GOMS Models - Simplified Cognitive Architectures

David E. Kieras

University of Michigan

Introduction to the Session

GOMS models are a well-developed tool for Model-based Evaluation that in their current modern form, are based on simplified cognitive architectures.

This session presents the most important forms and uses of GOMS models.

One computational tool for using GOMS will be summarized; Bonnie John will present another in Session 3b.

GOMS Models

GOMS Models: A Family of Engineering Models

GOMS Model Family as Simplified Cognitive Architectures

GOMS Models: A Family of Engineering Models

Definition of GOMS model:

An approach to describing the knowledge of procedures that a user must have in order to operate a system.

- Proposed by Card, Moran, & Newell (1983).

Goals - what goals can be accomplished with the system.
Operators - what basic actions can be performed.
Methods - what sequences of operators can be used to accomplish each goal.
Selection Rules - which method should be used to accomplish a goal.

GOMS is limited to describing procedural knowledge.

Interfaces have other aspects!

Family summarized here:

Keystroke-Level Model (KLM).
Critical-Path Method GOMS (CPM-GOMS).
Natural GOMS Language (NGOMSL)/Cognitive Complexity Theory.
Executable GOMS Language (GOMSL)/GLEAN.

See John & Kieras (1996) papers for more complete presentation.

Including example applications.

GOMS Model Family as Simplified Cognitive Architectures

Simple Stage Model Architecture.
A sequence of stages, with characteristic times, produces behavior.
Keystroke-Level Model.

Elementary Human Performance Architecture.
Cognitive system executes a procedural knowledge representation.
A shared working memory for temporary task information.
Simple stage model for rest of activity.
Original CMN-GOMS, CCT, NGOMSL models.

Multiple Parallel Processor Architecture.
Perceptual, cognitive, motor processes run in parallel.
- E.g. Card, Moran, & Newell (1983) Model Human Processor.

CPM-GOMS model.
GOMSL/GLEAN.

The Keystroke-Level GOMS Model

The Keystroke-Level Model (KLM) Method

Operators and Times for the Keystroke-Level Model

Example: File Deletion in MacOS, Original Design, Experienced User

Example: Command Key File Deletion, Experienced User

Summary: The Keystroke-Level Model

The Keystroke-Level Model (KLM) Method

Adapted from Card, Moran, & Newell (1983)

1. Choose one or more representative task scenarios.
2. Have design specified to the point that keystroke-level actions can be listed.
3. List the keystroke-level actions (operators) involved in doing the task.
4. Insert mental operators for when user has to stop and think.
5. Look up the standard execution time to each operator.
6. Add up the execution times for the operators.
7. The total is the estimated time to complete the task.

Operators and Times for the Keystroke-Level Model

K - Keystroke (.12 - 1.2 sec; use .28 sec for ordinary user).
Pressing a key or button on the keyboard.
Different experience levels have different times.
Pressing SHIFT or CONTROL key is a separate keystroke.
Use type operator **T(n)** for a series of **n K**s done as a unit.

P - Point with mouse to a target on the display.
Follows Fitts' law - use if possible.
Typically ranges from .8 to 1.5 sec, average (text editing) is 1.1 sec.

B - Press/release mouse button (.1 sec; click is .2).
Highly practiced, simple reaction.

H - Home hands to keyboard or mouse (.4 sec).

W - Wait for system response.
Only when user is idle because can not continue
Have to estimate from system behavior
Often essentially zero in modern systems

M - Mental act of thinking.
Represents pauses for routine activity (not problem-solving).
New users must often pause to remember or verify every step.
Experienced users pause and think only when logically necessary.
Estimates ranges from .6 to 1.35 sec; 1.2 sec is good single value.

