D7.1 INTEGRATIVE MODELS ON THE IMPACT OF DIDIY

Project Acronym	DiDIY
Project Name	Digital Do It Yourself
Grant Agreement no.	644344
Start date of the project	01/01/2015
End date of the project	30/06/2017
Work Package producing the document	WP7: Integrative modelling, guidelines and tools for the transferability of results
WP Lead Partner	MMU
Other Partner(s) involved	all
Deliverable identifier	D7.1
Deliverable lead beneficiary	MMU
Due date	M27 (March 2017)
Date of delivery	31/03/2017
Version	1.0
Author(s)	MMU
License	Creative Commons Attribution ShareAlike 4.0
Classification	PUBLIC
Document Status	APPROVED
This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 644344.	
Disclaimer: The views expressed in this document do not necessarily reflect the views of the EC.	

Disclaimer

This document is provided "As Is"; it is a study introducing the main research topics in the presented context. Any feedback, suggestions and contributions to make this document better and more useful are very welcome. Please let us know through the contact page http://www.didiy.eu/contact. We will seek to incorporate relevant contributions in the document and add your name to the list of contributors.

Executive summary

Deliverable D7.1, "Integrative models on the impact of DiDIY", summarises the progress in Integrative Modelling (IM) within the DiDIY Project up to month 27. It relates to a number of other Work Packages within the Project, as in particular reported in deliverables D3.2, summarising relevant results from WP3 on Work and Organisation, and D4.8, summarising relevant results from WP4 on Research and Education. They determined the direction of the simulation modelling described here. This simulation modelling will relate to and complement two other strands of work, the second version of the Knowledge Framework, in D2.5, and the general conclusions, which will be described in D7.3 (DiDIY-related education processes), D7.2 (Social Adoption of DiDIY) and D7.4 (DiDIY-related policy recommendations).

The first section of this document discusses the requirements for the problem space that the simulated agents face – in other words, a simulation of making and its difficulties. It then describes the chosen solution: a 1D string world of objects that makes clear the distinction between things ("atoms") and bits (the plans of how to construct things out of elementary parts). This problem space framework is applied in the subsequent versions of the models. Section 3 describes the prototype model and its behaviour. This introduces the agents and some rudimentary interaction between them. Results obtained with the prototype model show how the trade-off between exploitation and exploration (DiDIY being an exemplar of the latter) changes the rate of innovation and hence subsequent "wealth" of agents. Section 4 applies and adapts the problem space to how DiDIY might impact upon the workplace by comparing the situation in a manufacturing unit between a more traditional way of organisation and a more DIY way of working. Section 5 looks forward to a version of the framework to investigate different kinds of communication flow (e.g., due to classroom structures, or from adding new internet-mediated communication) to makers. The deliverable ends with some preliminary conclusions.

In short, this document describes the first ever agent-based simulation of makers and making. It shows how DiDIY-related phenomena might emerge from the interaction of individual makers and allows experimentation with counter-factual or observed alternatives.

Revision history			
Version	**Date**	**Created / modified by**	**Comments**
0.0	23/03/2017	MMU	First, incomplete draft.
0.1	27/03/2017	MMU	Extensions and fixes.
0.2	28/03/2017	MMU	Extension and fixes.
0.3	29/03/2017	MMU	Draft circulated amongst partners.
0.4	30/03/2017	MMU	Extensions and fixes.
0.5	31/03/2017	MMU	Extensions and fixes.
1.0	31/03/2017	LIUC	Approved version, submitted to the EC Participant Portal.

Table of Contents

1. Introduction

This document summarises the progress in Integrative Modelling (IM) within the DiDIY Project up to month 27. The timing of this deliverable is not ideal as it necessarily does not include the work done on Integrative Modelling that will occur during the last 3 months of the Project. However, we will try to anticipate those developments to a limited extent here.

The purpose of the IM within the DiDIY Project is to produce specific, but dynamic and complex simulations that illustrate the abstract and informal characterisations of DiDIY-related phenomena. This can clarify the reference of informal natural language accounts, revealing new questions and gaps in our knowledge, but more positively, it can show how complex DiDIY phenomena can emerge and change. We will not recap the contrast between simulation and discursive accounts of phenomena here, as we already did that in D3.2, but mention how simulation modelling complements accounts in natural language, and can give a semantically-thinner but more holistic account of how complex micro-macro phenomena can come about.

This strand of work relates to a number of other packages of work within the Project, and in particular deliverables D3.2, summarising relevant results from WP3 on Work and Organisation, and D4.8, summarising relevant results from WP4 on Research and Education. These determined the direction of the simulation modelling described here. This simulation modelling will relate to and complement two other strands of work, the third and final version of the Knowledge Framework, in D2.5, and the general conclusions that will be described in D7.3 (DiDIY-related education processes), D7.2 (Social Adoption of DiDIY) and D7.4 (DiDIY-related policy recommendations). This is illustrated in Figure 1.

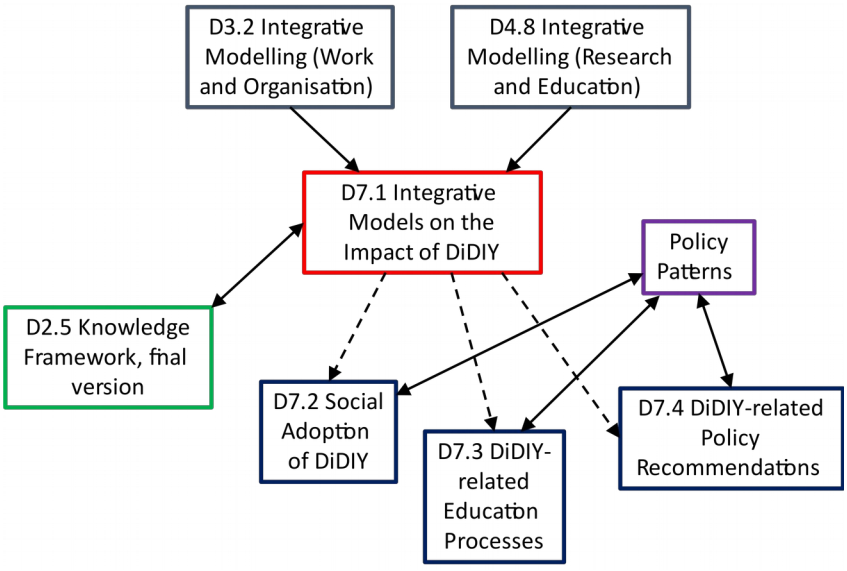

Figure 1 – The relationship of D7.1 to other deliverables.

As an intermediary form, between the formalised world of simulations and the discursive recommendations of D7.2, D7.3, and D7.4, the Project aims at collectively developing a number of "Policy Patterns". These are short, semi-formalised descriptions of a problem with an outline of one possible solution to it. At the very least, the policy patterns will relate to the content of these three deliverables (D7.2 to D7.4), and complement them. Thus, the goals and contents of D7.1 should be

seen in the light of these other parts of the Project that could be considered as other kinds of "Integrative Modelling" in the widest sense.

The goals of D7.1 are three-fold:

- to illustrate how the synergies involved in DiDIY might come about, in contrast to more traditional ways of working;
- to provide specific instantiations of DiDIY concepts that might be located within the Knowledge Framework;
- to make explicit some of the conceptual distinction between parts of DiDIY phenomena, for example the Macro vs. the Micro, conditions vs outcomes, or emergent vs. immergent processes.

The first section of this deliverable briefly recaps on the formalised problem world of making that all models use. The second section reports on the sensitivity analysis of the prototype model, giving some ideas as to its underlying characteristics. The next section compares a traditional way of working with a DiDIY way within a simulated "machine-shop" environment. The following section looks at some preliminary results concerning different topologies of communication, which relate to different types of learning flow. The document concludes with a preview of future work and some tentative conclusions, as related to the above goals.

