Modeling and Abstraction in Multilevel Simulation

- Once the desired resolution of a simulation model is determined, it is still necessary to determine a level of *abstraction*.
- Abstraction is the separation of qualities and properties from particular instances.
- An analytical model measuring arrivals and departures in a simple system may not require any abstraction.
- Domains such as computer networks are sufficiently complex that some abstraction is required to make the simulation tractable.
  - A network simulation with packet-level resolution may have several levels of abstraction.
  - For example, packets may be simulated at the workstation level, the subnet level, or higher.
- The appropriate level of abstraction is dependent upon the objectives of the simulation run, the capability of the system, and the ability and interests of the researcher.
Level of Abstraction

• In many cases, the desired level of abstraction changes dynamically.
• This has led to the development of multilevel simulation.
  – A multilevel simulation is able to accommodate multiple levels of abstraction.
  – Levels of abstraction may be hierarchical, model-based, resolution-based, or hybrid constructions.
• One way in which levels of abstraction may be changed dynamically is through aggregation.
  – Unfortunately, aggregation is a one-way process unless lower level state information is saved.
  – Deaggregation can only produce approximations of the components originally aggregated if the original components are not maintained.
  – Therefore, deaggregation often results in some loss of fidelity.
  – The aggregation/deaggregation problem is one of the major issues in simulation scalability.
Multilevel abstraction

• The purpose of abstraction is to isolate aspects of the problem that are important and suppress those that are unimportant [Rumbaugh 91].
• Multilevel simulation is based on multiple levels of abstraction and is a strategy to extend the utility of a simulation model.
• Levels of abstraction have several meanings, all of which may coexist in the same simulation.
  – A base model is capable of accounting for all of the input-output behavior of the real system [Zeigler 76].
  – The base model is a hypothetically complete explanation of the system behavior.
  – For any non-trivial system, it is axiomatically impossible to fully specify the base model.
  – Abstraction is used to specify a higher level representation of the system.
Model Abstraction

- **Model abstraction serves two purposes.**
  - First, abstraction may increase understanding of models and model behavior.
  - Second, abstraction may increase the computational efficiency of the implementation of the model [Sevinc 91].

- **Understanding of the model can increase when we discard unnecessary model details.**
  - There are limits to this approach, of course.
  - In textbooks we often see abstract communications models similar to that shown below:

  ![Diagram](sender - channel - receiver)

  - This abstraction is useful for introducing students to communications models, but is not sufficiently detailed for useful simulation study.
Why abstraction is useful in simulation: [Fishwick 88]

- An abstract model is usually less computationally complex than the base model.
- As long as the trade-off between complexity and data sufficiency is a favorable one, then the abstract model remains useful.
- The abstract model is easier to understand than the base model in most cases.
  - Since an abstraction involves a reduction in process component(s), the model is more easily created and modified.
- The creation of an abstract model permits us to build a library of different models which represent the same process.
  - In many cases, the library of models for one process represents an evolutionary path for the modeling process.
  - Processes often are modeled using simple methods at first and more complex methods later as more knowledge is gained by the simulationist.
  - Additionally, validation helps to prune away inferior models.
• We abstract up and refine down, as shown above.
• In this example, a strict hierarchy is illustrated which implies a totally ordered relationship.
• However, this need not be the case, as some models may simply be mapped to other models.
Abstraction Network

- Two models with different perspectives, but not different levels of abstraction.
- Abstraction passes essential information from a lower level model to a higher level model.
- Unnecessary information is not passed out.
- When additional information is required, then the model is refined to a lower level closer to the base model.
Hierarchical modeling

- [Iwasaki 92] offers four dimensions of abstraction:
  - **Structural**: Abstraction by lumping together a group of components that are physically close.
  - **Functional**: Abstraction by lumping together a group of components that collectively achieve a distinct, higher-level function.
  - **Temporal**: Abstraction by ignoring behavior over a short period of time.
  - **Quantitative**: Abstraction by ignoring small differences in variable values.

