Modeling and Simulation

Dr. G.H.J. Lanel

Lecture 1

Outline

Outline

What is A model?

A Representation of an object, a system, or an idea in some form other than that of the entity itself.
(Shannon)

Types of Models:

- **Physical:** Scale models, prototype plants, . . .
- **Mathematical:** Analytical queuing models, linear programs, simulation.

Types of Models:

- **Physical:** Scale models, prototype plants, . . .
- **Mathematical:** Analytical queuing models, linear programs, simulation.

What is Simulation?

- A Simulation of a system is the operation of a model, which is a representation of that system.
- The model is amenable to manipulation which would be impossible, too expensive, or too impractical to perform on the system which it portrays.
- The operation of the model can be studied, and, from this, properties concerning the behavior of the actual system can be inferred.

What is Simulation?

- A Simulation of a system is the operation of a model, which is a representation of that system.
- The model is amenable to manipulation which would be impossible, too expensive, or too impractical to perform on the system which it portrays.
- The operation of the model can be studied, and, from this, properties concerning the behavior of the actual system can be inferred.

What is Simulation?

- A Simulation of a system is the operation of a model, which is a representation of that system.
- The model is amenable to manipulation which would be impossible, too expensive, or too impractical to perform on the system which it portrays.
- The operation of the model can be studied, and, from this, properties concerning the behavior of the actual system can be inferred.

Applications:

- Designing and analyzing manufacturing systems.
- Evaluating H/W and S/W requirements for a computer system.
- Evaluating a new military weapons system or tactics.
- Determining ordering policies for an inventory system.
- Designing communications systems and message protocols for them.

Applications:

- Designing and analyzing manufacturing systems.
- Evaluating H/W and S/W requirements for a computer system.
- Evaluating a new military weapons system or tactics.
- Determining ordering policies for an inventory system.
- Designing communications systems and message protocols for them.

Applications:

- Designing and analyzing manufacturing systems.
- Evaluating H/W and S/W requirements for a computer system.
- Evaluating a new military weapons system or tactics.
- Determining ordering policies for an inventory system.
- Designing communications systems and message protocols for them.

Applications:

- Designing and analyzing manufacturing systems.
- Evaluating H/W and S/W requirements for a computer system.
- Evaluating a new military weapons system or tactics.
- Determining ordering policies for an inventory system.
- Designing communications systems and message protocols for them.

Applications:

- Designing and analyzing manufacturing systems.
- Evaluating H/W and S/W requirements for a computer system.
- Evaluating a new military weapons system or tactics.
- Determining ordering policies for an inventory system.
- Designing communications systems and message protocols for them.

Applications: Ctd

- Designing and operating transportation facilities such as freeways, airports, subways, or ports.
- Evaluating designs for service organizations such as hospitals, post offices, or fast-food restaurants.
- Analyzing financial or economic systems.

Applications: Ctd

- Designing and operating transportation facilities such as freeways, airports, subways, or ports.
- Evaluating designs for service organizations such as hospitals, post offices, or fast-food restaurants.
- Analyzing financial or economic systems.

Applications: Ctd

- Designing and operating transportation facilities such as freeways, airports, subways, or ports.
- Evaluating designs for service organizations such as hospitals, post offices, or fast-food restaurants.
- Analyzing financial or economic systems.

Steps in Simulation and Model building:

1. Define an achievable goal.
2. Put together a complete mix of skills on the team.
3. Involve the end-user.
4. Choose the appropriate simulation tools.
5. Model the appropriate level(s) of detail.
6. Start early to collect the necessary input data.

Steps in Simulation and Model building:

1. Define an achievable goal.
2. Put together a complete mix of skills on the team.
3. Involve the end-user.
4. Choose the appropriate simulation tools.
5. Model the appropriate level(s) of detail.
6. Start early to collect the necessary input data.

Steps in Simulation and Model building:

1. Define an achievable goal.
2. Put together a complete mix of skills on the team.
3. Involve the end-user.
4. Choose the appropriate simulation tools.
5. Model the appropriate level(s) of detail.
6. Start early to collect the necessary input data.

Steps in Simulation and Model building:

1. Define an achievable goal.
2. Put together a complete mix of skills on the team.
3. Involve the end-user.
4. Choose the appropriate simulation tools.
5. Model the appropriate level(s) of detail.
6. Start early to collect the necessary input data.

Steps in Simulation and Model building:

1. Define an achievable goal.
2. Put together a complete mix of skills on the team.
3. Involve the end-user.
4. Choose the appropriate simulation tools.
5. Model the appropriate level(s) of detail.
6. Start early to collect the necessary input data.

Steps in Simulation and Model building:

1. Define an achievable goal.
2. Put together a complete mix of skills on the team.
3. Involve the end-user.
4. Choose the appropriate simulation tools.
5. Model the appropriate level(s) of detail.
6. Start early to collect the necessary input data.

Steps in Simulation and Model building: Ctd

7. Provide adequate and on-going documentation.
8. Develop a plan for adequate model verification. (Did we get the "right answers?")
9. Develop a plan for model validation. (Did we ask the "right questions?")
10. Develop a plan for statistical output analysis.

Steps in Simulation and Model building: Ctd

7. Provide adequate and on-going documentation.
8. Develop a plan for adequate model verification. (Did we get the ”right answers?”)
9. Develop a plan for model validation. (Did we ask the ”right questions?”)
10. Develop a plan for statistical output analysis.

Steps in Simulation and Model building: Ctd

7. Provide adequate and on-going documentation.
8. Develop a plan for adequate model verification. (Did we get the ”right answers?”)
9. Develop a plan for model validation. (Did we ask the ”right questions?”)
10. Develop a plan for statistical output analysis.

