AUTOMATED SYSTEMS OF ENTERPRISING SOFTWARE

THE EVOLUTION OF AUTOMATION

A Thesis Prospectus
In STS 4500
Presented to
The Faculty of the
School of Engineering and Applied Science
University of Virginia
In Partial Fulfillment of the Requirements for the Degree
Bachelor of Science in Computer Science

By
Brandon Bremer

November 1, 2021

On my honor as a University student, I have neither given nor received unauthorized aid on this assignment as defined by the Honor Guidelines for Thesis-Related Assignments.

ADVISORS

Catherine Baritaud, Department of Engineering and Society

Daniel Graham Department of Computer Science

The COVID-19 pandemic paved the way for the automation of business. Labor shortages, inability to keep staff on-site, and demands for increased wages contributed to a 16.7% increase in spending on equipment (Lynch). COVID-19 discriminately impacted low-income workers' employment and their families due to a lack of proper access to resources (Center on Budget and Policy Priorities 2021). However, automation was not constrained solely to low-income employment with low-code development gaining a foothold in the software industry and increasing 23% to a market total of $13.8 billion (Gartner 2021). Hesitancy and fears regarding automation, artificial intelligence, and low-code development are prevalent (Schmelzer 2019). Meanwhile, the role played by society and government in the shaping of these evolving technologies is only beginning to materialize. Questions of labor displacement in response to automation is a contentious subject with economists taking adverse stances on the subject (Lynch 2021). These rapid developments coalesce in governments being urged by the public to take a stronger stance on the issues of automation and artificial intelligence and begin to directly moderate them (Fournier-Tombs 2021).

I will be exploring the overall state of automation with a particular focus on high-skill and algorithmic automation and its effects on the workforce. More specifically research will be focused on the debate regarding whether or not automation will result in a mass exodus of labor, the possibilities of cohabitation by man and machine in the labor force, and the impacts of algorithms and their biases on labor. This research will be coupled with the state-of-the-art overview of low-code development platforms and other forms of automation occurring within the computer science industry. The benefits and concerns surrounding low-code development platforms will be addressed, such as security, capabilities for enterprising software, and questions of efficiency. The primary coupling here will be based on the impacts these forms of

automation have on their associated labor and how society can construct these technologies so they continue to work for us rather than us work for them.

STATE OF AUTOMATION WITHIN THE SOFTWARE DEVELOPMENT INDUSTRY

With low-code development and automated testing on the rise, questions are raised regarding the security, sustainability, and effectiveness of these technologies. It is also worth considering who these technologies benefit and harm, how they are affected by biases, and what issues stand in the way of their acceptance. For the purposes the definition of automation is not strictly the complete replacement of a human by a machine or technology. More often than not a technological innovation will reduce the total number of required laborers by improving productivity, defined as the amount of work produced by a single laborer. A good example of this is the cotton gin which greatly reduced the labor required to separate cotton fibers from seeds. While a human laborer was still required to operate the machinery, it could do the work of and effectively displace multiple workers. In this same sense, low-code development platforms and automated testing environments can be viewed as automation that threatens to displace workers.

My experience with low-code development has been primarily developing large-scale software applications with the platform Mendix. While low-code development systems are not all the same they do generally have specific characteristics that are shared and can be used to group and compare them(Sahay & co. 2020). The shared features of low-code development platforms allow me to use Mendix as a representative example of these features. Low-code development platforms can be broken down into a few key components for how they model the user interface, the data, and they handle the data.

The first component of any low code development platform is the model. The user interface is represented and edited with a direct representation of the interface as is shown in Figure 1.