- Iwasaki notes that these dimensions are neither complete nor necessarily independent of each other.
- Abstraction by ignoring behavior or small differences is akin to simplification strategies.
- Abstraction by lumping together denotes aggregation, which will be discussed further.
Defining resolution of a model

- A manufacturing simulation of automobiles may have a resolution of stock numbered parts as illustrated above.
  - That is, specific parts with specific identification stock numbers are modeled.
  - Common screws, molding clips, or any items without a specific stock number are not numbered.
- The resolution of this model, the level of detail this model can input and output, is set.
- There are many potential levels of abstraction. The base case occurs when there is no abstraction above the level of resolution.
- In such a case the simulation is designed to model the part-by-part assembly of an automobile.
Hierarchical abstraction of an automobile assembly

- It is often the case that the base level is unwieldy, hence we might want to model in terms of assemblies and subassemblies.
Assemblies and Subassemblies

- This is analogous to the whole-part relationships
- We can mask the part-by-part details of the base model and produce a model with a higher level of abstraction.
- Assuming the resolution remains fixed, the underlying simulation will still be executing with part-by-part detail.
- However, the information presented to the analyst will be at a higher level of abstraction.
- We can apply this mask somewhat arbitrarily since we are currently working with a fixed-resolution model.
- We may wish to focus on brake pads and consider the rest of the simulation at the major assembly or sub-assembly level.
- Since the resolution of the model is part-by-part, we may focus on brake pads and consider the rest of the assembly process at higher levels of abstraction.
Multilevel simulation model

- Abstraction hierarchies enable analysts to concentrate on particular parts of a complex model.
- Multiple levels of abstraction contribute to general purpose simulation models.
- A simulation created to study brake pads is unlikely to be easily extended for other study.
- However, a multilevel simulation model provides greater flexibility to allow analysts to focus on different parts of the model depending upon the current problem of interest.
Multimodels

- Models that are composed of other models, in a network or a graph, are called *multimodels* [Fishwick 95].
- Multimodels can combine the power of well-known modeling methodologies such as finite state automata (FSAs), Petri nets, block models, differential equations, and queuing models [Lee 93].
- Multimodels may integrate different models at the same or varying levels of abstraction.
Multimodeling Validated Models

• Typically, we abstract or refine models of the same type. If our base model was a FSM, we would expect abstractions from that model to also be FSM-based.
• Parts of a large system are often already well-modeled using various model types.
  – For example, it might not be desirable to rewrite an existing Petri net model as an FSA model in order to use the models together.
• Multimodeling is often appealing since existing, validated models can continue to be used.
  – Combining different valid model types in the same simulation can be very beneficial if we can maintain fidelity and if the resulting combined model can be validated.
As shown previously, we illustrate an abstraction network in which one type of model could be mapped to another.
Political Campaign Multimodeling

- Consider a simulation of a political campaign for President of the USA.
  - For our purposes, we are interested in gauging popular support for a candidate and how that may translate into electoral votes.
  - Campaign managers are interested in a variety of predictive models.
  - Polling is nearly continuous (although using a differential equation to model polling may be extreme) and there are a variety of models used to predict polling information.

(It is worth noting that simulation has become a very important part of US campaign strategies.)
Resolution and Abstraction

• Multilevel models incorporate components at multiple levels of resolution [Popken 96].
  – Resolution-based abstraction was discussed in detail in Chapter 7.
  – To quickly review, in one type multiresolution model, some entities may operate from event lists providing event-by-event detail while others may run based on PDFs.

• When resolution is fixed at the base model level, a bound is placed on the ability to accept inputs at a level of abstraction finer than that specified.
Resolving a Hierarchy of Inputs

In designing a combat model, for example, we may specify inputs and outputs from a *resolution hierarchy* as shown below.

- We may decide to set the model resolution at the company level.
- In this case, the model can neither accept input nor produce results at a resolution lower than selected without the assistance of some *synthesizing function*.

- A company-level model cannot directly accept individual soldier inputs nor provide results about individual soldiers.
- Higher levels of resolution may be approximated probabilistically, but with some possible loss of fidelity.
Modeling Thermopylae

- Two Million Persians
- Three Hundred Spartans
- Seven Day Stand

“Go tell the Spartans, stranger passing by, that here obedient to their laws we lie.”
- Simonides

“The fox knows many tricks; the hedgehog one good one.”
- Archilochus
Resolving a Hierarchy of Inputs

- The simulation designer may wish to build base models which can accommodate multiple levels of resolution.
- Multiple resolution is often desirable because of the flexibility it provides to the analyst.
- The analyst is empowered to select one of the provided levels of resolution that provides the information he or she requires.
- This capability has been demonstrated by Wall and was covered in detail in Chapter 7.
Aggregation and Deaggregation