Example: File Deletion in MacOS, Original Design, Experienced User

Assumptions:

Experienced user thinks of selecting and dragging an icon as a single operation.
Finding the to-be-deleted icon is still required, since it is different every time.
Moving icons to the trash can is highly practiced:

- The trash can does not have to be located, so finding the trash can is overlapped with pointing to it.
- Finding the trash can is overlapped with pointing to it.
- Verifying that the trash can has been hit is overlapped with pointing to it.
- Final result (bulging can) is not checked since it is redundant with verifying that the can has been hit.

Pointing to the original window is overlapped with finding it.

General procedure

Find the file icon to be deleted and drag it to the trash can.

Operator sequence

1. initiate the deletion **M**
2. find the file icon **M**
3. point to file icon **P**
4. press and hold mouse button **B**
5. drag file icon to trash can icon **P**
6. release mouse button **B**
7. point to original window **P**

Total time = 3P + 2B + 2M = 5.9 sec

Example: Command Key File Deletion, Experienced User

Assumptions:
User operates both mouse and command key with right hand.
Right hand starts and ends on the mouse.

General procedure
Select the file icon to be deleted and hit a command key.

Operator sequence
1. initiate the deletion **M**
2. find the icon for the to-be-deleted file **M**
3. point to file icon **P**
4. click mouse button **BB**
5. move hand to keyboard **H**
6. hit command key **KK**
7. move hand back to mouse **H**

Total time = P + 2B + 2H + 2K + 2M = 5.06 sec
Only slightly faster, due to need to move the hand!

Summary: The Keystroke-Level Model

Illustrates how to consider the details of user behavior.
- Specification of task situations (scenarios, task instances).
- Specific procedure user must follow.
- Exploration of alternative situations and procedures.

A simple, useful method.
- No reason not to use it when it applies.
- Extensive record of successful application.

CPM-GOMS

CPM-GOMS Concept

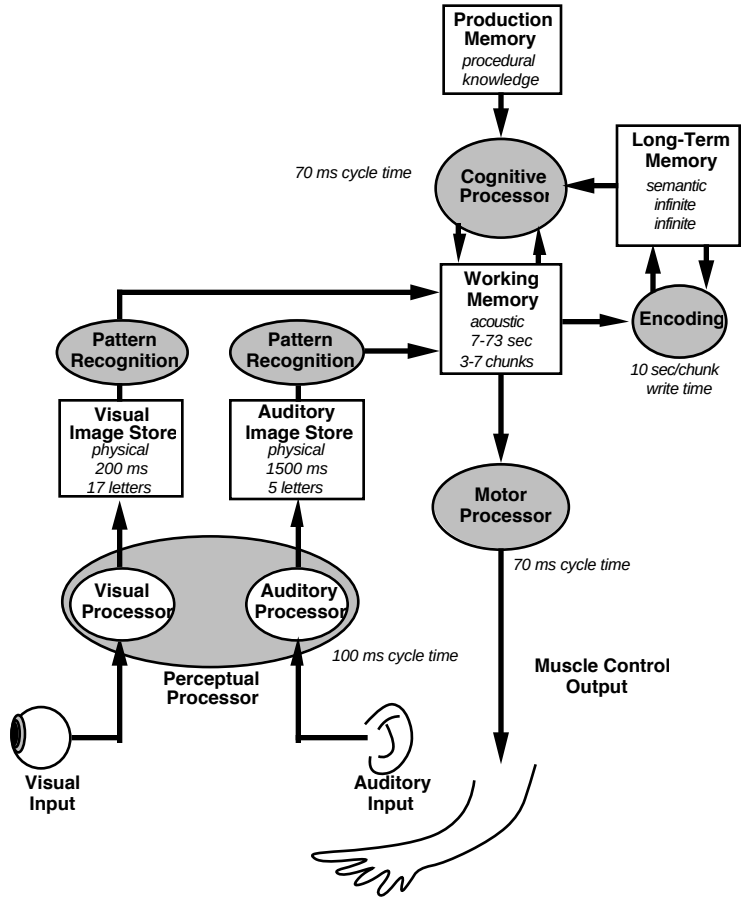
The CPM-GOMS Method

Example Templates

Fragment of a CPM-GOMS Model

CPM-GOMS Concept

Analyze task based on Card, Moran, & Newell (1983) Model Human Processor.
Cognitive, ***P***erceptual, ***M***otor GOMS, or ***C***ritical ***P***ath ***M***ethod ***GOMS***



An informal architecture of cognitive, perceptual, and motor processors with some basic characteristics and time parameters - my slightly modified version.

