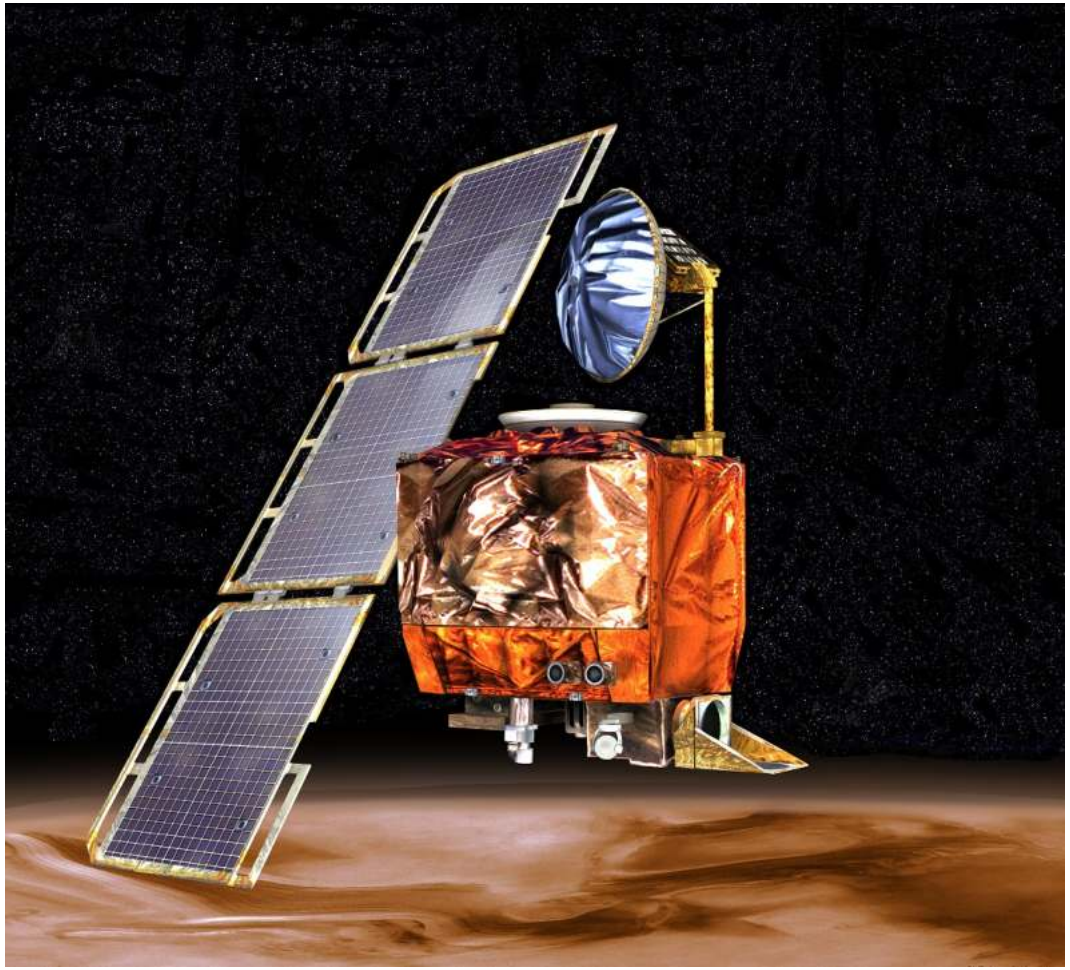
Mismeasure For Measure

PART 1 OF N

Mateusz Pusz
June 28, 2023

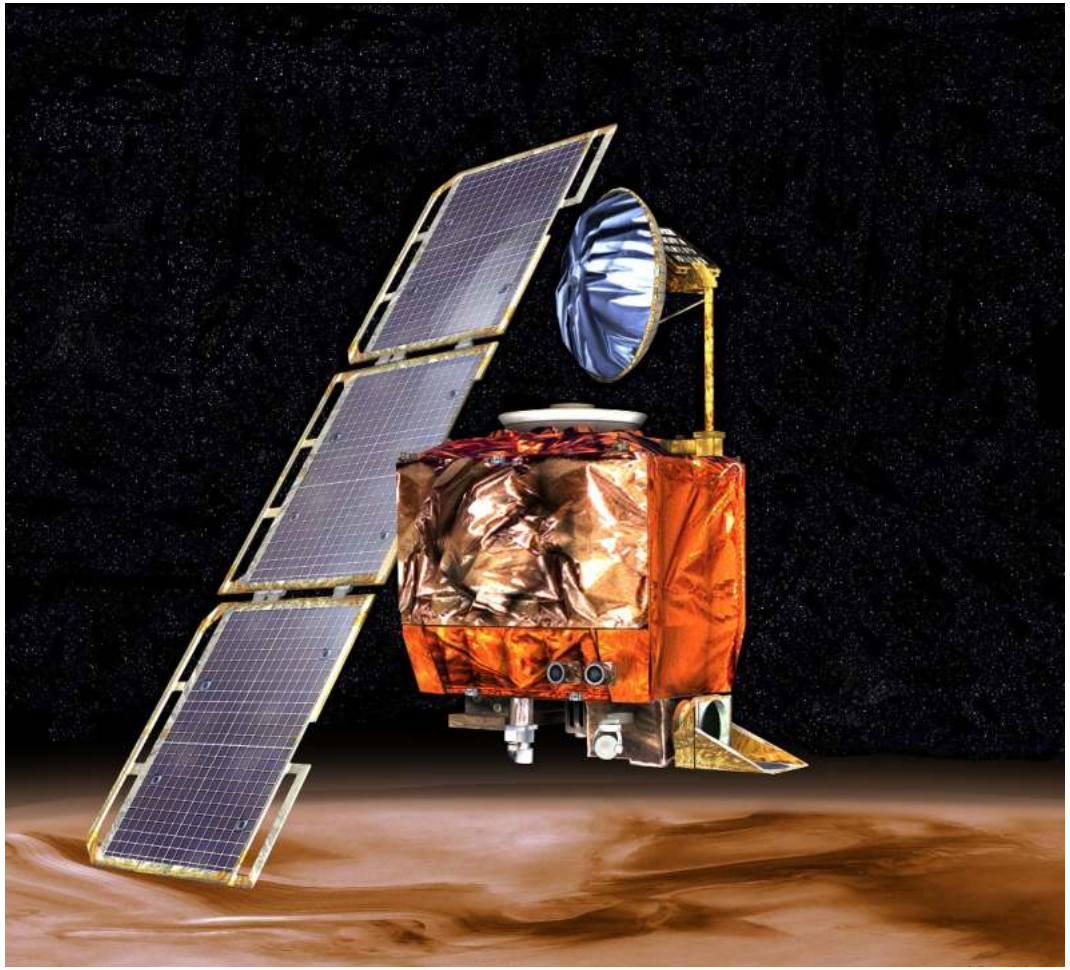