• Closely related to levels of abstraction are the concepts of aggregation and decomposition.
• Generally, to move to a higher level of abstraction (less detail) involves aggregation, or the representation of several more detailed components by a single equivalent component.
• Grouping components and aggregating variables is quite a well-known procedure.
• Conversely, to move to a lower level of abstraction (more detail) the model has to be further decomposed to adequately describe the modeled physical entities.
• This procedure is more difficult and has largely been an unsolved problem.
Aggregation

Aggregation is conceptually a subset of abstraction [Fishwick 88].
- Collecting lower-level entities and joining them into a higher-level entity is certainly a form of abstraction.

Consider the system shown below. If we collect and combine the model elements of (a) and replace it with the single element in (b), we have aggregated our original model.
- When the original model elements are not saved somewhere in the system, the aggregation is destructive.
- Not to be confused with representation issues.
- A network simulation could present the analyst with a view of the model in (b) when in fact the simulation is continuing to execute with the individual components of (a).
- This representational aggregation is merely an aggregation of results.

- This can be useful, but it does not produce any computational gains.
Representational Aggregation

- Representational aggregation achieves complete consistency.
- This should not be surprising since only the outcomes of the lower-level components are aggregated.
- As long as the original, lower-level outcomes are saved in the system, loss of consistency may be avoided. In many cases this is not a feasible strategy.
- Simulations may have a very large number of entities.
  - The U.S. Army’s Prairie Warrior ‘94 simulation exercise had more than 50,000 entities, and that should not be considered close to an upper bound [MITRE 95].
  - It is very difficult and computationally expensive to constantly track so many individual entities.
Consistency

- Intuitively, we can think of consistency as similar to moving from one map scale to another in the same family of maps [Davis 91].
- Model consistency may be more rigorously defined in the manner suggested by Hillestad and Juncosa [Hillestad 93].
- We define the mapping function $Z_A$ and mapping function $Z_B$ to produce comparable output from models $A$ and $B$, respectively.
- Since the outputs can be represented as vectors, consistency may be defined in terms of a scalar norm of the differences [Hillestad 93]. Initially, consistency and at time $t$, $\varepsilon(t) = \left\| \mathbf{Z}_A(t) - \mathbf{Z}_B(t) \right\|$ and at time $0$, $\varepsilon(0) = \left\| \mathbf{Z}_A(0) - \mathbf{Z}_B(0) \right\|$

- Acceptable consistency may then be defined as the acceptable size of $\varepsilon$ for all $t$ in the range of interest. For example, inconsistencies occurring before steady state is reached in both simulations may be unimportant.
Consistency in Terms of Outputs

• Figure below illustrates consistency between two models at time 0 and time $t$. As mentioned previously, some sort of mapping or transition function may be required to compare the outputs of two different models.
Consistency Example

- Let model A be the historical event script which has recorded the number of packets on an Ethernet segment.
- Model B uses a probability distribution function to generate packets on an Ethernet segment.
- The functions $Z_A$ and $Z_B$ convert the outputs from the respective models into common units and map the output onto a common time interval.
- We can analyze the differences between the models by comparing the number of packets each model produces at discrete time points.
Aggregation and Resolution

• We often think of aggregation in terms of hierarchical structures, but this need not be the case.
• Consider two network simulation models.
  – Model A has a resolution at the message level
  – Model B has a resolution at the octet level.
• In order to evaluate the consistency of the two models, we need to map their output into common terms.
• Typically we are interested in packets which are composed of octets and constructed by a network’s transport layer mechanism to move octets from one application to another.
  – We could decide to map both model A and model B to produce packets and compare the consistency of the output as illustrated previously.
  – We may choose to simply map one model directly to another, in which case only one mapping function is required.
    • Model B could simply have its octets aggregated into messages. In this case no mapping function $Z_A$ is needed for model A and map A would be null.
Absolute Consistency

• When *absolute consistency* is desired, that is, when then any destructive aggregation is strictly a one-way street, as shown below.