2. StringWorld – a problem space for addressing DiDIY simulation issues

Drawing on the second version of the Knowledge Framework (D2.4) and the preceding descriptions of the issues relevant for simulation modelling from WP3 and WP4 (D3.2 and D4.8), we abstract the following fundamental characteristics for the problem environment that DiDIYers face:

- there are individual agents that have to make decisions as to alternative kinds of action;
- there is a universe of things that can be constructed out of other things by agents or machines;
- some things are available as the building blocks from which to make other things ("resources");
- how to perform actions and in what sequence to produce a particular thing must not be obvious to an agent, but require either (a) some research via trial-and-error experimentation or (b) to follow some plan given to them. In other words, the plan to make something should not always be obvious from an inspection of the thing that results from executing the plan and the sequencing of actions in plans is important;
- only some actions may be executed by an agent – determined by its access to tools and the affordances of the things themselves. Not all agents have an ability to do the same actions;
- agents have a different set of skills and knowledge of how to do things;
- plans may be communicated between agents separately from things;
- things can be passed between or bought/sold between agents, but not duplicated without essentially reconstructing the thing in a comparable process to how the original was made;
- agents aim to produce certain kinds of thing – those that have utility to themselves or others ("targets");
- there is some cost to the actions involved in making things (monetary or temporal);
- tools (either other things or machines) can be used to transform some things into other things.

Furthermore, there are some constraints coming from the wish to represent these problem characteristics within a Simulation Framework:

- the framework must be computationally feasible, so that programming and simulation execution are possible within the period of the Project;
- these things must be able to have a complex make-up so that one can see how the plans, actions and the constituency of things relate but different things can have the same constituency;
- things must be relatively easy to visualise.

A "problem space" within which to program DiDIY Integrative Models was decided upon and presented to the Project at month 6 for comment. This meets the above criteria. It is a world of "strings" following an earlier model (Edmonds 2007). Each thing is a separate entity associated with a sequence of letters. Thus if the elements "A" and "B" are available, the following strings can be obtained: "A", "B", "AB", "BA", "AA", "BB", "AAA", "AAB", "ABB", etc. In the following, we will dispense with the quote marks when mentioning such strings, their nature being clear.

To see the complexity that this can result in, even with only two such elements, imagine the following world of possibilities:

- the world is naturally abundant in the resource strings A, AB, and B,
- and you want to make the string BA,
- but only have the following actions:
    1. stick a string ending in A to a string starting in A, thus AAA to ABA to get AAAABA (join-A);
    2. stick a string ending in B to a string starting in B, thus AAB to BBA to get AABBBA (join-B);
    3. “rotate” a string of 4 letters long, e.g., from ABAX to XABA to AXAB to BAXA (rotate);
    4. split a string into two equal portions (split).

In order to make a BA you might need to do something like the following: (1) join A to AB to make AAB (2) join AAB to B to make AABB (3) rotate AABB to get BAAB (4) split BAAB into BA and AB. The result is one BA plus a waste product, AB. This is illustrated in the following plan (Figure 2).

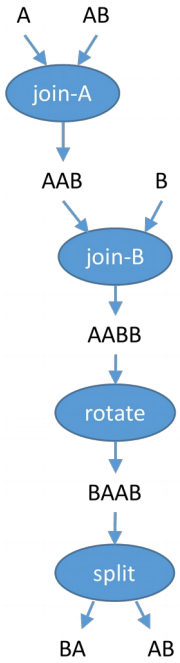

Figure 2 – Making string BA, from resources A, AB and B.

Just looking at the thing of form BA, it is not obvious how to make it given the available resources and actions. This lack of transparency in how to make things is important for modelling DiDIY phenomena, otherwise knowledge for how to make things (e.g., in the form of plans such as in Figure 2) is not valuable and not worth communicating. In such an environment who can communicate knowledge to whom, who is allowed to use plans and how plans can be converted from “bits” to “atoms” is important.

Having to do all those steps is quite different from having a machine that does all the steps in Figure 2 quickly and automatically, feeding in lots of As, ABs and Bs and getting out BAs and ABs (which one might recycle back into the in-feed. This is also different from having the equivalent of an additive manufacturing technology where one just fed in As and Bs and the pattern “21” and got out BA (or any other string you wanted) – a 1D printer!

This basic problem structure is used in the model versions that follow, but with different actions and kinds of goals available in each one.

3. A prototype model of making

Overview

The model described here is the prototype model already discussed in D3.2, but included for completeness and to present and discuss new results obtained since then. It directly instantiates the StringWorld problem space described in the previous section.

The purpose of this model was to develop and provide the simulation infrastructure needed in order to model the activity of making. That is individuals using resources they can find in their environment plus other things that other individuals might sell or give them, to design, construct and deconstruct items, some of which will be of direct use to themselves, some of which they might sell or give to others and some of which might be used as a tool to help in these activities. It explicitly represents plans and complex objects as separate entities in the model – embedding the "Atoms – Bits" distinction highlighted within the DiDIY Project. This allows plans to be shared between agents, which give the steps of how to make objects of use – either on a commercial or a free basis.

The framework is the basis upon which the subsequent models have been constructed, allowing the exploration of a variety of "what if" or counterfactual possibilities and thus give a concrete but dynamic and complex instantiation of the issues and situations discussed within the DiDIY Project. In a sense this model is a "bits" representation of the ideas discussed.

Model description

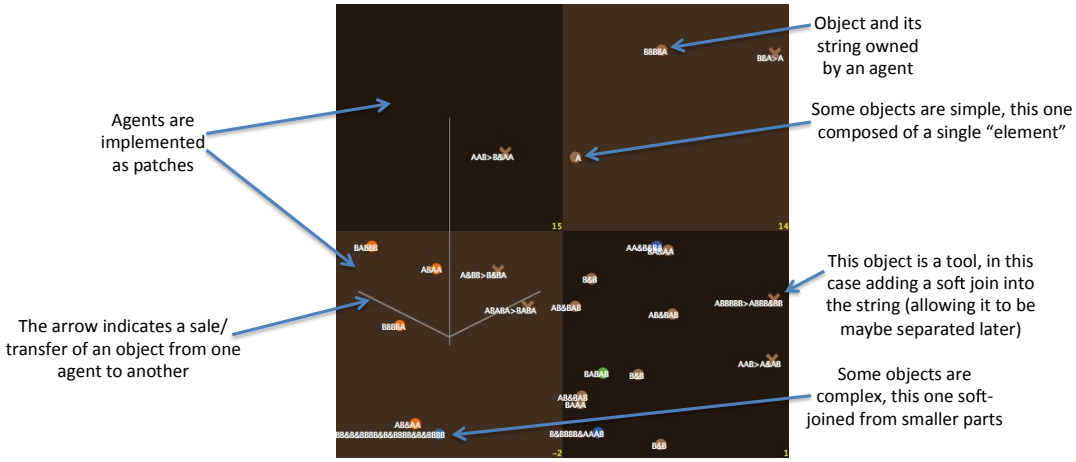

Figure 3 – A view of the model with agents, objects and a sale.

There are three main kinds of entity in the model, agents, things and plans.

1. *Agents*. There are a fixed number of agents that do the making and decision making in the model. These are individually represented as "patches" in the model. They own and hold things. They can (depending on the nature of the things) act upon these things to make new things. At the moment their position is not important and they can swap/trade things with any other agent. They also hold in their mind a number of plans which they have either learnt themselves (through trial and error) or obtained form another agent.

2. *Things*. Things are individually represented and tracked within the model (from creation to destruction). They each have the nature of a 1D string of symbols, composed of a fixed number of "elements" (the letters "A", "B" etc. depending on the parameter which determines the number of elements) and the two symbols "&" and ">". "&" indicates a soft join, that is a join that an agent can make (or break) without a specific tool. ">" indicates that the item can also be used as a tool, that is it can be used to "transform" the string on the left of the ">" into the one on the right in another string.
3. *Plans*. Agents remember sequences of actions they used to construct/deconstruct strings and the cost/benefit of the result as explicit plans. These plans are a tree structure of actions and the strings that resulted. The ability to remember these plans allows agents to repeat successful plans and also allows the possibility of plans being shared/licensed between agents.

The world of 1D strings is sufficiently complex to make the process of working out what sequences of actions would result in which valuable strings is a hard problem. Which strings are available in the environment and which strings have inherent "use" value are randomly determined at the start of the simulation. Which subset of strings are available to each agent and which subset can be redeemed by each agent can be varied, so as to be able explore the impact on the heterogeneity of resource availability and agents needs. This hardness is what makes plans valuable and so worth sharing.

