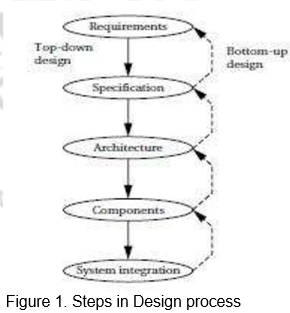
**1. Embedded System Design Process Overview**

* Design follows a structured sequence: **Requirements → Specification → Architecture → Components → Integration**.
* Follows **Top-Down** (abstract to concrete) and **Bottom-Up** (components to system) design approaches.
* Design goals include **low manufacturing cost**, **high performance**, and **low power consumption**.
* Every step includes **analysis**, **refinement**, and **verification** to meet system goals.

**Diagram:**



**2. Requirements**

* Gathered directly from the **customer**; includes both **functional** (what system does) and **non-functional** (cost, size, power) needs.
* Functional: defines **output based on input**; Non-functional: covers **speed**, **cost**, **battery life**, **size**, etc.
* Sets the foundation for building a **specification** and helps define measurable design targets.
* Example: In a GPS system – destination input, route display, power limits, and size constraints.

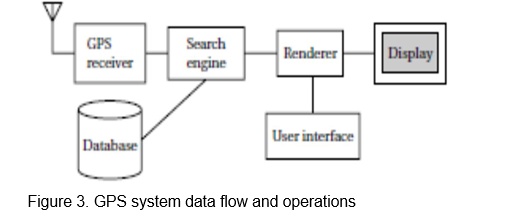


**3. Specification**

* Refines requirements into **clear, measurable, and testable descriptions** for system design.
* Acts as a **contract between customer and developer** to avoid ambiguity in expectations.
* Should define **data sources**, **interfaces**, **operations**, and **user interaction** in detail.
* Example: For a GPS – details of satellite data, UI behavior, map updates, and background tasks.

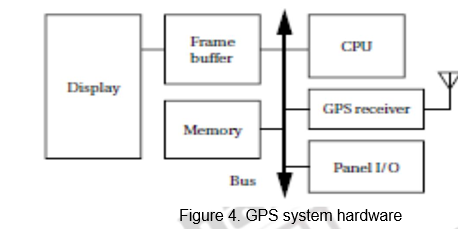
**4. Architecture Design**

1. Converts **specifications into a functional block diagram**, showing how components work together.
2. Architecture is divided into **System**, **Hardware**, and **Software** views.
3. Must satisfy both **functional** goals (like correct navigation) and **non-functional constraints** (like power, speed).
4. Example: GPS system block diagram includes GPS receiver, user input, map database, and renderer.



**5. Hardware & Software Components**

* Hardware: includes processor, memory, sensors, displays, etc. (some components may be ready-made).
* Software: includes embedded programs, drivers, UI code, and control logic based on system tasks.
* Component design must respect **memory**, **speed**, and **power** limits set in earlier steps.
* Example: GPS receiver as hardware; topographic map software with efficient memory usage as software.



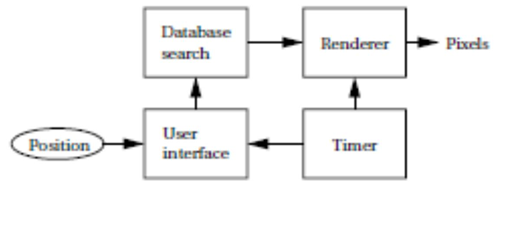


Figure 5. GPS system software

**6. System Integration**

* All components are connected and tested as one system – this phase often reveals hidden **bugs**.
* Start with **small module testing**, then integrate and debug **in stages** to find issues early.
* Use proper **debugging tools** (like breakpoints, logs, JTAG) as embedded systems offer limited visibility.
* Careful planning and step-by-step testing lead to a smoother and faster integration process.

**Design Views: Top-Down vs Bottom-Up**

* **Top-Down**: Start with high-level system idea → refine into components → then into code/hardware.
* **Bottom-Up**: Start from tested modules (e.g., sensor driver) → combine into subsystems → complete system.
* Both are used in real designs to balance creativity, accuracy, and reusability.
* Helps manage uncertainty (future constraints or requirements) and reuse existing parts efficiently.

**Key Design Goals**

* **Cost Efficiency**: Reduce hardware cost, development time, and maintenance overhead.
* **Performance**: Ensure timely processing, fast response to inputs, and efficient multitasking.
* **Power Management**: Optimize code and hardware to reduce energy use (essential for battery devices).
* **Reliability**: System must function under real-world conditions without failure.