Synchronising Agent Populations when Combining Agent-Based Simulations

Bhagya N. Wickramasinghe
RMIT University
Melbourne, Australia
bhagya.wickramasinghe@rmit.edu.au

Dhirendra Singh
RMIT University
Melbourne, Australia
dhirendra.singh@rmit.edu.au

Lin Padgham
RMIT University
Melbourne, Australia
lin.padham@rmit.edu.au

ABSTRACT

In this paper we systematically investigate the problem of integrating two agent-based simulations (ABMs) that conceptually model the same agent population. The set up progressively steps the two simulations via an external controller, and synchronises the agent populations at the end of each step in such a way that they remain equivalent. Synchronisation involves changes to state variables only inside each simulation, resulting in additions and deletions to the list of agents, to agent properties, and to their relationships to each other. The key challenge is to ensure that the both the underlying simulations as well as the global simulation are always in a logically consistent and meaningful state. We present a conceptual framework to discuss these issues and solutions, in the context of integrating two substantially complex ABMs from literature. Our integration approach guarantees that the validated logic of the participating simulations is preserved.

Author Keywords

Agent-based Modelling, Integration, Synchronisation, Coupling

ACM Classification Keywords

I.2.11 Distributed Artificial Intelligence: Multiagent systems

INTRODUCTION

Agent-based models (ABMs) and social simulation are increasingly being used for understanding complex systems of interacting individuals, groups, and organisations, for such domains as land use [7] and residential choice [6] modelling. Since ABMs for a given domain typically capture similar or closely related concepts, is it often desirable, when building ABMs for a new problem in the domain, to `reuse` existing validated models as much as possible. In practice however, such reuse is not commonplace, due to a range of issues that make integration of ABMs challenging [18, 14].

In this paper, we dissect the problem of integrating two ABMs where the same conceptual agents exist in both simulations.

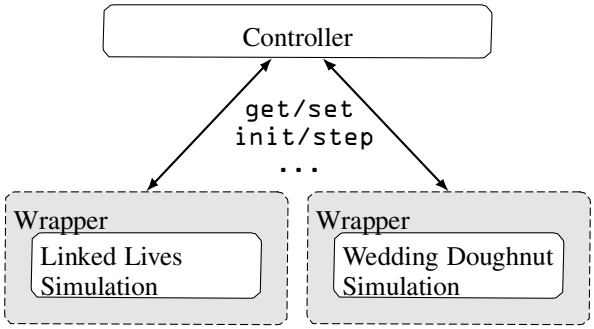

Figure 1. Architecture of the combined simulation

For this, we use two existing implementations – the Wedding Doughnut (WD) [16], an ABM on partnership formation, and Linked Lives (LL) [9], an ABM looking at how demographics impact changes in demand and supply of social care. These simulations, while being fairly complex in themselves, share several common concepts between them. Integrating these two simulations allows us to study the impact of social pressure induced marital partnerships on social care. Integration of two models also reinforces individual models and improves the reliability by extending their functionalities.

Our approach is to run the participating simulations independently, synchronising them at the end of each time step[1] using an external controller (such as OpenSim [18]), as shown in Figure . The intent is to leave the logic of the validated simulations untouched, and only use a control interface (e.g., functions to init/step each model) and data interface (e.g., functions to get/set shared variables), for monitoring and synchronisation.

In contrast to our previous work [18, 14] that looked at synchronising duplicated resources, in this work we describe synchronisation challenges with duplicated agents in a concrete setting. To our knowledge, this is the first report in literature to carefully examine the issues in synchronising duplicate agents across two simulations. Further, in this paper we present a conceptual framework to aid conversations about these general issues related to synchronising agents, and offer insights into possible solutions to these based on our investigation with the WD and LL ABMs.

[1]Both simulations are time-stepped, and each atomic step represents a duration of one year.