The Mars Climate Orbiter

- **Robotic space probe** launched by NASA on December 11, 1998



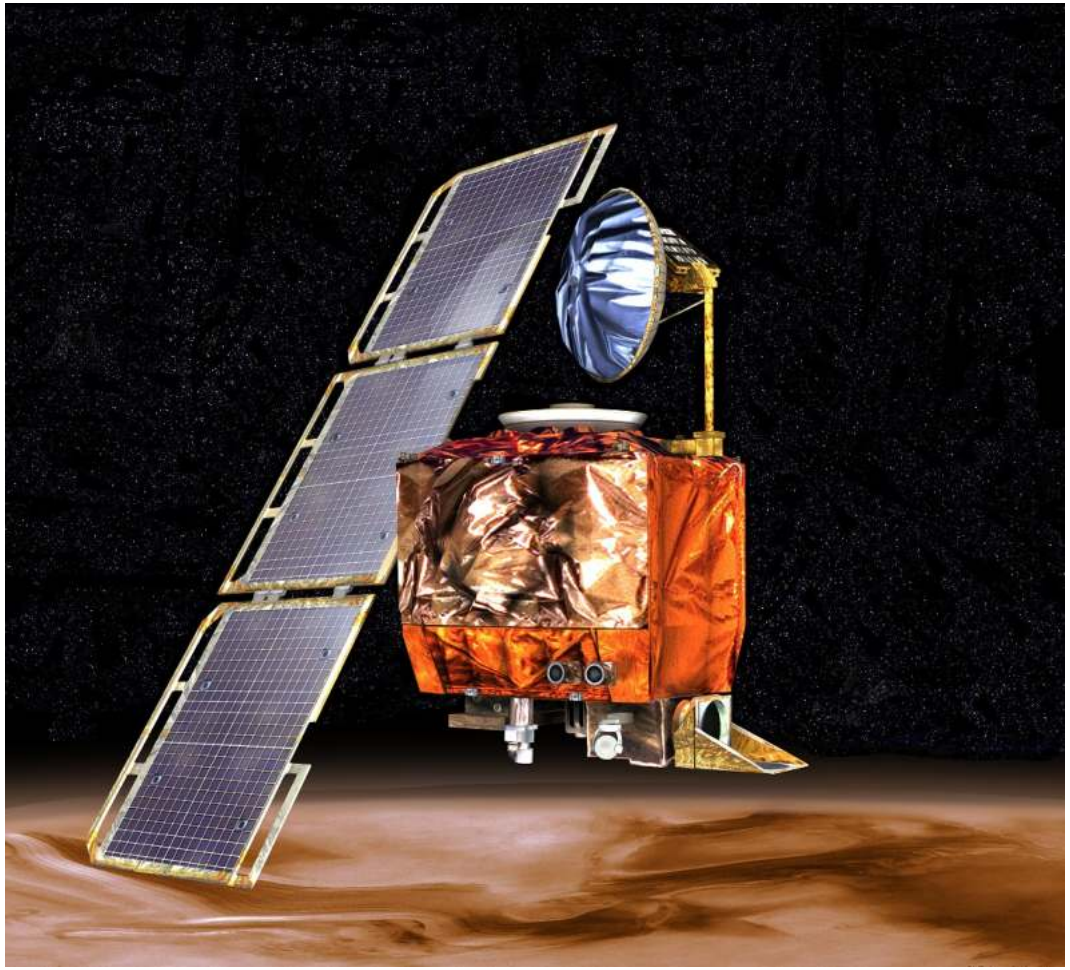