The CPM-GOMS Method

Start with a task decomposition into basic activities, such as READ-SCREEN or ENTER-COMMAND.

Express these activities in terms of Model Human Processor.

Use a PERT chart tool to describe activities in MHP terms.

John & Gray have described many common activities as "templates" - fragments of PERT charts that can be assembled into more complex models.

First assemble the activities sequentially, then attempt to interleave them where possible.

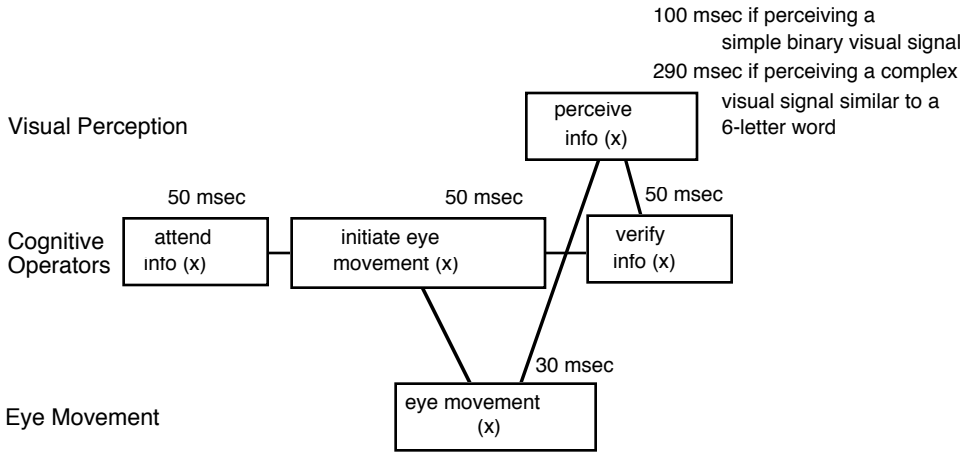
PERT chart tool identifies critical path and required execution time.

Especially useful when concurrent perceptual/motor activity may be limiting execution speed.

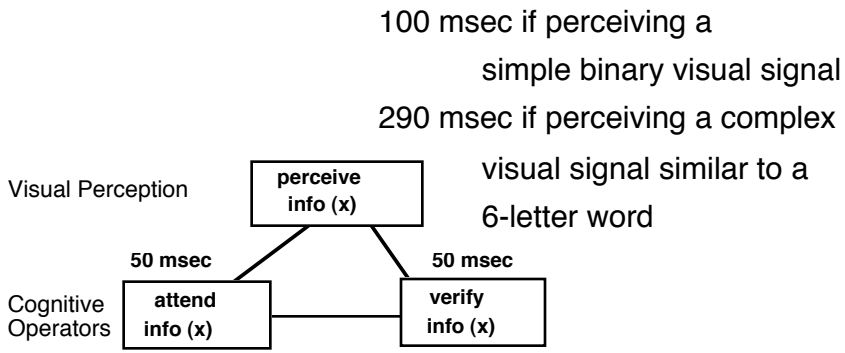
Example Templates

See John & Gray (1995)

Goal: READ-SCREEN, when an eye-movement is required in the task.



Goal: READ-SCREEN, when no eye-movement is required.



Fragment of a CPM-GOMS Model

From John & Kieras (1996), including annotations.

