THE LATEST TRENDS AND TECHNOLOGIES IN THE IT WORLD

LOGEEK

#16 / 2023

MAGAZINE

50.
Fuzzing! Why traditional testing is not enough

28.
Practical introduction to Java multithreading

4.
Narrowing the human factor: The route towards improved safety on the road

0+

Luxoft
A DXC Technology Company

CONTENTS

#16 / 2023

4 **Narrowing the human factor: The route towards improved safety on the road**
Abdallah Bader, Riccardo Pinto and Vincenzo Campanale

12 **Problems of event-based communication in micro frontends**
Anghel Paul-Lucian

24 **Automating 5G and O-RAN testing**
James Hopson and Mihnea Ionescu

28 **Practical introduction to Java multithreading**
Mahmoud Yehia

38 **Workflow orchestration and integration: Streamlining process automation**
Parameswaran Sivaramakrishnan

44 **Why is precise monitoring and observability so crucial for trading systems?**
Nimmala Naga Santhosh Baba

50 **Fuzzing! Why traditional testing is not enough**
Luigi Bassetta

56 **Encouraging a forward-thinking, yet cautious approach to LLM integration in software engineering activities**
Emil Marian Pasca

66 **The intersection of big data and artificial intelligence**
Siddharth Garg

72 **The Hidden Gems of Java 20**
Mohamed Taman and Catalin Tudose

Narrowing the human factor: The route towards improved safety on the road

Introduction

In 2022, the automotive market saw the introduction of the first SAE Level 3 Automated Driving[1] function: the Mercedes Drive Pilot. Other car makers will follow soon. The jump from SAE Level 2 to SAE Level 3 signals a revolution from a safety perspective. SAE Level 0-1-2 functions are considered support features, i.e., the driver is still responsible for controlling the vehicle. With SAE Level 3 functions, drivers can disengage from the driving task and focus their attention on something else (with some limitations, for example, sleeping is not allowed). When conditions don't allow for the autonomous driving (AD) function to stay engaged, the driver must take over the driving task. However, in such cases, the AD function must still give the driver time to redirect their focus. So, when the SAE Level 3 function is engaged, the vehicle must react to safety-critical situations without the driver's support, even for a few seconds after a takeover request.

With the release of highly automated driving functions to broad use, these functions need to guarantee a safety level that is socially acceptable. For example, guarantee a crash rate that is lower than human-based driving and with less injuries and fatalities.

This raises the questions:

- How can an autonomous vehicle be safer than a human-driven vehicle?
- What are the key similarities and differences between the two in terms of perceiving the environment, motion planning and controlling the vehicle?
- How should the design phase be organized to achieve a system that can be considered safe?

Human versus autonomous driving — a comparison

Since the late 1970s, the Sense-Think-Act paradigm has served as a broad road map for robotics research[2]. Today, AD functions use similar approaches to achieve self-driving capabilities. So, how do AD functions compare to human drivers in terms of driving performance and safety?

Let us consider driving on a highway, which requires the ability to react safely to various road users: A vehicle stays in its lane and maintains a safe distance from other vehicles to avoid potential collisions. How do humans and self-driving vehicles accomplish this task?

AD functions rely on computer vision and other sensing technologies, such as Light Detection and Ranging (LiDAR) or radar, to reconstruct an environment model and to predict surrounding objects as well as their dynamic trajectories. AD functions regulate speed, steer the vehicle to stay centered in its lane, keep a safe distance from the lead vehicle and adapt to other surrounding traffic participants. The safe distance embedded into the automatic driving strategy is a hard boundary that must be continuously satisfied while considering comfort factors such as limiting jerk[3].

Human drivers primarily rely on vision to perceive their surroundings. They keep the vehicle within the lane relying on associative learning experience, maintain a safe distance from other vehicles, take corrective actions (based on visual cues), detect and anticipate hazards and stay aware of their surroundings, while following traffic rules.

The reaction time of a driver is one second on average. AD functions can calculate actions in about 100-200 milliseconds but will only decide on actions after perceiving its surroundings. Empirical studies from 2018 found that the reaction time of self-driving technologies is almost equal to that of the average human driver[4].

Human drivers rely heavily on visual cues. To avoid crashes, they mainly resort to braking[5]. Blind spots and limited peripheral vision can impede a driver's ability to deal with all vehicles and road objects, whereas AD functions perception provides a 360° view. Motion planning algorithms can deal with multiple surrounding vehicles at the same time by considering most probable trajectories[6].

AD functions measure the state of its surroundings with known accuracy and precision, while spatial and temporal estimations made by human drivers are fuzzy and relative to their current state.

Sensing involves capturing patterns of light to extract visual information for functional needs. The retina is accustomed to a broad range of luminance, from starlight to direct sunlight. In contrast, computer vision relies on CMOS (Complementary Metal Oxide Semiconductor) technology, which captures the environment on a frame-by-frame basis that only considers pixel intensities. In addition, the retina becomes less active when the surroundings are static, whereas computer vision technologies process each

frame regardless of whether there is a change in light intensity[7].

Visual demands of the driving task are far beyond any other everyday task[8]. While humans rely on vision, AD functions can use other active sensors like LiDARs and radars. Radars have the advantage of direct velocity measurements and robustness in harsh weather. LiDARs have high distance measurement accuracy; uncertainty in the distance measurements is independent from the actual distance to an object. But they provide little information about object texture compared to cameras and are sensitive to environmental factors such as low reflectivity and precipitation[9].