SpringSim-ADS, 2015, April 12 - 15, Alexandria, VA, USA

Existing integration standards like the High Level Architecture [2] and OpenMI [5, 8] are unsuitable for the kinds of issues that are specific to the integration of ABMs, such as those we discuss in this paper, despite offering an architecture similar to that of Figure . The key difference is that they do not allow duplication of resources or agents. [13] integrate two traffic simulators that cover mutually exclusive spatial areas in a map. 'Vehicle' agents in them can move from one simulated area to another but there is no notion of two duplicated/different instances of the same agent residing in both simulations at the same time. We cater for duplicity of resources and agents specifically, as this ability is essential if validated ABMs are to be coupled in this way. Multi-resolution modelling [3, 12] techniques also discuss some similar issues to what we discuss here. But, they mainly target coupling models that are similar but different in resolutions (e.g. [1, 4]). Our work focuses on developing new composite models by combining several different existing models.

The remainder of the paper is as follows: in the next section we describe WD and LL ABMs and their agents, their relationships to each other and to space, the modelled population dynamics, and the expected outcomes from integration. Using concrete examples from this integration, we then describe the identified problems under Issues In Synchronising Agents section. The 4th section, titled Conceptual Framework, presents a framework for talking about these integration challenges. Finally, we conclude with a discussion on solution.

EXAMPLE INTEGRATION

The Linked Lives (LL) [9] and Wedding Doughnut (WD) [16] ABMs have several concepts in common, such as processes modelling births and deaths in the population, partnership formation, and people relocating based on changed circumstances. Table 1 summarises these shared concepts between the two ABMs, as well as the expected functionality from the combined simulation.

Linked Lives Simulation

LL models the effect of changes in demographic structure on the demand and supply of social care. The simulation progresses in yearly time steps. The main concepts modelled in LL include reproduction, ageing, partnerships, divorces, relocations and deaths. Social care provision is evaluated based on distance between care needer (e.g. old parent) and care provider (e.g. daughter). The supply of social care is influenced by family structures, composition of a household and dispersion of family members over geopolitical divisions like towns.

LL encompasses both cohabiting and non-cohabiting relationships in its partnership formation model that pairs randomly selected couples based on age and gender specific probabilities. Agents that are in cohabiting relationships live together in the same house. Fertility is governed by a fixed probability for females between age 17 and 45. A newborn child is co-located in the same house as the mother. The information recorded for a new agent (child) includes both parents, age, and gender. Agents re-locate from one house to another for the following reasons: starting a cohabiting partnership, young adults moving out of parent's house, employment, searching for partners, or couples separating due to a divorce. Divorces occur according to a fixed annual age specific probability and results in the male agent moving to a new house. Mortality is modelled according to the Gompertz-Makeham model that stipulates an exponential increase in death rates with age. In case of death, the agent is marked as such and removed from the list of living agents and from the house it occupied. All agents with existing relationships with the agent are also updated by removing the relationship.

The spatial representation used in LL consists of 45 squares, representing the various townships, juxtaposed to form the shape of the UK, and where each square is a grid of size $25 * 25$ representing 625 houses in each town.

Wedding Doughnut Simulation

WD captures the role of social pressure in the formation of partnerships. Social pressure is proportional to the number of agents in a given agent's social neighbourhood who have existing partnerships. The higher the social pressure, the more likely a single agent is to form a partnership. The likelihood of two agents forming a partnership is proportional to the extent of overlap in their social connections. WD also models the demographic process of births, ageing, and deaths, and progresses in yearly time steps similar to LL.

WD captures cohabiting partnerships only. Births, and deaths, are modelled using a combination of historical statistical data and Lee-Carter demographic model. A newborn child is added to the list of children for the mother, and also the father in the case of cohabiting partnerships, and placed at a location adjacent to the mother. Females who are not in a cohabiting relationship can still get selected to give birth, and in these cases the identity of the father is not known. This group of females could potentially be in non-cohabiting partnerships or be single-mothers, but this distinction is not made explicit in WD. Deaths are registered by removing agents from the simulation and removing relevant relationship links in other agents.

The spatial representation used in WD is a 72×72 grid with wrapped boundaries resulting in a doughnut-shaped toroidal space. Conceptually, this captures the social network of all agents, where the distance between two agents is directly proportional to the strength of their relationship. Agent movements in this space are limited to occurrences of partnership formation. Two agents starting a partnership move to a location between them, where the distance moved is inversely proportional to the number of agents in their neighbourhood (agents move less when they have more social ties).

ISSUES IN SYNCHRONISING AGENTS

In the integrated simulation setup of Figure , conceptually the same agent population resides in both participating simulations. Our approach progresses the two simulations step-by-step using an external controller, while synchronising changes to the agent populations in the two simulations at the end of each step. This setup assists in retaining the validity of the independent ABMs within the integrated arrangement,

Concept	Linked Lives	Wedding Doughnut	Desired Combined Functionality
Mortality	Gompertz-Makeham	Lee-Carter/Census	Adopt Lee-Carter/Census
Fertility	Flat reproductive probability for females aged 17-45	Lee-Carter/Census	Adopt Lee-Carter/Census
Adoptions	Adopted by a random couple	Not modelled	Use as is from LL
Partnerships	Age/gender based probability; Cohabiting and non-cohabiting	Function of social connections; Cohabiting only	Adopt WD for cohabiting and LL for non-cohabiting partnerships
Divorces	Age specific probability	Not modelled	Adopt from LL
Movements	When seeking and forming/breaking partnerships, looking for employment, and reaching adulthood	When forming partnerships	Conceptually mapping two spaces so movements in each representation can be translated to the other
Space	2D (UK map)	Toroidal (Social connections map)	Avoid unrealistic spatial connections

Table 1. Comparison of concepts in LL and WD

since we make no changes to the models, other than those required to enable read and write access for the above operations. Synchronising involves updating in each simulation the list of agents, their properties, and their relationships, in such a way that overall the two populations are equivalent. We now describe in some detail the concrete challenges we experienced in this integration.

Births and Deaths

At a basic level, births and deaths in both ABMs involve similar operations: creating/destroying agent objects, and setting/clearing properties and relationships with other agents (such as mother-child relationship) and things (such as the relationship to the house occupied). However, synchronising changes due to births and deaths in the two populations is complicated by the fact that WD and LL use processes based on different statistical methods for simulating fertility and mortality (see Table 1), so the choice of which parents will give birth and which agents will die in a given year is different in each simulation. This leads to inconsistencies like differing numbers of births and deaths, or an agent being alive in one simulation but unborn or dead in the other.

One approach to handling this issue is to use the birth/death process from one simulation while disabling the other. In this case we chose the Lee-Carter/Census model from WD as the governing process for the integrated simulation, as it is considered to be the more informed method out of the two.

This choice, however, poses another subtle challenge for the case of births in WD where the females are not in cohabiting partnerships and the identity of the male partner/father is not available in the model. When synchronising such births from WD to LL, we are required to select a father and assign the father-child relationship, for LL to work correctly. We do not encounter this problem when synchronising newborns of cohabiting couples since both simulations represent them in a conceptually similar manner.

Overall, the solution provides an ability to *disable a competing process in one simulation*, such as by overwriting its effects, *perform additional bookkeeping*, such as by recording the history of partnerships, as well *compute and update states and relationships*, such as when selecting a suitable father/partner for a newborn when the male parent is unknown in WD.

Partnership formation and dissolution

WD and LL differ in the way partnerships are formed and dissolved. Whereas WD only models cohabiting partnerships, LL models both cohabiting and non-cohabiting partnerships. LL models divorces whereas WD does not. Finally, changes in partnerships can have knock-on effects. For instance, formation of partnerships in both models results in movements of those agents, representing physical relocation in LL, and a change in social ties in WD.[2] The outcome we would like for the integrated simulation is to use the more accurate WD model for cohabiting partnerships (it is informed by an agent's social connections rather than a fixed age/sex based probability), while using the LL model for non-cohabiting partnerships (WD has no concept of such relationships.)

Synchronising cohabiting partnerships from WD to LL introduces a new challenge. The formation of a cohabiting relationship in LL results in changes to the living arrangements of the couple such that they start living together, in the house that has the lesser number of occupants. While conceptually this is not problematic, there was the technical issue that the code that implements this logic in LL existed not as a separate function, but tightly coupled in a function that performs other agent movements. In this case there was no choice but to first *decouple this logic into a new function* and validate that this change did not alter the model's logic. This can be done in a fairly systematic way, by running existing tests for the ABM before and after the change and ensuring that the results remain unaffected.

Non-cohabiting relationships in LL may be migrated to WD by linking both agents as partners and moving them closer in the social networking space, but perhaps not as close as they would have been if they lived together. This calculation *requires some form of spatial approximation* using an appropriate technique informed by new domain knowledge, since the notion of non-cohabiting relationships, and therefore the

[2]We cover the issue of movements separately in next sub-section, Synchronising agent movements.

logic to compute movements for such, does not exist in WD. One way to implement this could be to first force the "normal" cohabiting movements for the pair by calling the appropriate function in WD, and then retracting the end points to a new position somewhere along the line joining their pre and post positions.

A related issue occurs in the case of dissolution of partnerships, when we try to migrate divorces in LL, to WD that does not model divorces. In LL, when a couple gets divorced, the male partner relocates, while any children remain with the female in their existing house. Translated to the social network space of WD though, moving only one of the partners is not accurate because separation affects the social network of both people. The challenge again is to approximate the new locations of the pair in this social space, and the calculation must necessarily be informed by new domain knowledge.

An undesired side-effect of introducing separation of couples in WD is that it can lead to inadmissible subsequent partnerships, such as between a parent and a child. This is because the model logic does not explicitly check that the prospective couple does not have a parent-child relationship.[3] This issue reinforces the need for *additional checks and balances* in the ABM wrapper to monitor and revert such cases.

Synchronising agent movements

The spatial representation in WD and LL captures social and physical landscapes respectively. Translation between these spaces therefore, needs to account for how geographical movements cause changes in social connections, and vice versa. In LL, agents move for various reasons like the formation or dissolution of cohabiting partnerships, employment, and reaching adulthood. The only existing equivalent in WD is movements when forming cohabiting partnerships. For meaningful integration, all other types of movements in LL should also be mapped to WD. As suggested in previous section, this can take the form of a spatial approximation, tuned by some parameter informed by domain experts.

The discussion so far assumes that we want to migrate changes in one model to the other. However, spatial conversions present a new kind of opportunity and challenge: of *accepting compatible movements from both spaces*. An example is when an agent relocates in LL due to employment, while it moves (adjusts its social landscape) in WD upon forming a partnership. In may be perfectly acceptable that both these types of events be allowed to occur in a given year. So as a result, in LL, the agent would relocate such that it co-resides with the new partner in the town where it is employed, while in WD the social landscape would be adjusted to account for both events (such as by moving to a point along the line joining the two end-points).

An artefact of the different spatial representations used in the two ABMs is that *boundary conditions may become problematic*. The issue comes from the fact that the LL space is a 2D grid with known boundaries, while in WD it is toroidal and therefore continuous. This can lead to configurations where two agents that are close to each other in WD, end up along opposite borders of the 2D grid when their coordinates are converted to the LL space. In turns out that this is quite acceptable in our case, as there is no underlying correlation between the relative distances in the two spaces. For instance, neighbouring families that live close to each other in LL do not necessarily have close ties, and ties to family members are often unaffected by physical distances.

[3]It is possible in WD for a single mother and male child of eligible age to form a partnership, with a very small probability. This is probably an oversight in the model. This effect is amplified when divorces are introduced, as it increases the instances of single mothers and male children.

CONCEPTUAL FRAMEWORK

In this section, we define the application independent problem of synchronising two agent populations, based on the problems that emerge from the previously discussed integration of WD and LL, and develop a general framework for framing such problems.

Synchronising instances of the same conceptual agent is an essential part in building a versatile methodology for integrating heterogeneous ABMs. As briefly explained earlier, the proposed architecture consists of wrappers that encapsulate the ABMs, and a central controller program to orchestrate the integrated simulation (Figure).

The first task in building the wrappers is handling restricted access issues related to state variables and simulation control in the ABMs. This process involves building an application programming interface (API) to the underlying ABM, with functions to access and update its state variables (getters and setters for agents and their properties), as well as control of the simulation (such as initialisation and stepping). Some refactoring of the code may also be required, such as separating out the code from a large function into several smaller ones that can be called individually by the controller. Importantly, these code changes should not alter the model logic, preserving the validation effort that went into creating it initially. We verify this by testing the ABMs in isolation before and after any code changes (using sensitivity analysis techniques), and confirming that the results are not altered in any way. After this step, we do not interfere with the ABM code in any way.

The wrapper does not intervene during step execution and its tasks are limited to before and after executing a simulation step. The controller maintains a mapping table that links duplicated instances of the same agent in the participating simulations. The controller is also responsible for resolving conflicts that arise because of incompatible changes in the simulations in a given time step. All integration logic resides in the wrappers and the controller.

The integration process involves linking the duplicated instances of the same agent in the participating simulations, then progressively stepping the models and synchronising changes in a way that all instances remain consistent across the system. Achieving consistency can be challenging because agent representations may be quite different in each simulation.

We define an agent state as a combination of *attributes, relationships* and *resources* that it shares with others. Agent states are typically implemented as variables that hold certain values. An *attribute* represents a concept that is local to the agent. For instance, in the WD and LL simulations, the concept of ageing is represented using a variable called **age** in the agent, which can hold integer values in a fixed range. *Relationships* represent the connections that an agent has with other agents in the simulation. This is how concepts of social structures such as parent-child relationships are captured. *Resources* are objects and things in an agent's external world and common to multiple agents in the simulation. For example, in LL, a house is a resource in the simulation that is shared by a family of agents.

Synchronising agent attributes

Synchronising concepts based on the agent attributes involves converting a set of attributes from one instance to another set of attributes in the other instance, and takes the general form of below equation, where X is the input and Y is the output set of agent attributes.

$$f(X) = Y$$

This facilitates conversions of the following types: one-to-one (e.g., translating coordinates between two spatial representations), one-to-many and many-to-one (e.g., between a population of **males** and **females**, and **people**), and many-to-many (e.g., between a population of **males** and **females**, and age categories like **children**, **adolescents** and **adults**). Conversions can result in loss of information when using less expressive representations (e.g., from continuous to discrete), partial representations (e.g., partnerships in WD are cohabiting only, while in LL they can also be non-cohabiting), performing dis-aggregations (e.g., one-to-many), and making approximations (e.g., interpolating or extrapolating values).

In the integration of WD and LL, we see the following kinds of conversions:

1. Conversion between different representations (one-to-one)

   An example is how death of an agent is represented in the two simulations. In LL it is a Boolean variable named **dead**, while in WD it is an integer variable called **yearOfDeath**. An example function that converts the concept of death from WD to LL is as follows (it sets **dead** to `True` in LL if the **yearOfDeath** is not 0 in WD, else sets it to false):

   $$dead_{LL} = \begin{cases} True, & \text{if } yearOfDeath_{WD} > 0 \\ False, & \text{otherwise} \end{cases}$$

2. Conversions to specialisations (one-to-many) or aggregations (many-to-one) of a concept

   For example, agent population can be captured by two integer variables as **male** and **female** in one simulation while another one captures the same concept using an integer variable named **people**. The formal representation of the function that adds the number of **male** and **female** agents to produce total number of **people** looks as follows:

   $$f(male_{s1}, female_{s1}) = people_{s2}$$

These particular conversions were relatively straightforward, but conversions requiring disaggregation (such as from people, to numbers of males and females) are necessarily more difficult as missing information must somehow be inferred.

Synchronising agent relationships

Representation of concepts in different simulations can have differences with respect to agent relationships. For instance, in LL, both maternal and paternal relationships are recorded for every newborn, whereas for newborns of single-mothers only the maternal relationship is recorded in WD since it does not explicitly model non-cohabiting partnerships. When synchronising births from WD to LL for single-mothers, we are required to find a suitable father for the child and form the father-child relationship, but also form the non-cohabiting partnership relation between the male and female if it does not already exist.

When translating a relationship between two agents from one simulation to another, first we have to identify second simulation counterparts of the two agents and the corresponding representation of the relationship. Once that relationship is created, the next step is creating other dependent relationships that are required to maintain consistency of the simulation. Eligible agents to form those relationships are identified based on requirements already defined in the model.

In a broader context, translation of relationships from one domain to another also needs to allow relationships to be deleted. In the case of WD and LL, marital relationships are dissolved in WD when agents divorce in LL. Though not covered in these two models, a divorce could also dissolve the relationship between an agent and its in-laws. The framework we presented here captures both additions and deletions of agent relationships in integrated ABMs. For deleting a relationship, we have to perform the same set of steps as earlier, but this time relationships are deleted instead of created.

Enforcing agent behaviours and model functions

In certain cases, a concept in an ABM is modelled as a phenomenon of several agent-behavioural functions, and model level functions that iterate through agents updating them. For instance, in LL, the concept of marital partnerships is captured by both **doMarriages** and **doMovingAround** behaviours, which change the agent's relationship status and location respectively. When an agent-behaviour is replaced by a more advanced implementation of the same concept, the integrated simulation has to execute other auxiliary behaviours and functions in order to maintain consistency. For instance when replacing LL marriage with WD's implementation, which only updates the relationship status of cohabiting couples, we have to relocate them to a common house by enforcing LL agent-instances' movements related to marital partnerships, so they actually cohabit in LL's world.

Enforcing an agent-behaviour or a model function requires isolating the target logic. If the target logic is in an independent function, it is a simple case of calling that function

from the model wrapper. However, if the target logic is only a small portion of a larger function, which is the case for LL agent movements, we have to re-factor the code by extracting target logic into model wrapper and replacing existing code with a statement calling the newly created function in the wrapper. Then the newly created function can be called arbitrarily without affecting the other agent movements. After re-factoring the model can be validated by running existing tests to ensure that there are no alternations in the model's logic.

The problem of synchronising cohabiting partnerships of WD and LL can be solved according to the above framework, by extracting the code related to relocating LL cohabiting couples in to the LL wrapper. This feature is also of benefit in finding a partner for new single mothers who have never been in a relationship. This approach advocates minimum possible code changes within a component simulation, ensuring no functional or logical changes within a component, observable to model users[4].

DISCUSSION

Given the conceptual framework of previous section that discusses synchronisation of duplicate agents via adjustments to their states and relationships, let us now see how these ideas can be applied to solve some of the problems we described under section Issues In Synchronising Agents.

The general process of migrating changes from simulation A to simulation B involves *(i)* identifying agents (by checking their properties) in both simulations which exhibit the change in question, *(ii)* reverting the change (and side-effects) in simulation B, and *(iii)* applying simulation A changes (and side-effects) to simulation B.

Consider the case where we want to migrate deaths from WD to LL as outlined in Table 1. Mortalities are identified in WD by selecting all agents who have changed their **yearOfDeath** variable value to the current year, while in LL it involves checking if an agent's **dead** Boolean variable has been set to `true`. Reverting deaths in LL therefore involves setting those **dead** variables back to `False`, as well as reverting side-effects like adding a revived agent back to the list of **livingPeople**, as well as the **occupants** list of the house it occupied, and if it had a partner, then setting its partner's **partner** variable to point back to the agent. Applying deaths from WD to LL involves finding the corresponding agent in LL and performing relevant changes to the same variables.

In the case of introducing divorces to WD, the separating couple's **partner** variable is set to `null` and their **location** updated to reflect the separation of their social networks. For instance, they could be relocated to their locations in the social landscape prior to their marriage, or moved to a location close to a majority of their old connections (who may have moved in the time being). The function for computing the new locations, as well as any additional data required for that computation (such as previous locations), resides in the WD wrapper (Figure), and as before, the desired outcome is achieved only by injecting new values into the state variables of the simulation. At noted before, the wrapper code also checks for inadmissible instances of parent-child partnerships after each simulation step, and reverts any that are found.

An example of synchronising relationships is when a child is born. Consider when we want to migrate the birth of a child in WD, to LL. For cohabiting relationships, this is trivial, as there exists a one-to-one mapping in LL, for the mother-child and father-child relationships in WD. For children of single-mothers the father is unknown in WD. If the female has a known partner in LL (i.e., she is in a non-cohabiting partnership there), then the male is selected as the father. Otherwise, we consult the lists of last known partners that we maintain in the model wrappers, and if a match is found, assign that male as the father (and also form a non-cohabiting partnership between the male and female). In the final case where there is no known partnership for the female, we invoke the non-cohabiting match-making functionality in LL to find a suitable partner for the female, and then assign this male as the father.

The relationship between the spatial representations – the social landscape in WD and a physical landscape in LL – is indirect and not easily quantifiable, because there is no clear way to correlate the relative placement of agents in the social space to their relative placement in the physical space. Incidentally, when the simulations are initialised, we copy the WD population to LL, and place the agents randomly in the physical space. Similarly, when couples form a cohabiting relationship, they move closer in their social space and start cohabiting in the physical space. However, there is no correlation between the extents of the movements in the two spaces, since the same degree of movement in WD can coincide with a relocation of arbitrary distance in LL. Given this, our approximation for mapping movements between the two spaces works as follows: when migrating cohabiting partnership formations from WD to LL, we relocate the agents such that they both reside in one of their houses (the one with the lesser number of occupants), but do not compute movements in LL for non-cohabiting partnerships; when migrating divorces from LL (where the male relocates to a random address) to WD, we adjust the locations of the pair in the social space such that they both move closer to the locations they occupied before they formed a relationship (and therefore away from each other).

In our previous works we have looked at building simulations from parts using specialised modules such as for human modelling and traffic simulation [11, 10], explored the specific issue of sharing resources between ABM modules in a combined simulation [14, 17], and proposed two alternative controllers for such integrations [15, 18]. In those works, as in this one, *our integration approach has been to keep the validated logic of the ABMs intact*, and instead use a data and control interface for accessing and updating state variables, and controlling the participating simulations via wrap-

[4]The concept of no observable change applies to the component only when it is executed by itself. Of course, when integrated with other components, which add new complexity, the outcomes within a component may also change.

pers (Figure). This allows the integration logic to exist solely in the controller and the ABM wrappers, which facilitates retaining the validity of the participating ABMs. We do make modifications to the ABMs, for data and control interfacing and to refactor code if needed, but check that the logic remains unchanged.

Integrated simulations built adhering to the proposed architecture are easily extensible because wrappers hide implementation details. For instance, implementation level differences of LL (a single threaded Python program) and WD (a multi-threaded Java program) were never considered as issues during the integration process. Communication between the central coordinator program and wrappers is encoded in platform independent JSON (XML is another option) and messaging is done via network connections between the controller and the wrappers. This allows any new model to be integrated with minimum adjustments to the existing integrated simulation. This architectural approach of integrating with wrappers is similar to the HLA, and our own OpenSim [18].

The example integration of WD and LL also highlights the futility of the alternative approach of building a composite ABM from scratch, by incorporating the logic of the participating ABMs. First, that would require a thorough understanding of the implementation details of both ABMs. Second, the relevant code and functions could not be literally imported, since the two models are written in different programming languages, Python and Java. To address this, one would have to convert the program from one programming language to the other. Third, in contrast to LL which is a single-threaded program, WD is a multi-threaded program with a thread for each agent. In this case parts of the code would have to be redesigned, including data structures and execution flows, to fit the threading paradigm chosen. Fourth, one would still need to address some of the issues we have discussed in this paper, such as combining the spatial representation in a meaningful way. Finally, future extensions by adding functionality from new ABMs become even more complicated, since addition of new functionality requires retracing most of the previous steps.

Our work in this paper has dissected the intricate issues in coupling existing ABMs with duplicate agent populations. We showed why such couplings can be challenging, by looking at the specific integration of the Wedding Doughnut (WD) and the Linked Lives (LL) simulations. It is our strong belief that it is precisely these kinds of issues that severely hinder reuse of ABM systems. We have presented a conceptual framework within which we can frame these issues and discuss solutions. Our ongoing work in this area aims to build the tools and related methodology to support such reuse.

REFERENCES

1. Baohong, L. A Formal Description Specification for Multi-Resolution Modeling Based on DEVS Formalism and its Applications. *The Journal of Defense Modeling and Simulation: Applications, Methodology, Technology 4*, 3 (July 2007), 229–251.
2. Dahmann, J., Fujimoto, R., and Weatherly, R. The DoD High Level Architecture: an update. In *Winter Simulation Conference. Proceedings (Cat. No.98CH36274)*, vol. 1, IEEE (1998), 797–804.
3. Davis, P., and Hillestad, R. Families of Models that Cross Levels of Resolution: Issues for Design, Calibration and Management. In *Proceedings of 1993 Winter Simulation Conference - (WSC '93)*, IEEE (1993), 1003–1012.
4. Davis, P. K., and Bigelow, J. H. *Experiments In Multiresolution Modeling (MRM)*. 1998.
5. Gregersen, J. B., Gijsbers, P. J. a., and Westen, S. J. P. OpenMI: Open modelling interface. *Journal of Hydroinformatics 9*, 3 (July 2007), 175.
6. Huang, Q., Parker, D. C., Filatova, T., and Sun, S. A review of urban residential choice models using agent-based modeling. *Environment and Planning B: Planning and Design 41* (2014), 661–689.
7. Matthews, R. B., Gilbert, N. G., Roach, A., Polhill, J. G., and Gotts, N. M. Agent-based land-use models: a review of applications. *Landscape Ecology 22*, 10 (2007), 1447–1459.
8. Moore, R. V., and Tindall, C. I. An overview of the open modelling interface and environment (the OpenMI). *Environmental Science & Policy 8*, 3 (June 2005), 279–286.
9. Noble, J., Silverman, E., Bijak, J., Rossiter, S., Evandrou, M., Bullock, S., Vlachantoni, A., and Falkingham, J. Linked lives: The utility of an agent-based approach to modeling partnership and household formation in the context of social care. In *Winter Simulation Conference (WSC)*, no. 2011, IEEE (Dec. 2012), 1–12.
10. Padgham, L., Nagel, K., Singh, D., and Chen, Q. Integrating BDI Agents into a MATSim Simulation (2014). 1–6.
11. Padgham, L., Scerri, D., Jayatilleke, G., and Hickmott, S. Integrating BDI Reasoning into Agent Based Modeling and Simulation. In *Winter Simulation Conference* (2011), 345–356.
12. Reynolds, P. F., Natrajan, A., and Srinivasan, S. Consistency maintenance in multiresolution simulation. *ACM Transactions on Modeling and Computer Simulation 7*, 3 (July 1997), 368–392.
13. Rieck, D., Schünemann, B., Radusch, I., and Meinel, C. Efficient traffic simulator coupling in a distributed V2X simulation environment. In *Proceedings of the 3rd International ICST Conference on Simulation Tools and Techniques*, ICST (2010).
14. Scerri, D., Drogoul, A., Hickmott, S. L., and Padgham, L. An architecture for modular distributed simulation with agent-based models. In *Proceedings of Autonomous Agents and Multi-Agent Systems (AAMAS)* (2010), 541–548.

15. Scerri, D., Hickmott, S., Zambetta, F., Gouw, F., Yehuda, I., and Padgham, L. Bushfire BLOCKS: a modular agent-based simulation. In *Proc. of 9th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2010)*, no. AAMAS 2010 (2010), 9–10.
16. Silverman, E., Bijak, J., Noble, J., Cao, V. D., and Hilton, J. Semi-artificial models of populations: connecting demography with agent-based modelling. In *World Congress on Social Simulation*. Taipei, Taiwan, 2012.
17. Singh, D., and Padgham, L. A Rollback Conflict Solver for Integrating Agent-based Simulations. In *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems* (2014), 1399–1400.
18. Singh, D., and Padgham, L. Opensim: A framework for integrating agent-based models and simulation components. In *Proceedings of the European Conference on Artificial Intelligence (ECAI)* (Prague, Czech Republic, Aug. 2014), 837–842.