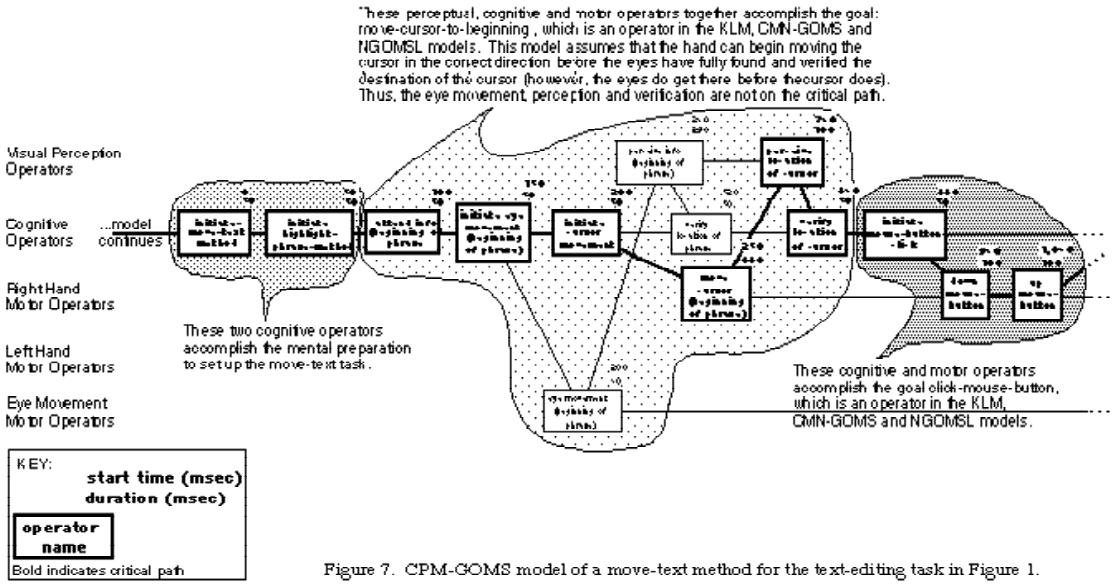


Figure 7. CPM-GOMS model of a move-text method for the text-editing task in Figure 1.

Complete model hardcopy often several feet long.

Very labor-intensive, but can be extremely effective.
E.g. save millions of dollars (Gray, John, Atwood, 1993).

A research goal: Get the same information, but computationally!

NGOMSL

NGOMSL Models

NGOMSL Methodology

NGOMSL Example:
Methods for Recording a Program with a VCR

Methods for Recording a Program (continued)

NGOMSL Models

A high-level structured natural language procedural representation of a simple production-rule cognitive architecture.

"***N***atural" ***GOMS L***anguage

Cognitive Complexity Theory (CCT) - GOMS as production rules.

Kieras & Polson (1985) implementation of basic GOMS concept in production-system form.

- Production-rule actions contain Keystroke-Level Model operators.
- Research shows considerable empirical validity.
  Predicts both learning and execution time from number of rules involved.
- Assumes that GOMS methods are strictly hierarchical and sequential.

NGOMSL Model - GOMS as a ordinary programming language.

Based on CCT research results, but suitable for practical use.

- Production systems are too difficult for use in routine design.
- NGOMSL notation looks like an ordinary programming language.
  One NGOMSL statement is one CCT production rule (basically).

Predicts:

- Relative learning time from a specified design.
- Execution time given a specified design and task scenario.
- Especially sensitive to the consistency of the procedures.

Useful for many conventional desktop computing interface design situations.

NGOMSL Methodology

Top-down breadth-first task decomposition.

Start with the user's top-level goals.

Write a step-by-step procedure for accomplishing each goal in terms of subgoals or keystroke-level operators.

- Use NGOMSL syntax for the procedure.

Recursively write a method for each subgoal until all methods contain only keystroke-level operators.