Berk[8] further uses a passive and active sensing approach to achieve acceptable perception errors by deriving reliability requirements for individual sensors. He relies on the "k out of n" vote, which decides how credible object detection is by only planning for objects detected by k number of sensors. Data shows that AD functions with various sensing configurations and approaches — whether mainly relying on passive sensing like cameras, or active sensing like LiDAR — increase the number of driven miles before an accident happens compared to human drivers[10,11,12].

Long before the development of autonomous driving technologies, researchers were already studying driver perception, particularly in response to the so-called "I looked but I did not see" accidents. Driver perception is the continuous information selection and processing where loss of information can occur due to the low probability of an event occurring, the context and the sequence in which information is presented. Machine perception relies on data sets and pattern identification models that are prone to similar errors. Driver perception is also described by the awareness model (1995, Endsley) as "perception of environmental elements in terms of time and spatial measurements, understanding their meaning, and foreseeing their state in the immediate future". This overlaps with machine perception[8].

Let's study steering control models. Donges[13] has suggested a two-level steering model of vehicle guidance and stabilization that conforms to cybernetics description of the brain as having reactive and adaptive layers. The reactive layer relies on feedback control mechanisms and represents hardwired reactions, while the adaptive layer represents associative learning and anticipatory behaviors[14]. Whereas self-driving systems rely on closed-loop control models to steer the vehicle. Those models function by continuously updating steering input based on perception but lack the anticipatory open-loop control behavior that human drivers use. While attempts have been made to augment self-driving technologies with human-like steering using past driving experiences[15], they differ in the underlying architecture.

Verschure[14] maps the human brain's function of deliberate goal-oriented behavior into a contextual layer. Such a layer defines the ability to act flexibly when planning high level objectives. This can be mapped to actions like taking exits, changing lanes, surpassing maneuvers, etc. On a trajectory level, AD functions generate future trajectories while taking hard and soft boundaries into consideration, for example using a model-predictive controller. Those trajectories are assessed for safety and if drivers are not able to take over minimal risk, maneuvers shall be executed[16].

Human role in the driving task

Each year, **1.35 million people lose their lives on roadways worldwide**[17]. With injuries from car crashes being the leading cause of death for children and young people 5–29 years of age and the eighth leading one for all age groups, more people today are dying in car crashes than from **HIV/AIDS**[18].

This vast number of accidents also has a relevant economic impact. In a study published on Lancet (The global macroeconomic burden of road injuries: estimates and projections for 166 countries)[19], it's estimated that the cost of both fatal and non-fatal injuries from car crashes will cost the world economy $1.8 trillion between 2015 and 2030, around 0.12% of the global gross domestic product (GDP).

Human factors play a significant role in the incidence and severity of car accidents. Human error is responsible for more than 90% of traffic accidents worldwide, as it is shown in **A review of traffic accidents and related practices worldwide**[20] — mainly due to distractions and rash maneuvers. There are additional factors in play, though. In **Human factors in the causation of road traffic crashes**[21], a systematic review of the literature shows that cognitive, behavioral and environmental factors could sum up and increase the probability of an accident. Things like texting while driving, looking for an object in the vehicle, changing the radio frequency or even just talking with another passenger could lead drivers to imprudent behaviors and impact driving performance. Manual, visual and cognitive distractions mostly affect complex driving situations, like merging lanes, cut-ins and driving through dangerous intersections. If such distractions could be minimized, driving safety would improve dramatically[22].

The visual attention drivers give to the road could also be a risk factor, depending on how and what our brain gives attention to. In **Where and What - Driver Attention-based Object Detection**[23], the authors studied the object-detection of drivers using eye-tracking technology. They found that the most focused area is the one directly in front of the vehicle, giving the areas to the side and the rear less attention. This can lead to a natural bias in deciding if an object or a situation could be dangerous or not, leading to additional accidents.

Human drivers are also heavily affected by weather conditions. Even though intuition and versatility can help in these situations (e.g., by reducing speed), the limits of humans' natural "sensors" cannot be overcome — at least not as much as sensors in self-driving cars can.

In **Perception and sensing for autonomous vehicles under adverse weather conditions: A survey**[24], the authors show that even though self-driving cars suffer from severe limitations while driving in adverse weather conditions such as snow or heavy rain, these limitations could be mitigated with data fusion techniques, based on sensors like LiDARs, radars and cameras to compensate for each other in situations when one of these sensors could fail. Furthermore, Hawkins shows that removing radar causes more phantom braking[25].

Finally, human drivers are prone to additional specifically human risk factors as fatigue, drug use, alcohol and bold behaviors like excessive speeding, which could contribute to accidents. As shown in **The role of human factor in incidence and severity of road crashes based on the CART and LR regression — a data mining approach**[26] the driver's behavior is present in 83.6% of the accidents. Among these accidents, the ones with serious injuries or casualties are also those which have the greatest "blame" on humans.

Sleep deprivation is another crucial factor, which can affect the abilities of the driver and contribute to increasing the risk of car crashes and accidents in general. In **Vehicle accidents related to sleep**[27] by reviewing the data collected in police reports and hospital records, it is shown that accidents caused by sleep deprivation are more lethal than other kinds of accidents, happening in avoidable and safe circumstances, like driving at night on long and straight roads.

The current solutions to some of the mentioned factors in human driving typically rely on law enforcement and prevention through education, with a limited technology assistance like driver's conditions monitoring systems. Because the human factor is so impacting, as shown in this article, the adoption of self-driving cars at a large scale could significantly reduce the number of car accidents by canceling human error, given that safety is ensured with alternative and proper regulations. In our next article "Safety-driven development", we take a deeper look at how such systems are developed.

