A Comet Revisited
Lessons Learned from Philae’s Landing
András Balázs

From the Editors
Two years ago, we published a column on the software involved in landing on a comet. It’s clear by now that although the landing itself was an impressive accomplishment, not everything went as planned. We thank András Balázs for his thorough, honest analysis of what went wrong, what was done, and, importantly, what more could have been done. The software community could benefit from more such evaluations of the problems that so frequently occur in projects. —Michiel van Genuchten and Les Hatton

A Comet Revisited

AFTER A 10-YEAR journey across the Solar System and many maneuvers, the Rosetta spacecraft—carrying the Philae lander, a scientific mini-laboratory (see Figure 1)—smoothly approached comet 67P/Churyumov-Gerasimenko. Rosetta then flew a multitude of low- and high-altitude orbits around the comet, performing scientific experiments and mapping the comet’s shape and surface in detail never seen before.

The Philae mission comprised these phases:

1. cruise (onboard Rosetta);
2. separation, descent, and landing;
3. first comet science;
4. hibernation; and
5. long-term science.

On 12 November 2014, Rosetta initiated the ballistic delivery of Philae to the comet’s nucleus (see Figure 2a). The launch occurred approximately 500 million km from Earth, approximately 3 astronomical units (AUs) from the sun, and 22.5 km from the comet.

Upon first touching down after a descent phase of 7 hours, the lander couldn’t attach itself to the comet, owing to unexpected, probably systematic failures in both parts of the dual-redundant anchoring subsystem and a malfunction of the nonredundant hold-down thruster.1 However, thanks mostly to the comet’s gravitational attraction, Philae still completed its touchdown. After several hours of bouncing and uncontrolled tumbling, Philae reached its final parking position, tilted to one side (see Figure 2b) roughly 1.2 km from the planned landing site. Philae remained functionally intact, despite its anomalous landing. It also managed to maintain radio contact with Rosetta, which served as a relay station between Philae and the flight operations control center on Earth.

Philae’s batteries supplied energy for doing science on the comet’s surface for roughly 60 hours on the first run. Thereafter, the lander went into hibernation owing to the disadvantageous thermal and solar-illumination conditions at its parking site. After about 6 months of hibernation, Philae woke up at 1.8 AU from the sun. Its central onboard computer (CDMS) then autonomously entered the long-term-science phase.

Philae’s Operation Control
A previous Impact department article2 and a more detailed reference paper3 elaborate on the technical, hardware, software, and operational
requirements and their implementation in the CDMS.

Once *Philae* landed, the starting times and durations of the radio visibility windows between *Philae* and *Rosetta* depended on such factors as *Rosetta*’s flight track, the comet’s 12.6-hour rotation period, and *Philae*’s location and orientation on the comet. Although these time windows were nominally calculable, *Philae*—in particular, its CDMS—had to be prepared for deviations from the predictions. We, as the CDMS hardware and software developer group, anticipated that the flight operators might have sporadic, time-restricted opportunities for intervention. The primary requirements were flexibility and onboard autonomy for serial and parallel sequencing, and control and harmonization of the operation of the five subsystems and nine scientific instruments.

The energy available for the science programs from the primary and secondary batteries was limited. The CDMS also had to cope with extreme environmental and operational conditions throughout the long-term-science phase. That phase, driven solely by solar power and the rechargeable battery, required comprehensive thermal control and power flow management.

**When Good Intentions Face Reality**

As the International Academy of Astronautics noted, the *Philae* mission was “the first-ever trajectory development for a ballistic comet landing, the first on-comet operations, and the first cometary in-situ science collection.”4 However, not everything went as planned. Here, I explore some lessons learned from our and the *Philae* team’s experiences with problems that occurred in the hardware and software and in mission operations control.

**Lesson 1: No One Can Conquer Fate**

Originally, this statement from Douglas Adams inspired us to “conquer fate”:

\[
\text{The major difference between a thing that might go wrong and a thing that cannot possibly go wrong is that when a thing that cannot possibly go wrong goes wrong, it usually turns out to be impossible to get at and repair.}^5
\]

Fault tolerance and the graceful degradation of vital subsystems in *Philae*, and particularly in the CDMS, were primary design goals. To help achieve these goals, we implemented a multilevel hardware and software scheme in the CDMS. The first level involved constructing a “thing” (system) that “cannot possibly go wrong.” Such a system—for example, the CDMS, has these elements:

- redundant hardware and the proper architecture of redundant elements;
- self-repairing capability, supported by the hardware architecture and specific software;
- robust fallback software with limited functionality (that is, communication capability); and
- reprogrammability of the acting software equipped with full functionality.