Write a selection rule to specify which method to use if more than one for a goal.

Count number of statements in methods to predict learning time.

Consistency can be directly reflected by presence of re-used submethods, reducing learning time.

Similar methods also reduce learning time.

For a specific task scenario, count number of statements and operators executed to predict execution time.

If methods have been written to be general, any subsumed scenario can be handled by the model.

Not limited to specific sequences of actions.

NGOMSL Example: Methods for Recording a Program with a VCR

```
Selection rule set for goal: record a program
        If you are present when the program starts
          and you will be present when the program ends
          then accomplish goal: record a program manually
        If you are present when the program starts
          and you will not be present when the program ends
          and you know how long the program lasts
          and the VCR clock is set
          then accomplish goal: record a program with One-Touch Recording
        If you will not be present when the program starts
          and you know when the program will start
          and you know how long the program lasts
          and the VCR clock is set
          then accomplish goal: record a program with Timer Recording
        Return with goal accomplished
```

Methods for Recording a Program (continued)

Method for goal: record a program manually

```
  Step 1.  Wait for program to start.
  Step 2.  Hold down REC button.
  Step 3.  Press PLAY button.
  Step 4.  Release both buttons.
  Step 5.  Verify that display shows "REC" and arrow is moving
  Step 6.  Wait for program to end
  Step 7.  Press STOP button.
  Step 8.  Return with goal accomplished.
```

Method for goal: record a program with One-Touch Recording

```
  Step 1.  Wait for program to start
  Step 2.  Press OTR button.
  Step 3.  Press OTR button.
  Step 4.  Decide: If time shown is less than length of program, go to Step 3.
  Step 5.  Return with goal accomplished.
```