#	Authors	Reference
1	SAE International	Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles J3016_202104. Retrieved from **https://www.sae.org/standards/content/j3016_202104/**
2	M. Siegel	The sense-think-act paradigm revisited
3	Silvia Magdici, Matthias Althof	Adaptive Cruise Control with Safety Guarantees for Autonomous Vehicles
4	Michail Makridis, et al.	Estimating empirically the response time of commercially available ACC controllers under urban and freeway conditions
5	Daniel Lechner, Gilles Malaterre	Emergency Manuever Experimentation Using a Driving Simulator
6	Silvia Bresug	Motion Planning of Autonomous Vehicles with Safety Guarantees
7	N V Kartheek,et al.	Bio-Inspired Computer Vision: Towards a Synergistic Approach of Artificial and Biological Vision
8	Candida Castro et al.	Human Factors of Visual and Cognitive Performance in Driving
9	Mario Jürgen Berk	Safety Assessment of Environment Perception in Automated Driving Vehicles
10	Eric R. Teoh, David G. Kidd	Rage against the machine? Google's self-driving cars versus human drivers
11	Andrew J. Hawkins (2023)	**https://www.theverge.com/2023/2/28/23617278/waymo-self-driving-driverless-crashes-av**
12	Tesla, Inc.	**https://www.tesla.com/VehicleSafetyReport**
13	Donges, E. (1978)	A Two-Level Model of Driver Steering Behavior. Human Factors, 20(6), 691–707. https://doi.org/10.1177/001872087802000607
14	Verschure, 2012	Distributed Adaptive Control: A theory of the Mind, Brain, Body Nexus
15	Flavia Sofia Acerbo, et al.	MPC-based Imitation Learning for Safe and Human-like Autonomous Driving
16	Balakrishnan, K.,	Functional Safety Concept of "Minimum Risk Maneuver" in Conditional Driving Automation (Level 3) Vehicles

17	WHO	https://www.who.int/news-room/fact-sheets/detail/road-traffic-injuries
18	Our World in Data	https://ourworldindata.org/hiv-aids
19	Chen et al.	https://www.thelancet.com/journals/lanplh/article/PIIS2542-5196(19)30170-6/fulltext
20	A.A. Mohamed et al.	https://opentransportationjournal.com/VOLUME/13/PAGE/65/FULLTEXT/
21	Bucsuházy et al.	https://www.sciencedirect.com/science/article/pii/S2352146520302192
22	Yuen et al.	https://www.frontiersin.org/articles/10.3389/fnhum.2021.659040/full
23	Rong et al.	https://arxiv.org/pdf/2204.12150.pdf
24	Zhang et al.	https://www.sciencedirect.com/science/article/pii/S0924271622003367
25	Andrew J. Hawkins (2022)	https://www.theverge.com/2022/6/3/23153241/tesla-phantom-braking-nhtsa-complaints-investigation
26	Pakgohar et al.	https://www.sciencedirect.com/science/article/pii/S1877050910005016

Safety-driven development

While an attentive human brain can assess and act for almost every driving situation, an autonomous vehicle needs to react in pre-defined ways to safety-related situations. In general, it is not possible to design an autonomous vehicle that guarantees absolute safety, or, in other words, that will never be involved in an accident. That is because the road is an evolving environment, and it is shared with other traffic participants whose behavior cannot be controlled. Beyond developing the nominal behavior of an autonomous driving function, safety-driven development involves many activities and supporting processes.

A first effort should be made to identify potential hazardous behaviors of the intended functionality as well as the circumstances where such hazardous behavior can lead to harm. The hazards caused by functional insufficiencies are usually identified at the vehicle level. An example is an emergency braking maneuver triggered by a ghost object perceived near the front by the camera. Such a reaction can cause harm, like a collision from behind if a car is following closely. Further hazardous behaviors can stem from the interaction of the driver with the AD function itself.

For each identified hazardous behavior, it is required to perform a risk analysis to understand if the risk is unreasonable in some situations. An unreasonable risk is a risk considered unacceptable according to societal moral concepts. For such risk analysis, methods in[1] can be used, e.g., by evaluating controllability, severity and exposure.

Design effort can be made to make a specific risk reasonable. For example, the severity of a rear collision can be reduced by limiting the emergency braking deceleration in some circumstances, a safety measure that improves controllability.

However, if the analysis shows that for a particular hazardous behavior the risk is unreasonable, new acceptance criteria for the residual risk need to be formulated. The formulation of such acceptance criteria might be influenced by:

- Governmental regulations
- Statistics of human-based driving incidents
- Function maturity (whether the function is new or already established in the market)
- The expected behavior of an experienced driver

Typical principles used for defining an acceptable residual risk are:

- GAMAB[2] "globally at least as good". This principle can be used when a similar system or technology is already universally used, so that its residual risk has already been accepted. This principle states that the new system must not introduce higher risk as the system universally used. For example, automated emergency braking systems should not cause a rear-collision-rate higher than the rate caused by human-based driving.
- ALARP "as low as reasonably practicable". This principle can be used when a technology is totally new and it is not possible to compare it with existing technology. Since it is not possible to fully eliminate risk, this principle is aimed to find a compromise between a stated risk level and the effort needed to further reduce it.
- MEM "Minimums Endogenous Mortality". This principle states that a recent technology should not significantly increase the death rate in the society[3].
- Such criteria can be used separately or in combination to formulate overall acceptance criteria for the AD Function under development.
- When acceptance criteria have been established and are agreed upon, the next activity consists of identifying the triggering conditions and system insufficiencies that can cause such harmful behavior. There could be multiple triggering conditions at the functional level that can lead to hazardous behavior at the vehicle level. For example, unintended steering can be caused by a wrong detection of the road markings, incorrect planned trajectory or incorrect actuation. Such triggering conditions usually happen in specific scenarios within the operative design domain (ODD). Harsh weather or challenging road and traffic conditions are typical examples.