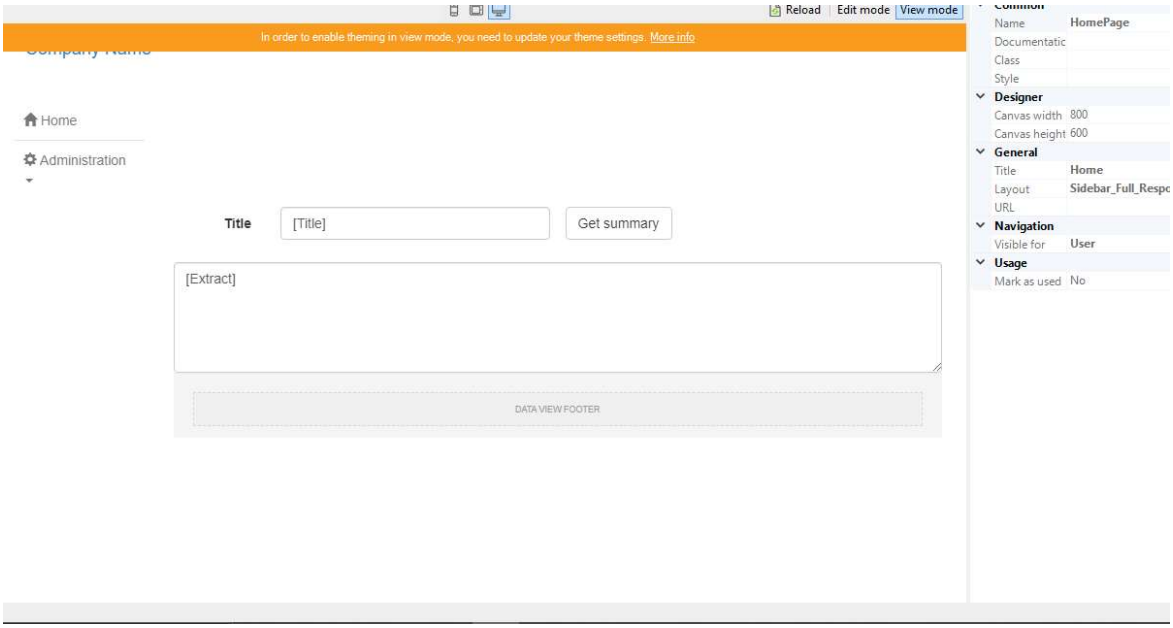

Figure 1Mendix UI Modeler From Mendix 7.23.19

The developer manipulates this system by adding widgets that take in data from the data modeler and can then display it based on which widget is selected. The widgets added by the developer would then appear on a user's screen in the exact same positions. The data itself is represented

