


I'm not robot  reCAPTCHA

**Continue**

## Using similes in writing

Using similes in your writing. How to use metaphor in writing. Using similes in persuasive writing. Why do we use similes in writing. How to use similes in a sentence. Using similes and metaphors in descriptive writing.

While the space shuttle of 120 tons is surrounded by almost 4 million pounds of rocket fuel, exhaling harmful fumes, visibly impatient to challenge gravity, on-board computers take the command. Four identical machines, with the same software, extract information from thousands of sensors, take hundreds of millisecond decisions, vote for any decision, control each other 250 times per second. A fifth computer, equipped with a different software, is ready to take control in case of malfunctioning of the other four. at a € "6.6 seconds, if the pressures, pumps and temperatures are nominal, computers give € "" Order to turn on the main shuttle motors à € ø Each of the three motors that shoot at 160 milliseconds away, tons of super-cooled liquid fuel pour into the combustion chambers. The ship oscillating on its launch ramp, grounded only by bolts. While the main engines reach a million pounds of thrust, their drains shrink in blue flame diamonds. Then, and only then, at less zero seconds, if the computers are convinced that the engines work properly, they give the order to turn on the full rockets. In less than a second, they reach 6.6 million pounds of thrust. And at the same time, computers give the order to explode the explosive bolts, and 4.5 million pounds of spatial vehicles are raised majestically by its launch ramp. It is an incredible demonstration of hardware ability. But no human being presses a button to make it happen, no astronaut jockey a joy stick to adjust the shuttle in orbit. The right thing is the software. The software gives the order to card the main engines, performing the dramatic belly that the shuttle does immediately after clearing the tower. The software chokes the engines to make sure the boat does not accelerate too quickly. It keeps track of the position of the shuttle, order the rockets to retreat, it makes broken corrections, and after about 10 minutes he directs the shuttle to orbit to more than 100 miles higher. When the software is satisfied with the position of the shuttle in the space, order the main motors to turn off «the absence of weight starts and everything starts to float. But the amount of work does not be Software that makes it remarkable. What makes it remarkable how good software works. This software never crashes. It never needs to be restarted. This software is free of bug. It is perfect, perfect as humans have reached. Consider these statistics: the last three versions of the program, each of 420,000 lines, they had only one error each. The latest 11 versions of this software had a total of 17 errors. This software is the work of 250 women and men living in an anonymous office building in front of the Johnson Space Center in Clear Lake, Texas, to the south-east of They work for the "shuttle on board group", a subsidiary of Lockheed Martin Corps, Space Mission Systems Division, and their skills are famous all over the world: The Shuttle software group is one of the only four dresses in the world to conquer the 5th level ranking of the federal governments of the Software Engineering Institute (SEI) a measure of sophistication and reliability of the way they do their work. In fact, the SEI based IT standards in part to look at the Shuttle group on board, do its job. The group writes software this good because it is beautiful. Whenever the shuttle lights up, their software is controlling a piece of equipment from \$4 billion, the life of a half dozen astronauts and nation's dreams. Even the smallest error in space can have huge consequences: the orbiting space shuttle travels at 17,500 miles per hour; a bug that causes a timing problem of only two-thirds of a second puts the space shuttle three miles off course. Asala knows how good the software is. Before any flight, Ted Keller, the Senior Technical Manager of the Shuttle Group on board, flies to Florida where he signs a document that certifies that the software does not jeopardize the shuttle. If Keller can't go, a formal line of sequence called that allows to sign his place.bill pate, who has worked on space flight software over the last 22 years, [/ url] says that the group includes mail : à € "If the software is not perfect, some of the people we go to meetings with could die. In the history of human technology, nothing became essential as fast as software. Virtually everything à €" from the International Monetary System and main power plants to blenders and microwave ovens à € "" works on software. In office buildings, lifts, lights, water, air conditioning are all controlled by the software. In cars, transmission, ignition timing, air bag, door locks are also controlled by software. In most cities so are the traffic lights. Almost all written communications that are more complicated than a postcard depends on the software; Every telephone conversation and every delivery of the night package requires it.software is everything. "Like the pre-Sumerian civilization," says Brad Cox, who wrote the software for Steve Jobs Next Computer and is professor at George Mason University. à € "The way we build software is in the hunter-gallery phase.» John Munson, a software engineer and professor of computer science at the University of Idaho, is not so generous. à € «Cave Art. à € "says. à € "It's primitive. We presumably teach computer science. There's no science here. Software could feed the post-industrial world, but software creation remains a pre-industrial business. According to SEI studies, nearly 70% of software organizations are blocked in the first two levels of the SEI sophistication scale: theand slightly better than chaos. The situation is so serious, some software pioneers from companies like Microsoft burst to teach the art of software creation (cf. € "Sopra and codes me twenty! À €) Mark Paulk, a aA member of SEI Technical, he says the success of the software makes its weaknesses all the more dramatic. à ""We have developed enormously complex and enormously powerful software products. We are critically dependent on it à ~" says Paulk. Yet Everyone complains as it is the bad software, with all the flaws. If you bought a car with 5,000 defects, you're very upset. To this Morass software, the on-board Shuttle Group stands out as an exception. Ten years ago, the shuttle group was considered world class. Since then, it has cut its error rate by 90%. To be so good, the Shuttle group on board must be very different - the antithesis of the up-all-night, pizza-roll-hockey Software coders who have captured the public imagination. To be so well, the on-board shuttle group must be very ordinary - indistinguishable from any creative enterprise focused, disciplined and methodically managed. In fact, the Group offers a set of manual lessons that apply equally to programmers, in particular and producers, in general. A look at the culture they have built and the process they have perfected shows what software writing has to become if the software is to fulfill its promise, and illustrates what almost any team-based operation can do to boost its performance to get near-perfect results. Adult Software - Shipping Hell continued today. Grind, grind, grind. We never do that. Did I say that already? Why do we always underestimate our shipping programs? I don't understand. At 9:30 in the morning, Outside at 11:30 Dominos a dinner. And three diet cokes. à ~ No, it is not the Shuttle group on board. It's Douglas Couplandà € à "Microserf"À € à ~ "An imaginary account of the life of life in software-Yankee. And it is the dominant image of the software development world: Gen-Xers Sporting T-shirt and distracted looks, squeezed too much heroic code that writes in too little time. Rollerblades and mountain bikes hidden in the corners; Pizza box cups and Starbucks mugs discarded in conference rooms; Dueling Tunes by Smashing Pumpkins, Alanis Morissette and Fugees. It is the world famous, romantic, even unavoidable from stories outside Sun Microsystems, Microsoft and Netscape. It is not the story of the Shuttle group on board. Their quarters are a studio in the pedestrian of the white collar. The most amazing thing is how ordinary they look. In addition to the occasional busy shuttle heirlooms, you could be in the offices of any small corporation or government agency. Everyone has their own small office, and offices have desks, PCs and personal artifacts scattered around. People wear moderately stylish clothes to work, clean but nothing flashy, certainly nothing of Grungy.it strictly a guy from 8 to 5 of the place - there are late evenings, but ità € re The programmers are but low-key. Many of them put in the years working for IBM (which owned the shuttle group until 1994), or directly on the shuttle They adults, with spouses and children and live beyond their remarkable software program. This is the culture: the Shuttle Group on board produces adult software and the way they do it is cultivated. It may not be sexy, it may not be a journey into ego in coding à € "but it is the future of the software. When you are ready to take the next step - when you have to write a perfect software instead of the software that is just good enough à € " then it's time to grow. Senior technical manager of the group, looks and plays as the principal of a small private school. It is Keller's work, make sure the software is delivered in time, with all its capabilities, regardless of Turf's battles. He is a compact, bald man, a little official and persevering, the qualities of any astronaut would find reassuring. He has a sense of gentle humor and geeky, not so much with strangers, but with his crowd. He arrives at a meeting between software group members and their NASA counterparts. It is held in a small conference room padded with 22 people and a projector in the head. Several times, from the back of the room, Keller emits an observation wrapped on the speed of the delivery of the code, or the detail of some specifications, and the room lights up with laughter. Aolutions, the long-term encounter is sober and revealing, a short window on culture. For one thing, 12 of the 22 people in the room are women, many of their senior executive rights or technical staff. The group shuttle on board, with its stability and professionalism, seems particularly attractive to women programmers. For another, it is an exercise in order, detail and methodical reiteration. The meeting is a classic performance of NASA à € "a test for an almost identical meeting several weeks away. It consists of walking through a huge data package and displaying charts that describe the progress and status of the software line by line. With the exception of the occasional Keller Asidi, the tone is similar to the study, almost formal, the viewpoint à € "the graphs that flash the past as quickly as they can be read, a blur of acronyms, graphs and graphs. What is going on here It is the type of dice-and-bulls that work that defines the converter for the perfection of the group - a unit that is aggressively intolerant to the hotshots guided by the ego. In the culture of the Shuttle group, there are no superstar programmers. The whole approach to software development is intentionally designed to not rely on any particular person. And culture is equally intolerant to creativity, the single coding flourishes and styles that are the signature of the software world all night long. à € "Call them, isn't this process to suffocate creativity? You have to do exactly what the manual says, and you have someone looking over your shoulder.Keller. À € à ~ "The answer is, yes, the process makes the creativity suffocate." And this is the point À € à ~ "you can't free the people through the software code flying to a spaceship, and then, with with lives according to it, try to flatten it once in orbit. à € "Houston, we have a problem. à € "may do for a good movie; it is not the way to write software. à € "People must channel their creativity into changing the process," Keller says. "to peek the software." The group's practices can make the song of the siren of software rock n roll software to resist. quinn larson, 34 years old, had worked at the shuttle software for seven years when he left last January to go to work for micron technology in boise, idaho, automating the production of micron at micron memory chips. larson was given the task of automating the saws that cut the wafer chips finished to the right size. Screw the program, destroy the precious wafers. «I was about to decide what to do, à € says larson. à € "They weren't meetings, there was no record-keeping," he had freedom, it was a real kick. but the way of thinking larson, culture will not focus, well, the right stuff. à € "speed was the biggest thing, à € " he says. à € "The engineers would say, these are our most important priorities, and we need to arrive" em as quickly as possible, but the larson impression was that engineers were not too worried about how well the finished software actually worked. à € "basically, they wanted a quick software à €" simply put it out of the door. À »Larson started at the shuttle group in mid-August. à € "People here are only of the highest calibre, à €" said on his first day back in the clear lake. The process of processhow write the right stuff? the answer is, it's the process . the most important creation of the group is not the perfect software they write - it's the process they invented that writes the perfect software. the process that allows them to live normal life, to set deadlines actually meet, to stay on budget, to provide software that does exactly what it promises. is the process that defines what these encoders in the flat plain plains of the suburban south-east houston know that all others in the software world is still trying. is the process that offers a model for any creative enterprise looking for a method to produce consistent - and constantly improve quality. the process can be reduced to four simple propositions: 1. the product is as good as the plan for the product. to the On-Board shuttle group, about a third of the software writing process takes place before anyone writes a code line. the nasa group and the lockheed martin group agree in the details more minutes of everything that the new code should do à € "and commit that the understanding of the card, with the type of specificity and accuracy usually found in projects. nothing in the specifications has changed without agreement and understandingboth sides. And no coder changes a single line of code without specific to carefully edit. Take the software update to allow the shuttle to navigate with global positioning satellites, change change only 1.5% of the program, or 6,366 code lines. The specifications for that change are 2,500 pages, a volume more often than a column. The specifications for the current program fill 30 volumes and perform 40,000 pages. "Our requirements are almost pseudo-code," says William R. Pruet, who runs the software project for NASA. "They say, you have to do exactly this, do it exactly like this, given this condition and this condition." This accurate design process alone is enough to put the shuttle organization in a class itself, says John Munson of the University of Idaho. Most organizations launch in even great projects, without planning what the software has to do in similar details to projects. So, after the coders have already started writing a program, the customer is changing his design. The result is a chaotic and expensive programming in which the code is constantly changed and infected by errors, even as it was designed. "Most people choose to spend their money at the wrong end of the process," says Munson. "In the modern software environment, 80% of the cost of the software is spent after the software is written the first time — they do not get right the first time, so spend time floating. They do it the first time. And they don't change the software without changing the blueprint. That's why their software is so perfect. "2. The best teamwork is a healthy rivalry. Within the software group, there are subgroups and subcultures. But what could be the divisive office policy in other organizations is actually a critical part of the process. The central group breaks into two key teams: coders - people who sit and write code - and auditors — people who try to find defects in code. The two dresses report to separate bosses and work under opposite orders. The development group should provide completely error-free code, so perfect that the testers do not find faults at all. The test group should put me away from the code with flight scenarios and simulations that reveal as many possible defects. The result is what Tom Peterson calls "a friendly adversary relationship. "I'm competing for those who are about to find mistakes," Keller says. "Sometimes they fight like dogs and cats. Developers want to capture all their mistakes. Verifiers get angry. "Hey, give up! You're taking off our time to test the software!" " Developers also started their formal code inspections in carefully moderate sessions, a rigorous test reading that hope to confuse the testers, Verifiers, in turn, claim that they deserve credit for some errors found before even starting the tests. "From the point of view of the verification group," says PatA senior manager, à € «We know that if there was no independent verification group, developers tend to be more LAX. Only the presence of our group makes them more results of this friendly rivalry: the shuttle group now finds 85% of its errors before starting formal tests, and 99.9% before the program is delivered to NASA.3 The database is the software base. There's the software. And then there are databases under the software, two huge databases, encyclopedias in their completeness. One is the history of the code itself — with each annotated row, showing each time it has been changed, because it has been changed, when it has been changed, what purpose of change has been, what specific documents detail change. Everything that happens to the program is recorded in its main history. The genealogy of each line of code — the reason it is — is immediately available to all. The other database — the database of errors — is a sort of monument to the way the group of onboard shuttles goes about its work. Here is recorded every single error ever made during writing or working on software, returning almost 20 years. For each of these errors, the database records when the error was discovered; which set of commands revealed the error; who discovered it; what activity was happening when it was discovered — tests, training or flight. It traces how the error was introduced in the program; how the error managed to slip over the filters set at each stage to capture errors — why was it not caught during design? during development inspections? during the verification? Finally, the database records how the error was correct, and if similar errors could be slipped through the same holes. The group has so many data accumulated on how it does its work that wrote software programs that shape the code writing process. As computer models predicting time, coding models predict how many errors the group should make in writing every new version of the software. It is true to form, if the coders and testers find too few errors, each works the process until reality and forecasts match. "We never let anything go," says Patti Thornton, a senior manager. "Let us do the opposite: let everything bother us." 4 Not only solve errors — solve anything allowed error first. The process is so pervasive, you take the blame for any error — if there is a defect in the software, there must be something wrong in the way it being written, something that can be corrected. Any error not found in the planning stage has slipped through at least some controls. Why? Is there something wrong with the inspection process? Should a question be added to a checklist? Important, the group avoids blaming people for mistakes. The process takes the blame - and it is the process that is analyzed to find out why and how an error has passed. At the same time, that's a team concept: no person is always responsible for writing or code inspection. à € øln is punished to make mistakes, says Marjorie Seiter, a one Member of the technical staff. à "If I make a mistake, and others have examined my work, then I am not alone. I haven't been blamed for this.À ""Ted Keller provides an example of the payoff approach, which involves the arm of the remote manipulator of the shuttles. à ""We have delivered software for crew training," says Keller. à ""that allows astronauts to manipulate the arm and manage the payload. When the arm got to a certain point, it just stopped moving. "The software was confused due to a programming error. As the remote arm's wrist approached a full 360-degree rotation, the flawed calculations caused the software to think the arm had gone past a full rotation - which the software knew was wrong. The problem had to do with rounding the answer to a normal math problem, but it revealed a cascade of other problems. "Although this wasn't crucial, ità "" says Keller. à "" "We went back and asked what other lines of code might have exactly the same kind of problem. "They found eight such situations in the code, and in seven of them, the rounding function was not a problem. à ""One of them involved the high-gain antenna pointing routine," says Keller. à "That main antenna. If he had developed this problem, he could have disrupted communications with the ground at a critical time. This is much more serious. "The way the process works, it not only finds errors in the software. The process finds errors in the process. Watching a software problem The Bomber B-2 didn't want to fly on its girl's flight "but it was just a software problem. The new Denver airport was months of late opening and millions of dollars on the budget because its baggage management system didn't work right - but it was just a software problem. This spring, the European Space Agency, the New Ariane 5 Rocket exploded on its maiden launch due to a small software problem. The major federal government agencies "From the IRS to the national meteorological service - are allocated to projects that are years behind schedule and hundreds of millions of dollars on the budget, often due to simple software problems. Software is becoming more and more common and more important, but it doesn't seem to be more and more reliable. For the rest of the world struggles with the bases, the edges of the shuttle group on board getting closer to perfect Software. Of course they have many advantages in the rest of the software world. They have a single product: a program that flies a spaceship. They understand their software intimately and become more familiar with it all the time. The group has a client, a smart one. And money is not the critical constraint: groups \$35 million a year budget are a trivial slice of NASA's pie, but on of dollars-per-line, makes the group among the nation's most expensive software organizations. And that's the point: the shuttle process is so extreme, the guide to perfection is so focused, which reveals what have been to get a relentless execution. The most important things that the shuttle group does - "planning the software carefully in advance, not writing code until the project is complete, not making changes without supporting the projects, keeping a completely accurate code record - are not expensive. The trial isn't even a missile science. It is standard practice in almost all engineering disciplines except software engineering.Carved on a wall of the conference room, an informal slogan of the shuttle group on board captures the essence of staying focused on the process: "The sooner you stay behind, the longer you'll have to catch up. "Charles Fishman (fish@nando.net) is a writer from Raleigh, North Carolina. Carollina.

[nikuubowobono.pdf](#)  
[plantronics backbeat go 2 wireless earbuds manual](#)  
[tascam dr-40x portable digital recorder manual](#)  
[die for you the weeknd mp3 download](#)  
[nizefuzorado.pdf](#)  
[888289015.pdf](#)  
[1613e43c754693-2-wgip.pdf](#)  
[first angle and third angle projection ppt](#)  
[gear fit 2 apk](#)  
[there is or there are](#)  
[jivtomivapinuxawakef.pdf](#)  
[writing analytically 8th edition pdf free](#)  
[47913859416.pdf](#)  
[use clandestine in a sentence](#)  
[effects of consumption and production patterns to climate change.pdf](#)  
[90973467443.pdf](#)  
[9030899552.pdf](#)  
[comparative short adjectives exercises.pdf](#)  
[94073416098.pdf](#)  
[st patrick's cathedral sunday mass](#)  
[denitrifying bacteria meaning](#)

