EXPLORING THE IMPACT OF DIGITAL DIY ON THE WORKPLACE

Ruth Meyer[1]

[1]Centre for Policy Modelling, Manchester Metropolitan University, Manchester, M15 6BH, UK

Introduction

Digital Do-It-Yourself (DiDIY) can be defined as the set of all manufacturing activities (and mindsets) that are made possible by digital technologies (DiDIY 2017). The availability and ease of use of digital technologies together with easily accessible shared knowledge may allow anyone to carry out activities that were previously only performed by experts and professionals. In the context of work and organisations, the DiDIY effect shakes organisational roles by such disintermediation of experts, something that can be recognised in the makers communities (Anderson 2012), and has been described with the term "democratization of manufacturing" (Tanenbaum 2013; Williams 2014).

A central question arising from the research related to this is "how the work of a worker in a manufacturing firm will be reshaped due to the influence of DiDIY" (Morris 2010). DiDIY in this context allows workers to overcome the traditionally strict organisational hierarchies by having direct access to relevant information, e.g. the status of machines via real-time information systems implemented in the factory. In order to investigate the impact this would have on the organisation of work, we decided to compare two versions of a simulation model with each other: (i) a version where workers ask their supervisor for information about the next task to perform, and (ii) a model version where all workers have access to the necessary information about machines and tasks so that they can decide themselves which of the outstanding tasks to work on next.

A simulation model of this general scenario needs to represent a more or less abstract manufacturing firm with supervisors, workers, machines and tasks to be performed. Its purpose is to capture the change in workflow that might happen due to the introduction of freely available information about which machines are in use and which tasks need to be finished within which deadlines. Experiments with such a model can then be run to investigate particular aspects of the central research question, including the following:

- If we allow the workers autonomy in the decisions concerning the order in which they want to perform outstanding tasks, does this improve the effectiveness of the production process?
- Would supervisors become superfluous since workers are self-organising their work?
- What is the impact on the manufacturing unit as a whole – it is more productive or differently productive?

StringWorld – a Problem Space for Addressing DiDIY Simulation Issues

The problem space within which the model is set is a world of "strings" following an earlier model (Edmonds 2007). Agents ("makers") use things they can find in their environment ("resources") plus things other agents might give or sell to them to construct new things using a limited set of operations. Each thing is a separate entity associated with a sequence of letters ("elements"). For example, you might have the following strings: "A", "B", "AB", "BA", "AA", "BB", "AAA", "AAB", "ABB" etc., if the elements "A" and "B" were available.

Agents aim to produce certain kinds of things ("targets"), but – since it is not obvious how to make these given the available resources and operations – require either doing some trial-and-error experimentation or following a plan, which states the sequence in which to apply operations. Things and plans are explicitly represented as separate entities, embedding the "Atoms – Bits" distinction highlighted within the DiDIY project.

This problem space meets the criteria of providing an environment to model DiDIY activities that is both computationally feasible and complex enough, so that plans on how to make things are worth sharing. A prototype implementation of the StringWorld is available online (Edmonds 2016).

Model Description

The *StringWorld Factory Model* consists of agents (workers) that are realised as patches in NetLogo, coloured in shades of brown. Any non-agent patches are coloured in black. These may hold resources, available machines (one patch each per different type of machine), produced targets (one patch per target type), or be empty, i.e. they are not used in the model.

Machines are tools providing one of a number of predefined string operations like "join" (sticking two strings together), "add-B" (adding a "B" to the end of a string), "prefix-A" (adding an "A" to the beginning of a string), or "envelope" (sticking one string at the beginning of a string and another at the end, thus "enveloping" the string with the two others). Resources, targets and any intermediate strings are separate objects (things). Operations use up input things to create a new output thing (plus potential by-products). In the current model version we assume that there are always enough resources to perform any operation that needs them.

The factory as a whole has the goal of producing a certain number of each of the targets. This is set via the model parameter *production-goals*. Both resources and targets are automatically chosen during model initialisation, when the factory setup is determined from the specified number of resources and targets (model parameters *num-resources* and *num-targets*) and the number of machine types (*num-machine-types*). Since we are modelling a factory, i.e. a manufacturing system that is producing certain products, the initialisation process has to ensure that it is always possible to make the chosen targets from the given resources with the available operations. This is not a given in the StringWorld problem space and thus a non-trivial problem.