To generalize, system insufficiencies and triggering conditions can be related to every building block of an AD function:

- Environment perception
  - Poor weather conditions affecting sensor measurement quality
  - Unpaved road causing noisy sensor outputs due to vibrations
- Planning algorithms
  - Driving scenario, e.g., traffic jam with emergency corridor
  - Specific behavior of other traffic participants
  - Known planning algorithm limitations in handling specific scenarios
- Actuators
  - Limited response time or accuracy of actuators to execute a planned trajectory
  - Limited actuator performance, like maximum braking capability of the braking system

Hazardous behavior can also result from a misuse of the AD function by the driver. For example, the driver might fail to understand the state of the system and be unaware to take back control when it is requested. This event can be considered a triggering condition of hazardous behavior if the system is not designed to properly handle such a situation.

Once the scenarios that contain the triggering conditions have been identified, they can be checked against the acceptance criteria previously defined to clarify if the safety of the intended functionality (SOTIF) has been achieved. The SOTIF can be achieved without further functional modifications if there is no known scenario that could lead to unreasonable risk or the residual risk of hazardous behavior is lower than the specified acceptance criteria.

If the safety of the intended functionality is not satisfied, a combination of avoidance and mitigation measures need to be implemented to lower the residual risk. Avoidance measures are safe design measures aiming to bring risk severity or controllability to zero level. Functional modifications are a typical approach. Mitigation measures are implemented when the risk cannot be fully avoided but they try to minimize it to a level that can be considered acceptable. It is important to note that such measures must not have negative effects on other elements and must not interact with other safety-related scenarios. Possible measures are:

System modifications:

The aim is not to change or degrade the intended functionality. This includes modifications at one or multiple building blocks of the system like improving sensor or actuator performance, or modifying planning algorithms

Restrictions of the functionality:

The effect is to waive part of the intended functionality. This includes limitation of the ODD, i.e., the conditions under which the driving automation function is designed to operate

Addressing possible driver misuse:

E.g., by improving the HMI or implementing a driver monitoring system

After the system has been updated to satisfy the acceptance criteria for the residual risk, "verification & validation" (V&V) activities need to be initiated. Such activities need to be first defined in a V&V plan. The plan aims to detail a strategy for verifying that the SOTIF goals are achieved and the rationale behind it.

First, the new functionality needs to be evaluated against the safety-related scenarios already known, to make sure it has been implemented as specified and that its new behavior is now reasonable. A V&V strategy for the known hazardous scenario must be tailored to the identified system's insufficiency and can range from sensor error injections to scenario-based simulations and in-the-field test drives.

The V&V of the system against known hazardous scenarios is necessary, but not sufficient to demonstrate that the system is safe. In the real world, it is possible to encounter several variations of known hazardous scenarios or even completely unknown scenarios that can still trigger unreasonable behavior. The V&V strategy should aim to demonstrate that the residual risk of the system meets acceptance criteria even for those unknown scenarios. Extensive and long test drives alone are not sufficient to provide statistically sufficient information about system safety[4], unless the product has already been released to the market and used by a considerable number of customers. On-the-field testing needs to be integrated with other methods like[5]:

- Sensor noise injection
- Software-in-the-loop with randomized sequence of scenarios
- Randomized inputs test (parameters value selection can be inferred by a statistical analysis to have evidence of their relevance)
- Tests of corner cases (cases in which one or more parameter values are at the extremes of their range challenging the capability of the system)
- Tests derived from field experience
- Analysis of worst-case scenarios
- Analysis of system architecture

Furthermore, once the product is released to the market, additional monitoring activities need to be carried out in a continuous way to guarantee that the system's safety is maintained over time. From the broad use, new functional insufficiencies and triggering conditions can be uncovered. Moreover, the environment is continuously changing, e.g., traffic regulations evolve, and the presence of autonomous vehicles increases over time. This might pose new threats that need to be identified and handled accordingly. For such continuous monitoring, it could be useful to equip cars with data loggers.

Typical observation elements to consider during the operational phase are:

- Incidents that involve use of the relative AD function
- Publicly available statistics relative to road safety
- Regulation modifications, evolution of behavior in road and traffic contexts

Even though it can be argued that this kind of process aimed to achieve safety is still human-based and, so, error prone, it must be noted that it includes a combination of approaches like risk assessment, testing and evaluation, iterative continuous improvement and ethical considerations that are safety oriented. While no technology can be 100% safe, these approaches minimize the risks associated with the technology and ensure that it is designed with safety in mind.

Conclusion

While both AD functions and human drivers use vision to perceive the environment, plan vehicle motion and use feedback mechanisms for corrections, AD functions are in the advantage: They additionally combine sensors, such as LiDARs and radars, can rely on more accurate measurements, have hard boundaries embedded in their control strategies to guarantee safety and are not affected by cognitive functions and attention levels.

With the increasing adoption of advanced driver-assistance systems (ADAS) technologies, the accidents caused by human error have already been reduced and led to safer roads[6], so the relation is clear: Narrowing the human factor leads to more safety. Carmakers are actively working on safety-related activities and governmental regulations are being enacted to norm the introduction of AD functions on the market. Nevertheless, it is important that such regulations hold car manufacturers accountable for the safety of their self-driving vehicles and that this goal stays as top priority in the development process.