The Mars Climate Orbiter

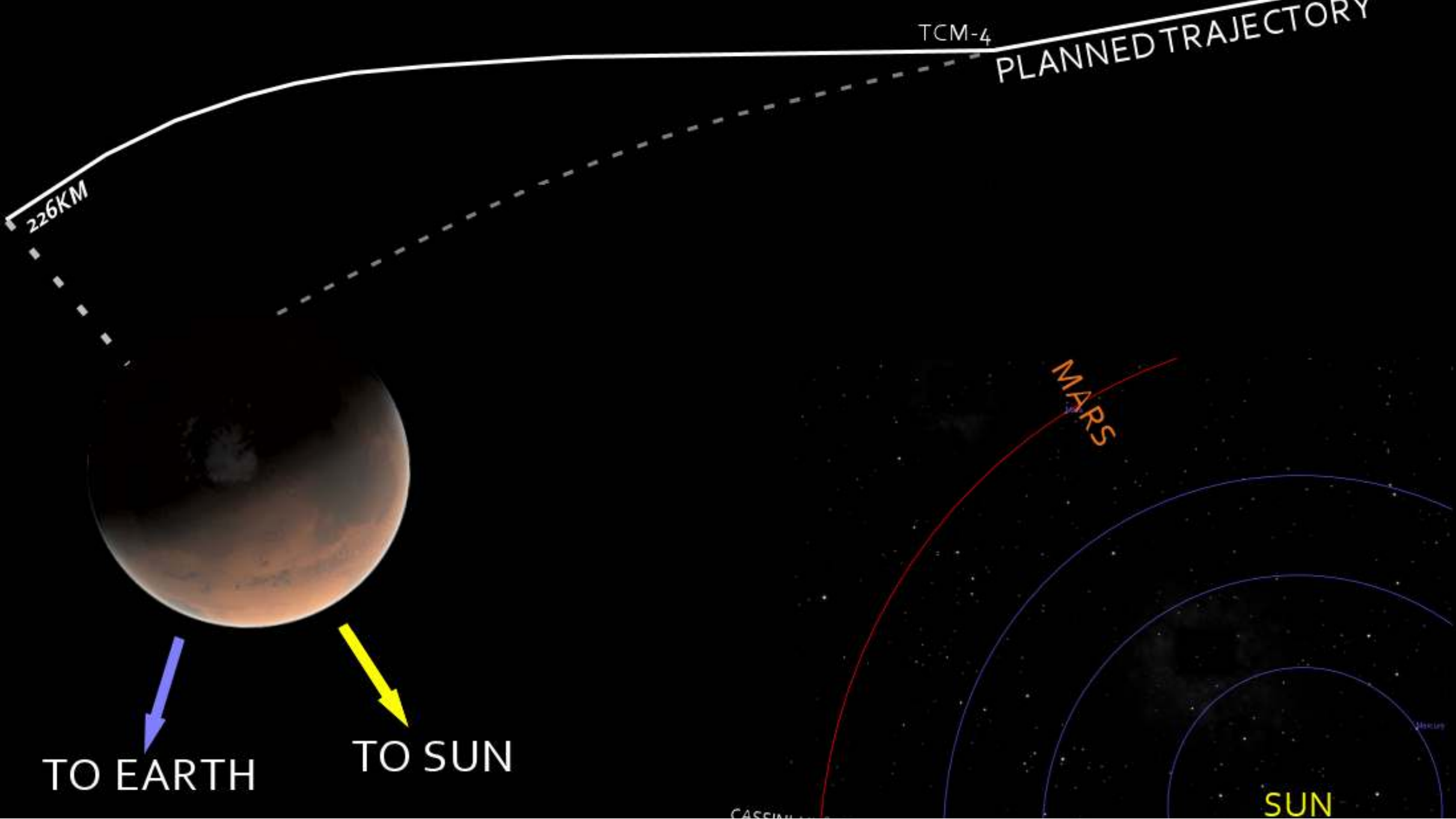
- **Robotic space probe** launched by NASA on December 11, 1998
- Project costs: **$327.6 million**
  - spacecraft development: $193.1 million
  - launching it: $91.7 million
  - mission operations: $42.8 million

