Excellent. Now we do this **for real**, exactly how it would be done on an **avionics Mission Computer (MC)** program.

What follows is **Phase-1 fully applied**, not theory.
This is what a senior expects you to *produce*, even if they never say it.

🔵 PHASE-1 APPLIED TO AN AVIONICS MISSION COMPUTER

(System Decomposition & Requirement Ownership)

1️⃣ Phase-1 Starting Point (Inputs)

From **Phase-0**, we already have:

- ✔ System purpose understood
- ✔ Failure consequences known
- ✔ System boundaries defined
- ✔ Criticality understood (Mission-critical / Safety-critical)

Now we move from **"WHY" → "WHAT MUST BE GUARANTEED"**

2️⃣ Functional Decomposition of the Mission Computer

🔹 Rule enforced

- **No CPUs**
- **No cards**
- **No buses**
- **No software names**

Only **functions**.

2.1 Top-Level Functions (Mission Computer)

ID	Function
F-01	Acquire avionics data
F-02	Validate incoming data
F-03	Process mission algorithms
F-04	Fuse multi-source data

ID	Function
F-05	Distribute mission data
F-06	Manage operational modes
F-07	Monitor system health
F-08	Detect and isolate faults
F-09	Report status and BIT
F-10	Handle reset and recovery

👉 If a fresher misses **F-06, F-07, F-08, F-10**, they **fail Phase-1**.

3️⃣ Convert Functions → Functional Requirements

Now we turn functions into **contractual "shall" statements**.

3.1 Core Functional Requirements

Examples (not exhaustive):

- **FR-01**:
  The Mission Computer shall acquire data from connected avionics subsystems.
- **FR-02**:
  The Mission Computer shall validate all incoming data for integrity and correctness.
- **FR-03**:
  The Mission Computer shall process mission data within specified real-time constraints.
- **FR-04**:
  The Mission Computer shall fuse data from multiple sources to produce consolidated outputs.
- **FR-05**:
  The Mission Computer shall distribute processed mission data to dependent subsystems.

3.2 Health & Safety Requirements (Where Professionals Stand Out)

- **FR-06**:
  The Mission Computer shall continuously monitor internal health parameters.
- **FR-07**:
  The Mission Computer shall detect, isolate, and report faults.
- **FR-08**:
  The Mission Computer shall support degraded modes of operation.

- **FR-09**:
  The Mission Computer shall support controlled reset and recovery mechanisms.

⚠️ Freshers focus on FR-01 to FR-05.
⚠️ Seniors watch FR-06 to FR-09.

4️⃣ Functional Flow Diagram (Mission Computer)

This is **mandatory**.

```
Avionics Inputs
      ↓
Data Acquisition
      ↓
Data Validation
      ↓
Mission Processing
      ↓
Data Fusion
      ↓
Data Distribution
      ↓
Avionics Outputs
```

Parallel flows (often missed):

```
          ┌──────────────┐
          │ Health Mon.  │
          └──────┬───────┘
                 ↓
          Fault Detection
                 ↓
           BIT / Status
```

👉 If the junior cannot draw this **without the spec open**, they do not understand the system.

5️⃣ Responsibility Allocation (Critical Phase-1 Skill)

Now we answer:

> "Who is responsible for guaranteeing each function?"

5.1 Allocation Table

Function	System	Hardware	Software
Data acquisition	✔	✔	✔
Data validation	✔		✔
Mission processing	✔		✔
Data fusion	✔		✔
Data distribution	✔	✔	✔
Mode management	✔		✔
Health monitoring	✔	✔	✔
Fault detection	✔	✔	✔
BIT reporting	✔		✔
Reset & recovery	✔	✔	✔

🔥 Key Lesson:

System responsibility NEVER disappears.
It only flows downward.

6️⃣ Identify Derived Requirements (Where Seniors Judge You)

These are **not written explicitly**, but are unavoidable.

6.1 Derived Requirements Examples

Primary Requirement	Derived Requirement
Real-time processing	Deterministic scheduling
High availability	Watchdog timers
Fault tolerance	Redundant paths
Data integrity	ECC memory

Primary Requirement	Derived Requirement
Continuous operation	Thermal monitoring
Reset control	Safe-state definition

If a fresher cannot identify these → they **don't think like an avionics engineer yet**.

7️⃣ Requirement Classification (Professional Structuring)

Each requirement is classified:

Requirement	Type
FR-01	Functional
FR-03	Performance
FR-04	Functional
FR-06	Safety
FR-07	Safety
FR-08	Availability
FR-09	Recovery / Maintenance

This classification later drives:

- Architecture
- Verification
- Certification effort

8️⃣ Ambiguity & Gap Analysis (Mandatory Output)

You **must raise issues**.

Examples for Mission Computer:

- Latency limits not defined for degraded mode
- Fault isolation time not specified
- Behavior during partial power loss unclear
- Reset allowed during mission phase unclear
- BIT reporting frequency not specified

🚨 Silence here = immaturity.

9️⃣ Phase-1 Evaluation Checks (Used by Seniors)

They may ask:

🔍 Check 1

> "Which function is most safety-critical?"

Expected:
👉 Fault detection / wrong data prevention

🔍 Check 2

> "Which requirement drives hardware selection?"

Expected:
👉 Real-time + availability + fault tolerance

🔍 Check 3

> "What happens if data validation fails?"

Expected:
👉 Data rejection + degraded mode + reporting

🔍 Check 4

> "Which requirement is hardest to test?"

Expected:
👉 Fault handling under real-time constraints

🔟 Phase-1 OUTPUTS (Non-Negotiable)

For the Mission Computer, Phase-1 must produce:

1. Functional decomposition list
2. Functional flow diagram
3. Functional requirements list
4. Responsibility allocation table
5. Derived requirements list

6. Ambiguity / gap list

No outputs → Phase-1 **not complete**.

1️⃣1️⃣ Phase-1 COMPLETION CHECK (Mission Computer)

You are Phase-1 complete if you can answer:

> **"Which Mission Computer function, if it silently degrades, is the most dangerous — and why?"**

If you say *"wrong data without fault indication"* → ✅ pass.

What Comes Next (Phase-2)

👉 **Phase-2: Architecture Definition for Mission Computer**

This is where:

- Single vs dual MC
- PPC vs ARM
- VPX / backplane choice
- Redundancy philosophy
- Partitioning HW vs SW

If you want, I can:

- Proceed directly to **Phase-2 (Mission Computer)**
- Create **Phase-1 evaluation tasks for freshers**
- Simulate a **real system design review**
- Turn this into a **training playbook PDF**

Just say the word.