

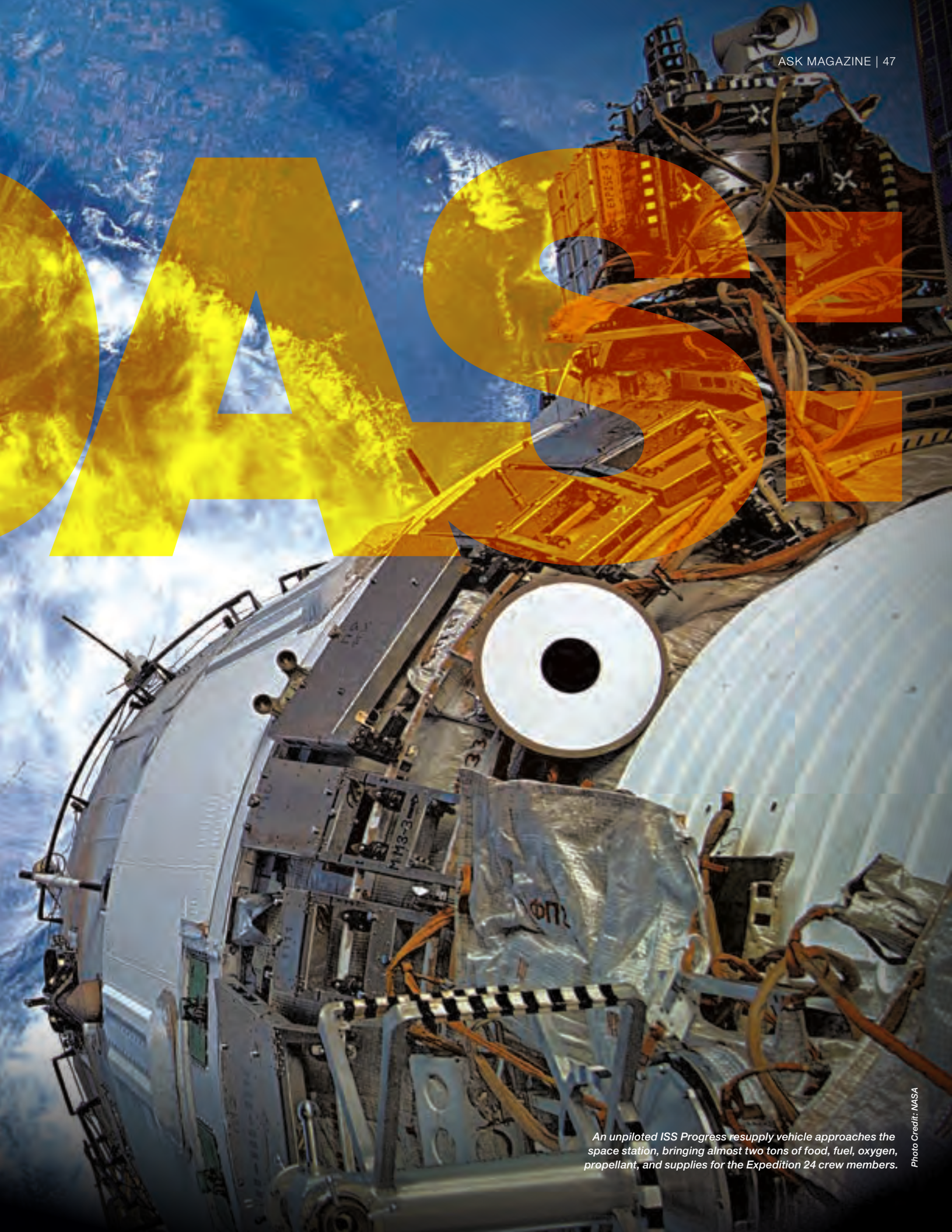
MIND

Keys to Software Success

BY JEFF CLINE

A 1995 Standish Group survey of 365 respondents spanning 8,380 software applications showed that only 16 percent of software development projects finished on time and on budget; 31 percent were canceled; and the remaining 53 percent overran costs by an average of 189 percent. Similar surveys predict that information technology projects are more likely to fail than succeed.

ASK



An uncrewed ISS Progress resupply vehicle approaches the space station, bringing almost two tons of food, fuel, oxygen, propellant, and supplies for the Expedition 24 crew members.