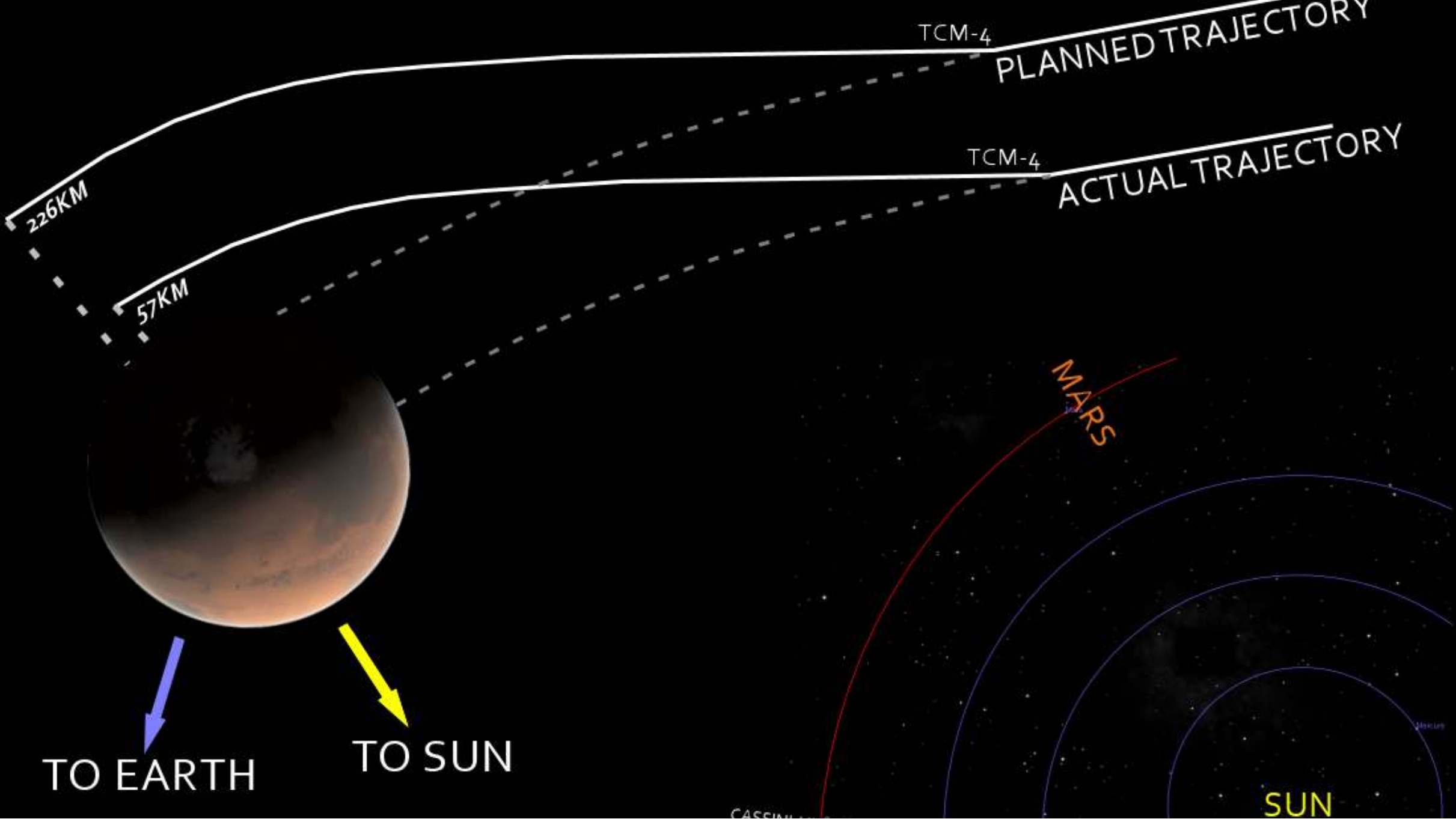


The Mars Climate Orbiter

- **Robotic space probe** launched by NASA on December 11, 1998
- Project costs: **$327.6 million**
  - spacecraft development: $193.1 million
  - launching it: $91.7 million
  - mission operations: $42.8 million
- Mars Climate Orbiter began the planned *orbital insertion maneuver* on September 23, 1999 at 09:00:46 UTC




TCM-4
PLANNED TRAJECTORY
226KM
MARS
TO EARTH
TO SUN
SUN


TCM-4
PLANNED TRAJECTORY
TCM-4
ACTUAL TRAJECTORY
226KM
57KM
MARS
TO EARTH
TO SUN
SUN

The Mars Climate Orbiter

- Space probe went **out of radio contact** when it passed behind Mars at 09:04:52 UTC
  - *49 seconds* earlier than expected
- Communication was never reestablished
- The **spacecraft disintegrated** due to atmospheric stresses or **re-entered heliocentric space** after leaving Mars' atmosphere

