- Library built on top of C++
- Originally developed as a C++ framework
  for modeling HW design (v1.0)
  An alternative to VHDL or Verilog

- SystemC v2.0: removed SystemC with more
  emphasis on system level design
  Still supports SystemC 1.0 code

- How does it work?
  SystemC incorporates an embedded simulation engine for
  providing concurrency, structural connecting, and event handling

System-Level Modeling

- Ports: connect to an interface of a channel
- Channels: communication channel that provides model for
  specific interfaces
  An interface must define a module (or concurrent process itself)
- Interfaces: Declares a set of functions that define
  the interface to a channel
Dynamic Sensitivity
- Used for synchronization/communication
- Allows for user defined events

```plaintext
wait();
wait(event); wait for event
wait(el1 | el2 | el3); wait for one event (first event)
wait(200, sc-ns); wait for 200 ns
wait(200, sc-ns, el1 | el2); wait for 200 ns of either event el1 or el2
```

**SC-METHODS**: execute completely every time it is activated
- Acts as a function called by simulation engine
- Cannot have `wait()` statement

**SC-THREADS**: executes once at beginning of `execute` (simulation)
- Must have a `wait` statement or it will only execute once
- Typically includes an infinite loop around it desired functionality
Transaction-level Modeling (TLM)

- separate computation from communication
- communicate through channels
- allows independent design/refinement of both
- very useful for system-level design
  especially for systems incorporating SW/HW

Languages for TLM: SystemC, SpecC

TLM is very useful for multiple design methods:
- iterative design and refine method

Model Types:
- Unimed: Functionality without timing
- Approx. Timed: Provides additional implementational details, provides timing information, but without relying on cycle accurate models or sim.
  Details may include system arch, mappings of comp. to processing elements, network organization/hierarchy, etc.
- Cycle-accurate: Provides sufficient low-level details to provide cycle-accurate timing information
  to RTL simulation, or instruction set simulator (ISS)
Bus Arbitration
- Bus transactions require multiple clock cycles
- Bus conflict will occur when two clients request to use
  in simultaneously
- Bus arbitration will manage bus access

Priority Arbitration
- Requests for bus contention is priority arbitrated that will determine
  which component and when to grant bus access
- Priority schemes
  1. Fixed Priority
  2. Rotating Priority

Dynamic/Class Arbitration
- Arbitration integrated into component bus interface
- Order of dynamic chaining will define priority