**Mars Climate Orbiter**

Team:  
• Sakka Mohamad-Mario  
• Al-Khalidy Essam  
• Zafar Azzam  
• Mirghani Mahmoud Abdelrahman

**What was the goal?**

* Put the Mars Climate Orbiter into orbit around Mars to study its weather and help relay signals for another lander.

**How did it fail?**

* A simple units mix-up: one team provided thruster data in pound‑seconds, while the navigation software expected newton‑seconds.
* That error made the spacecraft dive far too low during arrival and it was lost in the Martian atmosphere (September 23, 1999).

**Initial snapshot (budget, tech, team & strengths)**

* Budget: About $125M for the orbiter itself; the whole effort is often quoted around $327M including launch and operations.
* Tech: ~638 kg orbiter, launched December 11, 1998 on a Delta II rocket.
* Teams: Spacecraft by Lockheed Martin; navigation by NASA JPL—both highly experienced, which was a major strength.

**Why did it fail? (real reasons, not just “human error”)**

* The unit mismatch wasn’t caught by tests or reviews.
* Weak interface controls: the handoff between contractor and NASA didn’t strictly enforce units.
* Not enough end‑to‑end testing: the exact data files used in flight weren’t fully run through the full navigation chain ahead of time.
* Monitoring gaps: no strong “tripwires” to flag that the accumulated maneuvering didn’t match expectations.

**What happened because of it? (consequences)**

* Loss of the spacecraft and its science return.
* Loss of relay support for Mars Polar Lander.
* Process changes at NASA/JPL: tighter systems engineering, stricter unit policies, and a rethink of the “faster‑better‑cheaper” approach.

**How we would’ve prevented it (practical moves)**

* Use one unit system (metric) everywhere. If something isn’t metric, convert it automatically with a tested tool.
* Make interfaces check themselves. Write the specs so software can verify the units and fail fast if they’re wrong.
* Let code catch unit mistakes. Use libraries that track units; stop anything that doesn’t pass unit tests.
* Practice the full process. Before big maneuvers, run the real contractor files through the same pipeline you’ll use in flight.
* Add clear safety checks. Monitor total ΔV, compare predicted vs. actual path, and set a hard minimum altitude alarm that halts ops if crossed.
* Review together, with proof. Both teams show—using sample data—that units are handled correctly. Sign-off only after a live demo, not just paperwork.

**Work Breakdown Structure**

**A diagram of a software company

AI-generated content may be incorrect.**