ECS253 Scribe on May 30, 1997
Christina Chung

- Secure system
  - Specification: formal top-level security system proved to be internally consistent
  - Design: proved to meet specification
  - Implementation: argue rigorously that it matches design
- Hierarchical design methodology
  - divide system into layers of abstraction
  - specify system as state machines
  - SPECIAL:
    - v-functions: give values of state variables, can be primitive operations which obtain values directly or derived operations that obtain values base on the variables
    - o-functions: transitions
    - ov-functions: both
    - Example

```
Module stack;
vfun top(): integer;
     hidden;
     initially: top=0;
     exceptions: none;
vfun data(i:integer):integer;
     hidden;
     initially: ∀i, (data(i)=undefined)
     exceptions: (i<0) or (i>size)
ofun: push(x:integer);
     exceptions: top ≥ size
     effects:
          ‘top=top+1;
          ‘data(‘top)=x;
          ∀j≠’top, ‘data(j)=data(j)
end stack
```

Note: pop is a ovfun since it changes the state and the return value

    - Verification procedures:
      - Stage 1: Interface definition (detection principle, alteration principle)
      - Stage 2: Hierarchical decomposition (layer can only access its own layer or its neighbouring layer(s))
      - Stage 3: Module specification (formal specification of each module developed)

        - Stage 4: Mapping functions: (define functions to capture how they interact with lower level functions, ensure mapping is consistent and meet the specifications)
        - Stage 5: Implementation (start from lower layer to higher layer)
- UCLA Secure Kernel
    - top-down approach of specifications:
        - Top level specification (in informal language)
        - Abstract level specification
        - Low level specification
        - Code satisfying specifications (prove it meet with low level specification)
    - check all accesses are verified, implicit procedures / functions are disallowed
- DEC VMM
    - VMS ULTRIX (virtual machines)
      | |
      \ /
      \ /
      VMM (secure kernel)

      VAX (physical machine)
    - Virtual machine modes Real machine modes
      User ---------- User
      Supervisor ---------- Supervisor
      VM Executive ---------- Executive
      VM Kernel ---------- Executive
      Kernel
    - subjects: user processes, VMs
    - objects: disks, tapes, terminals
    - access classes: integrity level (intlvl), security (seclvl), integrity categories (intcat), security categories (seccat)
    - equality: class(A) = class(B)
    - dormination: A dorminates B if
    - seclvl(A) ≥ seclvl(B) AND
      seccat(A) ⊇ seccat(B) AND
      intlvl(A) ≤ intlvl(B) AND
      intcat(A) ⊆ intcat(B)
    - Levels:
        - 0 VAX hardware
        - 1 modified microcode (since there are 5 virtual machine modes)
        - 2 hardware interrupt handlers
        - 3 lower level schedulers (at VM manager level)
        - 4 I/O services (real I/O, scheduled by lower level schedulers)
        - 5 VM physical space manager (partition memory for VMs)
        - 6 VM virtual space manager

- 7 higher level scheduler (VM scheduling mechanism to schedule processes in VMs)
- 8 audit
- 9 files
- 10 volumes (virtual disk volumes at VM level)
- 11 virtual terminals (user talks to security kernel to request VM)
- 12 virtual printers (labelling secret, top secret ...)
- 13 kernel interface (virtual I/O)
- 14 secure server/virtual VAX
- 15 VM
- 16 user

Note: Level 0 to 14 is the secure kernel.