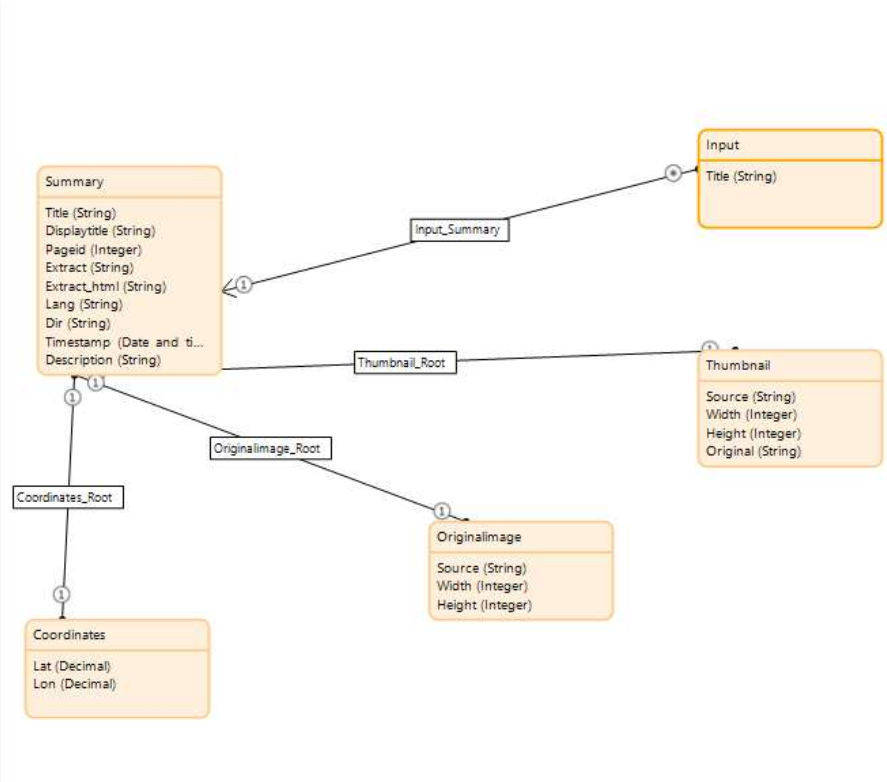

Figure 2Mendix Entity Modeler From Mendix 7.23.19

by a Domain modeling system such as Figure 2. The domain modeler consists of the pieces typically existing in a database, entities, their attributes, and their associations to other entities.

The data in low-code development platforms is manipulated via microflows/dataflows, such as the ones in Figure 3 which take in one or more entities and then make changes to them based on

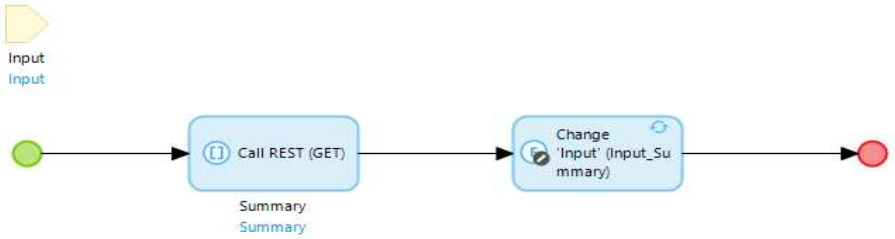

Figure 3Microflow From Mendix 7.23.19

conditionals or other factors set by the developer. Data can be queried from the model and then changed, and these changes will be dynamically reflected by the UI. Additionally, there is generally some sort of source control offered by the low-code development platform which eliminates the need for external source control such as git.

The benefits and costs of low code software development are still in debate and are subjects of research at this time. Researchers at Chulalongkorn University in Thailand performed a case study on the modeling software Odoo to measure the effectiveness of low code development platforms. The low-code version of Odoo was tested with, non-Odoo users, Odoo users, and Odoo developers and compared with traditional Odoo. Comparisons were performed between each user type and each of the two development models and the research concluded that the new low-code model led to significant improvements in performance and efficiency (Pichidtienthum & co. 2021). Complications with low-code software development can be analyzed through trends in stack overflow, a common forum for questions in computer science,

posts regarding low-code software development as researchers in Bangladesh did. Through this analysis, inferences can be made into which pieces of low-code development platforms are complex or difficult for developers to understand or utilize and which translate well from a traditional software engineering background. The research concluded that some topics such as databases presented little difficulty for individuals transitioning from a traditional computer science background to low-code development. Meanwhile, other subjects such as API integration and Dynamic Event handling did not translate as well and a lack of solid tutorials and support for these topics made them difficult for low-code developers (Abdullah & co. 2021).

Aside from low-code software development recent improvements in the automation of testing frameworks are an area of inquiry. Software testing is the practice of thoroughly testing each component of the software in a variety of ways, generally it is performed by testing different actions and inputs and checking that the outcomes are as intended. Software testing is often not performed by hand in modern software development. Instead, software testing frameworks have been developed to generate a software testing suite that effectively tests any given software (Winkler & co. 2010). This level of automation is frequent and greatly reduces the amount of time and effort required by developers to test their code thoroughly. Taking things one step further the generation of automated testing suites is being automated as well. Through a combination of machine learning and mutation-based testing, it is now possible to completely automate the software testing process (Shen 2020).