Such a scheme is in principle fault tolerant. However, in practice, it isn’t. What if, as an extreme example, a buggy software version is accidentally uplinked to such a system and falls into a deadlock without the system being able to communicate and thus receive the corrected software? Such a system, thought to be fault tolerant, would then fail. To enable intervention (“get at and repair”) from the Earth, we extended the CDMS with triple-redundant emergency telecommand decoders, along with other basic functions to get out of deadlocks.

The irony of fate was that exactly at the final step of design, when we thought we had achieved perfect fault tolerance, we introduced...
a source of error that remained unnoticed for a long time. A cyclic redundancy check code was supposed to protect the received telecommands from misinterpretation. Nevertheless, the telecommand decoders sometimes (but not often) misinterpreted bit-serially transmitted cross-coupled telemetry packets (which were feedback noise caused by improper harnessing) as true emergency telecommands. The software workaround turned out to be surprisingly simple (but energy consuming): both receiver units were kept powered simultaneously, thus eliminating the receiver buffers’ vulnerability to cross-coupled noise.

Lesson 2: Being Prepared for the “Unbelievable”
During the cruise phase, we faced an astonishing incident. In a very narrow temperature range around the CDMS’s thermal equilibrium (~27 degrees C), one of the dual-redundant processor units remained unpowered after being turned on.

As I mentioned before, the CDMS was required to be fully operable under extreme environmental conditions. Careful design and screening turned this requirement into practice. But in spite of the CDMS passing all the environmental qualification tests, the problem remained hidden and only came to light accidentally during flight. Fortunately, the other processor unit automatically took over the missing one’s functionality, demonstrating redundancy’s vital role in critical systems. Moreover, the unpowered processor unit could be brought to life—”thanks to Douglas Adams” (see Lesson 1)—by a hardware-decoded emergency telecommand. Even more fortunately, the CDMS always worked outside the critical temperature range during comet operations.

Lesson 3: When Even Redundancy Is Useless
Because the success of anchoring was a mission-critical requirement, the anchoring-subsystem developer team and we had devoted much effort to its design, quality assurance, and implementation. In spite of all this preparation, a string of surprising engineering issues occurred during the cruise phase:

• One of the two touchdown event detectors—an accelerometer—was unusable because its sensitivity range coincided with the vibration of Philae’s flywheel.
• The touchdown sensor status was cleared after each evaluation attempt, eliminating the chances for repeated evaluation to exclude any transient errors.
• Doubts arose as to whether the sequence and timing of actions in the anchoring algorithm would be efficient enough to shoot the lander’s harpoons.

These issues gave the two teams strong incentives to revise the anchoring strategy and control. We didn’t have access to Philae’s flight hardware; all we had was the changeable software of the CDMS. With the telescope sensors in Philae’s dumping mechanism as the basis, an alternative touchdown detection algorithm was designed as a replacement for the unusable detector. By implementing an “Or-Majority” (Touchdown = A or MajorityOf(B, C, D)) voting scheme for four (A, B, C, D) touchdown event sources or paths, we made the detection of the touchdown event more robust against transmission errors and false alerts. We together with the anchor team also completely reworked the anchoring-control software algorithm, tested it on the ground, and uplinked it to the CDMS.
Even so, and even though Philae properly detected the touchdown event, the anchoring failed, which severely affected the rest of the mission.

Lesson 4: Conflicts Can Exist between Safety and Science
For a long time, the Philae team felt that the delivery strategy should meet the requirements of both maximal scientific throughput and battery redundancy (parallel use of the primary and secondary batteries). The latter requirement would have necessitated a brief descent.

To not violate Rosetta’s safety margins, the spacecraft ejected Philae relatively far from the comet, which resulted in a prolonged descent. The candidate scientific experiments for the descent required more energy than the secondary battery could provide. So, science won out over safety in terms of a redundant battery supply.

