Multitasking Under Time-Pressure and Uncertainty

Michael Freed

NASA Ames Research Center

Multitasking skill

- Delay answering phone until finished typing sentence

  heuristic: prefer to delay interrupt until good stopping point
- Pull over to side of road before studying map
- Drive back onto road (but don't drive to start point)
- Do something useful when stopped at a red light
- ...

Multitasking can be viewed as skilled behavior for managing task interactions based on learned ***tactics*** *– domain-dependent applications of general heuristics.*

Challenge

Create agents with human-level ability to employ diverse multitask management tactics

- General heuristics underlying tactics

  architecture mechanisms

- Task-specific knowledge

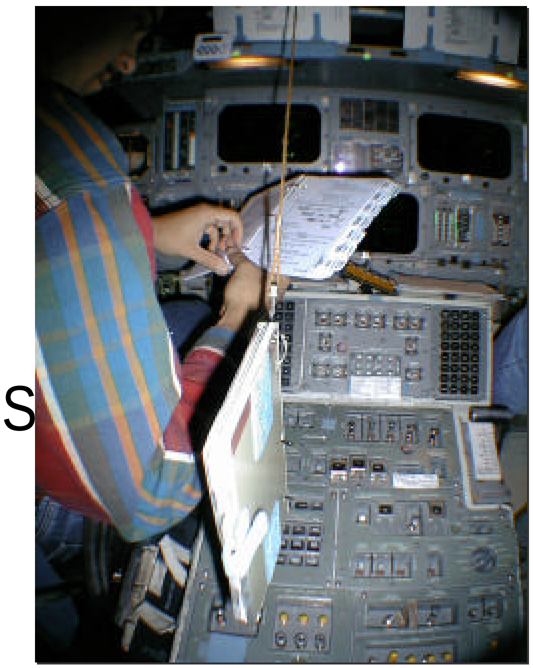
  specialized representation elements

  task representation methodology

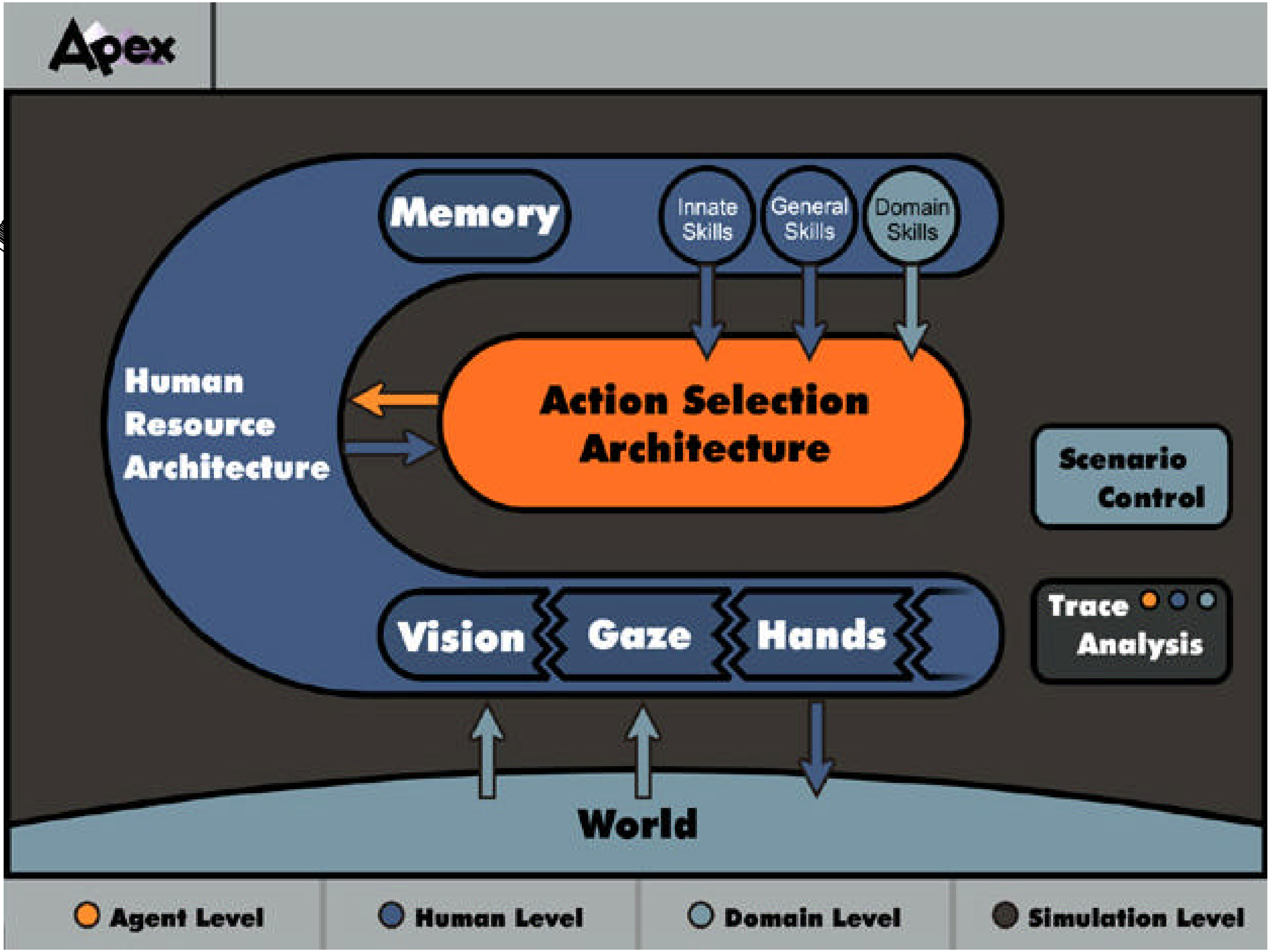
Apex projects: all involve multitasking

Human-level competence
in aviation tasks

Interface evaluation
based on CPM-GOMS



Autonomous robots


Apex
Memory
Innate Skills
General Skills
Domain Skills
Human Resource Architecture
Action Selection Architecture
Scenario Control
Vision
Gaze
Hands
Trace Analysis
World
Agent Level
Human Level
Domain Level
Simulation Level

Action selection architecture

requirements

Many domains of practical interest are demanding in the sense that a skilled agent must :

- Cope with time-pressure
  - Can't deliberate endlessly
- Cope with uncertainty
  - Can't completely know or predict world state
  - Actions may fail or produce undesirable side-effects
  - Preconditions may become unsatisfied
  - Resource requirements may change during execution