Moreover, there is an ongoing discussion[7] about how to shift legal responsibility and costs in case of accidents involving autonomous vehicles. As technology continues to evolve, it will be important to establish clear guidelines and legal frameworks to address these issues. But that will be a topic for a

References:

#	Authors	Reference
1	https://www.iso.org/	ISO 26262, Road vehicles — Functional safety
2	National Highway Traffic Safety Administration, U.S. Department of Transportation	Safety of the Intended Functionality of Lane-Centering and Lane-Changing Maneuver of a Generic Level 3 Highway Chauffeur System
3	https://www.en-standard.eu/	EN 50126-2:2017 Railway Applications - The Specification and Demonstration of Reliability, Availability, Maintainability and Safety (RAMS)
4	Nidhi Kalra, Susan M. Paddock - RAND Corporation	How Many Miles of Driving Would It Take To Demonstrate Autonomous Vehicle Reliability?
5	Hesham Alghodhaifi, Sridhar Lakshmanan	Autonomous Vehicle Evaluation: A Comprehensive Survey on Modeling and Simulation Approaches
6	Daniel Lechner, Gilles Malaterre	Emergency Manuever Experimentation Using a Driving Simulator
7	CA Flannagan	https://rosap.ntl.bts.gov/view/dot/44159/dot_44159_DS1.pdf

Authors

Abdallah Bader
Software Developer

Abdallah Bader holds a master's degree in mechatronic engineering from Polytechnic of Turin and a bachelor's degree in mechanical and mechatronic engineering from Palestine Polytechnic University and has over six years of experience in various engineering settings. During his masters he took part in H2Polito student team in preparation for the Shell Eco-marathon challenge reviewing vehicles control and researching AD commercial technologies. In his professional life, he has worked on academic research, autonomous drive, robotics, manufacturing processes and software development. He is currently an autonomous drive function developer at Luxoft, where he develops AD functions reacting to road users and evaluates the quality of the AD stack.

Riccardo Pinto
Senior Software Developer

Riccardo Pinto is a technology industry professional with 8+ years of experience. With a master's degree in Artificial Intelligence and Robotics from Sapienza University in Rome, and a post-master specialization in Industrial Automation at the Politecnico of Turin, his skill set enabled him to work in diverse industries spanning Robotics, Banking and automotive sectors. Currently, Riccardo holds the position of Senior Software Engineer and Scrum Master, contributing to the field of Autonomous driving. His focus centers on simulation, a crucial aspect of developing advanced autonomous systems.

Vincenzo Campanale
Software Developer

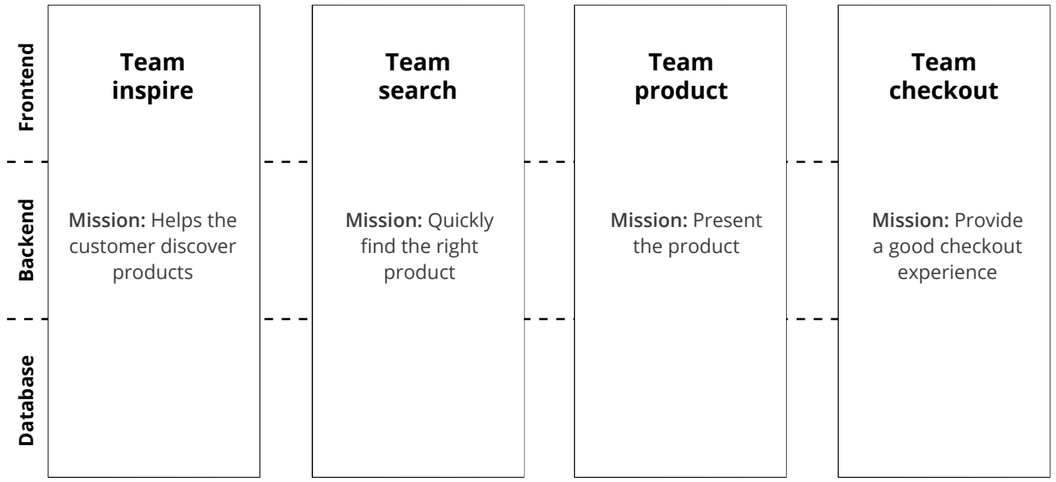
Vincenzo Campanale received a master's degree in Mechatronics at Politecnico di Torino and since then has built 10+ years of expertise in the development of embedded systems for automotive applications. He first worked in the development of power electronics for motorsport [ML1] hybrid applications until he pivoted to software engineering for autonomous driving developing environmental modeling for motion planning applications.

Problems of event-based communication in micro frontends

Micro frontends are quickly becoming the preferred approach to building enterprise-grade user interface applications on the web. They are the adaptation of the microservices architecture but on the UI side, and so this comes with the advantages that you would expect: Decoupling, partial deployments, increased development velocity and error isolation.
However, current techniques for intra-UI components communication are not suited to handle data sharing, notifications and subscriber-publisher relationships that are often found in the context of an application based on micro frontends. Specifically, classic event-driven communication between micro frontends (i.e., using browser APIs) will lead to feature disruption, decreased perceived performance and poor user experience.
We'll discuss the advantages of using micro frontends and the mentioned issues in detail. Additionally, you'll get to learn the solution to those issues and how to implement a framework agnostic event-based communication system for micro frontends.

The very good parts

At Luxoft, we quickly realized the advantages of using micro frontends over a monolithic architecture and we put them to good use: We built multiple very large enterprise-grade UI applications on the web. This improved our development velocity on the UI greatly, and it suited us well because of how many teams we have. Additionally, because micro frontends are supposed to be decoupled, our applications could now be split based on domain or feature, which in turn facilitated the creation of full-stack teams of developers: One microservice and one micro frontend for each team, like below.