Steps in Simulation and Model building: Ctd

7. Provide adequate and on-going documentation.
8. Develop a plan for adequate model verification. (Did we get the ”right answers?”)
9. Develop a plan for model validation. (Did we ask the ”right questions?”)
10. Develop a plan for statistical output analysis.

Put together a complete mix of skills on the team:

We need:

- Knowledge of the system under investigation.
- System analyst skills (model formulation).
- Model building skills (model Programming).
- Data collection skills.
- Statistical skills (input data representation).

Put together a complete mix of skills on the team:

We need:

- Knowledge of the system under investigation.
- System analyst skills (model formulation).
- Model building skills (model Programming).
- Data collection skills.
- Statistical skills (input data representation).

Put together a complete mix of skills on the team:

We need:

- Knowledge of the system under investigation.
- System analyst skills (model formulation).
- Model building skills (model Programming).
- Data collection skills.
- Statistical skills (input data representation).

Put together a complete mix of skills on the team:

We need:

- Knowledge of the system under investigation.
- System analyst skills (model formulation).
- Model building skills (model Programming).
- Data collection skills.
- Statistical skills (input data representation).

Put together a complete mix of skills on the team:

We need:

- Knowledge of the system under investigation.
- System analyst skills (model formulation).
- Model building skills (model Programming).
- Data collection skills.
- Statistical skills (input data representation).

Put together a complete mix of skills on the team: Ctd

We need:

- More statistical skills (output data analysis).
- Even more statistical skills (design of experiments).
- Management skills (to get everyone pulling in the same direction).

Put together a complete mix of skills on the team: Ctd

We need:

- More statistical skills (output data analysis).
- Even more statistical skills (design of experiments).
- Management skills (to get everyone pulling in the same direction).

Put together a complete mix of skills on the team: Ctd

We need:

- More statistical skills (output data analysis).
- Even more statistical skills (design of experiments).
- Management skills (to get everyone pulling in the same direction).

Involve the end user

- Modeling is a selling job!
- Does anyone believe the results?
- Will anyone put the results into action?
- The End-user (your customer) can (and must) do all of the above BUT, first he must be convinced!
- He must believe it is HIS Model!

Involve the end user

- Modeling is a selling job!
- Does anyone believe the results?
- Will anyone put the results into action?
- The End-user (your customer) can (and must) do all of the above BUT, first he must be convinced!
- He must believe it is HIS Model!

Involve the end user

- Modeling is a selling job!
- Does anyone believe the results?
- Will anyone put the results into action?
- The End-user (your customer) can (and must) do all of the above BUT, first he must be convinced!
- He must believe it is HIS Model!

Involve the end user

- Modeling is a selling job!
- Does anyone believe the results?
- Will anyone put the results into action?
- The End-user (your customer) can (and must) do all of the above BUT, first he must be convinced!
- He must believe it is HIS Model!

Involve the end user

- Modeling is a selling job!
- Does anyone believe the results?
- Will anyone put the results into action?
- The End-user (your customer) can (and must) do all of the above BUT, first he must be convinced!
- He must believe it is HIS Model!

Assuming Simulation is the appropriate means, three alternatives exist:

1. Build Model in a General Purpose Language.
2. Build Model in a General Simulation Language.
3. Use a Special Purpose Simulation Package.

Assuming Simulation is the appropriate means, three alternatives exist:

1. Build Model in a General Purpose Language.
2. Build Model in a General Simulation Language.
3. Use a Special Purpose Simulation Package.

Assuming Simulation is the appropriate means, three alternatives exist:

1. Build Model in a General Purpose Language.
2. Build Model in a General Simulation Language.
3. Use a Special Purpose Simulation Package.

Assuming Simulation is the appropriate means, three alternatives exist:

1. Build Model in a General Purpose Language.
2. Build Model in a General Simulation Language.
3. Use a Special Purpose Simulation Package.

Modeling with general purpose languages

- Advantages:
  - Little or no additional software cost.
  - Universally available (portable).
  - No additional training (Everybody knows(language X)!)
- Disadvantages:
  - Every model starts from scratch.
  - Very little reusable code.
  - Long development cycle for each model.
  - Difficult verification phase.

Modeling with general purpose languages

- Advantages:
  - Little or no additional software cost.
  - Universally available (portable).
  - No additional training (Everybody knows(language X)!)
- Disadvantages:
  - Every model starts from scratch.
  - Very little reusable code.
  - Long development cycle for each model.
  - Difficult verification phase.

Modeling with general purpose languages

- Advantages:
  - Little or no additional software cost.
  - Universally available (portable).
  - No additional training (Everybody knows(language X)!)
- Disadvantages:
  - Every model starts from scratch.
  - Very little reusable code.
  - Long development cycle for each model.
  - Difficult verification phase.

Modeling with general purpose languages

- Advantages:
  - Little or no additional software cost.
  - Universally available (portable).
  - No additional training (Everybody knows(language X)!)
- Disadvantages:
  - Every model starts from scratch.
  - Very little reusable code.
  - Long development cycle for each model.
  - Difficult verification phase.

Modeling with general purpose languages

- Advantages:
  - Little or no additional software cost.
  - Universally available (portable).
  - No additional training (Everybody knows(language X)!)
- Disadvantages:
  - Every model starts from scratch.
  - Very little reusable code.
  - Long development cycle for each model.
  - Difficult verification phase.

Modeling with general purpose languages

- Advantages:
  - Little or no additional software cost.
  - Universally available (portable).
  - No additional training (Everybody knows(language X)!)
- Disadvantages:
  - Every model starts from scratch.
  - Very little reusable code.
  - Long development cycle for each model.
  - Difficult verification phase.

Modeling with general purpose languages

- Advantages:
  - Little or no additional software cost.
  - Universally available (portable).
  - No additional training (Everybody knows(language X)!)
- Disadvantages:
  - Every model starts from scratch.
  - Very little reusable code.
  - Long development cycle for each model.
  - Difficult verification phase.