The most important process is that of agents picking one of their plans (some of which are default, "try something random", plans) and then doing its steps. The cost/value of the result of doing these plans are remembered (basically value – costs in making the object) so that later plans that are better can be preferentially chosen in this process. Thus this combines some trial-and-error with developing a focus on plans that worked better. The main process is outlined in Figure 4.

Each tick the following happens:
1. Some variables and links are cleared
2. **Main loop**. Each agent does the following in a random order of agents:
   2.1. Initialise variables, compile a choice list to speed up plan choice
   2.2. *Until* a plan succeeds or a given number of attempts have been exhausted:
      2.2.1. Choose a target string from plan memory, find a (given) number of plans from the memory that would result in this string
      2.2.2. Try these plans one at a time until one succeeds or they are exhausted:
         2.2.2.1. *If* plan seems to be possible to do, then do plan
   2.3. If are remembering too many plans, maybe forget one with a bias towards the least valuable
3. Update display, clear dead objects from can-buy lists, do stats and timing

Figure 4 – The main process.

This model aims at representing the process of making as directly as possible, given a universe of objects and tasks that allow for this to be meaningfully complex, so that communicating plans is worthwhile, and that there is motivation for trade and/or sharing objects.

Objects are represented individually and explicitly. Some objects can be extracted from the environment by some agents at a cost (depending on their composition). Some objects can be

directly used by some agents so they gain value (depending on their composition). Objects can be soft-joined using an "&" (e.g., A and B to A&B) or split at a "&" (e.g., B&AA to B and AA). Some objects can be used to transform other objects, those with a ">" in them (e.g., AB>BA used once on the object ABB would result in BAB. Objects can be passed from one agent to another agent but can only be owned by a single agent at a time. Once used up they disappear from the simulation.

Plans are mental representations of the steps needed to construct/deconstruct strings. They reside in the agent's memory. They can be communicated with other agents, potentially either freely (following some license conditions) or on a commercial basis.

Initialisation

The number of agents are fixed and are initialised as "patches" on a 2D grid. Each agent is initialised with the same default plans, which are given values. At the moment these plans and value pairs are: [-2 "get-random"] [-0.75 "apply-random"] [-0.5 "realise-random"] [-0.5 "sell-random"] [-2 "buy-random"] [-1.5 "split-random"] [-1 "join-random"], where [-2 "get-random"] means extract a possible string from the environment using the notional value for the plan of -2 when comparing it with other plans.

The main variation in terms of initialisation comes in what resources and redeemable target strings are available. The total "menu" of resources and targets are determined as follows.

Environmental Resources. This is a list of strings that can be got from the environment and the cost of doing so. This is determined as follows:

1. the basic "elements" (characters not "&" or ">") are given, the number set by parameter;

2. random strings using these elements are constructed with a length determined by a Poisson distribution whose mean is given by parameter;

3. a given proportion of these strings have a soft-join (a "&") inserted with a given probability;

4. an additional number of random tools are generated with the same length distribution;

5. all of these are then attributed a random cost picked from a Poisson distribution with a given mean (if the item is available to an agent in terms of extraction from the environment, then it is with this cost it is extractable);

6. each agent is then allocated access to a random selection of these, the number determined by a given proportion of the totality.

Target Affordances. To ground the value of strings for agents some strings are allocated a redemption value – that is, an agent can get this value by "consuming" this string. Again, like resources, agents may have a different selection of such strings, representing different needs and goals, but always with the same value. The process for initialising these targets and their values is as follows:

1. the basic "elements" (characters not "&" or ">") are given, the number set by parameter;
2. random strings using these elements are constructed with a length determined by a Poisson distribution whose mean is given by parameter;
3. all of these are then attributed a random value picked from a Poisson distribution with a given mean (if an agent is able to directly redeem this string, then this is the redemption value);

4. each agent is then allocated access to a random selection of these, the number determined by a given proportion of the totality.

Thus agents have a hard problem, how to make strings that they can redeem/sell given the strings that they have as a resource/buy.

For example the list of strings that might be extractable from the environment might be: A; A>; AA; AB; B>; BA; BB; A&A; A&B; AAA; AB>BA … and the list that is redeemable in terms of value be: AB; A&B; AAA; AAB; ABA; B&A; BBA; BBB; A&AA …. Inspecting these one can see that A&B might simply be both extractable and consumable; A&AA could be made by joining A and AA, both of which might be available in the environment; to make B&A one would have to obtain A&B, split it apart and rejoin it as B&A; and to make AB one might have to apply the tool AB>BA on the item AB. Making items of high value might require a very complex series of steps, and some maybe not possible to make. This is more difficult because each agent will only be able extract a different subset of resources and redeem a different set, which provides a motivation for commerce and sharing of items.

The parameters that control this environmental framework of resource cost/distribution and target value/distribution determine how hard the making problem is for agents and whether it will be useful for them to trade. A very hard framework would make the discovery of plans that allow the construction of valuable items important, a great heterogeneity in terms of resources encourage trade, but the heterogeneity of targets might encourage the emergence of streamlined production (i.e., manufacturing).

For further details on the model we refer to its ODD description given in the appendix of Deliverable 3.2. The model itself including documentation is also freely available online (Edmonds 2016).

Some results

Typical behaviour over 10 runs

First we look at some typical behaviour over 10 sample runs with the following settings:

- action-cost: 0.25
- av-income: 0
- choice-bias: 2
- cost-resources: 2
- len-resources: 2
- len-targets: 5
- max-time: 0
- max-tries: 20
- nat-tool-premium: 1
- num-agents: 4
- num-alternatives: 3
- num-elements: 2
- num-resources: 100
- num-targets: 50

- prop-breaks: 0.4
- prop-nat-tools: 0.25
- prop-resources-each: 0.75
- prop-targets-each: 0.75
- storage-cost: 0.01
- tool-use-cost: 0.25
- value-targets: 3

We look at the variation of a number of measures for each of the 10 runs, to give an idea of the general behaviour. The runs are set up the same but with different random seeds so different choices may be made. The same colours are used for the same runs in each chart.

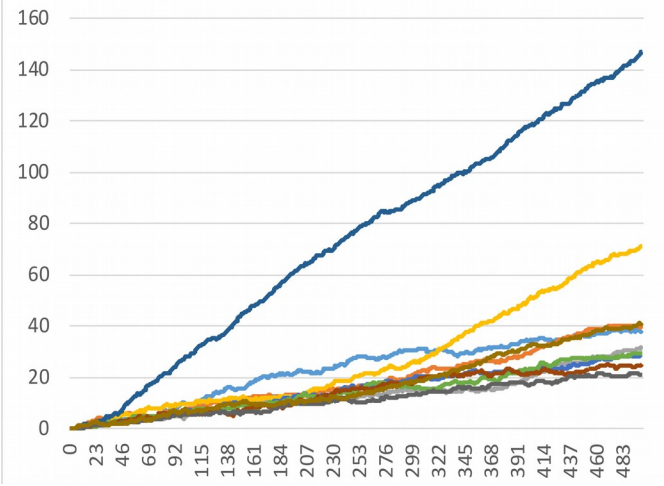

Figure 5 – Number of things made in 10 different runs.

The number of things made can be very different in some runs, and the rate of production change (as in some of the runs above). However, as we see below, most of these are identical copies of the same kind of thing (has the same string).

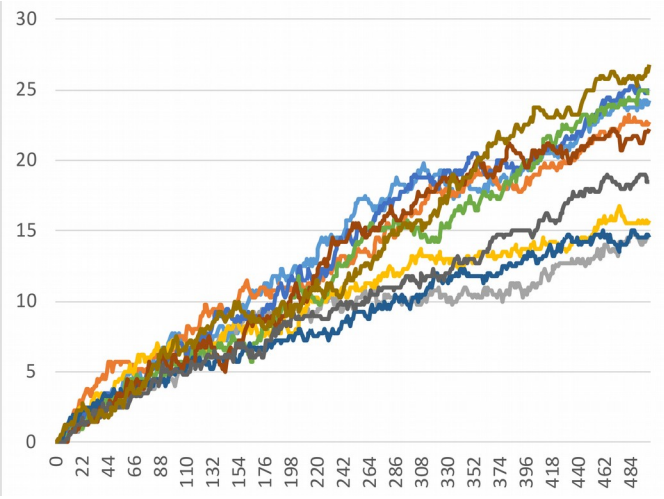

Figure 6 – Number of distinct things in the 10 runs.