Splitting systems with micro frontends and microservices

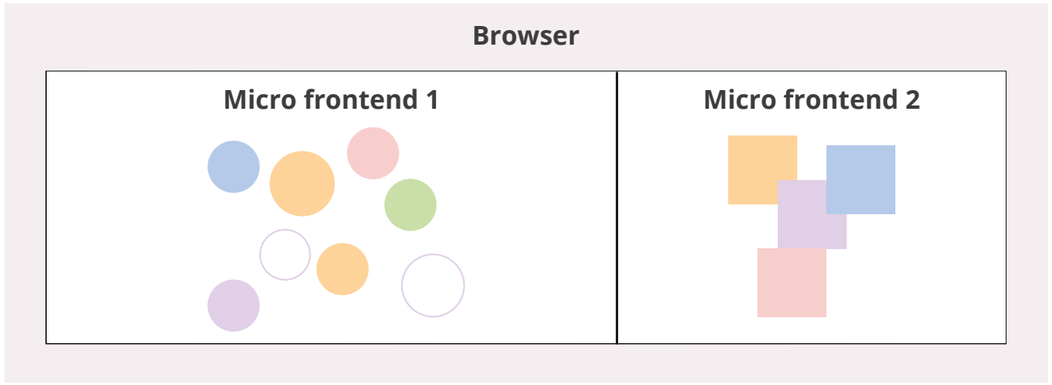
Our teams are now independent of each other from the perspective of development and feature introduction. The only contract each micro frontend team adheres to is to not modify interfaces already exposed and in use by other components of the app. Besides that, every team has great flexibility, and this allowed for fast feature development and deployment.

There are also other advantages that are self-evident: integration testing is now done by partially deploying one single micro frontend in a test environment; a micro frontend can be quickly integrated and used in other existing applications; failures are not propagated to the entire application.

The problem: Event-based communication between asynchronous components

Certain problems arise when MFEs need to communicate with each other, and we discovered this in our own applications.

First, a bit of background regarding how an MFE-based application might be organized from an architectural perspective: micro frontends are not pages. Different micro frontends can compose a page, each representing a specific component. For instance, in a networking application, one MFE might display information regarding inbound and outbound traffic of a network, while another might display the constituent objects of the network (switches, VMs, hypervisors, etc.).



Two MFEs on the same page

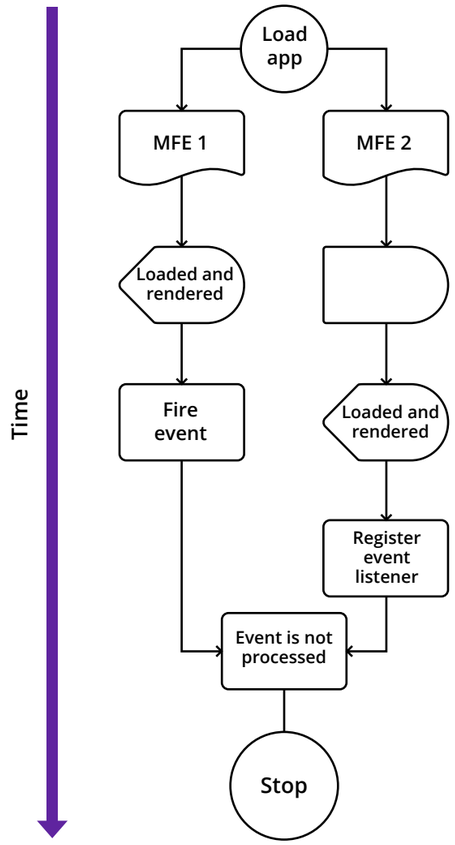
One characteristic of micro frontends is that each is its own bundle file, meaning the user's browser will finish downloading them from the server at different points in time. This asynchrony has certain advantages:

- Asynchronous loading of micro frontends enables the application to be reactive as soon as possible to user interactions
- Network and hardware load are distributed over a certain timeline, which increases perceived performance
- Asynchronous loading avoids the head-of-line blocking problem if a micro frontend requires an abnormally large amount of time to load

So, the order in which the bundle files are downloaded is non-deterministic. Thus, the same load order on each refresh is not guaranteed. The load times may vary from MFE to MFE depending on size, network characteristics, whether a server is overwhelmed, load, and other factors.

Why is this relevant? In a complex UI application (e.g., an enterprise one) it is wise to assume that operations such as scaffolding, data binding, data sourcing, and information sharing between components will take place along the lifecycle of the application. The problem in the context of MFEs arises when these operations take place at the time the former are initialized and are done through communication via events. Even though MFEs are decoupled, some need to communicate with each other.

What happens when a subscriber MFE was not yet initialized at the time a publisher MFE has fired an event? Exactly nothing.



Subscriber MFE missing events from publisher MFE

The flow above starts when an application composed of multiple MFEs is accessed in the browser. There are two vital components: An MFE that fires an event (i.e., a publisher), an MFE that registers a listener for that event (i.e., a subscriber). Both components are initialized and loaded at different points in time, more specifically: the publisher is loaded faster than the subscriber. This means that the event the publisher has fired was not intercepted by any event listener. So, what happens is...nothing. The event is freed by the garbage collector, so the object cannot be processed by any other component in the application.