Overview of GLEAN

GOMSL and GLEAN

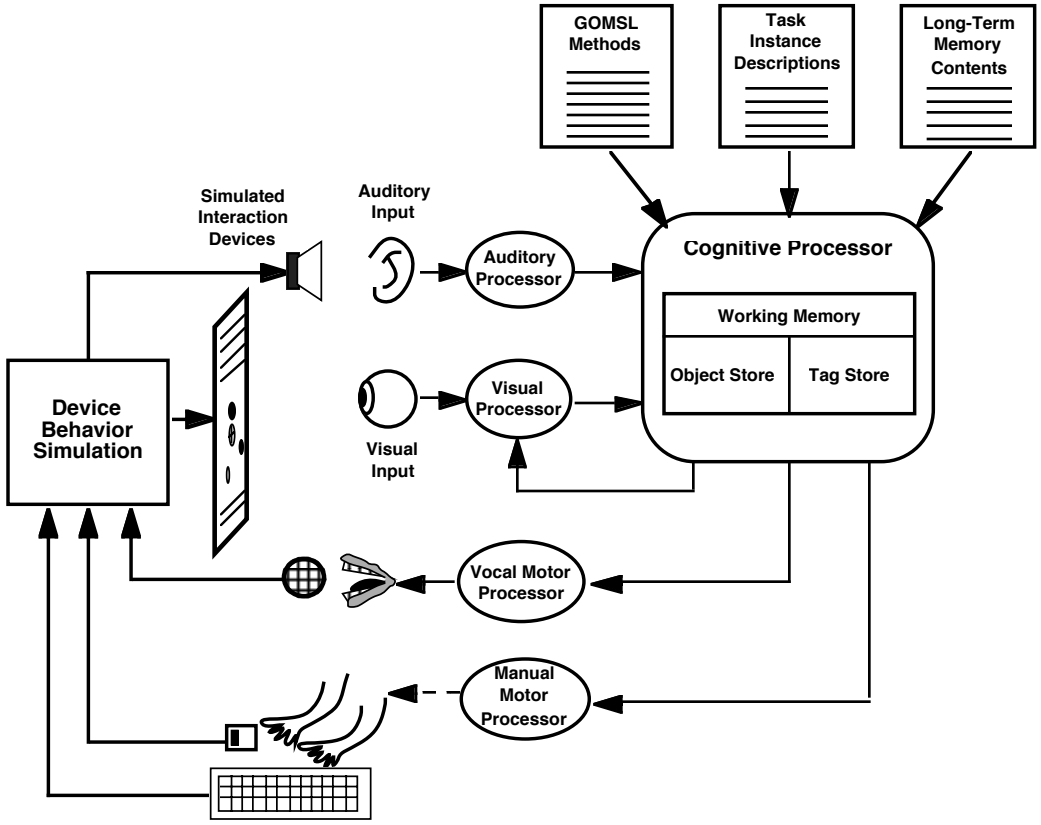
Example of GOMSL

Steps in Using GLEAN

GOMSL and GLEAN

GOMSL - a formalized and executable version of NGOMSL.
GOMS L anguage

GLEAN - a simplified version of the EPIC simulation system.
GOMS ***L***anguage ***E***valuation and ***AN***alysis
Keystroke-Level model representation of perceptual & motor timing.



Example of GOMSL

```
Method_for_goal: Close Track_data_window
  Step 1.   Accomplish_goal: Click_on Button using "Close".
  Step 2.   Return_with_goal_accomplished.

Method_for_goal: Click_on Button using <click_on_button_label>
  Step 1.   Look_for_object_whose Label is <click_on_button_label>,
            and Type is Button and_store_under <click_on_button>.
  Step 2.   Point_to <click_on_button>.
  Step 3.   Click Left_mouse_button.
  Step 4.   Delete <click_on_button>; Return_with_goal_accomplished.
```

Steps in Using GLEAN

1. **Choose and represent the benchmark tasks.**
2. **Write the GOMS model entailed by the interface design.**
3. **Describe the device behavior as needed.**
   A passive "dummy" device may be adequate.
   A scenario-driven interactive device may be required.
4. **Debug the model by running it with GLEAN.**
   Look for correct task performance, intended strategies followed.
5. **Obtain predictions by running final version of model.**
   Time-stamped actions on device, workload profiles.
6. **Examine predictions, profiles, GOMS methods to identify problems and possible improvements.**
7. **Modify the design and GLEAN representations, and obtain new predictions.**

Example Application of GLEAN

Example Application of GLEAN

Collaborator: Tom Santoro, NSMRL

Sponsored by: ONR SC-21 Manning Affordability Initiative

Domain is CIC teams, and how future computer systems could support them better, enabling reduced number of watch-standers on the job.

MultiModal Watch Station (MMWS)

- Glenn Osga, SPAWAR

Basic task: monitor radar display, take various actions, working as a team.

Verify ID of new tracks, monitor suspicious tracks, issue ESM reports, warnings, queries, engagement instructions.

Project Goal:

Model aspects of the design of the MultiModal Watch Station and its software to determine whether and how GOMS and GLEAN can contribute to the design of complex systems.

Use simplified version of interface and system to target key issues quickly.

Details of design and usage procedures were constantly changing.
Could scale up to more complete versions later when design solidifies.

Current Typical Watch Station

MultiModal Watch Station (MMWS)

