editors. Present and ulterior software engineering. Cham: Springer International Publishing; 2017. p. 195–216.
8. Kan SH. Metrics and Models in Software Quality Engineering. 2nd ed. Boston: Addison-Wesley Professional; 2002.
9. Taghizadegan S: Essentials of Lean Six Sigma, Amsterdam, NL: Elsevier Science; 2010.
10. Harrison NB, Avgeriou P, Zdun U. On the impact of fault tolerance tactics on architecture patterns. Proceedings of the 2nd International Workshop on Software Engineering for Resilient Systems. New York, NY, USA: ACM 2010 Apr 15;12–21. https://doi.org/10.1145/2401736.2401738.
11. Bass L, Clements P, Kazman R. Software Architecture in Practice. Subsequent. Boston: Addison-Wesley Professional; 2003.
12. Minn MJ, Zandieh AR, Filice RW. Improving radiology report quality by rapidly notifying radiologist of report errors. J Digit Imaging. 2015 Aug;28(4):492–8.
13. Liao GJ, Nagy PG, Cook TS. The impact of imaging informatics fellowships. J Digit Imaging. 2016 Aug;29(4):438–42.
14. Lindsköld L, Wintell M, Lundberg N. Pitfalls in radiology informatics when deploying an enterprise solution. In: Liu BJ, Boonn WW, editors. Medical Imaging 2010: Advanced PACS-based Imaging Informatics and Therapeutic Applications. SPIE. 2010 Mar 11;76280Q. https://doi.org/10.1117/12.843670.
15. Kalra A, Chakraborty A, Fine B, Reicher J. Machine learning for automation of radiology protocols for quality and efficiency improvement. J Am Coll Radiol. 2020 Sep;17(9):1149–58.
16. Juluru K, Shih H-H, Keshava Murthy KN, Elnajjar P, El-Rowmeim A, Roth C, et al. Integrating Al Algorithms into the Clinical Workflow. Radiol Artif Intell. 2021 Nov;3(6):e210013.