• If we try to move from model A back to model B, the best we can achieve is an approximation of the expected state when the simulation resumes execution at the base level.
Aggregating and Deaggregating Models

- Noted RAND researcher, Paul K. Davis [Davis 92], outlines the challenges associated with aggregating and deaggregating models:
  - Getting concepts and names right.
  - Defining the reference model.
  - Determining relationships and mappings.
  - Determining the form of reasonable aggregate equations relative to detailed equations.
  - Finding conditions under which aggregation equations might be reasonably valid.
  - Deciding on cases to be distinguished and how to make calibrations for each case – e.g., how to determine weighting factors over case and time so that the calibrations will be appropriate for the context of the larger application.
  - Most traditional simulation systems are limited because they are unable to handle models with different degrees of aggregation.

- As [Wall 93] observes, the appropriate level of aggregation may not be known when the simulation system is built. It is necessary to be able to move dynamically through different levels of aggregation.
Dynamic Aggregation

• Current systems are fairly inflexible with respect to changing levels of aggregation.
• The difficulty arises from the problem of ensuring consistency among all active levels [Wall 93].
• The use of an open architecture provides a means of constructing simulation models with the requisite flexibility to accommodate multiple levels of aggregation.
Open System Architecture

• A software system is said to be open if its behavior can be easily modified and extended [Wegner 92].
  – Openness can be achieved dynamically through reactivity or statically through modularity.
  – A reactive (interactive) system is one that can react to stimuli by modifying its state and emitting a response.
  – A modular (encapsulated) system is one whose number of components and/or functionality can be statically extended by an external agent.
  – Wegner goes on to note that most flexible open systems are both reactive and modular.

• An open architecture is liberally defined as any design that has published specifications.

• A more useful definition of an open architecture is an architecture that allows for the interoperability of different systems [Sochats 92].
  – Published specifications are an important requirement but do not guarantee interoperability.
  – An open system architecture is the combinational framework that supports the integration of multiple hardware and software systems.
Evaluating Distributed Simulation Design

• In order to evaluate a simulation design on a distributed platform we first establish a framework which allows us to analyze the delays in the message transmission.

• The following measurements provide some means of evaluating the distributed implementation.
  – Schedulability
  – Complexity of Schedulability
  – Buffer Requirement
  – Stability
Schedulability

• Schedulability: This is a direct measure of the capability of meeting message delay requirements. There are two possible ways to measure it:
  – The worst case achievable utilization. This is a threshold utilization below which the delay requirements are always met.
  – The probability of meeting delay requirements for a given load. In both cases, the higher the measure, the better the performance is.
Schedulability Testing/ Buffer Requirements

• Complexity of schedulability testing:
  – Sometimes it is necessary to know if a particular delay requirement can be met.
  – This is done by schedulability testing. A lower level of complexity makes schedulability an easier evaluation to apply.

• Buffer requirement:
  – A message will be lost and never be delivered if a buffer overflows.
  – Different network scheduling methods may result in different buffer requirements. An upper bound of buffer size must be derived for each scheduling method proposed.
Stability

• Stability: This reflects the system’s sensitivity to change in configuration.
  – One would prefer that a small change in the system configuration (e.g., a slight increase on a node) have minimal impact on the system’s capability of meeting message delay requirements.
Summary

• There is a fundamental need for variable resolution models (or families of models) in which there is true consistency across levels.
  – Concepts and methods are needed to enable cross-resolution work, including work with models not originally designed to be compatible [Davis 91].
• It is easy to confuse level of resolution with level of abstraction. In fact, both are closely related.
  – Resolution is the level of detail a model can process.
  – The same model may have multiple levels of abstraction.
  – Aggregation is a powerful form of abstraction but can result in the loss of resolution and/or the loss of detail.
  – When moving down an aggregation hierarchy, we can usually only approximate the expected state of the deaggregated components.
• The ability to mix and match different forms of objects representing the same real entity is dependent upon the definition of appropriate classes of objects having the same interface.
Conclusion

• A clean architecture is an absolute requirement to bring distributed computing to bear on the problem
  – Interfaces must be designed early and at all levels.
  – It is well understood that physically connecting hardware isn’t sufficient to achieve interoperability.
  – In an open systems structure the software must be modular.
    • The interfaces must be built and then the underlying application built.
    • The interfaces must be designed early in the process since a subsequent requirement to change the interfaces may require major modification of the underlying application.

• In order to apply the power of integrating multiple models at multiple levels of abstraction with multiple resolution, a strong simulation infrastructure must underpin the design.

• Combining underspecified models will often lead to “patches” and one-time fixes, which will make the simulations inflexible & brittle.