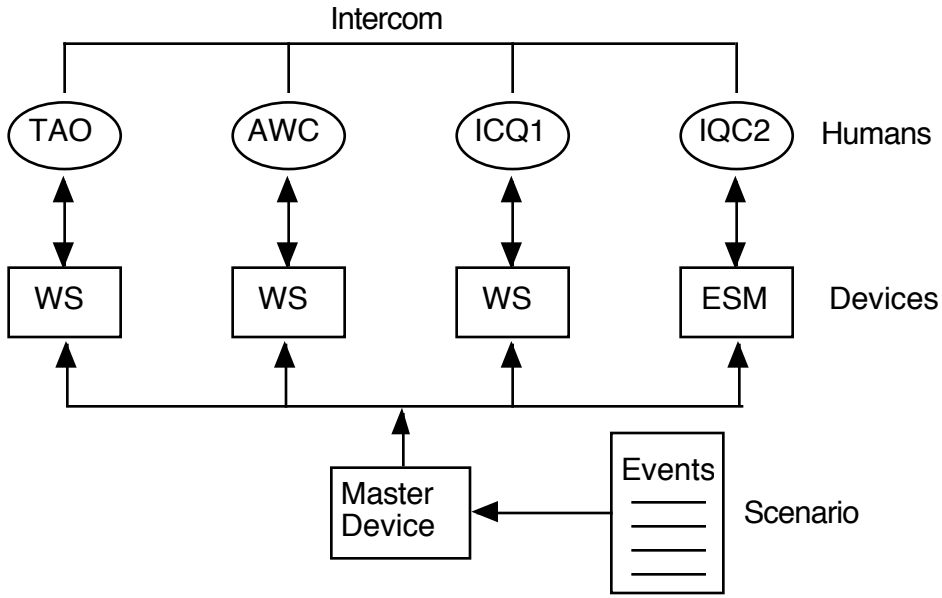
MMWS Main Display

Modeling a Team with a Team of Models

Evaluate team design for a reduced-manning CIC using MMWS.
A model run consisted of 1.5 hr of simulated time at a 1 ms time grain.

Four simulated humans:
Each with a GOMS model for their role, communicating with each other by speech.
Methods based on stated rules of engagement.

Four simulated devices:
Driven by a single input scenario of about 650 external events.
State of displays was dynamically computed 1/sec; dozens of objects displayed.



Team Model Results

Model compared to data obtained from six actual teams.

In verification stage, data from first half of scenario used to refine model:

Must complete actions on critical tracks.
Estimate scaling from GLEAN-generated workload values to workload assessments in data.

In validation stage, model tested against data from second half of scenario.

Average absolute error of prediction, variance accounted for, determined.

Team Model Results (continued)

Performance Measure	Verification	Validation
Critical tracks actions complete	11 of 11	13 of 14
New track report time (abs. error)	13.0%	33.27%
Query time (abs. error)	32.7%	45.50%
Warning time (abs. error)	9.6%	8.89%
ESM report time (abs. error)	13.9%	9.80%
Workload variance accounted for	53%	31% (40% post-hoc)

Data is problematic in many ways, but model is promising.

Small sample size - actual intact Navy teams were used.

The model was based on a specified team design concept, but humans were apparently using their own, different, team procedures.

- A problem in validation for models in complex real-world tasks.

Queries turn out to have not been well-behaved in human data.

Not tightly determined by rules of engagement.

Principled prediction of workload measures from model looks promising.

General Lessons Learned

GOMS really is relatively easy for non-specialists.

Computational GOMS models can scale to complex domains.

Modeling a team can be done with a team of models.
- A GOMS model for each human.
- Models interact using the specified team procedures.
- Should work well anywhere the team's activity consists of routine procedures.

Data on real tasks at this level of complexity is expensive to obtain and likely to be problematic for model validation.
- Later discussion in Practical Issues.

Using GOMS in Design of Functionality

Need for Systematic Design of Functionality

Connecting Task Analysis, Functionality, and Performance

High-Level GOMS Models

Example High-Level GOMS Model

Using a High-Level GOMS Model to Analyze Functionality

Need for Systematic Design of Functionality

Usual design process: Choose functionality, then design user interface.
Functionality often chosen haphazardly, independently of usability.

But choice of functionality is most important factor in usability.
A superior interface can't compensate for an inadequate set of functions!
Examples:
- A word processor with no footnote function - can't fix in the interface!
- First generation digital diaries - no time/date or alarm functions!

Indirect damage due to poor choice of functionality: Divert development resources to useless functions.
Need to know what functions are important!
Examples:
- Database that after usability improvements turned out to be unnecessary.
- Word processor with useless multiple-column functionality.