In fact there is almost an anti-correlation between runs, which make many things, and those that make more different things. As we see in the next chart very few of these things are tools.

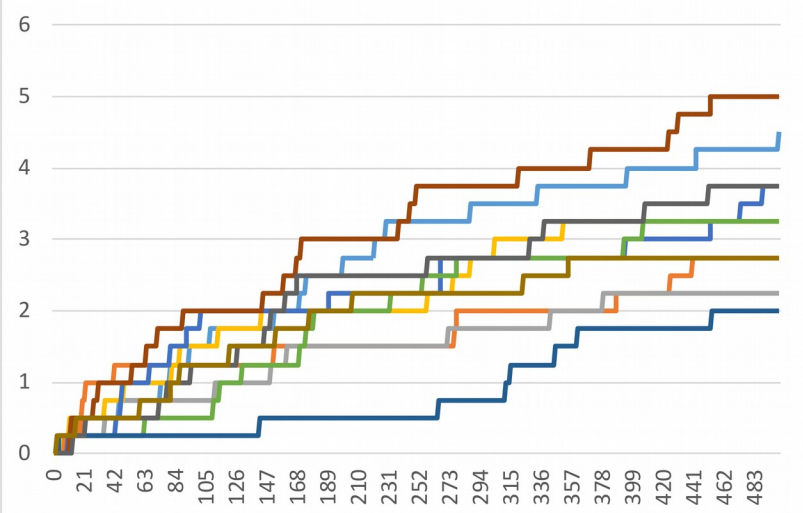

Figure 7 – Number of tools made/found in 10 runs.

The average string lengths of agents form quite an interesting pattern.

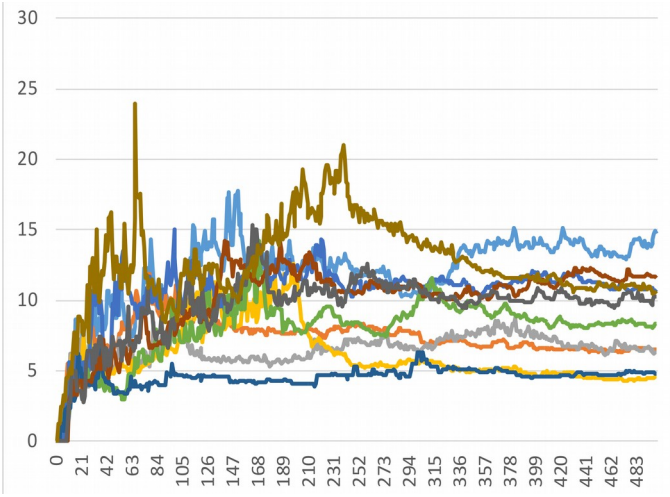

Figure 8 – Average String Length in 10 runs.

The run that made the most objects had a consistently low string length, so might have made a lot of simple things. The runs with more distinct things had generally higher string lengths.

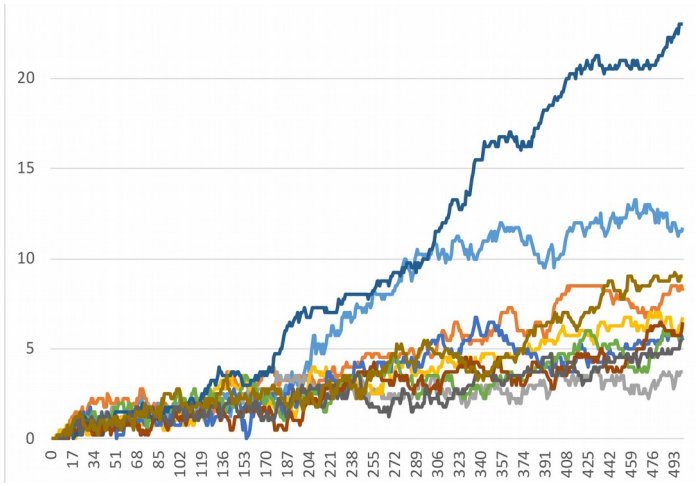

Figure 9 – Average number of things for sale each of 10 runs.

Looking at the number of items for sale, the run with lots of short strings has a lot of these for sale but very few are actually sold.

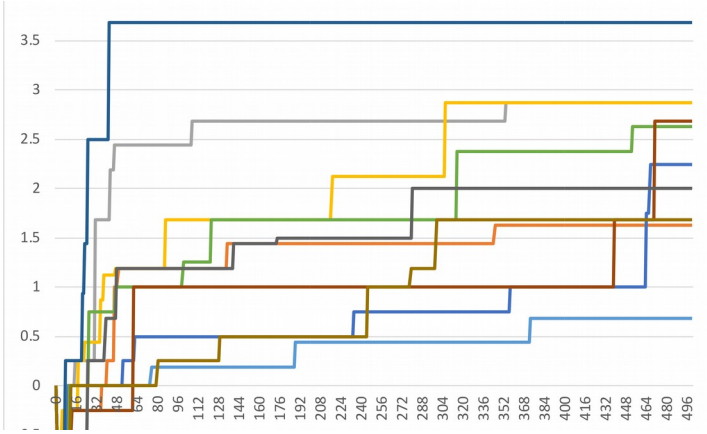

Figure 10 – Average maximum plan value in each of 10 runs.

Figure 10 shows the average maximum valued plan (over the 4 agents) for each of the 10 runs. The value of plans discovered increases over time, and can vary greatly in each run. The run where lots of things were made had a plan discovered early on, probably involving a process that made lots of small strings as a side effect. This seems to be confirmed in the next chart.

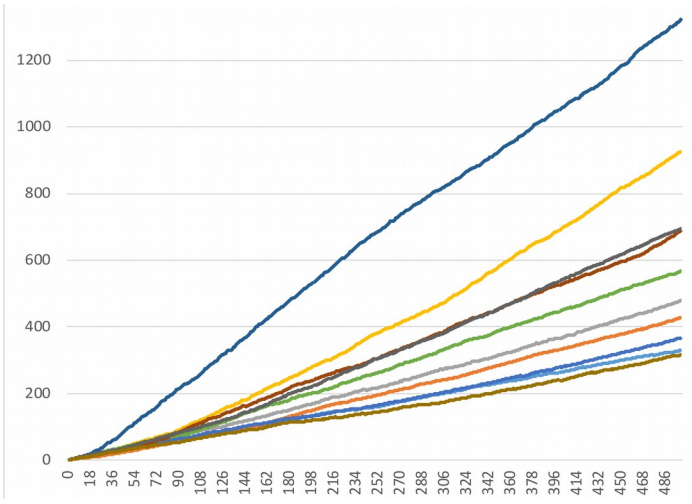

Figure 11 – Average Total Wealth in each of 10 runs.

The run with the most wealth was also one of the most unequal. In some runs the rate of change of inequality, such as in the yellow line below, indicates that other agents in this run started to catch up with the innovation of the leader. In about half of the runs inequality was low.

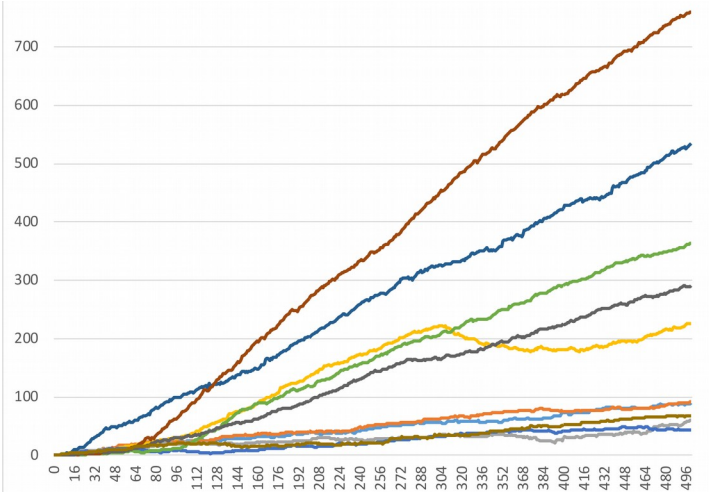

Figure 12 – Standard deviation of wealth in each of 10 runs.