General purpose languages used for simulation

- FORTRAN: Probably more models than any other language.
- PASCAL: Not as universal as FORTRAN
- MODULA: Many improvements over PASCAL
- ADA: Department of Defense attempt at standardization
- C, C++ : Object-oriented programming language

General purpose languages used for simulation

- FORTRAN: Probably more models than any other language.
- PASCAL: Not as universal as FORTRAN
- MODULA: Many improvements over PASCAL
- ADA: Department of Defense attempt at standardization
- C, C++ : Object-oriented programming language

General purpose languages used for simulation

- FORTRAN: Probably more models than any other language.
- PASCAL: Not as universal as FORTRAN
- MODULA: Many improvements over PASCAL
- ADA: Department of Defense attempt at standardization
- C, C++ : Object-oriented programming language

General purpose languages used for simulation

- FORTRAN: Probably more models than any other language.
- PASCAL: Not as universal as FORTRAN
- MODULA: Many improvements over PASCAL
- ADA: Department of Defense attempt at standardization
- C, C++ : Object-oriented programming language

General purpose languages used for simulation

- FORTRAN: Probably more models than any other language.
- PASCAL: Not as universal as FORTRAN
- MODULA: Many improvements over PASCAL
- ADA: Department of Defense attempt at standardization
- C, C++ : Object-oriented programming language

Modeling with general purpose simulation languages

- Advantages:
  - Standardized features often needed in modeling
  - Shorter development cycle for each model
  - Much assistance in model verification
  - Very readable code
- Disadvantages:
  - Higher software cost (up-front)
  - Additional training required
  - Limited portability

Modeling with general purpose simulation languages

- Advantages:
  - Standardized features often needed in modeling
  - Shorter development cycle for each model
  - Much assistance in model verification
  - Very readable code
- Disadvantages:
  - Higher software cost (up-front)
  - Additional training required
  - Limited portability

Modeling with general purpose simulation languages

- Advantages:
  - Standardized features often needed in modeling
  - Shorter development cycle for each model
  - Much assistance in model verification
  - Very readable code
- Disadvantages:
  - Higher software cost (up-front)
  - Additional training required
  - Limited portability

Modeling with general purpose simulation languages

- Advantages:
  - Standardized features often needed in modeling
  - Shorter development cycle for each model
  - Much assistance in model verification
  - Very readable code
- Disadvantages:
  - Higher software cost (up-front)
  - Additional training required
  - Limited portability

Modeling with general purpose simulation languages

- Advantages:
  - Standardized features often needed in modeling
  - Shorter development cycle for each model
  - Much assistance in model verification
  - Very readable code
- Disadvantages:
  - Higher software cost (up-front)
  - Additional training required
  - Limited portability

Modeling with general purpose simulation languages

- Advantages:
  - Standardized features often needed in modeling
  - Shorter development cycle for each model
  - Much assistance in model verification
  - Very readable code
- Disadvantages:
  - Higher software cost (up-front)
  - Additional training required
  - Limited portability

Modeling with general purpose simulation languages

- Advantages:
  - Standardized features often needed in modeling
  - Shorter development cycle for each model
  - Much assistance in model verification
  - Very readable code
- Disadvantages:
  - Higher software cost (up-front)
  - Additional training required
  - Limited portability

General purpose simulation languages

- GPSS:
  - Block-structured Language
  - Interpretive Execution
  - FORTRAN-based (Help blocks)
  - World-view: Transactions/Facilities
- SIMSCRIPT II.5:
  - English-like Problem Description Language
  - Compiled Programs
  - Complete language (no other underlying language)
  - World-view: Processes/ Resources/ Continuous

General purpose simulation languages

- GPSS:
  - Block-structured Language
  - Interpretive Execution
  - FORTRAN-based (Help blocks)
  - World-view: Transactions/Facilities
- SIMSCRIPT II.5:
  - English-like Problem Description Language
  - Compiled Programs
  - Complete language (no other underlying language)
  - World-view: Processes/ Resources/ Continuous

General purpose simulation languages

- GPSS:
  - Block-structured Language
  - Interpretive Execution
  - FORTRAN-based (Help blocks)
  - World-view: Transactions/Facilities
- SIMSCRIPT II.5:
  - English-like Problem Description Language
  - Compiled Programs
  - Complete language (no other underlying language)
  - World-view: Processes/ Resources/ Continuous

General purpose simulation languages

- GPSS:
  - Block-structured Language
  - Interpretive Execution
  - FORTRAN-based (Help blocks)
  - World-view: Transactions/Facilities
- SIMSCRIPT II.5:
  - English-like Problem Description Language
  - Compiled Programs
  - Complete language (no other underlying language)
  - World-view: Processes/ Resources/ Continuous

General purpose simulation languages

- GPSS:
  - Block-structured Language
  - Interpretive Execution
  - FORTRAN-based (Help blocks)
  - World-view: Transactions/Facilities
- SIMSCRIPT II.5:
  - English-like Problem Description Language
  - Compiled Programs
  - Complete language (no other underlying language)
  - World-view: Processes/ Resources/ Continuous

General purpose simulation languages

- GPSS:
  - Block-structured Language
  - Interpretive Execution
  - FORTRAN-based (Help blocks)
  - World-view: Transactions/Facilities
- SIMSCRIPT II.5:
  - English-like Problem Description Language
  - Compiled Programs
  - Complete language (no other underlying language)
  - World-view: Processes/ Resources/ Continuous

General purpose simulation languages

- GPSS:
  - Block-structured Language
  - Interpretive Execution
  - FORTRAN-based (Help blocks)
  - World-view: Transactions/Facilities
- SIMSCRIPT II.5:
  - English-like Problem Description Language
  - Compiled Programs
  - Complete language (no other underlying language)
  - World-view: Processes/ Resources/ Continuous