IMPACTS OF AUTOMATION WITHIN SOFTWARE DEVELOPMENT

Similar to how the labor shortage during 2019-2021 encouraged businesses to invest in automated technologies the labor shortage in software development led to increased automation. Whether these innovations will in the long term deplete the number of jobs available to aspiring

software developers or match the demands of a voracious market remains to be seen. Those most likely to be replaced within the context of automation are individuals performing invisible labor, labor which is primarily unacknowledged and unrewarded (Scroggins & Pasquetto 2020). This form of labor is frequently wiped out by the effects of automation within data-intensive science (Scroggins & Pasquetto 2020). In computer science, roles such as software testers is one role where there was once demand for these individuals which has been supplemented by the effects of automation. Testing teams are quickly becoming a relic of another era and the work they did is done by a developer who writes a testing suite.

It is irresponsible to claim that automation in software development is new or unexpected. Processes required in the development of code have been slowly automated, from the early days where wires had to be arranged by hand, to machine code, to modern-day compilers the arrangement of binary numbers has gradually become abstracted (Jesiek 2006). This abstraction has grown rapidly as the field of computing has advanced, and the separation between computer engineering and computer science has progressed to a point where they are not even coupled (Jesiek 2006). There is a very valid concern that low-code development is a further level of abstraction than traditional programming as entities, modules, and graphics are utilized to make the process of software development more intelligible to humans. This abstraction makes a goal of codesign between those developing hardware and software systems difficult and this lack of codesign can lead to inefficient systems in hardware and software (Jesiek 2006).

THE STATE OF AUTOMATION

Automation has been a feared technology for decades now, despite having existed in various forms for millennia. The pulley made it so one man could do the work of hundreds yet it is only in the context of complete automation that concerns are typically raised. In the wake of

pandemic, thousands are left jobless and machines rise to the front of industries (Lynch 2021). Distrust of corporations in conjunction with advancing technologies coalesce in a great deal of concern regarding the displacement of laborers. While generally automation is presumed to displace low-skill workers by directly replacing them with machines there is a great deal of automation in complex labor and even managerial positions (Acemoglu & Restrepo 2017). Displacement of high-skill-laborers, such as those with four-year university degrees, in turn, affects the low-skill labor market as they forced to compete for their salary against the displaced laborers (Acemoglu & Restrepo 2017). As all of this occurs economists delight that productivity is at a peak as labor in conjunction with automation generates more value than ever before with a minimum number of employees (Lynch 2021).

It is important to remember that we are not yet at a point of labor erasure in our automated capabilities. There are arguments that the end is not nigh and that instead we are approaching an era of human-robot cohabitation in the workplace. In this context studies of human and computer interaction become significantly more relevant. Systems that are automated tend to have a bias against or toward certain types of interaction (Fereidunian & co. 2007) The bias within automated systems is largely correlated with the ethical frameworks of the profession or business designing that system and these can impact further the ways in which computation and human interaction shape one another (Fleischmann & Wallace 2006). Even in AI the human component has not been entirely eliminated as can be seen through the development of micro-work systems (Tubaro & co. 2020). The automation of managerial networks has been shown to impact worker-manager relationships, and its prevalence in gig-work, another form of labor on the rise due to the pandemic, has left workers exploitable (Jarrahi & co. 2021). For all these

reasons there have been calls for governments to take action and reign in artificial intelligence and automation to prevent their potential abuses (Fournier-Tombs 2021).

AN ANALYSIS OF AUTOMATION

Many of the issues regarding the potential of automation stem from issues of who will be consulted in the development of the technology. The development of automated technologies is often thought to be carried out through the model seen in Figure 4