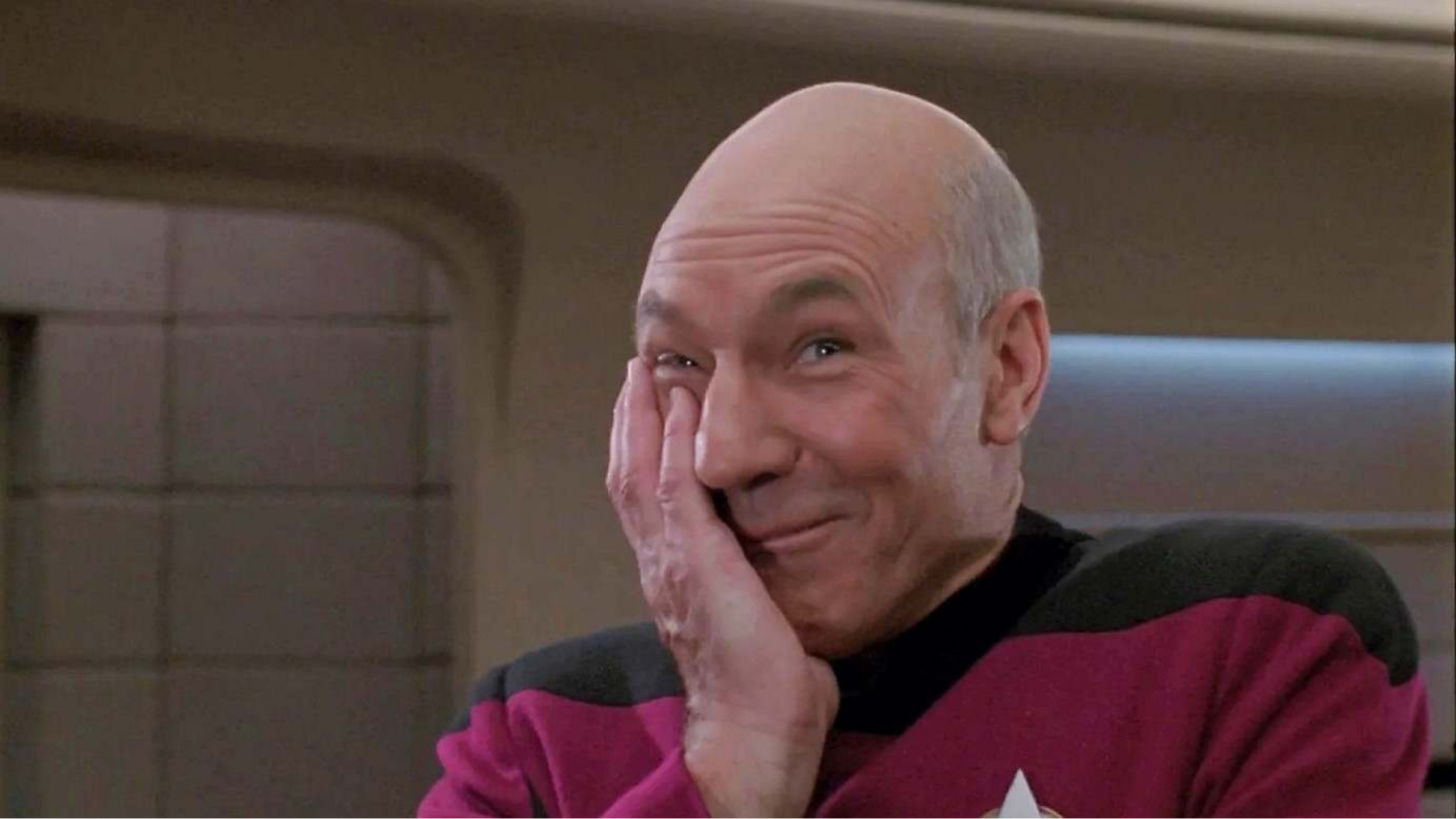
What went wrong?

What went wrong?

- The **primary cause** of this discrepancy was that
  - one piece of ground software supplied by Lockheed Martin produced results in a *United States customary unit*, **contrary to its Software Interface Specification** (SIS)
  - second system, supplied by NASA, expected those results to be in *SI units*, **in accordance** with the SIS

What went wrong?

- The **primary cause** of this discrepancy was that
  - one piece of ground software supplied by Lockheed Martin produced results in a *United States customary unit*, **contrary to its Software Interface Specification** (SIS)
  - second system, supplied by NASA, expected those results to be in *SI units*, **in accordance** with the SIS
- Specifically
  - software that calculated the *total impulse produced by thruster firings* calculated results in **pound-force seconds**
  - the trajectory calculation software then used these results to *update the predicted position* of the spacecraft and expected it to be in **newton seconds**
  - the quantities of those two units should use a **conversion factor of 4.45** if done properly



Taking the Responsibility

- **NASA does not place the responsibility on Lockheed** for the mission loss
- Various officials at NASA have stated that **NASA itself was at fault for failing to make the appropriate checks and tests** that would have caught the discrepancy