General purpose simulation languages

- GPSS:
  - Block-structured Language
  - Interpretive Execution
  - FORTRAN-based (Help blocks)
  - World-view: Transactions/Facilities
- SIMSCRIPT II.5:
  - English-like Problem Description Language
  - Compiled Programs
  - Complete language (no other underlying language)
  - World-view: Processes/ Resources/ Continuous

General purpose simulation languages: Ctd

- MODSIM III:
  - Modern Object-Oriented Language
  - Modularity Compiled Programs
  - Based on Modula2 (but compiles into C)
  - World-view: Processes
- SIMULA:
  - ALGOL-based Problem Description Language
  - Compiled Programs
  - World-view: Processes

General purpose simulation languages: Ctd

- MODSIM III:
  - Modern Object-Oriented Language
  - Modularity Compiled Programs
  - Based on Modula2 (but compiles into C)
  - World-view: Processes
- SIMULA:
  - ALGOL-based Problem Description Language
  - Compiled Programs
  - World-view: Processes

General purpose simulation languages: Ctd

- MODSIM III:
  - Modern Object-Oriented Language
  - Modularity Compiled Programs
  - Based on Modula2 (but compiles into C)
  - World-view: Processes
- SIMULA:
  - ALGOL-based Problem Description Language
  - Compiled Programs
  - World-view: Processes

General purpose simulation languages: Ctd

- MODSIM III:
  - Modern Object-Oriented Language
  - Modularity Compiled Programs
  - Based on Modula2 (but compiles into C)
  - World-view: Processes
- SIMULA:
  - ALGOL-based Problem Description Language
  - Compiled Programs
  - World-view: Processes

General purpose simulation languages: Ctd

- MODSIM III:
  - Modern Object-Oriented Language
  - Modularity Compiled Programs
  - Based on Modula2 (but compiles into C)
  - World-view: Processes
- SIMULA:
  - ALGOL-based Problem Description Language
  - Compiled Programs
  - World-view: Processes

General purpose simulation languages: Ctd

- MODSIM III:
  - Modern Object-Oriented Language
  - Modularity Compiled Programs
  - Based on Modula2 (but compiles into C)
  - World-view: Processes
- SIMULA:
  - ALGOL-based Problem Description Language
  - Compiled Programs
  - World-view: Processes

General purpose simulation languages: Ctd

- MODSIM III:
  - Modern Object-Oriented Language
  - Modularity Compiled Programs
  - Based on Modula2 (but compiles into C)
  - World-view: Processes
- SIMULA:
  - ALGOL-based Problem Description Language
  - Compiled Programs
  - World-view: Processes

General purpose simulation languages: Ctd

- SLAM:
  - Block-structured Language
  - Interpretive Execution
  - FORTRAN-based (and extended)
  - World-view: Network / event / continuous
- CSIM:
  - process-oriented language
  - C-based (C++ based)
  - World-view: Processes

General purpose simulation languages: Ctd

- SLAM:
  - Block-structured Language
  - Interpretive Execution
  - FORTRAN-based (and extended)
  - World-view: Network / event / continuous
- CSIM:
  - process-oriented language
  - C-based (C++ based)
  - World-view: Processes

General purpose simulation languages: Ctd

- SLAM:
  - Block-structured Language
  - Interpretive Execution
  - FORTRAN-based (and extended)
  - World-view: Network / event / continuous
- CSIM:
  - process-oriented language
  - C-based (C++ based)
  - World-view: Processes

General purpose simulation languages: Ctd

- SLAM:
  - Block-structured Language
  - Interpretive Execution
  - FORTRAN-based (and extended)
  - World-view: Network / event / continuous
- CSIM:
  - process-oriented language
  - C-based (C++ based)
  - World-view: Processes

General purpose simulation languages: Ctd

- SLAM:
  - Block-structured Language
  - Interpretive Execution
  - FORTRAN-based (and extended)
  - World-view: Network / event / continuous
- CSIM:
  - process-oriented language
  - C-based (C++ based)
  - World-view: Processes

General purpose simulation languages: Ctd

- SLAM:
  - Block-structured Language
  - Interpretive Execution
  - FORTRAN-based (and extended)
  - World-view: Network / event / continuous
- CSIM:
  - process-oriented language
  - C-based (C++ based)
  - World-view: Processes

General purpose simulation languages: Ctd

- SLAM:
  - Block-structured Language
  - Interpretive Execution
  - FORTRAN-based (and extended)
  - World-view: Network / event / continuous
- CSIM:
  - process-oriented language
  - C-based (C++ based)
  - World-view: Processes

Modeling with special purpose simulation languages

- Advantages:
  - Very quick development of complex models
  - Short learning cycle
  - No programming–minimal errors in usage
- Disadvantages:
  - High cost of software
  - Limited scope of applicability
  - Limited flexibility (may not fit your specific application)

Modeling with special purpose simulation languages

- Advantages:
  - Very quick development of complex models
  - Short learning cycle
  - No programming–minimal errors in usage
- Disadvantages:
  - High cost of software
  - Limited scope of applicability
  - Limited flexibility (may not fit your specific application)

Modeling with special purpose simulation languages

- Advantages:
  - Very quick development of complex models
  - Short learning cycle
  - No programming–minimal errors in usage
- Disadvantages:
  - High cost of software
  - Limited scope of applicability
  - Limited flexibility (may not fit your specific application)

Modeling with special purpose simulation languages

- Advantages:
  - Very quick development of complex models
  - Short learning cycle
  - No programming–minimal errors in usage
- Disadvantages:
  - High cost of software
  - Limited scope of applicability
  - Limited flexibility (may not fit your specific application)

Modeling with special purpose simulation languages

- Advantages:
  - Very quick development of complex models
  - Short learning cycle
  - No programming–minimal errors in usage
- Disadvantages:
  - High cost of software
  - Limited scope of applicability
  - Limited flexibility (may not fit your specific application)