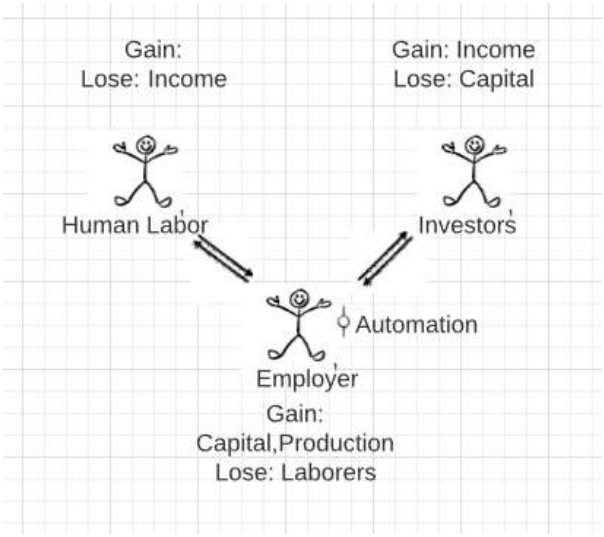

Figure 4Technology and Social Relationship Model For Automation

in which the end-user is considered to be the corporation utilizing the technology. In this model, those most likely to interact with the technology, laborers, tend to have almost no voice in the technology's development. The workers tend to be viewed as replaceable by the technology itself and thus their input is only needed as far as identifying what tasks the machine would have to be able to successfully carry out. Meanwhile, investors could purchase some of this labor capital in order to generate revenue for themselves. This view of automation and the future of labor includes the displacement of humans by machines but is highly beneficial to the employer as the minimize labor cost and to investors as they get a direct stake in the company.

Instead of the previous corporate-centered model of technology and social relationships, I would promote a socially constructed model such as Figure 5. In this model, the various social groups

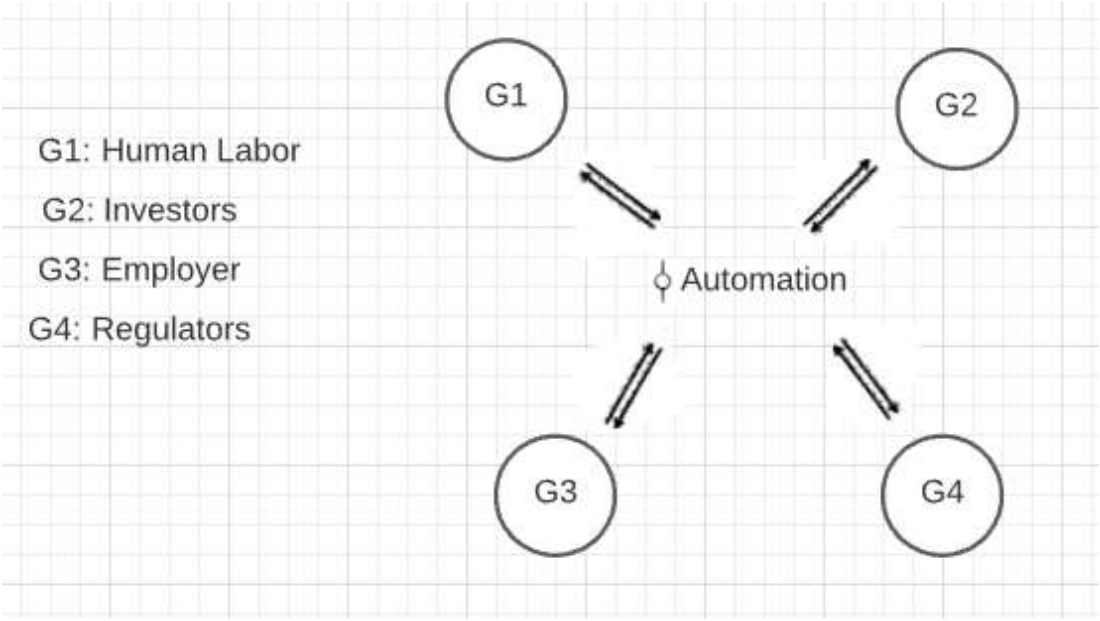

Figure 5 Social Construction of Technology Model for Automation

surrounding the creation and use of automated technology all play a contributing role. The laborers provide to the engineer all of the necessary information on how to make the automated technology best for improving their productivity and in turn receive a technology that best fits their needs rather than replaces them. Investors still get to invest in the labor-capital as they did before but now this labor capital is combined with human capital to produce greater returns on investment. Employers still gain capital and increase productivity but do so without losing their labor force. Finally, the regulators ensure that the automation mechanisms involved and developed never become predatory in nature, this role is especially important in work where there is a great deal of separation of all of the other social groups such as gig-work. The engineer in this instance will responsible for primarily the regulator and laborers' needs rather than the employer and investors whose primary need is revenue. This model is most effective if the

assumption that automation designed with input from the workers to best work alongside them outpaces the traditional model in terms of production.