This is a major problem, and it cannot be solved by existing browser APIs. Imagine a page composed of multiple micro frontends, there is a publisher-subscriber relationship between two of them, and the subscriber performs a critical feature whenever an event is fired from the publisher. If the events are not intercepted:

- Features will be disrupted
- User experience will suffer
- Users will find the application performing poorly and unresponsive to their inputs

Bad solutions

Let's think about a way to solve this problem in an inappropriate manner. There are a few possibilities:

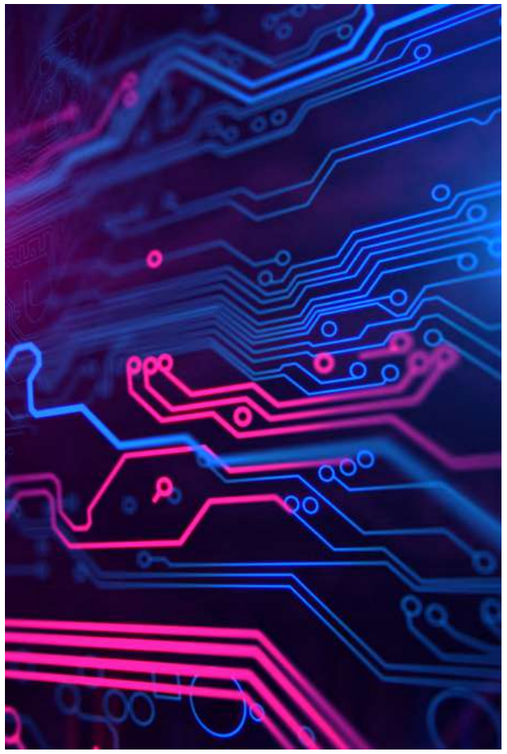
1. Wait for subscribers to be loaded before firing any event
2. Ditch events and use a global state to share data
3. Remove any kind of communication between MFEs

Let's see why all of these three are impractical.

First things first: Waiting for all subscribers to be present before publishing any event. This is obviously detrimental to perceived performance, and it destroys the advantage of asynchronous MFE loading. Not only that, micro frontends can be both subscribers and publishers. The dependencies that MFEs will have between themselves will be hard to manage, which defeats the purpose of compartmentalization.

Second: Ditching events and using a global state for data sharing and interaction between two MFEs. This creates coupling between MFEs and subsequently, teams. "Why is your MFE modifying the state like this?!" is a question you and your colleagues will ask very frequently if you take this approach to communication. Micro frontends should be as decoupled as possible, but full decoupling is not always feasible, which leads us to the next point.

Third: Remove any kind of communication between MFEs. For a non-mediocre application, eliminating communication is not practical. This is because features (i.e., MFEs) depend on each other. There are relationships between the components of a system such that interaction with one of them will trigger reaction in another. At Luxoft, in each of our MFE-based applications there are these kinds of relationships: Click on an entry in a table -> display detailed information regarding that entry in another part of the page. A pan and a stove are very different objects, but you can't cook your pancakes without one or the other.

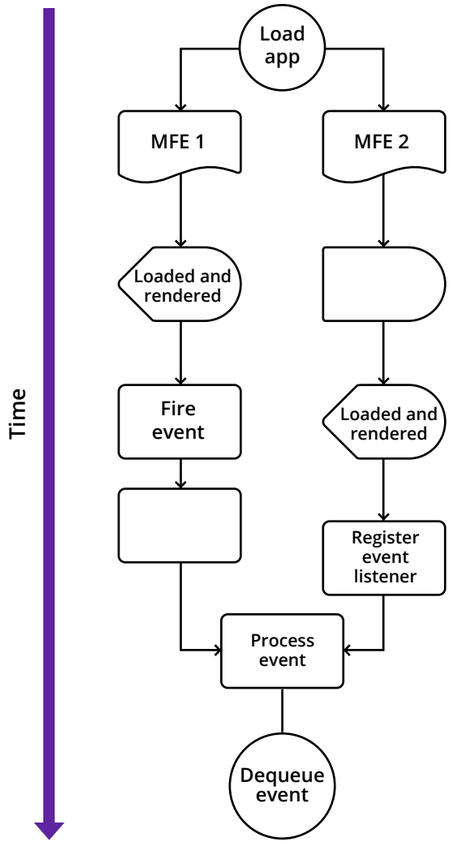


The solution: Event queues

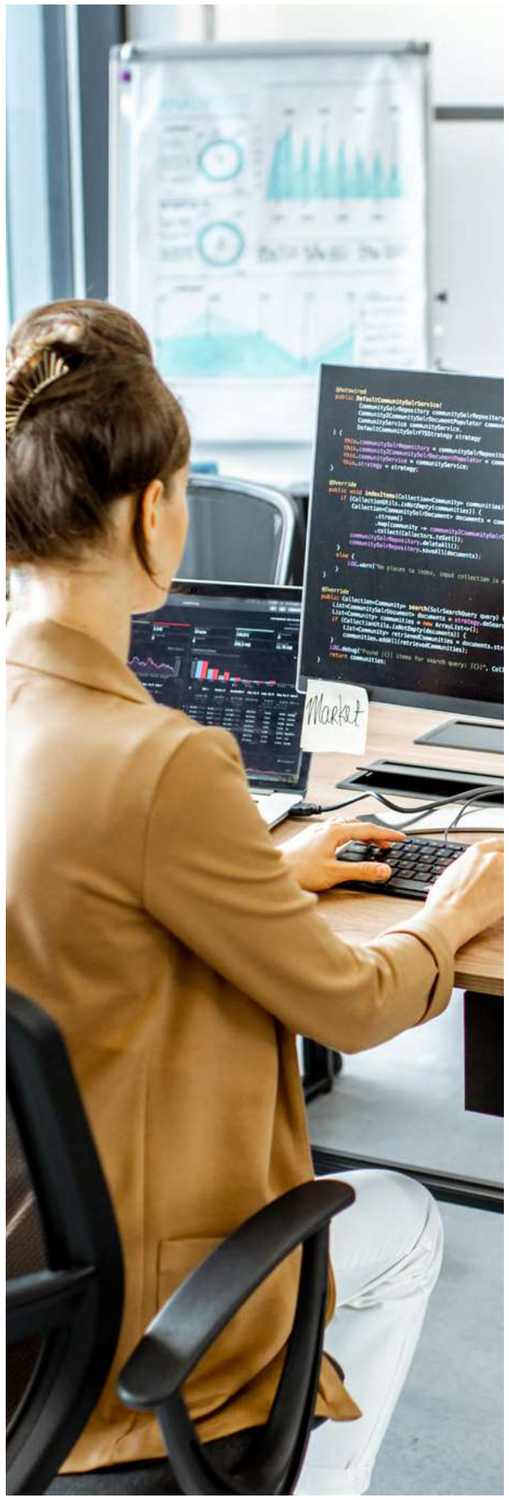
The solution to this problem consists of using event queues.