Modeling with special purpose simulation languages

- Advantages:
  - Very quick development of complex models
  - Short learning cycle
  - No programming–minimal errors in usage
- Disadvantages:
  - High cost of software
  - Limited scope of applicability
  - Limited flexibility (may not fit your specific application)

Special purpose packages used for simulation

- NETWORK II.5: Simulator for computer systems.
- OPNET: Simulator for communication networks, including wireless networks
- COMNET III: Simulator for communications networks
- ADA: Department of Defense attempt at standardization
- SIMFACTORY: Simulator for manufacturing operations

Special purpose packages used for simulation

- NETWORK II.5: Simulator for computer systems.
- OPNET: Simulator for communication networks, including wireless networks
- COMNET III: Simulator for communications networks
- ADA: Department of Defense attempt at standardization
- SIMFACTORY: Simulator for manufacturing operations

Special purpose packages used for simulation

- NETWORK II.5: Simulator for computer systems.
- OPNET: Simulator for communication networks, including wireless networks
- COMNET III: Simulator for communications networks
- ADA: Department of Defense attempt at standardization
- SIMFACTORY: Simulator for manufacturing operations

Special purpose packages used for simulation

- NETWORK II.5: Simulator for computer systems.
- OPNET: Simulator for communication networks, including wireless networks
- COMNET III: Simulator for communications networks
- ADA: Department of Defense attempt at standardization
- SIMFACTORY: Simulator for manufacturing operations

Special purpose packages used for simulation

- NETWORK II.5: Simulator for computer systems.
- OPNET: Simulator for communication networks, including wireless networks
- COMNET III: Simulator for communications networks
- ADA: Department of Defense attempt at standardization
- SIMFACTORY: Simulator for manufacturing operations

The real cost of simulation

Many people think of the cost of a simulation only in terms of the software package price.

There are actually at least three components to the cost of simulation:

1. Purchase price of the software
2. Programmer / Analyst time
3. "Timeliness of Results"

The real cost of simulation

Many people think of the cost of a simulation only in terms of the software package price.

There are actually at least three components to the cost of simulation:

1. Purchase price of the software
2. Programmer / Analyst time
3. "Timeliness of Results"

The real cost of simulation

Many people think of the cost of a simulation only in terms of the software package price.

There are actually at least three components to the cost of simulation:

1. Purchase price of the software
2. Programmer / Analyst time
3. ”Timeliness of Results”

The real cost of simulation

Many people think of the cost of a simulation only in terms of the software package price.

There are actually at least three components to the cost of simulation:

1. Purchase price of the software
2. Programmer / Analyst time
3. "Timeliness of Results"

The real cost of simulation

Many people think of the cost of a simulation only in terms of the software package price.

There are actually at least three components to the cost of simulation:

1. Purchase price of the software
2. Programmer / Analyst time
3. ”Timeliness of Results”

Terminology

- System:
  - A group of objects that are joined together in some regular interaction or interdependence toward the accomplishment of some purpose
  - Entity
  - An object of interest in the system.
  - E.g., customers at a bank

Terminology

- System:
  - A group of objects that are joined together in some regular interaction or interdependence toward the accomplishment of some purpose
  - Entity
  - An object of interest in the system.
  - E.g., customers at a bank

Terminology

- System:
  - A group of objects that are joined together in some regular interaction or interdependence toward the accomplishment of some purpose
  - Entity
  - An object of interest in the system.
  - E.g., customers at a bank

Terminology

- System:
  - A group of objects that are joined together in some regular interaction or interdependence toward the accomplishment of some purpose
  - Entity
  - An object of interest in the system.
  - E.g., customers at a bank

Terminology: Ctd

- Attribute:
  - a property of an entity
  - E.g., checking account balance
- Activity:
  - Represents a time period of specified length.
  - Collection of operations that transform the state of an entity
  - E.g., making bank deposits

Terminology: Ctd

- Attribute:
  - a property of an entity
  - E.g., checking account balance
- Activity:
  - Represents a time period of specified length.
  - Collection of operations that transform the state of an entity
  - E.g., making bank deposits

Terminology: Ctd

- Attribute:
  - a property of an entity
  - E.g., checking account balance
- Activity:
  - Represents a time period of specified length.
  - Collection of operations that transform the state of an entity
  - E.g., making bank deposits

Terminology: Ctd

- Attribute:
  - a property of an entity
  - E.g., checking account balance
- Activity:
  - Represents a time period of specified length.
  - Collection of operations that transform the state of an entity
  - E.g., making bank deposits

Terminology: Ctd

- Attribute:
  - a property of an entity
  - E.g., checking account balance
- Activity:
  - Represents a time period of specified length.
  - Collection of operations that transform the state of an entity
  - E.g., making bank deposits

Terminology: Ctd

- Event:
  - change in the system state
  - E.g., arrival; beginning of a new execution; departure
- Activity:
  - Define the state of the system
  - Can restart simulation from state variables
  - E.g., length of the job queue.

Terminology: Ctd

- Event:
  - change in the system state
  - E.g., arrival; beginning of a new execution; departure
- Activity:
  - Define the state of the system
  - Can restart simulation from state variables
  - E.g., length of the job queue.

Terminology: Ctd

- Event:
  - change in the system state
  - E.g., arrival; beginning of a new execution; departure
- Activity:
  - Define the state of the system
  - Can restart simulation from state variables
  - E.g., length of the job queue.

Terminology: Ctd

- Event:
  - change in the system state
  - E.g., arrival; beginning of a new execution; departure
- Activity:
  - Define the state of the system
  - Can restart simulation from state variables
  - E.g., length of the job queue.

Terminology: Ctd

- Event:
  - change in the system state
  - E.g., arrival; beginning of a new execution; departure
- Activity:
  - Define the state of the system
  - Can restart simulation from state variables
  - E.g., length of the job queue.