  - New, urgent tasks can arise at any time

Two approaches that don't work

Classical planners

- input: current world state, goal state
- output: detailed action sequence to achieve goal state
- can find solutions to hard problems

Classical schedulers

- input: set of actions to do and constraints on order/timing
- output: schedule specifying when to do each action
- can seek optimal solutions

Problems: (1) slow; (2) intolerant of uncertainty

Reactive Planners

- **Coping with time-pressure**
  - Stored plan library
  - Heuristic or single-rule plan refinement
- **Coping with uncertainty**
  - *Action decisions deferred until just before execution*
  - Integrated contingency handling

...but not very good at discovering optimal solutions or solving hard, novel problems

Procedure Description Language (PDL)

```
(procedure
   (index (hold-altitude using mcp))
   (profile right-hand)
   (step s1 (clear right-hand))
   (step s2 (find-loc alt-hold-button => ?loc))
   (step s3 (press-button ?loc right-hand)
      (waitfor (empty right-hand)
               (location alt-hold-button ?loc)))
   (step end (terminate)
      (waitfor (illuminated alt-hold-button))
   (step aux1 (restart ?self)
      (waitfor (resumed ?self))))
```

- concurrency
- reactivity
- hierarchy
- contingency-handling

Multitasking in Apex

- Concurrency control
- Rational interruption and resumption
- Graceful interruption and resumption
- Efficient use of resources

Concurrency Control

PDL idioms

Converge

```
(procedure
  (index (do-it))
  (step s1 (do-A)
  (step s2 (do-B)
  (step s3 (do-C)
     (waitfor ?s1 ?s2)
  (step s4 (terminate)
     (waitfor ?s3)))
```