The solution we implemented in the Factory Model uses a 'possible products' network of nodes and links inspired by the firms' skills universe introduced by Taylor and Morone (2005) in their paper on modelling innovation. Starting from the resources, we represent each possible (intermediary) product as a node. Incoming links represent the requirements of a node, i.e. in our case, the inputs necessary to produce the product it represents. The number of inputs determines the type of operation (with one, two or three inputs). A new node forms links with one, two or three existing nodes determined by a simple random distribution, which takes into account the respective number of available string operations. Once a new node is established as part of the network, it becomes available as a potential input for another node, and so forth. The total number of nodes in the network is calculated from the number of resources, machine types and targets. Figure 1 shows an example of such a network with three resources (blue) and five potential targets (white).

After building the network, we select as many leaf nodes as needed for the targets while making sure that every resource node is used at least once. In the example above, we might pick nodes 13, 15 and 17 if just three targets were required. We then assign suitable string operations to the (bundles of) incoming links, choosing from the pre-defined operations with one, two or three inputs to match the number of links. In the next step, each resource node is allocated a random short string made up of the letters "A", "B" or "C", e.g. "AA", "BA" and "CAB" for the nodes 1, 2 and 3 of Figure 1, respectively. With operations and starting points in place it is possible to work out which strings to assign to each intermediary and target node.

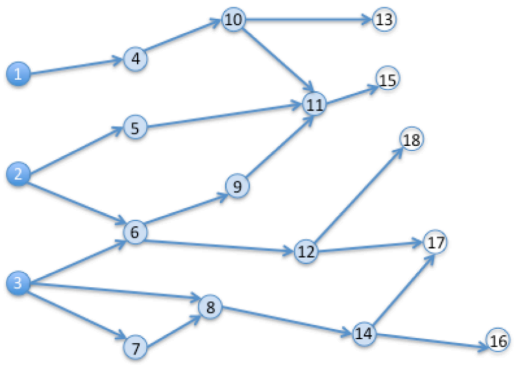

Figure 1: Example of a possible products network with resources on the left (blue nodes) and potential targets on the right (white nodes). Light blue nodes denote intermediary products.

This information is then used to create the necessary machines. It is also stored in form of a plan library that all agents may access. In the manufacturing context we are exploring with this model it makes sense that the information about how to produce the targets is (a) pre-existent, so the agents do not have to find it during the simulation run by trial and error, and (b) shared with everyone involved.

Another addition to the basic StringWorld framework is the explicit use of time in this model. Each operation takes a specific duration to complete during which the machine performing it is unavailable for any other use. This might lead to workers having to wait for a particular type of machine to become available, which in turn might influence their decision on what to produce next. Since in this model version no inherent costs or values are assigned to the resources or targets, time is the only "cost" factor taken into account.

We regard two variants of the factory model:

1. *With supervisor*. In this variant, the supervisor is in charge of deciding who is producing what. Each worker is assigned a target, and then assembles it by using the necessary resources and machines in the correct order. Whenever a job is finished, the worker asks the supervisor what to do next. The supervisor's decision process is influenced by the number of outstanding targets, the currently available machines and the overall time it takes to produce the outstanding targets. This is to ensure that the factory achieves its production goals in as short a time as possible. If any machines are available to start working on one of the outstanding targets the supervisor will choose one of these to assign to a free worker. Otherwise, the supervisor picks (with a certain probability) the target with the longest total production time (average production time of one target * number of outstanding targets).
2. *Without supervisor*. In this DiDIY variant of the factory model workers decide themselves what to do next. They have access to all the necessary information: currently available machines, outstanding targets, and which operations produce what from which inputs. For modelling their decision process, we have made the following assumptions:

    - Workers prefer to make something from the things they already have instead of starting from scratch (i.e. resources).
    - The more of their own stuff gets used, the better.
    - They prefer to use an available machine, i.e. not having to wait for a machine.
    - If this is not possible, they will pick the target that is most under-achieving at the moment.

During the first tests with this model variant we discovered that – depending on the particular factory setup – these criteria may not be sufficient and the workers may need to keep more of an eye on the overall production goals. Thus we added an overriding condition that if one or more of the targets hit an "under-achievement" threshold, a free worker will take it on with a certain "obedience" probability. Both of these values can be controlled via model parameters.

Preliminary Results

To be able to compare the two model variants we used the same factory setup in both and only varied the number of agents. Since time is the only "cost" factor implemented so far, we use the overall simulation time (i.e. the time necessary to achieve the production goals) and the average time agents spent waiting for a free machine as measures of performance.

The graphs in Figure 2 show averages over ten simulation runs each per number of agents with the two model variants. The factory layouts comprised of three resources, three targets and five different machine types each, with production goals of 200 for target 1, 300 for target 2 and 100 for target 3.

The overall shape of these graphs shows what is to be expected: production times go down, while wait times go up with the number of agents. Also not surprisingly, at least for small numbers of workers, the supervised variant performs better overall, since the supervisor assigns jobs with the interest of the whole system performance in mind. Interestingly, though, it seems that once a certain number of workers is reached, the self-organised model version manages to out-perform the supervised variant.