Next, we look at the sensitivity of the model to varying a number of the parameters. This gives one a feel for the behaviour of the model and the degree to which this varies depending on the values set. For each parameter value, we average the results over 10 independent runs. For indicative output we look at (a) the average length of strings that result from discovered plans and (b) the average maximum value of those plans. Each run was performed over 500 time steps so one can see the progression of the values over time.

The default values are as in the above list, i.e., these are the parameter values of all those settings that are not varied each time.

Number of Agents

The number of agents in each run was varied from 1 to 16 in steps of 1 (although only some of the values are shown for clarity).

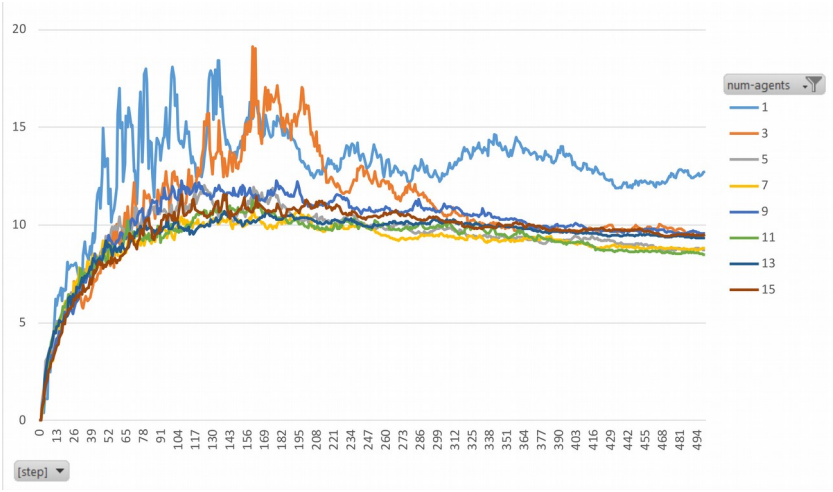

Figure 13 – Average String Length on Number of Agents.

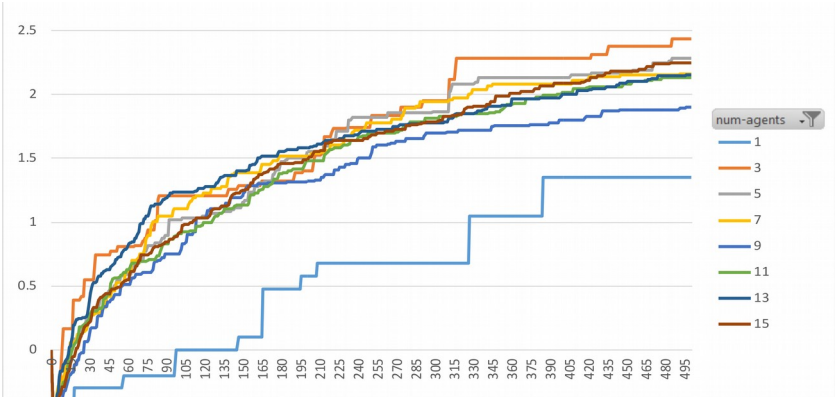

Figure 14 – Average Plan Value on Number of Agents.

Here we see considerable variation between the cases of 1 or 3 agents, but increasingly smooth average lengths and maximum values for greater numbers. Not all of this can be attributed to a simple averaging process, because the agents can buy and sell useful items, which can act to improve the value of plans and reduce the size of strings searched over.

Average Cost of Resources

Here we varied the average cost of the resources available from the environment.

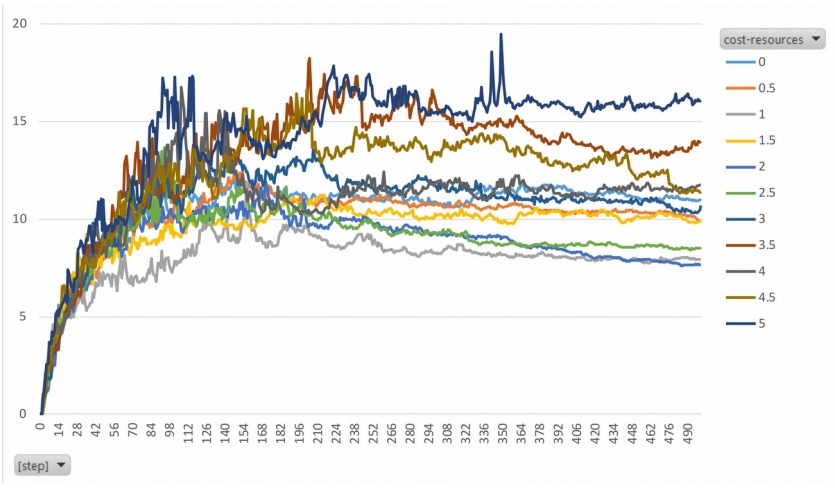

Figure 15 – Average String Length on Average Cost of Resources.

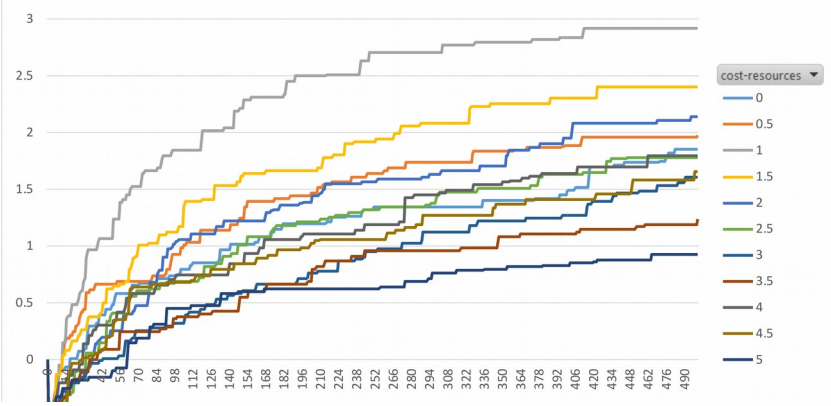

Figure 16 – Average Plan Value on Average Cost of Resources.

Unsurprisingly, increasing the average cost of the resources available from the environment has a direct impact upon the value of plans, though, surprisingly having a very low cost was not helpful as then there was less pressure to search for efficient plans (as is shown in the upper chart where the lowest lengths of plan results resulted from low but not smallest cost scenarios). Similar results were gained from varying the cost of using tools and action costs.

Number of Resources Available

Here we varied the richness of the kinds of resources available to agents. That is the number of different strings that could be simply “picked up” from the environment.

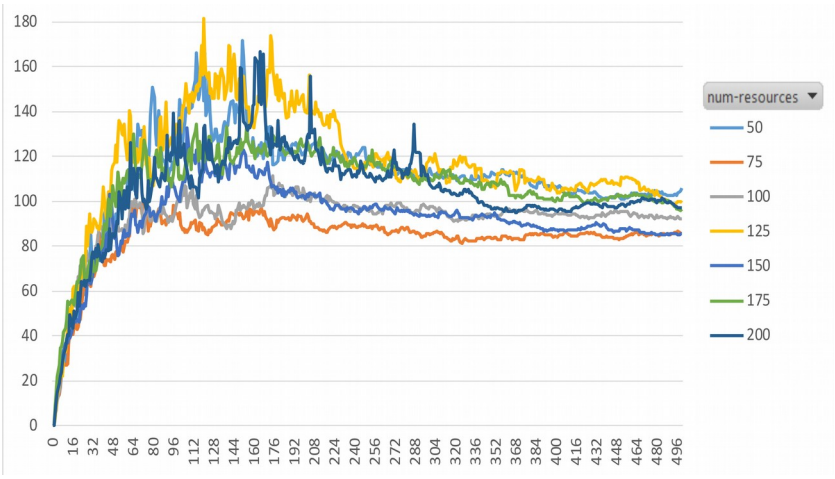

Figure 17 – Average String Length on Number of Resources Available.