Taking the Responsibility

- **NASA does not place the responsibility on Lockheed** for the mission loss
- Various officials at NASA have stated that **NASA itself was at fault for failing to make the appropriate checks and tests** that would have caught the discrepancy
- The **discrepancy between calculated and measured position**, resulting in the discrepancy between desired and actual orbit insertion altitude, **had been noticed earlier by at least two navigators**
  - their **concerns were dismissed because** they *"did not follow the rules about filling out form to document their concerns"*

mp-units

- A Physical Quantities and Units library for C++
- MIT License
- Hosted on Github at mpusz/mp-units
- Extensive documentation at mpusz.github.io/mp-units
- Available in the Compiler Explorer (Thank You Matt Godbolt!!!)

How to prevent such errors with mp-units? (https://godbolt.org/z/TnjbjPsYa)

```
namespace lockheed_martin {
  struct thruster {
    static quantity<usc::pound_force * si::second> get_firing_total_impulse()
    {
      auto q = 42 * (lbf * s);
      std::cout << "Sending total impulse of: " << q << '\n';
      return q;
    }
  };
} // namespace lockheed_martin
```

How to prevent such errors with mp-units? (https://godbolt.org/z/TnjbjPsYa)

```
namespace lockheed_martin {
  struct thruster {
    static quantity<usc::pound_force * si::second> get_firing_total_impulse()
    {
      auto q = 42 * (lbf * s);
      std::cout << "Sending total impulse of: " << q << '\n';
      return q;
    }
  };
} // namespace lockheed_martin
```

```
namespace nasa {
  struct trajectory {
    static void update_position(quantity<si::newton * si::second> q)
    {
      std::cout << "Received total impulse of: " << q << '\n';
    }
  };
} // namespace nasa
```

How to prevent such errors with mp-units? (https://godbolt.org/z/TnjbjPsYa)

```
namespace lockheed_martin {
  struct thruster {
    static quantity<usc::pound_force * si::second> get_firing_total_impulse()
    {
      auto q = 42 * (lbf * s);
      std::cout << "Sending total impulse of: " << q << '\n';
      return q;
    }
  };
} // namespace lockheed_martin
```

```
namespace nasa {
  struct trajectory {
    static void update_position(quantity<si::newton * si::second> q)
    {
      std::cout << "Received total impulse of: " << q << '\n';
    }
  };
} // namespace nasa
```

```
auto q = lockheed_martin::thruster::get_firing_total_impulse();
nasa::trajectory::update_position(q);
```

How to prevent such errors with mp-units? (https://godbolt.org/z/TnjbjPsYa)

```
namespace lockheed_martin {
  struct thruster {
    static quantity<usc::pound_force * si::second> get_firing_total_impulse()
    {
      auto q = 42 * (lbf * s);
      std::cout << "Sending total impulse of: " << q << '\n';
      return q;
    }
  };
} // namespace lockheed_martin
```

```
namespace nasa {
  struct trajectory {
    static void update_position(quantity<si::newton * si::second> q)
    {
      std::cout << "Received total impulse of: " << q << '\n';
    }
  };
} // namespace nasa
```

```
auto q = lockheed_martin::thruster::get_firing_total_impulse();
nasa::trajectory::update_position(q);
```

```
Sending total impulse of: 42 lbf s
Received total impulse of: 186.825 N s
```

How to prevent such errors with mp-units? (https://godbolt.org/z/xzYfEdrTK)

```
namespace lockheed_martin {
  struct thruster {
    static quantity<si::newton * si::second> get_firing_total_impulse()
    {
      auto q = 42 * (lbf * s);
      std::cout << "Sending total impulse of: " << q << '\n';
      return q;
    }
  };
} // namespace lockheed_martin
```

```
namespace nasa {
  struct trajectory {
    static void update_position(quantity<si::newton * si::second> q)
    {
      std::cout << "Received total impulse of: " << q << '\n';
    }
  };
} // namespace nasa
```

```
auto q = lockheed_martin::thruster::get_firing_total_impulse();
nasa::trajectory::update_position(q);
```

```
Sending total impulse of: 42 lbf s
Received total impulse of: 186.825 N s
```


CAUTION
Programming
is addictive
(and too much fun)