Terminology: Ctd

- Process:
  - Sequence of events ordered on time

Note: the three concepts(event, process,and activity) give rise to three alternative ways of building discrete simulation models

Terminology: Ctd

- Process:
  - Sequence of events ordered on time

Note: the three concepts(event, process,and activity) give rise to three alternative ways of building discrete simulation models

- Pure Continuous Simulation
- Pure Discrete Simulation
  - Event-oriented
  - Activity-oriented
  - Process-oriented
- Combined Discrete / Continuous Simulation

- Pure Continuous Simulation
- Pure Discrete Simulation
  - Event-oriented
  - Activity-oriented
  - Process-oriented
- Combined Discrete / Continuous Simulation

- Pure Continuous Simulation
- Pure Discrete Simulation
  - Event-oriented
  - Activity-oriented
  - Process-oriented
- Combined Discrete / Continuous Simulation

- Pure Continuous Simulation
- Pure Discrete Simulation
  - Event-oriented
  - Activity-oriented
  - Process-oriented
- Combined Discrete / Continuous Simulation

- Pure Continuous Simulation
- Pure Discrete Simulation
  - Event-oriented
  - Activity-oriented
  - Process-oriented
- Combined Discrete / Continuous Simulation

Examples of both types of models

- Continuous Time and Discrete Time Models:

  CPU scheduling model vs. number of students attending the class

Examples of both types of models

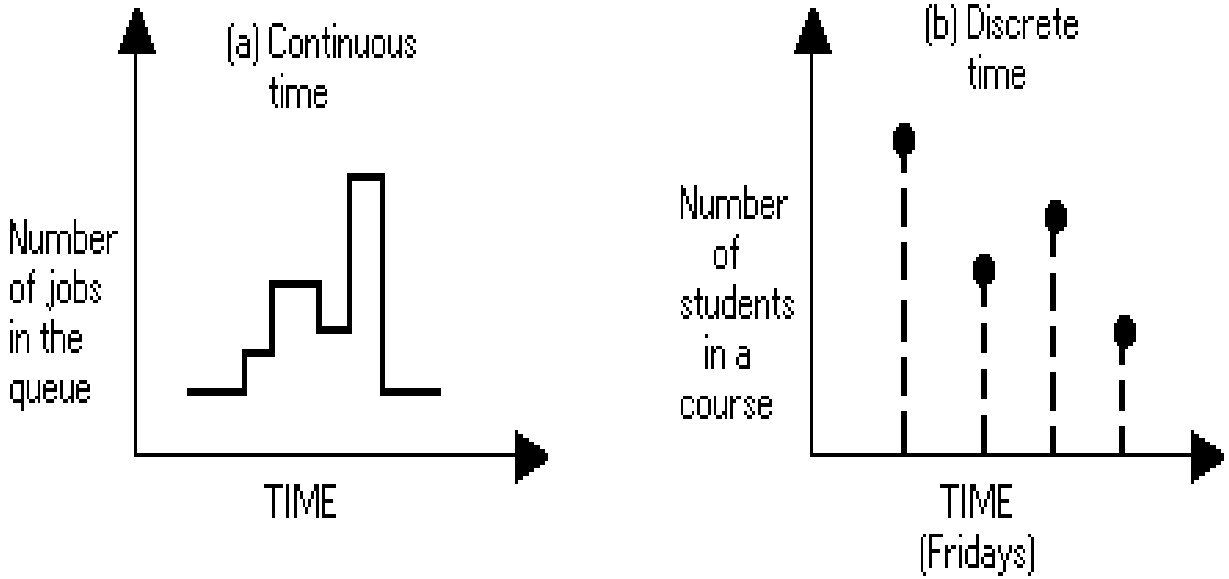
- Continuous Time and Discrete Time Models:

  CPU scheduling model vs. number of students attending the class

Examples of both types of models

- Continuous Time and Discrete Time Models:

  CPU scheduling model vs. number of students attending the class



Examples: Ctd

- Continuous State and Discrete State Models:

  Example: Time spent by students in a weekly class vs. Number of jobs in Q.

Examples: Ctd

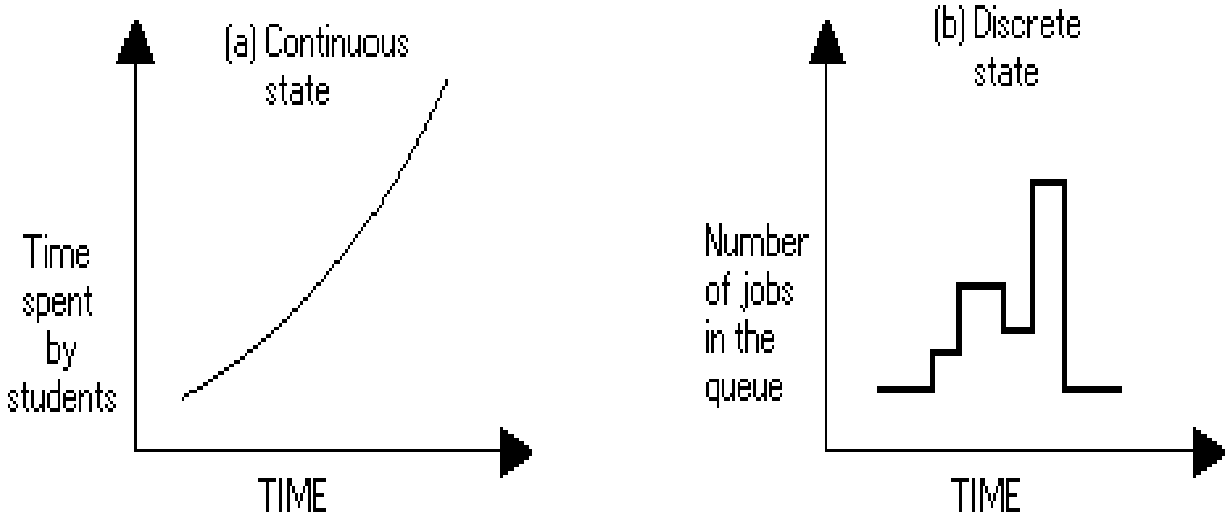
- Continuous State and Discrete State Models:

  Example: Time spent by students in a weekly class vs. Number of jobs in Q.