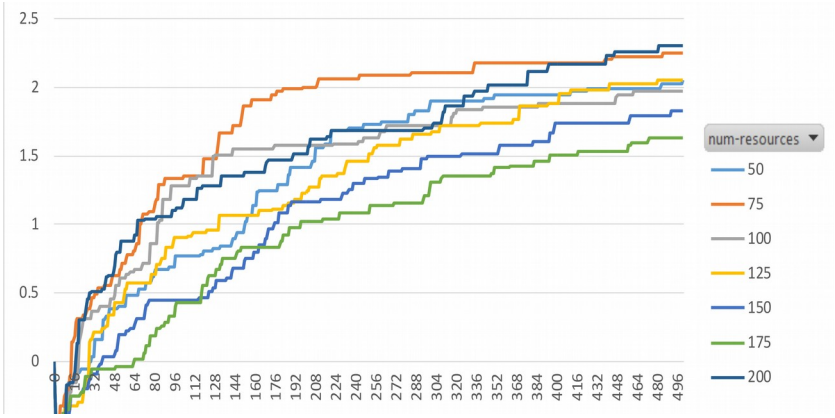

Figure 18 – Average Maximum Plan Value on Number of Resources Available.

Whilst the richness of the number of environmental resources seems to reduce the maximum values of the agents' plans, this is due to the search problem facing agents. With more resources there are more combinations to try, and so the learning is slow – they are "spoilt for choice" so to speak. Similarly, one can see the agents searching over longer strings when there are more resources.

Exploitative vs exploratory search bias

Here we vary the extent to which agents always go for the most valuable plans and shortest strings (high choice-bias), rather than exploring more at random (low choice-bias). In other words, their tendency to optimise and exploit or to explore possibilities that (up to then) had low values and produced long strings, and so might be thought less promising.

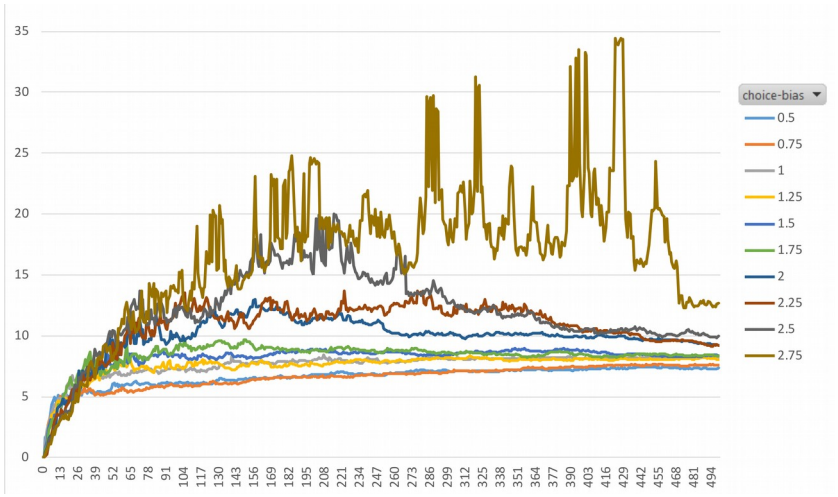

Figure 19 – Average String Length on Choice Bias.

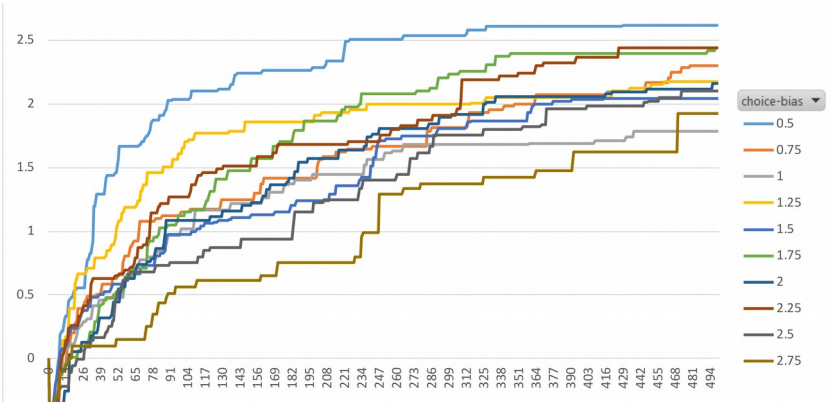

Figure 20 – Average Maximum Plan Value on Choice Bias.

Here an exploratory approach shows clear benefits in terms of both plan value and string length, with some sudden large strings dealt with in the exploitative mode (high choice bias). This is due to the space of possible actions being poorly explored in the exploitative mode, so it being more of a case of happenstance as to what is discovered first, since it is then hard to move beyond this.

Conclusion

The baseline model using the 1D StringWorld problem space already exhibits some interesting behaviour. Agents are clearly searching over the space of possible plans to look for better ways of making things and better things to make. This indicates the potential of the approach, that there is emergence in the model, showing some subtle synergies and dependencies between agents, and that the problem space for agents is roughly appropriate for its role.

4. Integrative Modelling of work and organisation aspects

Defining the scope of the model

WP3 set out to analyse how DiDIY is reshaping organisation and work. A key assumption has been that the DiDIY phenomenon is shaking up organisational roles by enabling the disintermediation of experts. As reported in D3.1, an extensive literature review identified several relevant research streams: digital manufacturing, entrepreneurship, and new organisational roles and competences, the first and last of which come together in the topic of workmen in the Industry 4.0 era. For further investigation with a simulation model, we have chosen this research topic.

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”. As can be seen from the case studies of the four firms, 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. 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?

In order to investigate the impact an accessible, real-time information system will have on the organisation of work, we decided to compare two versions of the 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.

Model description

The original simulation framework as described in section 3 above distinguishes between agents (makers, modelled as patches), things (including rudimentary tools, represented as strings), and plans (instructions on how to make particular things by applying one or more of the available actions or tools). To be able to model scenarios relating to workers in a manufacturing environment we developed this simulation framework further by shifting the focus from the process of making (i.e., agents finding ways to construct things) to the process of decision making (i.e., agents deciding when to do what).

We used the following mapping of existing model elements to scenario elements:

Agents	→	Workers and supervisors
Things	→	Tasks (Products to be produced)
Tools	→	Machines
Plans	→	List of necessary operations (machines) for completing a task

The way tools and actions are realised in the original model framework had to be expanded to allow for a suitable representation of machines. In contrast to the model of making, plans are usually pre-existing instead of being discovered by the agents through trial and error or learning. Another extension of the model framework is the explicit representation of the time it takes to perform particular actions. This is necessary to be able to determine when machines are free and if tasks can be completed within certain deadlines.

The *StringWorld Factory Model* consists of agents (workers) that are realised as patches. As in the prototype model, they are 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. Figure 21 shows an example factory model setup with three different resources, three targets and five types of machines.

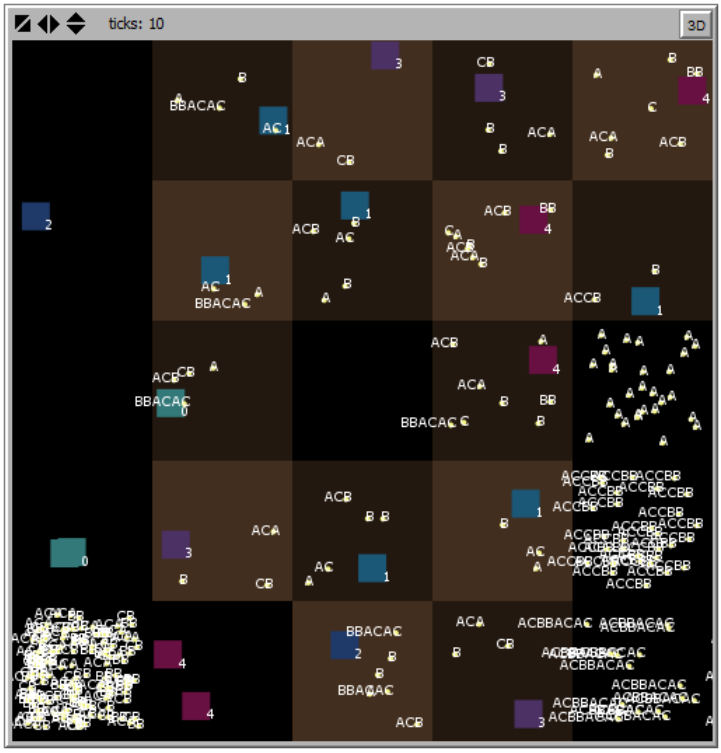

Figure 21 – The Factory Model with workers (brown patches) and machines (different coloured squares). Free machines are located in the black patches on the left. Resources are provided on the patch in the lower left corner, whereas the targets already produced are moved to the three black patches on the right. Workers keep things they are currently working on or that are by-products on their patch.