For landing-site-targeting reasons, regardless of whether Philae’s main or backup ejection mechanism was deployed, the team decided to set the changeable push-off velocity of the main ejection mechanism to be equal to that of the nonchangeable backup mechanism (0.19 m/s). The dominant factors in Philae’s delivery were Rosetta’s attitude and orbital velocity vector. Compared to these, Philae’s push-off velocity was nearly negligible. This suggests that both ejection mechanisms could have had a simpler but still robust design that produced an even lower nonchangeable push-off velocity.

Lesson 5: Being Unprepared for the Conceivable
The unexpected situation of an unanchored lander at an unknown location and attitude necessitated reshuffling—and to a large extent discarding—the originally planned timelines of the prestored science sequences. By assessing risks and re-prioritizing scientific objectives, the Philae team was obliged to make a series of ad hoc decisions under time pressure before each radio link session, in a continuous day–night work regime of three to four days.

Before the on-comet phases of Philae, the team extensively analyzed potential failure sources and prepared lengthy documents with recovery procedures. After the mission’s active phases, several papers were published on the impact of the failed anchoring, modeling Philae as a mechanical system. You might ask, why didn’t the team do that beforehand? During the cruise phase, so many problems had emerged in conjunction with Philae’s anchoring (see Lesson 3) that it was doubtful whether it would succeed. More important, the possibility that Philae would come to rest tilted to one side, instead of being fixed firmly on its feet, appeared nonnegligible.

And yet no one appreciated that such a scenario was realistic but didn’t necessarily have immediate fatal consequences. The entire Philae team suffered from a sort of groupthink and didn’t take measures in advance to prepare both the operational ground segment and (in particular) Philae’s onboard system.

On the whole, the team could have exploited the available battery energy more efficiently by shortening the standby periods by relying more on onboard autonomy. That would have allowed additional science experiments, even involving risk-taking actions if necessary. For example, we could have adjusted Philae’s attitude for better solar illumination and to take images from additional surface elements of the comet.

Lesson 6: The Revenge of Missed Opportunities
Throughout the first-comet-science phase, the entire Philae team was fully occupied with mastering Philae’s unexpected situation. So, it missed the opportunity to uplink science sequences and telecommands for subsystems control and autonomous scientific research tailored for the long-term-science phase. The first telemetry data after six months of hibernation reported that Philae was in good health, and no one reckoned on a stepwise degradation of its vital hardware subsystems, particularly the redundant radio communication units.

In addition, as the risk grew of completely losing Philae, the team arguably set the wrong priorities and adopted in part inadequate methods to achieve reliable contact with it. For example, because Philae was close to the sun at that time, the thermal and solar-power conditions were excellent, and the control software for quick battery recharging and even for an extended day–night work regime was obviously operable. Yet, the team had concerns that any premature battery discharging might lead to an irreparable deficit.

After not uplinking in advance the set of telecommands, the team showed again that it hadn’t realized the urgent need to establish favorable conditions—including flown orbits of Rosetta—for commanding Philae to collect and downlink scientific data, instead of focusing on investigating how and why the telecommunication units were degrading. It was of the utmost importance to get a second set of science data for drawing conclusions on comet evolution.

The awareness that the lander did nothing useful in standby mode for
at least 2.5 months from wake-up until its last contact with Earth was distressing. In the end, sadly, all the telecommunication units must have broken down before the team could deploy successful countermeasures.

The Philae mission was a jump into the unknown. Besides the standard systematic procedures and workflow, innovative, heuristic ideas had a specific role during software development. In the beginning, the requirements to achieve all the scientific objectives and the technical constraints in such a complex system weren’t fully clear. This led to our inability to complete the flight software in the less than three years of hardware and software design and implementation before the spacecraft launch. Afterward, during the cruise phase, we had to prepare the system for many nominal operational scenarios and emergency situations. All in all, we devoted more than 50 percent of development time to making the system—not only the CDMS but also Philae—as fault tolerant as possible.

Some of the lessons I described illustrate how design and implementation errors remained unrevealed before the mission launch, despite comprehensive on-ground testing and validation. This shows why software reprogrammability is so important.

The Philae team exploited the onboard autonomy and flexibility in many respects, but not to the extent that would have been—in retrospect—purposeful. In the end, we might have been able to compensate for the relatively slow hardware degradation, at least to partly save the long-term-science phase. In hindsight, this proved a bridge too far for all of us.

References

ABOUT THE AUTHOR
ANDRÁS BALÁZS is an embedded-hardware-and-software system engineer at the Wigner Research Centre for Physics. Contact him at balazs.andras@wigner.mta.hu.