Examples: Ctd

- Continuous State and Discrete State Models:

  Example: Time spent by students in a weekly class vs. Number of jobs in Q.



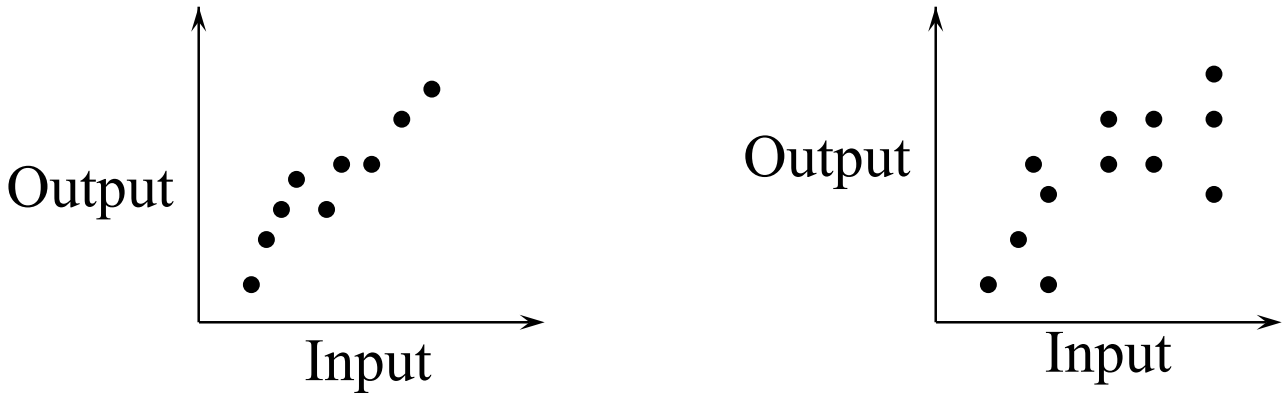
Other types of models

- Deterministic and Probabilistic Models:
- Static and Dynamic Models:

  CPU scheduling model vs. $E = mc^2$

Other types of models

- Deterministic and Probabilistic Models:



- Static and Dynamic Models:

  CPU scheduling model vs. $E = mc^2$

Other types of models

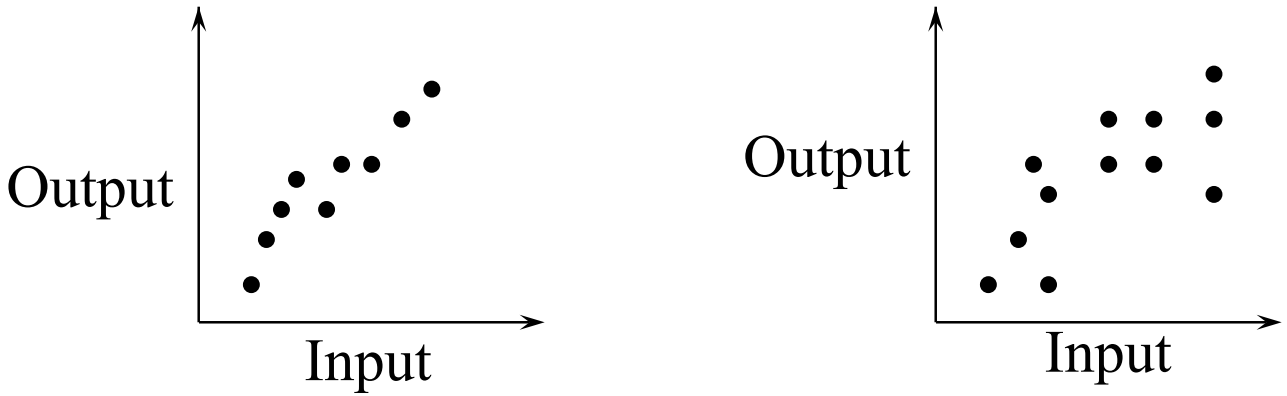
- Deterministic and Probabilistic Models:

- Static and Dynamic Models:

  PU scheduling model vs. $E = mc^2$

Other types of models

- Deterministic and Probabilistic Models:



- Static and Dynamic Models:

  PU scheduling model vs. $E = mc^2$

Other types of models

- Deterministic and Probabilistic Models:

- Static and Dynamic Models:

  CPU scheduling model vs. $E = mc^2$

Model the appropriate level(s) of detail

- Define the boundaries of the system to be modeled.
- Some characteristics of ”the environment” (outside the boundaries) may need to be included in the model.
- Not all subsystems will require the same level of detail.
- Control the tendency to model in great detail those elements of the system which are well understood, while skimming over other, less well-understood sections.

Model the appropriate level(s) of detail

- Define the boundaries of the system to be modeled.
- Some characteristics of ”the environment” (outside the boundaries) may need to be included in the model.
- Not all subsystems will require the same level of detail.
- Control the tendency to model in great detail those elements of the system which are well understood, while skimming over other, less well-understood sections.

Model the appropriate level(s) of detail

- Define the boundaries of the system to be modeled.
- Some characteristics of ”the environment” (outside the boundaries) may need to be included in the model.
- Not all subsystems will require the same level of detail.
- Control the tendency to model in great detail those elements of the system which are well understood, while skimming over other, less well-understood sections.