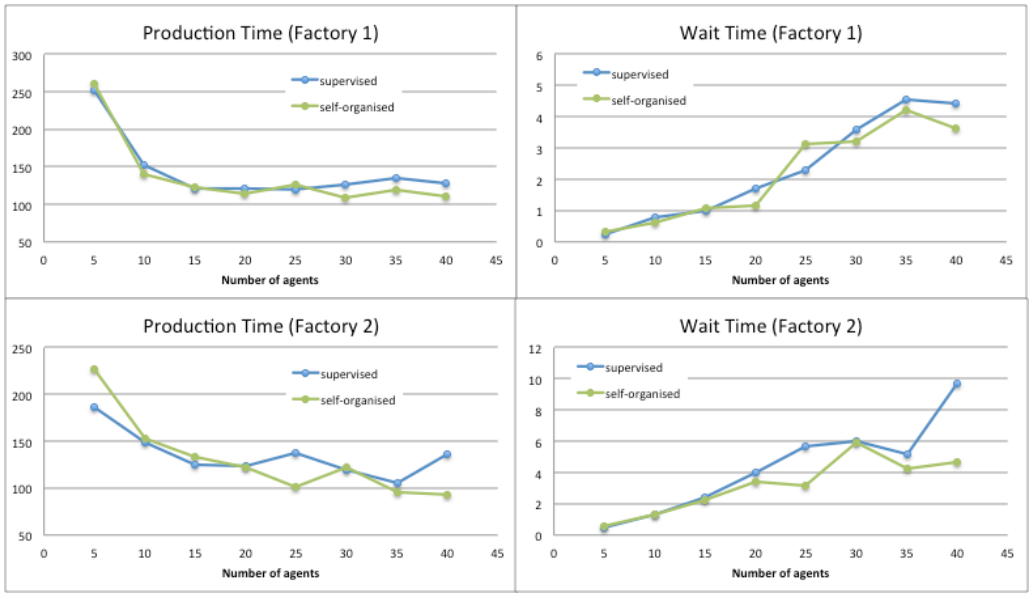

Figure 2: Comparison of the two model variants with regard to average production time (left) and average wait time (right) for two different factory layouts.

Discussion

The current version of the StringWorld Factory Model with its two variants already demonstrates the potential that DiDIY-related technologies and mindsets might have in the domain of work and organisations. Instead of the traditional top-down style of decision-making and job execution, which is implemented in the model variant with supervisor, the variant without supervisor allows workers to adopt a truly bottom-up style of both decision-making and execution of work. While this needed to be somewhat restricted by the overall production goals to make workers actually achieve those goals in certain settings, the way the workers' decision making is realised will easily allow to add another component important in the DiDIY context: that of cooperation between workers. At the moment, each worker only regards the things he/she has produced themselves or the joint resources, when deciding what to do next. With cooperation, they could extend this to include things other workers have produced to open up more possibilities, e.g. the development of a "chain production" in which several workers work together to produce one target. We will explore this option in a future version of the model.

References

ANDERSON, C. (2012). *Makers: The New Industrial Revolution*. Crown Publishing Group

DiDIY (2017). *What is DiDIY? Start here.* Home page of the DiDIY Project. http://www.didiy.eu/introduction

EDMONDS, B. (2007). Artificial Science - a Simulation to Study the Social Processes of Science. In Edmonds, B., Hernandez, C. and Troitzsch, K. G. (eds.) (2007) *Social Simulation: Technologies, Advances and New Discoveries*. IGI Publishing, 61-67. (http://cfpm.org/cpmrep138.html)

EDMONDS, B. (2016). A Model of Making (Version 6). CoMSES Computational Model Library. Retrieved from: https://www.openabm.org/model/4871/version/6

MORRIS, M.G., Venkatesh, V. (2010). Job characteristics and job satisfaction: understanding the role of enterprise resource planning system implementation. *MIS Quarterly*, 34(1), 143-161

TANENBAUM, J., Williams, A.M., Dejardins, A., Tanenbaum, K. (2013). Democratizing Technology: Pleasure, Utility and Expressiveness in DIY and Maker Practice. *Conference CHI 2013: Changing Perspectives*. Paris. France. pp. 2603-2612

TAYLOR, R., Morone, G. (2005). Innovation, Networks and Proximity: an Applied Evolutionary Model. *Proc. of the 4th European Meeting on Applied Evolutionary Economics (EMAEE)*, Utrecht, The Netherlands, 19-21 May 2005

WILLIAMS, A., Nadeau, B. (2014). Manufacturing for makers: from prototype to product. *Interactions*, 21(6) 64-67.