Photographed by an STS-131 crew member on Space Shuttle Discovery, the International Space Station is featured with Earth's horizon and the blackness of space as a backdrop.



For the past decade, a team led and managed by Barrios Technology, Ltd., at the Johnson Space Center has avoided the pitfalls associated with software development. In addition to launching the Mission Integration Database Applications System (MIDAS), a successful, large software application that supports the International Space Station (ISS) program, the team has a productive customer relationship, properly utilizes personnel, and empowers its employees, resulting in a highly functional software team.

What Is MIDAS?

MIDAS supports approximately twenty organizations that use the system to develop and manage a wide array of ISS products, including flight manifests, imagery plans, hazards and toxicity analyses, cargo packing plans, cargo certification, and consumables planning. The application consists of approximately 24 subsystems, 160 user-interface modules, and more than 330 database tables. It integrates the subsystems to allow each customer organization to use active, current data from other customers to help develop its products and make data about its own products available to others. This high level of data integration allows organizations to develop timely, high-quality products, increases cooperation among ISS organizations and partners, and provides a cost avoidance of approximately \$3 million annually for the ISS program.

The NASA application owner and driving force for MIDAS, Tim Brown, has said, “Not only do we have a system in place that benefits almost every corner of the ISS program, but that software has also acted as a ‘glue’ for the various areas within ISS. It is my firm belief that MIDAS has been a major contribution to the increased coordination and cooperation ISS now has among the various individual areas.”

Task Origin and Initial Release

In August 1999, Tim approached our company with several pages of high-level requirements for a flight-manifesting tool and asked us to consider bidding for the development of the application with a target release in fall of 2000. After reviewing the requirements with NASA and internally, we decided that

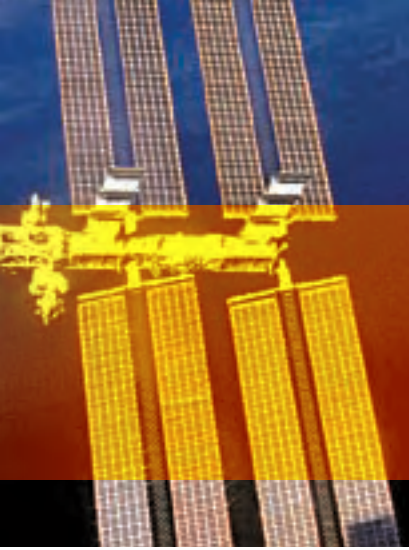
the task came with a high probability of failure, given immature requirements coupled with the task complexity and aggressive schedule. Recognizing the risk, but also the potential reward, we agreed to take on the work only if we could create a process that would give us the best opportunity for success.

We explained to NASA that we would prefer to estimate the cost of developing detailed requirements before submitting a build bid. We proposed that a select team of senior developers from another Barrios project meet a few hours a day for several months with NASA to develop a more detailed requirements document. We could then deliver a requirements document and a realistic build bid based on more mature data. To support this requirements-definition effort, we asked NASA to provide a dedicated MIDAS application owner who would have authority to make decisions and provide guidance.

NASA accepted our plan in November and identified Tim as our dedicated MIDAS application owner. In December we provided a schedule for the requirements-definition phase, which identified project tasks, external dependencies such as customer reviews and feedback, and milestones that would be necessary to produce a requirements document that would later inform our build bid.

In January 2000 our team began meeting with NASA to identify software requirements, evaluate target technologies, and demonstrate prototype designs. In mid-April we provided a draft requirements document for review. When the review comments came in later than the schedule allowed, we explained the importance of commitments being made and kept by both parties. The review comments were incorporated and the requirements document and build bid were delivered on time, but only after a few tense days as both sides defended their positions. This first speed bump was an unpleasant necessity that ultimately provided a good foundation for mutual trust and a very strong working relationship.

That painful event showed NASA that our schedules were real and that both parties were responsible for the success and on-time delivery of the project. This doesn't mean the schedule rules all else, but commitments and dependencies are often related and need to be coordinated. After this event



WHEN SCHEDULES ARE DEVELOPED FROM THE BOTTOM UP (BY THE EXECUTERS OF THE TASKS) INSTEAD OF FROM THE TOP DOWN (BY MANAGEMENT), THE DEVELOPMENT TEAM TAKES OWNERSHIP OF THE SCHEDULE.

both parties have always provided extremely timely support to the project.

NASA reviewed our build bid and authorized us to proceed. We developed a detailed schedule for design, development, testing, and deployment. The content was determined by collecting estimates from each software developer for the desired capabilities; integrating those inputs; adding time for integration testing, holidays, and vacation plans; and letting work management software predict the end date. When that date did not align with the customer's desired delivery date, we negotiated with NASA to remove content from the release and then updated the task list until the work management software predicted an October 27, 2000, release date.

This approach to schedule development has proven to be extremely valuable in several ways and provides key lessons:

- Software developers are best equipped to understand the effort required to develop and test software.
- When schedules are developed from the bottom up (by the executors of the tasks) instead of from the top down (by management), the development team takes ownership of the schedule. Because the team is committed to the schedule, members are invested in the project's success and willing to work extra hours if necessary. Conversely, when an unrealistic schedule is dictated from above, schedule risks can be viewed as "not my problem" by development staff, fostering resentment and adversely affecting team unity and performance.
- Resource loading the tasks in work management software and including vacations and holidays allows the program to provide an objective, realistic schedule prediction for the software delivery date.
- Investing the effort to develop a schedule this way creates a structured plan by which to communicate project progress and potential risk to both internal and external customers.
- Successfully executing a software release in accordance with an approved schedule creates additional trust between NASA and the contractor, demonstrating that our approved schedule is effectively a commitment, not a

plan. Repeated successful execution of these schedules over time increases customer confidence in the contractor.

- Following this approach, MIDAS enjoys a 100-percent on-time delivery rate for approximately 33 major software releases and 112 maintenance releases while running under budget.

In May of 2000 we began design and development of the manifesting tool, conducting numerous reviews to demonstrate progress and identify course corrections in our approach. We included key users in integration testing. This not only confirmed the tool was performing up to their expectations but also trained them in the new system. After substantial internal and external test support, MIDAS was delivered on November 3, 2000—one week later than the work management—software plan. Although MIDAS was ready for delivery on the original date of October 27, an unexpected flight freeze restricted software changes. The first release of MIDAS is considered an on-time delivery because the delivery date was altered by an external, unexpected event.

Extending MIDAS

We began to look up- and downstream of the manifest process itself to automate preceding and succeeding steps. For example, all manifest changes must be approved through a request process. By automating this step and previous steps, as well as those steps that follow flight manifesting (cargo packing, hazards analysis, cargo/transfer priorities, etc.), we have developed a fully integrated system of twenty-four subsystems that provides comprehensive traceability for hardware.

Organizations are often apprehensive of change, particularly when they comfortably work with internally developed tools such as a spreadsheet or database, but local tools isolate the data from other customers. By explaining the benefits of integrated data and committing to develop any MIDAS software upgrades without cost to candidate organizations, we were able to attract many organizations to our requirements table. We promised to provide them software funded by a specific NASA budget in exchange for their data and support of MIDAS. Integrating

A WRITTEN REQUIREMENT CAN BE INTERPRETED IN MANY WAYS, SO THE KEY IS WHETHER OR NOT THE SOFTWARE DOES WHAT THE USERS THOUGHT THEY WERE ASKING FOR, NOT WHAT WE, THE DEVELOPMENT STAFF, UNDERSTOOD THE REQUIREMENTS TO BE.

data from these organizations promotes stronger working relationships and contributes to job satisfaction for those involved in the product development.

While this level of growth and success has been wonderful for our users, team, and company, it has also created challenges. The development team in place in May 2000, still intact today, has fewer than five full-time people, who are now responsible for twenty-four subsystems and more than 900,000 source lines of code. Each person must have knowledge of five subsystems, on average, in order to ensure the system can be effectively sustained, and yet requirements for new features are identified every month and added to the queue for MIDAS releases. A typical software developer can maintain only about 50,000 source lines of code,¹ which suggests we should have eighteen software developers on staff.

Our small team is able to support so much complex software due to the successful development and implementation of many key lessons.

General Lessons

We've implemented several key elements into our structured processes that have proven to be very helpful in ensuring high-quality products, maintaining developer interest, and protecting our customer from single-point failures.

Employee Respect and Growth

The personnel we hire are highly trained adults, and we treat them as such. Our management approach is built upon trust and empowerment, not oversight or checkpoints. Once work is assigned to a developer, that developer is responsible for creating and meeting schedule estimates, performing testing, and managing requirements and user interaction.

If a customer's prioritized requirements cannot be accommodated by our team in the time requested, we negotiate a reduction in content or move the release date so our team can accommodate both content and schedule. This shows our employees that we value their professional *and* personal time. We want them to see this job as an enjoyable, satisfying career, not a twenty-four-hour-a-day obligation. As a result, our team members have never failed to step up when schedule challenges occasionally arise.

When software anomalies are identified, we focus on understanding the root cause of the problem and develop process changes to reduce or eliminate the potential for repeating the error rather than assessing blame. When necessary, we work with employees to improve a skill or revisit a process.

We demonstrate to our NASA customer that our people are the reason for our success and balancing their needs is just as important as the needs of the customer. People tend to experience stress over family, finances, and their job. If I can eliminate the

job stress from their life, we've given our team members more energy to focus on their more critical life concerns.

We also use rotational task assignments to provide new opportunities for staff development. Rotating personnel gives them new skills and a deeper and broader knowledge of our system design. A side benefit is our expanded ability to handle surges in requirements. Rotations also give staff expanded opportunities to learn from their coworkers.

Trusted Partnership with NASA

A strong working relationship with NASA allows our team to excel. We established early on that our word was our bond. By empowering our developers to own schedule estimates, consolidating those estimates into a scheduling tool to produce realistic schedules, working hard to honor those commitments, and admitting when we've made mistakes, we have created a working environment of mutual trust between NASA and Barrios. NASA trusts our schedule estimates are realistic. If we determine a new requirement is too complex for our technology, or not a fair effort-benefit trade, NASA trusts our assessment instead of assuming we are avoiding work.

As a result, NASA empowers us to identify and recommend ways to make the software better, trusts our opinion on which changes make the most sense, and often comes to the development team to discuss ideas before taking them to our user community.

Quality

We work hard to ensure that the software we deliver is of the highest quality. While we've been successful during the ten-year (and counting) delivery history of MIDAS, our philosophy is "our users will remember software was delivered on time and wrong long after they've forgotten it was delivered late but right."

Delivering what users need takes precedence over delivering on time. As users test our software, they often realize they really need something other than what they requested. We work with them to identify the difference between where we are and where we need to be, and develop a plan to respond. This may mean an update to the software prior to delivery, or we may deliver as is and on time if the software is usable but not optimal. In this latter case, we schedule a follow-up release to add features identified during testing.

Delivering "what they asked for" on time just because they agreed to that requirement three or four months ago doesn't mean that requirement is still accurate or appropriate. We don't want to deliver software if it isn't ready. We deliver only after our users have tested the software and agreed that it meets their expectations and appears to be bug free, or meets their expectations except for minor acceptable discrepancies. We ensure high quality through a four-phase integration-testing

approach that includes testing by the developer; testing by two other developers on the MIDAS team, one of whom is not familiar with the software; and testing by our customer-support group. Finally, key users test the software to ensure it meets their *expectations*. A written requirement can be interpreted in many ways, so the key is whether or not the software does what the users thought they were asking for, not what we, the development staff, understood the requirements to be.

A Good Team Is Like a Good Marriage

After ten years, I've found that our MIDAS team operates much like a good marriage. The keys to a successful relationship among our MIDAS team members include identifying each other's strengths and utilizing them, identifying each other's weaknesses and strengthening them, and identifying each other's hot buttons and avoiding them. Treating each person with respect and empowerment while providing a stable, interesting, and nurturing working environment promotes team unity and stability.

By working with such a high-caliber team of software developers and NASA for a decade, we've learned a tremendous amount about successful team development, customer relationships, and the importance of teamwork. The team's dedication allows our developers to sustain almost four times as much code as a typical developer. To replace this team of "almost five" would require more than twice as many new people of a similar skill level in order to barely get by. That is a testament to the power of positive team dynamics. ●

JEFF CLINE is an information technology manager and certified Project Management Professional for Barrios Technology. Supporting the ISS program for more than twenty-two years, he has served in various software-related capacities, including software architect, developer, and trainer, and he has led teams for the past fifteen years.



1. Tom Love, "10 Must Knows for CIOs About Software Development," November 16, 2006, www.itmpi.org/assets/base/images/itmpi/private/rooms/tomlove/MustKnows.pdf.