Model the appropriate level(s) of detail

- Define the boundaries of the system to be modeled.
- Some characteristics of ”the environment” (outside the boundaries) may need to be included in the model.
- Not all subsystems will require the same level of detail.
- Control the tendency to model in great detail those elements of the system which are well understood, while skimming over other, less well-understood sections.

Start early to collect the necessary data

- Data comes in two quantities:

  TOO MUCH!!

  TOO LITTLE!!

- With too much data, we need techniques for reducing it to a form usable in our model.
- With too little data, we need information which can be represented by statistical distributions.

Start early to collect the necessary data

- Data comes in two quantities:

  TOO MUCH!!

  TOO LITTLE!!

- With too much data, we need techniques for reducing it to a form usable in our model.
- With too little data, we need information which can be represented by statistical distributions.

Start early to collect the necessary data

- Data comes in two quantities:

  TOO MUCH!!

  TOO LITTLE!!

- With too much data, we need techniques for reducing it to a form usable in our model.
- With too little data, we need information which can be represented by statistical distributions.

Start early to collect the necessary data

- Data comes in two quantities:

  TOO MUCH!!

  TOO LITTLE!!

- With too much data, we need techniques for reducing it to a form usable in our model.

- With too little data, we need information which can be represented by statistical distributions.

Start early to collect the necessary data

- Data comes in two quantities:

  TOO MUCH!!

  TOO LITTLE!!

- With too much data, we need techniques for reducing it to a form usable in our model.

- With too little data, we need information which can be represented by statistical distributions.

Provide adequate and on-going documentation

- In general, programmers hate to document. (They love to program!)
- Documentation is always their lowest priority item. (Usually scheduled for just after the budget runs out!)
- They believe that"only wimps read manuals."
- What can we do?
  - Use self-documenting languages
  - Insist on built-in user instructions(help screens)
  - Set (or insist on) standards for coding style

Provide adequate and on-going documentation

- In general, programmers hate to document. (They love to program!)
- Documentation is always their lowest priority item. (Usually scheduled for just after the budget runs out!)
- They believe that"only wimps read manuals."
- What can we do?
  - Use self-documenting languages
  - Insist on built-in user instructions(help screens)
  - Set (or insist on) standards for coding style

Provide adequate and on-going documentation

- In general, programmers hate to document. (They love to program!)
- Documentation is always their lowest priority item. (Usually scheduled for just after the budget runs out!)
- They believe that”only wimps read manuals.”
- What can we do?
  - Use self-documenting languages
  - Insist on built-in user instructions(help screens)
  - Set (or insist on) standards for coding style

Provide adequate and on-going documentation

- In general, programmers hate to document. (They love to program!)
- Documentation is always their lowest priority item. (Usually scheduled for just after the budget runs out!)
- They believe that”only wimps read manuals.”
- What can we do?
  - Use self-documenting languages
  - Insist on built-in user instructions(help screens)
  - Set (or insist on) standards for coding style

Provide adequate and on-going documentation

- In general, programmers hate to document. (They love to program!)
- Documentation is always their lowest priority item. (Usually scheduled for just after the budget runs out!)
- They believe that”only wimps read manuals.”
- What can we do?
  - Use self-documenting languages
  - Insist on built-in user instructions(help screens)
  - Set (or insist on) standards for coding style

Provide adequate and on-going documentation

- In general, programmers hate to document. (They love to program!)
- Documentation is always their lowest priority item. (Usually scheduled for just after the budget runs out!)
- They believe that”only wimps read manuals.”
- What can we do?
  - Use self-documenting languages
  - Insist on built-in user instructions(help screens)
  - Set (or insist on) standards for coding style

Provide adequate and on-going documentation

- In general, programmers hate to document. (They love to program!)
- Documentation is always their lowest priority item. (Usually scheduled for just after the budget runs out!)
- They believe that”only wimps read manuals.”
- What can we do?
  - Use self-documenting languages
  - Insist on built-in user instructions(help screens)
  - Set (or insist on) standards for coding style

Provide adequate and on-going documentation

- Did we get the"right answers" (No such thing!!)
- Simulation provides something that no other technique does:
  - Step by step tracing of the model execution.
  - This provides a very natural way of checking the internal consistency of the model.

Provide adequate and on-going documentation

- Did we get the"right answers" (No such thing!!)
- Simulation provides something that no other technique does:
  - Step by step tracing of the model execution.
  - This provides a very natural way of checking the internal consistency of the model.

Provide adequate and on-going documentation

- Did we get the"right answers" (No such thing!!)
- Simulation provides something that no other technique does:
  - Step by step tracing of the model execution.
  - This provides a very natural way of checking the internal consistency of the model.

Provide adequate and on-going documentation

- Did we get the"right answers" (No such thing!!)
- Simulation provides something that no other technique does:
  - Step by step tracing of the model execution.
  - This provides a very natural way of checking the internal consistency of the model.

Develop a plan for model validation

- VALIDATION: ”Doing the right thing” or ”Asking the right questions”
- How do we know our model represents the system under investigation?
  - Compare to existing system?
  - Deterministic case?

Develop a plan for model validation

- VALIDATION: ”Doing the right thing” or ”Asking the right questions”
- How do we know our model represents the system under investigation?
  - Compare to existing system?
  - Deterministic case?

Develop a plan for model validation

- VALIDATION: ”Doing the right thing” or ”Asking the right questions”
- How do we know our model represents the system under investigation?
  - Compare to existing system?
  - Deterministic case?

Develop a plan for model validation

- VALIDATION: ”Doing the right thing” or ”Asking the right questions”
- How do we know our model represents the system under investigation?
  - Compare to existing system?
  - Deterministic case?

Develop a plan for statistical output analysis

- How much is enough? Long runs versus Replications
- Techniques for Analysis

Develop a plan for statistical output analysis

- How much is enough? Long runs versus Replications
- Techniques for Analysis

End!