Race

```
(procedure
  (index (do-it))
  (step s1 (do-A)
  (step s2 (do-B)
  (step s3 (do-C)
     (waitfor ?s1)
     (waitfor ?s2))
  (step s4 (terminate)
     (waitfor ?s3)))
```

Synchronize

```
(procedure
  (index (do-it))
  (step s1 (do-A))
  (step s2 (do-B)
     (waitfor (started ?s1)))
  (step s3 (terminate)
     (waitfor ?s1 ?s2)))
```

Rational interruption and resumption

Determining if tasks conflict

- Profile clause declares resource requirements

```
(profile (<resource> [tolerance]) …)
```

- Some tasks tolerate brief interruptions
- Conflict exists between tasks A and B if
  - A and B both require resource R, *and*
  - Expected Duration (A) > Tolerance (B)

    or Expected Duration (B) > Tolerance (A)

Rational interruption and resumption

Resolving task conflicts

- Compute priority based on task & situational factors:
  - **urgency** (U): measure of time until deadline
  - **importance** (I): cost of missing deadline (time cost)
  - **subjective workload** (S): measure of task crowding
- Urgency dominates if time enough to do everything
- Importance dominates if some deadlines cannot be met

Rational interruption and resumption

Resolving task conflicts

- Simple Priority = $S*I + (S_{max} - S)*U$
- Highest priority task gets resources

  Other tasks aborted or delayed
- Priority values set with priority clause

```
(priority <urgency> <importance>)
```

Graceful interruption and resumption

Capabilities

- Interruption tolerance
- Interruption suppression

```
(interrupt-cost <cost>)
```

- Transition behaviors
  - Interrupt-time, suspension-time, resume-time
  - Illustrates contingency-handling

Graceful interruption and resumption

PDL idioms for transition behaviors

```
(procedure
  (index (fly-cruise-leg using manual-control))
  (step s1 (maintain-altitude)
      (interrupt-cost 5))
      ...
  (step s12 (handoff-to-pilot-not-flying)
     (priority (importance 10) (urgency 10)))
     (waitfor (interrupted ?self)))
  (step s13 (monitor-pilot-not-flying)
     (waitfor (completed ?s12)))
  (step s14 (request-role-pilot-flying)
     (waitfor (resumed ?self)))
           ...)
```

Efficient use of resources

- Combine redundant tasks

```
(merge <condition> [<task pattern>])
```

- Online scheduling to exploit slack
  - Using slack time a scheduling problem
  - Apex scheduling mechanisms
    - Concurrent recursive decomposition => tasks
    - Priority-based allocation => schedule
  - Non-deliberative use of scheduler unusual!

Application-driven language development

There is no commitment to keep the language as-is. PDL is evolving as Apex modelers' needs become better understood.

- New syntax to simplify/abbreviate common patterns
  - e.g. sequential procedures
- Default behaviors to avoid pathological behavior
  - e.g. weak persistence tendency
- New architecture functionality and PDL constructs to access it when needed behavior difficult to represent
  - e.g. rank instead of priority
- Document new idioms as invented

Summary

- Multitasking ability founded on tactical knowledge derived from general heuristics
- Reactive planners can be extended to execute these heuristics in uncertain/time-pressured environments
- Specifically, extensions for concurrency control, interruption handling and resource management facilitate use of multitasking tactics
- Understanding of what needs to be represented and what notation is best for this purpose are improving as new Apex applications are developed

Apex 2.2

Apex is available at

ftp://eos.arc.nasa.gov/outgoing/apex/apex

Multitasking, time-pressure & uncertainty

Multitasking is inherently a problem of coping with time-pressure and uncertainty

- Time-pressure prevents serial execution by imposing deadlines
- Uncertainty limits knowledge of what tasks need to be coordinated and how they will interact

Any intelligent agent needs to multitask under these conditions

Approach

- Identify multitasking tactics used in everyday tasks and in tasks requiring specialized expertise
- Incorporate ability to carry out tactics in
  - Architecture (Apex)
  - Representation language for plan knowledge
  - Methodology for task representation