FEASIBILITY OF APPROACH

The case I am making in favor of automation is not unforeseeable or even unlikely. David Atour an economics professor from the Massachusetts Institute of Technology claims “You don’t fire workers and hire a robot. That happens exactly no times” (Lynch 2021). The case here is that even though productivity is greatly increased with a significant portion of the labor market still unemployed the productivity is not dependent on unemployment. If one laborer can now perform the work of ten it does not necessarily mean that 9 laborers should be fired, the 9 laborers were fired due to the restrictions and unforeseen issues resulting from a global shutdown. What it does mean is that if you now hire back all 10 laborers they are as productive as 100 laborers would have been before. Some businesses are thrilled to increase productivity through the conjunction of human and robot labor. This is exemplified by Amazon who plans to increase their minimum wage to $17/hour, hire more workers, and add more robots to their fulfillment centers in order to maximize their production (Lynch 2021). This is an example of the social construction of technology model of automation being utilized to supplement the labor force with automation rather than displace it. Social construction of technology reaps benefits from including all relevant social groups in the creation of a new technology and that human action determines the shape of technology more than technology shapes society. Which this model exemplifies through addressing often neglected social groups and demonstrating how society can utilize automation rather than being controlled by it.

A CONCLUSIVE COUPLING

Ideally, my case for low-code development platforms and testing frameworks as examples of automation and the demonstration of their potential pitfalls and benefits will be able to relate to my theory of socially constructed automation. Automation in software development is fast-paced and I hope to show in my research that studies indicate that greater production is achievable through human-robot cohabitation and a social construction of automated industry. In my experience low-code software development allows developers to put in more cognitive effort toward problem-solving while avoiding the monotonous portions of coding a system. If this can be evidenced and then generalized to automation as a whole it would strongly support my conclusions. Once research regarding automation is completed, I would like to be able to utilize this example of high-skill labor as a potential example of properly executed socially constructed automation. If I can relate my technical research to my STS research it would be a great boon toward showing the potential benefits of automation and this system of evaluating the construction automated systems.

WORKS CITED

Abdullah Al Alamin MD, Sanjay Malakar, Gias Uddin, Sadia Afroz, Tameem Bin Haider, and Anindya Iqbal. 2021. An empirical study of developer discussions on low-code software development challenges. *2021 IEEE/ACM 18th International Conference on Mining Software Repositories (MSR)* (2021). doi:http://dx.doi.org/10.1109/msr52588.2021.00018

Acemoglu, D., & Restrepo, P. (2017). Low-skill and high-skill automation. *SSRN Electronic Journal*. doi:10.2139/ssrn.3083552

Bremer, B. (2021). *Mendix UI Modeler from Mendix 7.23.19.* [1]. *Prospectus* (Unpublished undergraduate thesis). School of Engineering and Applied Science, University of Virginia. Charlottesville, VA.

Bremer, B. (2021). *Mendix Domain Model from Mendix 7.23.19.* [2]. *Prospectus* (Unpublished undergraduate thesis). School of Engineering and Applied Science, University of Virginia. Charlottesville, VA.

Bremer, B. (2021). *Microflow from Mendix 7.23.19.* [3]. *Prospectus* (Unpublished undergraduate thesis). School of Engineering and Applied Science, University of Virginia. Charlottesville, VA.