Machines are tools providing one of a number of predefined string operations like “join” (sticking two strings together), “add-B” (adding a “B” at the end of a string), “prefix-A” (adding an “A” at

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*), the number of machine types (*num-machine-types*), and the number of machines per type (*num-machines-per-type*). 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 (see section 2) 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 22 shows an example of such a network with three resources (blue) and five potential targets (white).

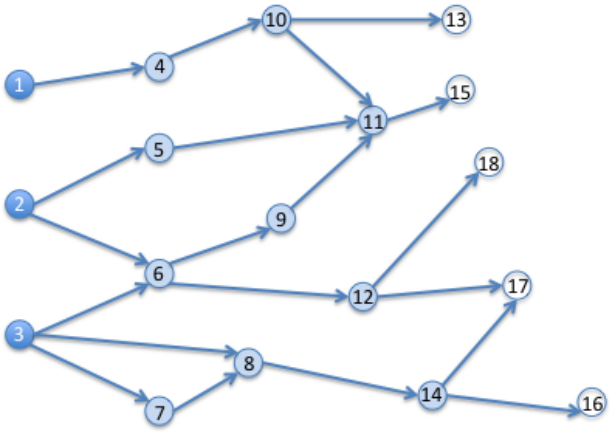

Figure 22 – Example of a possible products network with resources on the left (blue nodes) and potential targets on the right (white nodes). Light blue nodes denote intermediary products.

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 22, respectively. With operations and starting points in place it is now possible to work out which strings to assign to each intermediary and target node.

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 simulation 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 (in contrast to the prototype model of making), time is the only “cost” factor taken into account at the moment.

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 (production time of 1 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 hits 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 (*force-threshold* and *obedience*, respectively).

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

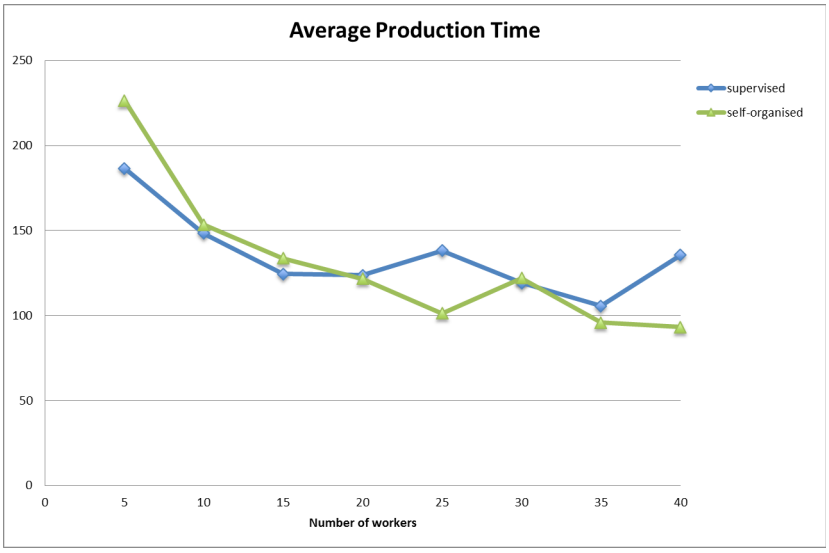

Figure 23 – Average overall production times dependent on the number of workers for the two model variants, with supervisor (blue) and without (green).

Figure 24 – Average time an agent spent waiting for a free machine dependent on the number of workers for the two model variants, with supervisor (blue) and without (green).

The overall shape of these charts 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 (>= 20 in the example), self-organised model version manages to out-perform the supervised variant.

As already noted above, the self-organised model variant may suffer from producing too much of a particular target while not producing enough of others, when the workers are left to decide what to do next solely depending on their own current situation. We therefore had to introduce a way to make the workers keep track of the production goals. Using the same factory layout as in the runs the previous charts are based on, we explored the effect the model parameters *force-threshold* and *obedience* have on the behaviour of the workers. Figure 25 shows how the over-production of target 1 evolves for different values of the under-achievement threshold combined with different values for the workers' propensity to obey this rule. All charts are averages over 5 runs with 15 agents each.

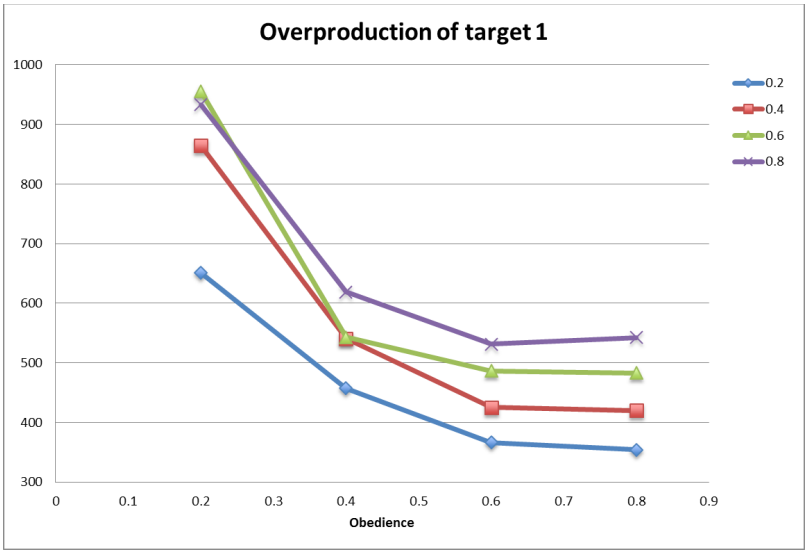

Figure 25 – Influence of the model parameters *obedience* and *force-threshold* on the overproduction of a target.

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 organisation. 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.

5. Integrative Modelling of research and education aspects

The version of the Integrative Model that deals with issues coming out of D4.8 and discussions with the partners at the London meeting in January 2017 is still being developed. Thus we will report on the direction of this development here, but not be able to discuss any results yet.

The main conclusion of D4.8 was that the Simulation Modelling should "focus on the issues of sharing of knowledge and/or tools and the possible change in the flow of learning from the traditional teacher-student direction to a more student-centred, peer-to-peer learning flow".

This emphasis on learning flow – how skills, information and examples might be communicated between agents – was taken up by other partners in the Project as being a generally important issue, beyond the classroom. This is because the ability to share over the Internet is what makes DIY activities into Digital DIY, and allows for a qualitative change in making activities. Showing the power of such sharing within simulation models would make this transition apparent and start a process whereby we might better understand the impact of such sharing.

This implies that we need to develop the basic modelling framework in two directions:

1. to introduce skills into the model. It already distinguishes things and knowledge (in the form of plans) – however, skills are different to (explicit) knowledge. Skills require practice at doing things, but can be aided by doing this in the presence of people with those skills, providing feedback and demonstrating;
2. different structures of communication and interaction need to be able to be imposed upon the model so as to compare the results of different ways of doing this – for example comparing a situation where agents can only talk to neighbours and where there is also an ability to share plans with anyone.

Skills

The suggested requirements for implementing skills within the modelling framework are as follows:

- skills are different from explicit knowledge (such as plans);
- skills are particular to certain actions, or related classes of actions;
- skills cannot be simply communicated over a network or by talking to anther agent;
- skills need practice to improve;
- the level of skill affects: (a) the probability of some making activities succeeding (the results do not have to be discarded) but also (b) the level of skill affects the quality of what results;
- the presence of someone with a skill facilitates the acquisition of that skill, shortening the practice time needed (but not eliminating it) and also improving the quality of the developing skill.