Functions need to be chosen based on task analysis.
Provide those functions that result in a product that is both useful and usable.

Connecting Task Analysis, Functionality, and Performance

Usually a disconnect between task analysis methods and modeling human performance.

Human Factors (HF) task analysis methods do not predict performance.

Though some HF experts think they should.

In fact, usually a "jump" between a task analysis and a human performance model.

But note close similarity between Hierarchical Task Analysis (HTA) and GOMS.

- HTA is most popular single method in HF.

Problem: How to combine task analysis, choice of functionality, and performance modeling?

Just guess, then see how it turns out?

Or can we move systematically from initial task analysis and functionality concepts to detailed design and human performance model?

Use High-Level GOMS models to represent task at basic functionality level.

High-Level GOMS Models

A GOMS model, but at a level above specific interface methods.

Goals:

Refer only to non-interface aspects of user's task.

Operators:

High-level mental operators.

- Think-of operators.
- Decide operators.

Invocations of system functions.

- Unanalyzed high-level operators.
- E.g., update the database.

May not be specific interface interactions.

- Much higher-level than keystroke.
- *Right*: update the database.
- *Wrong*: click on UPDATE button.

Methods

Show order in which mental operators and system functions are executed.

The model can include:

What information users need for system functions and decisions.
How users will detect and correct errors

Example High-Level GOMS Model

Method for goal: Verify circuit with ECAD system

```
  Step 1.  Think-of circuit idea
  Step 2.  Accomplish Goal: Enter circuit into ECAD tool
  Step 3.  Run simulation of circuit with ECAD tool
  Step 4.  Decide: If circuit performs correct function, then
           Return with goal accomplished
  Step 5.  Think-of modification to circuit
  Step 6.  Make modification with ECAD tool
  Step 7.  Go to 3
```

Method for goal: Enter circuit into ECAD system

```
  Step 1.  Invoke drawing tool
  Step 2.  Think-of object to draw next
  Step 3.  If no more objects, then Return with goal accomplished
  Step 4.  Accomplish Goal: draw the next object
  Step 5.  Go to 2
```

Selection rule set for goal: Draw an object

```
  Step 1.  If object is a component, then Accomplish Goal: draw a component
  Step 2.  If object is a wire, then Accomplish Goal: draw a wire
  Step 3.  ...
  Step 4.  Return with goal accomplished
```

Method for goal: Draw a component

```
  Step 1.  Think-of component type
  Step 2.  Think-of component placement
  Step 3.  Invoke component-drawing function with type and placement
  Step 4.  Return with goal accomplished
```

Method for goal: Draw a wire

```
  Step 1.  Think-of starting and ending points for wire
  Step 2.  Think-of route for wire
  Step 3.  Invoke wire drawing function with starting point, ending point, and route
  Step 4.  Return with goal accomplished
```

Using a High-Level GOMS Model to Analyze Functionality

Choose functionality based on high-level GOMS model of the task.

1. Write out high-level methods for user's top-level goals.
   - Methods should be independent of specifics of user interface design.
   - Methods can be used as the top level of final GOMS model of the interface.
2. Choose functionality that allows simple high-level methods.
   - Ensures that system will be both useful and at least reasonably usable.
3. Elaborate design by writing lower-level methods, down to keystroke-level.
   - Introduce interface design specifics in the lower-level methods.
   - Can now identify problems, predict learning and execution time.
4. Revise functionality, methods, interface choices as needed.

Smooth transition between task analysis to detailed design for usability, with a human performance model at the end.

A more precise, usability-oriented, form of conventional requirements analysis.

Summary

Examined a range of GOMS models, going from a trivial architecture to fully computational one.

Simplified architectures permit complex tasks to be modeled with a practical amount of effort.

GOMS methodology is useful from beginning of design to evaluation of different detailed designs.