Bremer, B. (2021). *Technology and Social Relationships Model for Automation.* [4]. *Prospectus* (Unpublished undergraduate thesis). School of Engineering and Applied Science, University of Virginia. Charlottesville, VA.

Bremer, B. (2021). *Social Construction of Technology Model for Automation.* [5]. *Prospectus* (Unpublished undergraduate thesis). School of Engineering and Applied Science, University of Virginia. Charlottesville, VA.

Tracking the COVID-19 economy's effects on food, housing, and employment hardships. Center on Budget and Policy Priorities. (n.d.). Retrieved October 20, 2021, from https://www.cbpp.org/research/poverty-and-inequality/tracking-the-covid-19-economys-effects-on-food-housing-and.

Fereidunian, A., Lucas, C., Lesani, H., Lehtonen, M., & Nordman, M. (2007). Challenges in implementation of human-automation interaction models. *2007 Mediterranean Conference on Control & Automation*. doi:10.1109/med.2007.4433895

Fleischmann, K. R., & Wallace, W. A. (2006). Ethical implications of values embedded in computational models: an exploratory study. *Proceedings of the American Society for Information Science and Technology*, *43*(1), 1–16. https://doi.org/10.1002/meet.14504301254

Fournier-Tombs, E. (2021). Towards a United Nations Internal Regulation for Artificial Intelligence. *Big Data & Society.* https://doi.org/10.1177/20539517211039493

Gartner forecasts worldwide low-code development technologies market to grow 23% in 2021. Gartner. (n.d.). Retrieved October 22, 2021, from https://www.gartner.com/en/newsroom/press-releases/2021-02-15-gartner-forecasts-worldwide-low-code-development-technologies-market-to-grow-23-percent-in-2021.

Jesiek, B. K. (2006). The sociotechnical boundaries of hardware and software: a humpty dumpty history. *Bulletin of Science, Technology & Society,* doi:10.1177/0270467606295002

Jarrahi, M. H., Newlands, G., Lee, M. K., Wolf, C. T., Kinder, E., & Sutherland, W. (2021). Algorithmic management in a work context. Big Data & Society. https://doi.org/10.1177/20539517211020332

Lynch, D. J. (2021, May 19). Hiring troubles prompt some employers to eye automation and machines. *Washington Post*. https://www.washingtonpost.com/business/2021/05/19/automation-labor-economy/

Pichidtienthum, S., Pugsee, P., & Cooharojananone, N. (2021). Developing module generation for odoo using concept of low-code development platform and automation systems. *2021 IEEE 8th International Conference on Industrial Engineering and Applications (ICIEA)*. doi:10.1109/iciea52957.2021.9436754.

Sahay, A., Indamutsa, A., Ruscio, D. D., & Pierantonio, A. (2020). Supporting the understanding and comparison of low-code development platforms. *2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*. doi:10.1109/seaa51224.2020.00036

Schmelzer, R. (2019). Should we be afraid of ai*? Forbes.* https://www.forbes.com/sites/cognitiveworld/2019/10/31/should-we-be-afraid-of-ai/?sh=468bf3e54331.

Scroggins, M. J., & Pasquetto, I. V. (2020). Labor out of place: on the varieties and valences of (in)visible labor in data-intensive science. *Engaging Science, Technology, and Society* 111-132. doi:10.17351/ests2020.341

Tubaro, P., Casilli, A. A., & Coville, M. (2020). The trainer, the verifier, the imitator: Three ways in which human platform workers support artificial intelligence. Big Data & Society. https://doi.org/10.1177/2053951720919776

Shen, Weijun, et al. Boundary sampling to boost mutation testing for deep learning models. Information and Software Technology, Jan. 2020. EBSCOhost, doi:10.1016/j.infsof.2020.106413

Winkler, D., Hametner, R., Östreicher, T., & Biffl, S. (2010). A framework for automated testing of automation systems. *2010 IEEE 15th Conference on Emerging Technologies & Factory Automation (ETFA 2010)*. doi:10.1109/etfa.2010.5641264