Thus the proposal is to implement the following:

- each agent has a table that tracks their ability to perform actions on different combinations of strings, each action-combination has a value from 0 (useless) to 1 (perfect);
- some actions (such as joining or splitting) are inherently easier than others (such as using tools);

- when faced with doing an action on a combination of strings already experienced, the level of skill (probabilistically) increases, with a rapid initial increase followed by smaller increases up to perfection;
- when faced with doing an action on a new combination of strings then it starts with a basic level of skill (related to the kind of action) but maybe start a bit higher if it has encountered similar strings before (used to handling such things) – this requires a record of what strings each agent has encountered before;
- each thing has a quality attached to it, the quality degrades a bit on each action step done by a human (maybe not by a machine), according to how imperfect their level of skill is;
- if a thing's overall quality drops below a certain threshold (a parameter) then it cannot be used in many other actions (such as selling it, using it to make something else etc).

This proposal captures some of the requirements for skills, but it is not perfect. The quality of a thing is represented as a single number, which is a definite simplification. How much a skill is transferable from one application of an action to another is a complex question and this proposal would need to make some assumptions about this aspect. The skills are attached to single actions (in a manner specific to what it is being applied to) rather than sequences of actions. However, it would capture many of the above requirements for skills and probably be good enough for simulating non-artistic domains.

Communication networks

The second aspect is the ability to impose different communication patterns upon the modelling framework. This is simpler than implementing skills. The proposal is to start by implementing different constraints upon who can communicate with whom and compare the impact of these upon discovery, the ability to make things, general wealth etc. Some of the possible patterns for such networks are illustrated in Table 1.

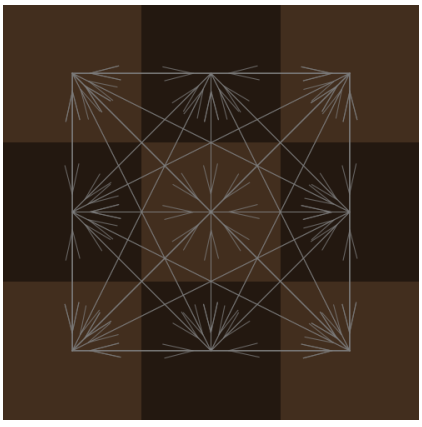

Figure 26 – All.

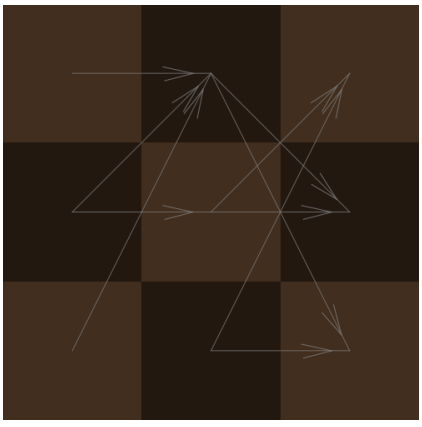

Figure 27 – Left to right (linear).

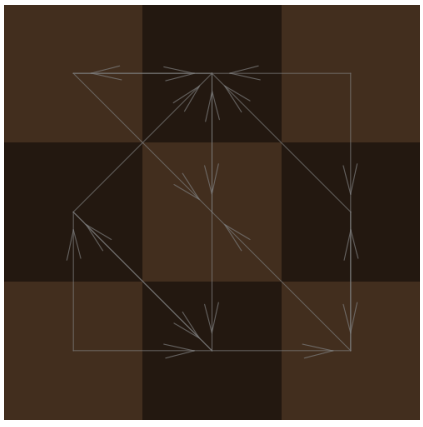

Figure 28 – Neighbours only.

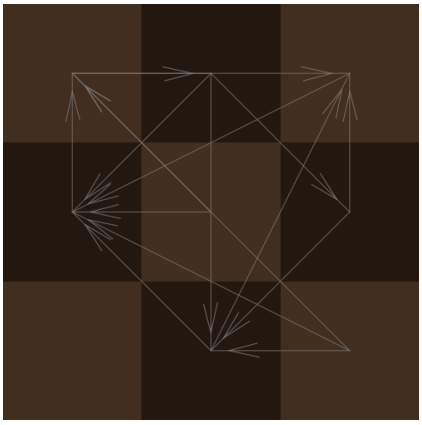

Figure 29 – Random.

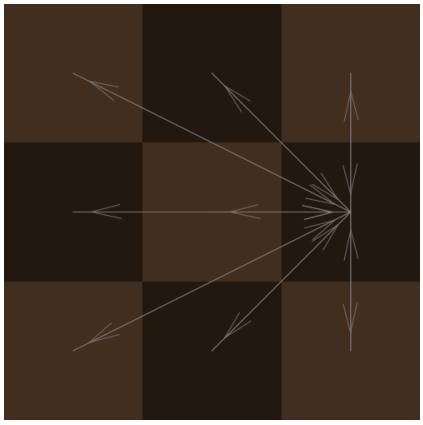

Figure 30 – Star.

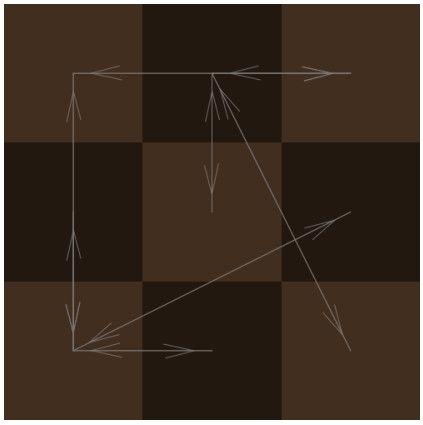

Figure 31 – Tree structured.

Table 1. Some possible patterns of communication between agents.

These structures are quite abstract but cover a number of possibilities. Once these have been investigated more specific patterns could be looked at, for example ones that might mimic patterns found in different kinds of classroom, or that result from the introduction of particular internet-based forums or plan-sharing sites.

One issue that is yet to be resolved is whether there should be different communication networks for swapping items, for sharing (i.e., communicating) plans and for facilitating skills. To start with, we propose that just one network structure is used, but that skills can only be facilitated with those in the network who occupy a location adjacent to the one with higher skill. This would reflect a coherent story that the network represents those one has potential contact with for discussions or buying/selling; constraining the interaction possibilities.

6. Future work and some conclusions

Future work

The integrative modelling work is not going to stop with this deliverable. In particular we plan the following:

- to refine the modelling framework and archive an update of this model;
- to refine and release a version of the factbase extension if this is further developed;
- to further develop the version of the model comparing traditional work patterns to alternative ways that integrate elements of DiDIY, doing a more rigorous comparison of these possibilities, and analysing the reasons for the different outcomes. Also writing a paper on this model for publication;
- to refine and document the extension that allows for queueing while waiting for an event within Netlogo for public releases. So far, this extension has been developed for the factory model;
- to develop the skills/communication version of the model for further analysis, development and comment, with the possibility of developing a joint paper on this;
- to discuss, with other partners, how this might be related to the third and final version of the Knowledge Framework;
- to actively explore what connections there could be in the other WP7 deliverables.

Conclusions

Conclusions from the Simulation Modelling are tentative at the moment. However we can say that:

- the simulation modelling of DiDIY type phenomena *is* possible (albeit complex);
- frameworks to aid the development of such complex models have been produced and released for general use by other modellers;
- such simulations can bridge the micro-macro gap, showing how the complex DiDIY phenomena could emerge and be constituted;
- key features of DiDIY phenomena shown include (a) its ability to explore a wide range of possibilities, and (b) its ability to adapt to new ideas, patterns, and opportunities;
- the potential for applying this kind of modelling is significant.

References

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

Meyer, R. (2016). Factbase – a NetLogo Extension. Centre for Policy Modelling Discussion Papers, CPM-16-227. Retrieved from: http://cfpm.org/discussionpapers/154/factbase-a-netlogo-extension

R. Taylor and G. Morone (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

Wilensky, U. (1999). NetLogo. http://ccl.northwestern.edu/netlogo/. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL.