Proceedings of the 2010 Winter Simulation Conference
B. Johansson, S. Jain, J. Montoya-Torres, J. Hugan, and E. Yücesan, eds.

DIVIDE AND CONQUER: A FOUR-FOLD DOCKING EXPERIENCE OF AGENT-BASED MODELS

S. M. Niaz Arifin
Gregory J. Davis
Steve Kurtz
James E. Gentile
Ying Zhou
Gregory R. Madey

Department of Computer Science & Engineering
University of Notre Dame
384 Fitzpatrick Hall
Notre Dame, IN 46556, USA

ABSTRACT

Verification and validation (V&V) techniques are used in agent-based modeling (ABM) to determine whether the model is an accurate representation of the real system. Docking is a form of V&V that tries to align multiple simulation models. In a previous paper, we described the docking process of an ABM that simulates the life cycle of *Anopheles gambiae*. Results showed that the implementations were docked for *adult* but not for *aquatic* mosquito populations. In this paper, following the 'Divide and Conquer' paradigm, we compartmentalize the simulation world to prohibit the propagation of errors between compartments. Using four separate implementations that sprung from the same core model, we describe a series of docking experiments, analyze the results, and show how they lead to a successful dock. The complete four-fold docking encompasses *verification* between the four implementations, as well as *validation* against the core model with respect to these implementations.

1 INTRODUCTION

Verification and validation (V&V) techniques are used in agent-based modeling (ABM) to determine whether the model is an accurate representation of the real system. The primary goal of verification is to compare multiple models with each other in order to ensure that the implementations are correct. Validation ensures that the model is semantically right, and it is indeed modeling the phenomena of the real system being simulated. **Docking**, also known as alignment or replication, is a form of V&V that tries to align multiple models in order to investigate whether they yield similar results. Docking is useful to confirm whether the claimed results of a given simulation are reliable, and can be reproduced by someone starting from scratch.

1.1 Background

In a previous paper (Arifin, Davis, Zhou, and Madey 2010), we described the verification and validation (V&V) process of an agent-based model (ABM) that simulates the life cycle of *Anopheles gambiae*. *Anopheles gambiae* is regarded as the primary vector for transmission of Malaria, which is one of the top three pathogen-specific causes of global mortality. Following a fixed version of the core (conceptual) malaria entomology model, we developed two agent-based implementations using Java and C++, the former having different versions which were developed in different phases. Both implementation simulated the life cycle of *Anopheles gambiae* by tracking attributes relevant to the vector population dynamics for each individual mosquito. We described the verification performed between the implementations, and the validation performed against the core model with respect to these implementations. We also described how the major findings helped to clarify concepts and eliminate ambiguities by identifying differences in model specification, interpretation, implementation and enhancement phases, and revealing semantic errors. The importance of rigorous docking was illustrated by the discovery of some incorrect assumptions and programming

errors, which, being unnoticed, led to erroneous results. However, as evident from the results of (Arifin, Davis, Zhou, and Madey 2010), the two implementations were not completely 'docked', especially in terms of the *aquatic* mosquito populations. The reason for this is explained below.

In a complex ABM, synergies arising from seemingly insignificant differences in separate implementations may lead to significant mismatch in overall model output. Hence, it is crucial, and sometimes necessary, to 'compartmentalize' the artificial simulation world, i.e. to separate it into isolated compartments so that errors in one specific compartment are not propagated and thus cannot influence the discovery (and correction) of errors in other compartments.

In this paper, we compartmentalize the mosquito world with respect to the adult and aquatic populations, and perform some experiments to identify and fix the sources of output mismatches in order to achieve a complete 'dock'. Following the 'Divide and Conquer' paradigm, we ensure that the pieces are working as intended, and then combine them to perform some other, more complex experiments. Using four separate implementations that sprung from the simplified core malaria entomology model, we describe the design of these experiments, analyze their results, and show how they lead to a successful dock between all four implementations, and hence achieve a complete four-fold docking of the core model.

This paper particularly addresses the following issues:

- How the results from four agent-based implementations compare to each other and to the corresponding results from theoretical models
- How to verify the age-structure of mosquito populations and the age-specific mortality rates (for both adults and aquatic populations)
- How to verify the oviposition mechanism and the calculation of *one-day old equivalent larval population* N_e at each step of the simulation (see Section 3.5.3), by removing all randomness (e.g. with fixed number of eggs to lay, single aquatic habitat, etc.)

The complete four-fold docking encompasses *verification* between the four implementations, as well as *validation* against the core model with respect to these implementations. The major findings, as described in Tables 2, 3, and 4, help to clarify model concepts and eliminate ambiguities.

This paper also serves as a case-study to illustrate the importance of docking complex simulations, since even the best simulation programs may imply dubious assumptions, leading to erroneous results. Rigorous docking can detect such problems, uncovering these incorrect assumptions (Will 2009). The four-fold docking process serves the dual purpose of increasing confidence to the core model and revealing conceptual errors in model implementations.

1.2 Paper Organization

The organization of this paper is as follows: Section 2 discusses some previous works involving docking. Section 3 discusses one fixed version of the core malaria model and its two slightly different variations used in this paper. Section 4 briefly describes the four separate implementations and the V&V workflow. Section 5 discusses the design of docking experiments, which are performed in three phases. For each phase, we analyze the results and identify issues to be updated for experiments in subsequent phases. We also show how the results lead to a successful dock between all four implementations, and include two additional experiments. Finally, section 6 concludes.

2 LITERATURE REVIEW

In this section, we re-emphasize two recent works that show the importance of rigorous docking for agent-based models. Edmonds and Hales (Edmonds and Hales 2003) replicated a published model involving co-operation between self-interested agents in two independent implementations to align the results and the conceptual design. The replication revealed a host of minor bugs and ill-defined implementation issues that otherwise appeared unnoticed. They concluded that aligning models can be very difficult, but very revealing, because simply implementing simulations with respect to a conceptual model and then analyzing outputs for consistency with the conceptual model and data series is insufficient to ensure the correctness of an implementation.

Will and Hegselmann (Will and Hegselmann 2008) showed the importance of replication by reporting a failure to replicate the results presented on a published model. Obtaining the source code from the original authors, Will (Will 2009) found that the simulation unintentionally implemented an assumption that was never mentioned in the

original model, and showed that the model crucially depended on that dubious assumption, and its removal leaded to dramatically different results.

This paper serves as a case study of docking multiple ABMs. It shows how to achieve a complete dock between multiple implementations, all of which are built on a sufficiently complex core model. It also highlights the importance of docking by showing that a successful dock may reveal conceptual and/or programming errors, and may eliminate dubious assumptions, reinforcing the findings in (Carley 2002), (Edmonds and Hales 2003), (Rand and Wilensky 2006), (Will 2009), and (Will and Hegselmann 2008).

3 THE AGENT-BASED MODELS

In this section, we briefly describe the fixed version of the simplified core (conceptual) malaria entomology model, and the four different implementations.

3.1 The simplified core models

The core model is primarily governed by the biology underlying *Anopheles gambiae*. However, for the purpose of docking, some simplifying assumptions have been made, which may deviate from biological plausibility. For different experiments performed in this paper (see Section 5), we use two slightly different versions of the core model. The major difference, as shown in Figure 1, is whether to use a single vs. multiple aquatic habitats.

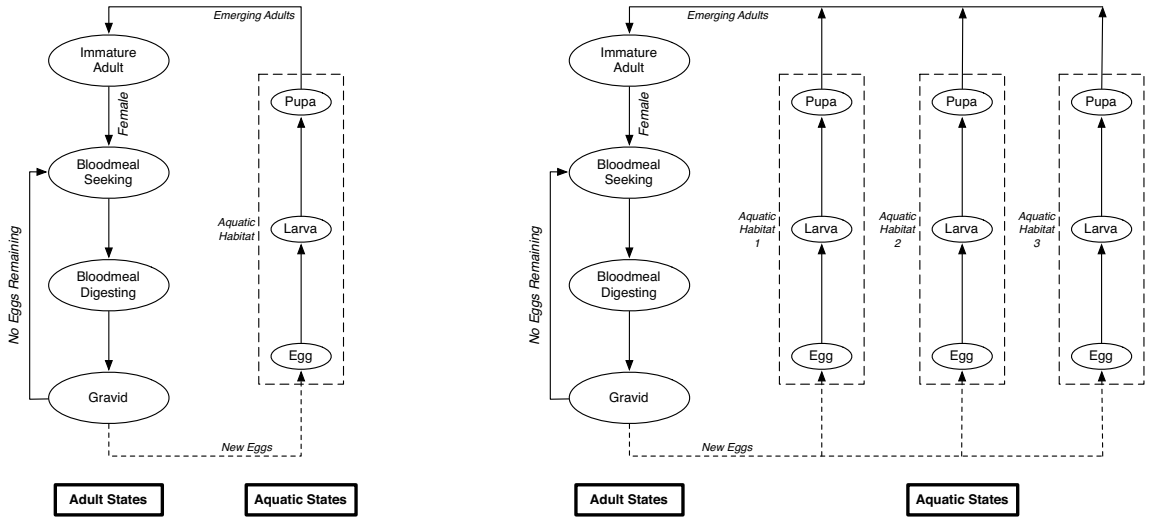

Figure 1: The simplified *Core Model 1* (on left) and the simplified *Core Model 2* (on right). *Core Model 1* has a single aquatic habitat and is used in experiments *E1*, *E2* and *E3*. *Core Model 2*, designed for experiments *E4* and *E5*, has three aquatic habitats.

The state durations for mosquito agents are defined in Table 1. One important, though biologically unnatural, assumption is illustrated in the state duration for the *Gravid* state: the duration (in days) that the female mosquito is forced to remain as *Gravid* equals the number of oviposition attempts taken to successfully lay all her remaining eggs.

3.2 The Agents

Anopheles gambiae mosquitoes are the only mosquito agents in the model. Like all other mosquito species, anophelines go through four major states in their life cycle: egg, larva, pupa, and adult (see Figure 1). The first three states are collectively termed as 'aquatic'. The adult state of the female mosquito is when the female is able to be a vector of the malaria parasite. Being autonomous agents, mosquitoes begin their life as eggs (except the initial adults created at the start of the simulation) and progress through various life cycle states.

Table 1: State durations for the mosquito agents. States marked with * apply only to *female* mosquito agents.

State	Duration (in Days)
Egg	1
Larva	Temperature-dependent; see Equation (1).
Pupa	1
Immature Adult	2
Bloodmeal Seeking*	1
Bloodmeal Digesting*	2
Gravid*	Stay until all eggs are laid; then stay 1, 2 or 3 days depending on the number of attempts taken that day.

3.2.1 Aquatic Mosquitoes

Eggs and pupae transition out of their current state after a 24-hour period. The growth and development of larvae, however, is much slower and depends on daily temperature. Following the stochastic thermodynamic models of Sharpe and DeMichele (Sharpe and DeMichele 1977) and Schoolfield et al. (Schoolfield, Sharpe, and Magnuson 1981), we use the equation

$$Dev_{day} = (Temp_{day} * 0.000305 - 0.003285) * 24 \quad (1)$$

where Dev_{day} is the larval development rate and $Temp_{day}$ is the average temperature (in °C) of that day. The threshold for the *Larva* $\rightarrow$ *Pupa* transition is deterministically set to 1.0 removing any randomness (see Section 5.1). Each day, each larva approaches its threshold by developing at a rate characterized by equation (1).

3.2.2 Adult Mosquitoes

For this paper, only *female* mosquitoes are considered, and all male mosquitoes are omitted. This allows an unbiased, uniform selection of agents from separate age-groups when the mosquito agents die out. The females feed on bloodmeals, lay eggs, and go back to feed on bloodmeals again. This is repeated until a female dies. Once a female enters the *Gravid* state, it remains there until all of her eggs are laid, which are then instantiated as new agents into the system. The states are described below.

In ***Immature Adult***, a female mosquito emerges as an immature adult after its aquatic development is complete. This is the starting state of the ABM where new mosquitoes are initially placed into the system and also when pupae turn into adults. In ***Bloodmeal Seeking***, the female seeks and finds a bloodmeal. In ***Bloodmeal Digesting***, the female rests while the blood is digested and eggs are developed. In ***Gravid***, the eggs are developed, and the female is ready to lay them. The maximum number of eggs a female may oviposit, $Eggs_{max}$, is deterministically set to 80, removing any randomness. All eggs are set to be females. Once a female enters the *Gravid* state, it remains there until all of her eggs have been laid, and then transitions back to the *Bloodmeal Seeking* state.

3.3 The Environment

The environment provides an artificial realm in which agents operate. Variables such as temperature and humidity factors, defined within the environment, govern how the mosquito agents develop. The aquatic states of *An. gambiae* require an ***aquatic habitat*** environment to develop, and are often found in transient, sunlit and small pools. These aquatic habitats control the development of aquatic mosquitoes and the oviposition rate for female adults, as described below. The ***carrying capacity*** of an aquatic habitat represents the repulsive force sensed by a *Gravid* female and limits the number of eggs she may oviposit in that habitat. Though not treated as a hard limit, this is an indication to the female that the habitat is full, and it may be more beneficial to lay eggs elsewhere.

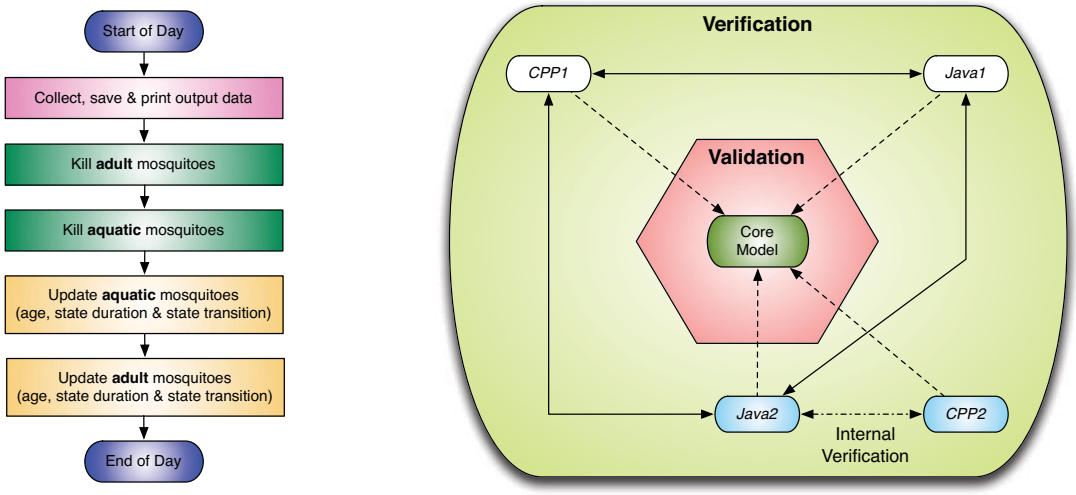

Figure 2: The daily processing order of the simulation (on left) and the V&V workflow (on right). In the workflow, bidirectional arrows indicate *Verification* relationships between the four implementations. The dashed arrow, between the re-factored implementations *Java2* and *CPP2*, indicates *Internal Verification*, since these two are docked with respect to each other. Unidirectional, dashed arrows indicate *Validation* relationships between the *Core Model* and the four implementations.

3.4 Age-specific Mortality Rates

In any ABM, the mortality of agents plays a crucial role in shaping the model's characteristics. Traditionally, ABMs that model malaria transmission assume age-independent vector mortality (i.e. non-senescence of the vector). However, as shown by recent studies (Styer, Carey, Wang, and Scott 2007), this leads to misleading predictions, and obscures age-dependent aspects of mosquito biology. This led us to use the concept of Age-Specific Mortality Rates (ASMRs) for the adults and larvae states. For these states, ASMR refers to the total number of deaths per day of a given age. We use a modified version of the Gompertz function, making the age-dependent component increase exponentially with age.

3.5 Population Dynamics

Mosquito populations are dynamic in nature. The population dynamics is governed by three key operations performed at each simulation day: killing adult mosquitoes, killing aquatic mosquitoes, and creating new mosquitoes.

3.5.1 Daily Processing Order

The present model discretizes time such that every agent is updated during a time step representing a single day (24-hour period). The major daily processing steps, modified and hence different from (Arifin, Davis, Zhou, and Madey 2010), are shown in Figure 2 (left).

3.5.2 Killing Adult Mosquitoes

As noted in Section 3.4, adult mosquitoes have a daily age-specific mortality rate. Newly emergent adults begin with a daily mortality rate of α. As the mosquito ages, the age-specific mortality rate $ASMR_{Age(adult)}$ for that age group changes based on the following equation:

$$ASMR_{Age(adult)} = \frac{\alpha * e^{\frac{Age}{\beta}}}{1 + \alpha * s * \beta (e^{\frac{Age}{\beta}} - 1)} \tag{2}$$

where α is set to the baseline daily mortality rate, β and s are constants defining the rate of increase and asymptote, respectively. The number of adults with a given age that are removed from the system, $ToKill_{Age(adult)}$, is then computed by:

$$ToKill_{Age(adult)} = ASMR_{Age(adult)} * Adults_{Age} \tag{3}$$

3.5.3 Killing Aquatic Mosquitoes

Aquatic mosquitoes (in the egg, larva, and pupa states) have different mortality rates. Both eggs and pupae have a constant mortality rate of 10%. Larval mortality rates, however, are dependent on both the age and the notion of *one-day old equivalent larval population*, N_e. Within a given aquatic habitat, N_e is computed by:

$$N_e = \sum_{Age=0}^{MaxAge} Age * Larvae_{Age} \tag{4}$$

where $MaxAge$ is the age of the oldest larvae group in the system. N_e is then used to calculate the age specific mortality rate for larvae using the following equation:

$$ASMR_{Age(larvae)} = \alpha * e^{\frac{N_e}{Age*C*R}} \tag{5}$$

where C is the carrying capacity of the aquatic habitat (see Section 3.3) and R is the rainfall coefficient (for now, we assume a rainfall coefficient of 1.0). The number of larvae with a given age that are removed from the system, $ToKill_{Age(larvae)}$, is then computed by:

$$ToKill_{Age(larvae)} = ASMR_{Age(larvae)} * Larve_{Age} \tag{6}$$

The *Biomass* of the aquatic habitat is defined as:

$$Biomass = Eggs_{habitat} + N_e + Pupae_{habitat} \tag{7}$$

where $Eggs_{habitat}$ and $Pupae_{habitat}$ are the number of eggs and pupae, respectively, in the selected habitat.

3.5.4 Creating New Mosquitoes

Beyond the initial creation of adult mosquitoes, new mosquitoes are created when females in the *Gravid* state visit an aquatic habitat and oviposit. Each day, *Gravid* females make a maximum of *three* attempts to lay the entirety of eggs. However, the potential number of eggs, $Eggs_{potential}$, that a female can actually lay in a given habitat is limited by the *Biomass* already present in the habitat:

$$Eggs_{potential} = Eggs_{max} * \left(1 - \frac{Biomass}{i * C}\right) \tag{8}$$

where $Eggs_{max}$ is the maximum number of eggs available to lay (see Section 3.2.2), N_e is the one-day old equivalent larval population of the habitat (see Section 3.5.3), i is the oviposition attempt number (i.e. 1, 2, or 3), and C is the carrying capacity of the habitat. If $Eggs_{potential} > Eggs_{max}$, then all available eggs are laid and the female transitions back to the *Bloodmeal Seeking* state. Otherwise, $Eggs_{max}$ is reduced by $Eggs_{potential}$ and another oviposition attempt occurs. If $Eggs_{max} \neq 0$ after three attempts, the female stays another day in the *Gravid* state, attempting to lay the remaining eggs.

4 Model Implementations

In this section, we briefly describe the four different implementations. Figure 2 (right) shows the V&V workflow, illustrating logical connections between the implementations and the core model.

CPP1, described in detail in (Zhou, Arifin, Gentile, Kurtz, Davis, and Wendelberger 2010), is the earliest implementation of the core model. It is developed in the C++ programming language utilizing the GNU Scientific Library (GSL) (Galassi, Davies, Theiler, Gough, Jungman, Booth, and Rossi 2003). This implementation is written to be portable, compiling and running on any platform for which an ANSI compliant C++ compiler is available. A major drawback of this implementation is the lack of built-in visualization tools.

The first Java implementation, *Java1*, is built over three phases using Java. It provides some key advantages such as built-in graphical visualization tools that allow the agents to be readily inspected (which aids in the debugging process), high efficiency and less error-prone code. For details about *Java1*, see (Arifin, Davis, Zhou, and Madey 2010).

The re-factored implementations, *CPP2* and *Java2*, reflect a unified architecture, and encode a mosquito's life cycle and behavior in a structure called a *strategy*. The strategy is flexible and can adapt to characterize new genus, species or variation within one species. The architecture is well-suited for parallelization across many cores or computers. For details, see (Gentile, Davis, Laurent, and Kurtz 2010). For the purpose of docking, *CPP2* and *Java2* are verified with respect to each other (internal verification), and a single output is compared with those of *CPP1* and *Java1*.

5 EXPERIMENTS AND RESULTS

In this section, we describe the docking experiments and their results. For brevity, experiments are labeled as ***E1***, ***E2***, and so on. We also describe the rationale behind designing these experiments from the docking point of view.

5.1 Model Assumptions

Experiments described in this paper make the following assumptions:

- All randomness are removed from the simulation in *E1*, *E2*, *E3* and *E4*. This allows direct model-to-model docking, and to compare the results against those of theoretical models.
- Carrying capacity of the aquatic habitat is reduced to 3000 (in *E1*, *E2* and *E3*), and to 3000, 4000, 5000 (in *E4* and *E5*; see Section 5.5).
- The *Mate Seeking* state is omitted altogether for simplicity.
- The oviposition mechanism has been simplified.
- All males are omitted from all experiments to allow uniform killing of agents from separate age-groups.

In *E1* and *E2*, following the 'Divide and Conquer' paradigm, we compartmentalize the core model with respect to the adult and aquatic populations. *E3* combines the adult and aquatic populations.

5.2 Experiment 1 (*E1*): Isolating *Adult* Mosquito Populations

E1 deals with the adult mosquito populations by isolating it from the aquatic mosquito populations. We use 2, 2 and 0 days as the state durations for *Immature Adult*, *Bloodmeal Seeking* and *Bloodmeal Digesting*, respectively. The female stays in the *Gravid* state until all eggs are laid, and transitions back to *Bloodmeal Seeking* the following day (see Figure 1). Starting with 100 female adults, *E1* ensures that the age-structure and age-specific mortality rates (for the adult states) match to the theoretical values. It also verifies the simplified oviposition mechanism.

5.3 Experiment 2 (*E2*): Isolating *Aquatic* Mosquito Populations

E2 deals with the aquatic mosquito populations by isolating it from the adult mosquito populations. Starting with 1000 female eggs that initially reside in the single aquatic habitat (with 3000 carrying capacity), it ensures that the age-structure of all aquatic states, the age-specific mortality rates of larvae, and the base mortality rates of eggs and pupae populations match to the theoretical values. It also verifies the temperature-dependent larval development rate (see Equation (1)).

5.4 Experiment 3 (*E3*): Combining Adult and Aquatic Mosquito Populations (Single Aquatic Habitat)

E3 combines the adult and aquatic mosquito populations. Starting with 100 female adults and 1000 female eggs (initially residing in the single aquatic habitat with 3000 carrying capacity), it verifies the transitions from aquatic to adult (i.e. from *Pupa* state to *Immature Adult* state) population. It also checks the oviposition mechanism, including actual number of eggs laid in the aquatic habitat, and the state durations (see Table 1).

E3 is performed in three separate phases as described below. After each phase, we compare the four-fold outputs, analyze and fix potential misinterpretations, and proceed to the next phase.

5.4.1 *Phase 1*

Figures 3 and 4 show the results of *Phase 1*. Table 2 discusses the issues discovered after analyzing these results.

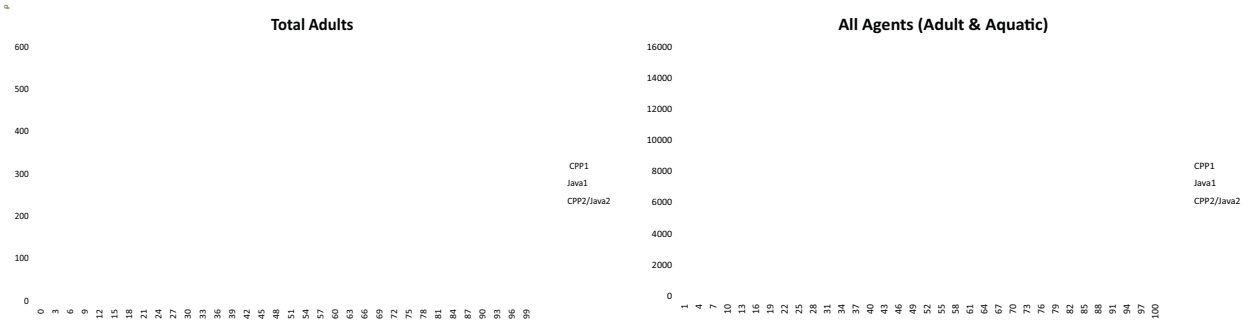

Figure 3: Four-fold docking results for *E1, E2* and *E3 (Phase 1)*. The graph on left shows the total number of (female) adults, and the graph on right shows the total number of agents (adults and aquatic). The *x*-axis denotes simulation days, and the *y*-axis denotes mosquito abundance.

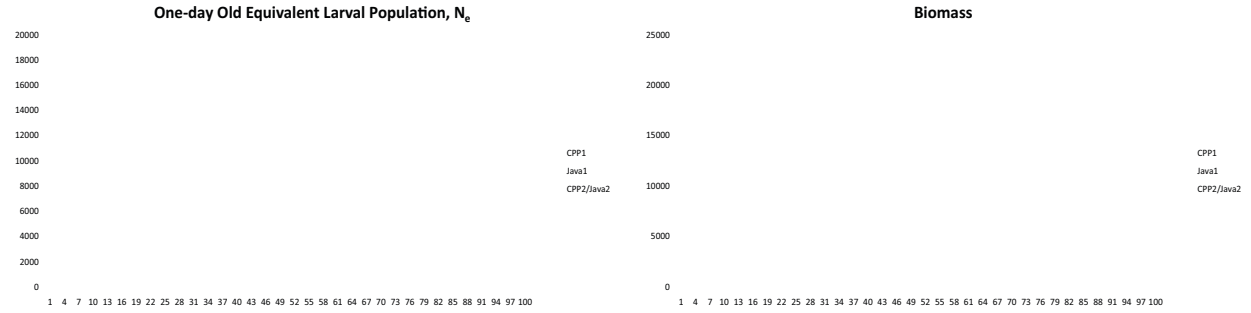

Figure 4: Four-fold docking results for *E1, E2* and *E3 (Phase 1)*. The graph on left shows the *One-day Old Equivalent Larval Population, N_e* (see Section 3.5.3). The graph on right shows the total biomass in the aquatic habitat. The *x*-axis denotes simulation days, and the *y*-axis denotes mosquito abundance.

5.4.2 *Phase 2*

Figures 5 and 6 show the results of *Phase 2* after the issues found in *Phase 1* are addressed. Table 3 discusses the issues discovered after analyzing these results.

Table 2: List of issues discovered and updated in *Phase 1*

Issue	Resolution
In *CPP1* and *Java1*, the adult populations are slightly less in number than those in *Java2* and *CPP2*. Also, the aquatic populations in *CPP1* and *Java1* are killed at a higher rate than suggested by the theoretical numbers.	Resolved after all implementations use the same carrying capacities for the aquatic habitat.
CPP1 and *Java1* have *Gravid* mosquitoes lay all of their eggs on the first oviposition attempt. This creates an egg-laying pattern where bursts of eggs are laid on the same day, followed by a few days when no eggs are laid, then another burst, and so on. In *Java2* and *CPP2*, however, the eggs are laid over a period of successive days.	This issue suggests a difference in oviposition code. It is partially resolved after all implementations ensure to use the same formulas for calculating how many eggs a female can lay in different oviposition attempts.
On the first oviposition attempt when eggs are laid, *Java1* and *CPP1* lay eggs one day sooner than *Java2* and *CPP2*.	*CPP1* and *Java1* lay eggs at the end of the *Bloodmeal Digesting* state, rather than waiting until the first day of being *Gravid*. They are updated to ensure that eggs are only laid while in the *Gravid* state.
In *Java1*, after all female mosquitoes lay all their (80) eggs in the *Gravid* state, they do not transition back to the *Bloodmeal Seeking* state on the same day.	*Java1* updates to ensure that the females transition back to the *Bloodmeal Seeking* state from the *Gravid* state on the same day once all eggs are laid.

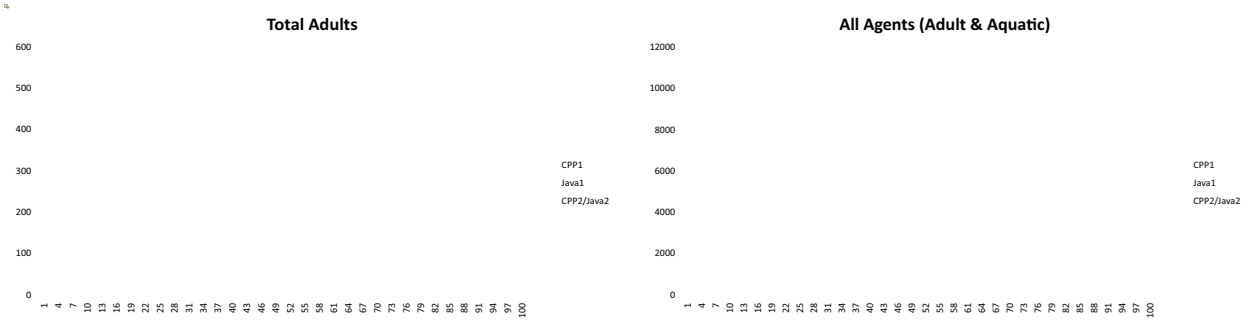

Figure 5: Four-fold docking results for *E3 (Phase 2)*. The graph on left shows the total number of (female) adults, and the graph on right shows the total number of agents (adults and aquatic). The *x*-axis denotes simulation days, and the *y*-axis denotes mosquito abundance.

5.4.3 *Phase 3*

After the issues found in *Phase 2* are addressed, we analyze the results again and discover a single issue (see Table 4). Once it is addressed, the results produce a **complete** four-fold dock, as shown in Figure 7.

5.5 Additional Results

We also perform two more experiments (*E4* and *E5*) that deal with *multiple* aquatic habitats and introduce some *randomness* into the simulation. *E4* combines the adult and aquatic populations in three aquatic habitats with carrying capacities 3000, 4000, 5000, respectively. For egg-laying, *Gravid* female mosquitoes deterministically select aquatic habitats. *E5* operates with similar settings except that the selection of aquatic habitats are made at random.

Results from *E4* and *E5* confirm to those of *E3*, and are omitted for brevity.

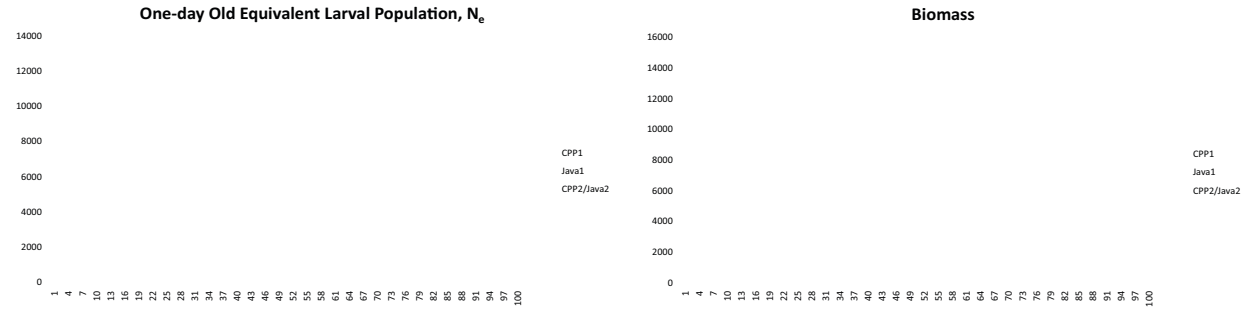

Figure 6: Four-fold docking results for *E3 (Phase 2)*. The graph on left shows the *One-day Old Equivalent Larval Population, N_e* (see Section 3.5.3). The graph on right shows the total biomass in the aquatic habitat. The *x*-axis denotes simulation days, and the *y*-axis denotes mosquito abundance.

Table 3: List of issues discovered and updated in *Phase 2*

Issue	**Resolution**
CPP1 and *Java1* enable the female mosquitoes to lay all (80) of their eggs on the first oviposition attempt at the beginning part of the simulation (on day 5 and 6). However, as suggested by Equation 8, laying all the eggs in a single oviposition attempt is possible only if the aquatic habitat is empty.	*CPP1* and *Java1* find that the egg-laying is indeed complete after three attempts, and performs a simple adjustment in the delay before the female mosquitoes can leave the *Gravid* state.
CPP2 and *Java2* place an upper bound of 80% on the larval mortality rate.	*CPP1* and *Java1* ensure the same.
In the calculations of Equation 8, *CPP2* and *Java2* use $Eggs_{remaining}$ (the number of eggs remaining to lay), instead of $Eggs_{max}$.	*CPP2* and *Java2* ensure to use $Eggs_{max}$.
In *Java1*, *Gravid* females incorrectly lay all (80) eggs, and the biomass of the aquatic habitat does not affect the number of eggs actually laid.	In the calculation of $Eggs_{laid}$ (the number of eggs allowed to lay), some *double* values are coerced to *integer* values, making the expression evaluating to 0 in these instances. This, in turn, affects the related variables ($Eggs_{potential}$, $Eggs_{laid}$, and $Eggs_{remaining}$). *Java1* updates this by using explicit typecasting to *double* values.

6 CONCLUSION

This paper serves as a case study of docking for ABMs. It shows how to achieve a complete four-fold dock between multiple implementations, all of which are built from a sufficiently complex core model. Following the 'Divide and Conquer' paradigm, we compartmentalize the agent-world with respect to the adult and aquatic populations. Isolating the agent-world into compartments allows designing specific experiments that suit for each compartment. We describe the design of these experiments and analyze their results. We show how the discovered issues help to clarify model concepts and eliminate ambiguities from different implementations. As indicated by the results, removal of these potential sources of output mismatches leads to a complete dock of the ABMs.

The complete four-fold docking encompasses *verification* between the four implementations, as well as *validation* against the core model with respect to these implementations. It produces incremental agreement in model outputs and serves the dual purpose of increasing confidence to the core model and revealing conceptual errors in model implementations.

Table 4: The single issue discovered and updated in *Phase 3*

Issue	Resolution
After 14 simulation days, *CPP1* and *Java1* kill a different number of adult mosquitoes, suggesting a rounding error.	*Java1* uses an extra '*a*' in the adult age-specific mortality rate function (Equation 2), and omits it to match the correct equation.

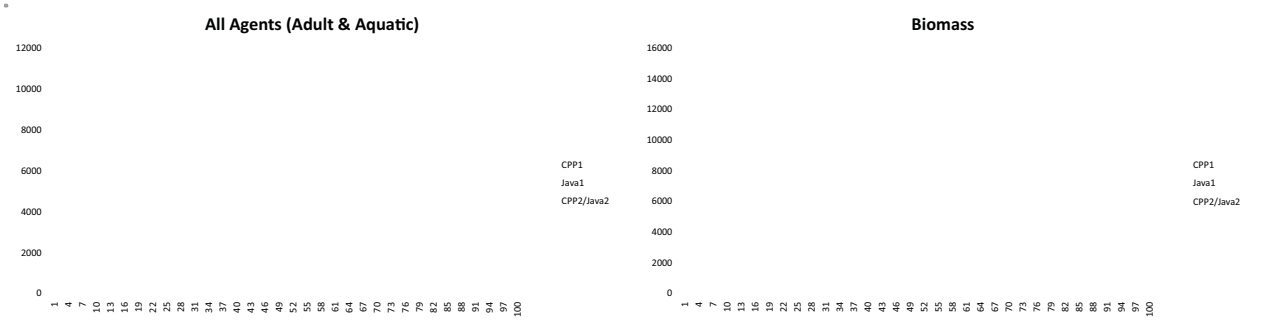

Figure 7: Four-fold docking results for *E3 (Phase 3)*. The graph on left shows the total number of agents (adults and aquatic). The graph on right shows the total biomass in the aquatic habitat. The *x*-axis denotes simulation days, and the *y*-axis denotes mosquito abundance. The complete overlap between the four implementations' results indicates a ***complete four-fold dock***.

ACKNOWLEDGMENTS

We would like to thank Frank H. Collins, George and Winifred Clark Chair of Biological Sciences at the University of Notre Dame, for his continuous support and expert domain knowledge of *Anopheles gambiae* and malaria.

REFERENCES

Arifin, S. M. N., G. J. Davis, Y. Zhou, and G. R. Madey. 2010, July. Verification & Validation by Docking: A Case Study of Agent-Based Models of *Anopheles gambiae*. In *SCSC2010: Proceedings of the Summer Computer Simulation Conference*. Ottawa, ON, Canada.

Carley, K. M. 2002. Computational Organizational Science and Organizational Engineering. *Simulation Modelling Practice and Theory* 10 (5-7): 253 – 269.

Edmonds, B., and D. Hales. 2003. Replication, Replication and Replication: Some Hard Lessons from Model Alignment. *Journal of Artificial Societies and Social Simulation* 6:4.

Galassi, M., J. Davies, J. Theiler, B. Gough, G. Jungman, M. Booth, and F. Rossi. 2003, February. *Gnu Scientific Library: Reference Manual*. Network Theory Ltd.

Gentile, J. E., G. J. Davis, B. S. Laurent, and S. Kurtz. 2010, July. A Framework for Modeling Mosquito Vectors. In *SCSC2010: Proceedings of the Summer Computer Simulation Conference*. Ottawa, ON, Canada.

Rand, W., and U. Wilensky. 2006. Verification and Validation through Replication: A Case Study Using Axelrod and Hammond's Ethnocentrism Model. *North American Association for Computational Social and Organization Sciences (NAACSOS)*.

Schoolfield, R. M., P. J. H. Sharpe, and C. E. Magnuson. 1981. Non-linear regression of biological temperature-dependent rate models based on absolute reaction-rate theory. *Journal of Theoretical Biology* 88 (4): 719–731.

Sharpe, P. J. H., and D. W. DeMichele. 1977. Reaction kinetics of poikilotherm development. *Journal of Theoretical Biology* 64 (4): 649 – 670.

Styer, L. M., J. R. Carey, J.-L. Wang, and T. W. Scott. 2007. Mosquitoes Do Senesce: Departure From The Paradigm Of Constant Mortality. *Am J Trop Med Hyg* 76 (1): 111–117.

Will, O. 2009. Resolving a Replication That failed: News on the Macy & Sato Model. *Journal of Artificial Societies and Social Simulation* 12:4.

Will, O., and R. Hegselmann. 2008. A Replication That Failed: On the Computational Model in 'Michael W. Macy and Yoshimichi Sato: Trust, Cooperation and Market Formation in the U.S. and Japan. Proceedings of the National Academy of Sciences, May 2002'. *Journal of Artificial Societies and Social Simulation* 11 (3): 3.

Zhou, Y., S. M. N. Arifin, J. Gentile, S. J. Kurtz, G. J. Davis, and B. A. Wendelberger. 2010, July. An Agent-based Model of the *Anopheles gambiae* Mosquito Life Cycle. In *SCSC2010: Proceedings of the Summer Computer Simulation Conference*. Ottawa, ON, Canada.

AUTHOR BIOGRAPHIES

S. M. NIAZ ARIFIN received his B.S. from Bangladesh University of Engineering and Technology (BUET) in 2004 and his M. S. from the University of Texas at Dallas in 2006. He is currently working as a Ph. D. student at the University of Notre Dame. His research interests include agent-based simulations, mathematical modeling in biology and data mining. His M.S. research focus was on artificial intelligence and natural language processing. He served as a software developer in the stereotactic breast cancer treatment project at Xcision Medical Systems, California and the Rails online database project at Sabre Holdings Corporation, Texas. His email address is <sarifin@nd.edu>.

GREGORY J. DAVIS is a graduate student at the University of Notre Dame pursuing a joint Ph. D. within the Psychology and Computer Science & Engineering departments. Before returning to academia as a student, he developed web-based data collection and reporting software for the Department of Epidemiology & Biostatistics at the University of Texas Health Science Center in San Antonio, developed statistical models of cancer risk and treatment benefits, and developed device drivers for mobile hardware platforms. His research interests in psychology involve understanding and modeling the mechanisms of human visual attention. His research interestes in computer science & engineering are in the area of agent-based modeling and simulation applied to biological populations, including simulating disease vectors. Additionally, his interests include developing simulation tools for non-programmers, web-based data collection and reporting, and mobile application development. His email address is <gdavis2@nd.edu>.

STEVEN KURTZ is a graduate student at the University of Notre Dame. He received his B. S. in Computer Engineering from the University of Notre Dame. His primary research interest is in computer architectures for emerging nanotechnologies. His email address is <skurtz@nd.edu>.

JAMES E. GENTILE is a graduate student at the University of Notre Dame. His research interests include agent-based modeling, global health and system analysis. His email address is <jegentile@gmail.com>.

YING ZHOU received the master's degree in Computer Science at Beijing University of Technology, Beijing, China, in 2004. After graduation, she worked as a software engineer in the State Information Center in china from 2004-2005. Now, Ying is a third-year Ph. D. student in the Computer Science and Engineering Department at University of Notre Dame. She is working on the project of Malaria Transmission Consortium, which is funded by the Bill & Melinda Gates Foundation. Ying's research efforts span agent-based modeling of the Anopheles mosquito life cycle, malaria epidemiology simulation and PDA-based Malaria Indicator Survey development. Her email address is <yzhou3@nd.edu>.

GREGORY R. MADEY received the Ph. D. and M. S. degrees in operations research from Case Western Reserve University and the M. S. and B. S. degrees in mathematics from Cleveland State University. He worked in industry for several firms, including Goodyear Aerospace, Gould Oceans Systems (now part of Northrup-Grumman), and Loral (now part of Lockheed Martin). He is currently faculty in the Department of Computer Science and Engineering at the University of Notre Dame. His research includes topics in agent-based modeling and simulation, emergency management modeling and simulation, web-services and service oriented architectures, bioinformatics, web portals for scientific collaboration, open source software, and cyberinfrastructure. He has published in various journals including, Communications of the ACM, IEEE Transactions on Engineering Management, IEEE Computing in Science & Engineering, The Journal of Systems & Software, BMC Bioinformatics, Computational & Mathematical Organization Theory, Nucleic Acids Research, Decision Sciences, The European Journal of OR, Omega, Expert Systems with Applications, and Expert Systems. He is a member of the ACM, AIS, IEEE Computer Society, Informs, and the Society for Computer Simulation. His email address is <gmadey@nd.edu>.