Since micro frontends are the adaptation of microservices to the UI, we can borrow patterns from one to the other. Event queueing already existed as a concept and in practice on the backend, not so much in the context of UI. Even so, this technique can be used to solve our problem in the most elegant way.

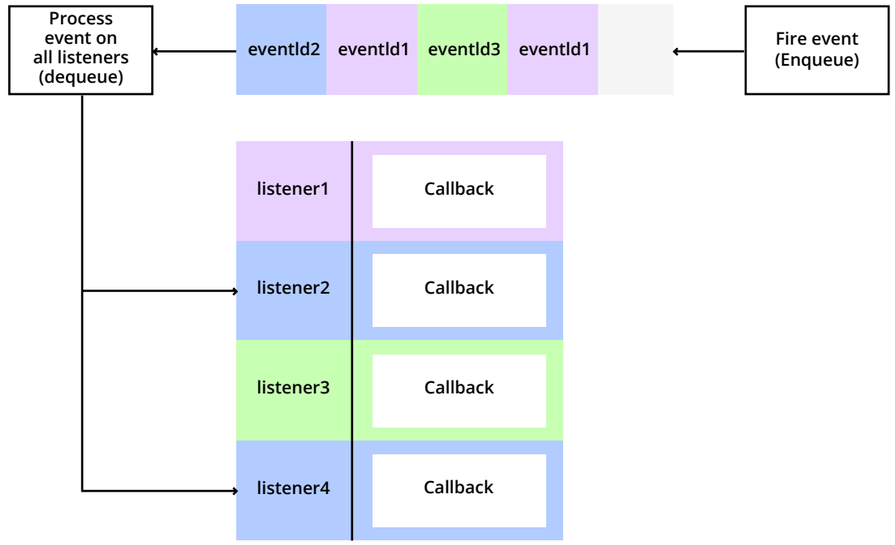
We need to modify our previous flow where the subscriber didn't process the event by adding a few more steps. **We store the event in a queue**, so events are propagated equitably (FIFO). This keeps a reference to the event and thus it will no longer be freed by the garbage collector. **We match the listener to the event in the queue**. Finally, it is processed by the listener and dequeued.



Subscriber MFE correctly intercepting events from publisher MFE



From a high-level, our shopping list has a queue and an arbitrary number of events in that queue. And we also need a place to hold listeners meant to process the events in the queue. Additionally, we need a method to match events to listeners, which can be done using some identifier or other. So, our architecture would look like this:



Messaging system architecture with event queue and listeners

Notice that an event can be processed by multiple listeners. After being processed by every appropriate listener, the event can be dequeued.

But wait, how should an event look like i.e., what's its interface?

```
const event = {
   identifier: «unique:id»,
   payload: {
      awesomeSciFiSeries: «Dune»
   }
}
```

The implementation

Implementing a messaging system that does this can be done using plain JavaScript. Making it such that every MFE-based application can use it regardless of the underlying UI framework.

First things first, we need a way to instantiate our messaging system. A class will do just fine.

Second, we need a queue.

Third, we need a structure that will hold our listeners. Without listeners, no event will be processed. You'll notice that this pattern that we're using is very similar to how browser events API works. This is deliberate, we don't need the overhead of learning to use a complex tool.

```
// We'll call our messaging system «Broker»
class Broker {
  constructor() {
    // Queue that should store our events such that we have a reference to them
    this.queue = [];
    // Object that will hold our listeners
    this.listeners = {};
  }
}
```

All well and good, but instantiating empty objects is basically useless. We need some more operations to make this system actually useful: a method for firing events, a method for registering listeners, and a way for the two to be matched such that events are processed by the appropriate listeners.

Registering listeners

Before writing the code for firing events, we need to decide how the messaging system will match listeners to events. We can take some inspiration from the Custom Events API already present in browser environments. When instantiating a custom event you can specify its type, which is a string that can be used to identify the event when registering a listener.

```
const event1 = new CustomEvent(«event1»);
obj.addEventListener(«event1», (event) => ...);
```

Adapting this approach to our own case, we can make it such that each fired event will have a string identifier specified by the developer. Any listener wishing to intercept the event need only know the identifier of the event:

```
class Broker {
  // ...
  registerListener(eventId, callback) {
    const listenerId = getRandomId();
    if (typeof this.listeners[eventId] !== «object») {
      this.listeners[eventId] = {
        [listenerId]: callback
      };
    } else {
      this.listeners[eventId][listenerId] = callback;
    }
    return listenerId; // return an identifier that can be used to unregister the listener afterwards
  }
// ...
```

this.listeners is a map that holds listeners for each possible event in the application. When an event is fired, Broker can check whether listeners exist for said event and run their callbacks. We're also returning a listener identifier to be used when wanting to remove the listener (always unsubscribe, don't waste memory in your applications).

A method for removing a listener could look as such:

```
class Broker {
  // ...
  removeListener(eventId, listenerId) {
    delete this.listeners[eventId][listenerId];
  }

// ...
```

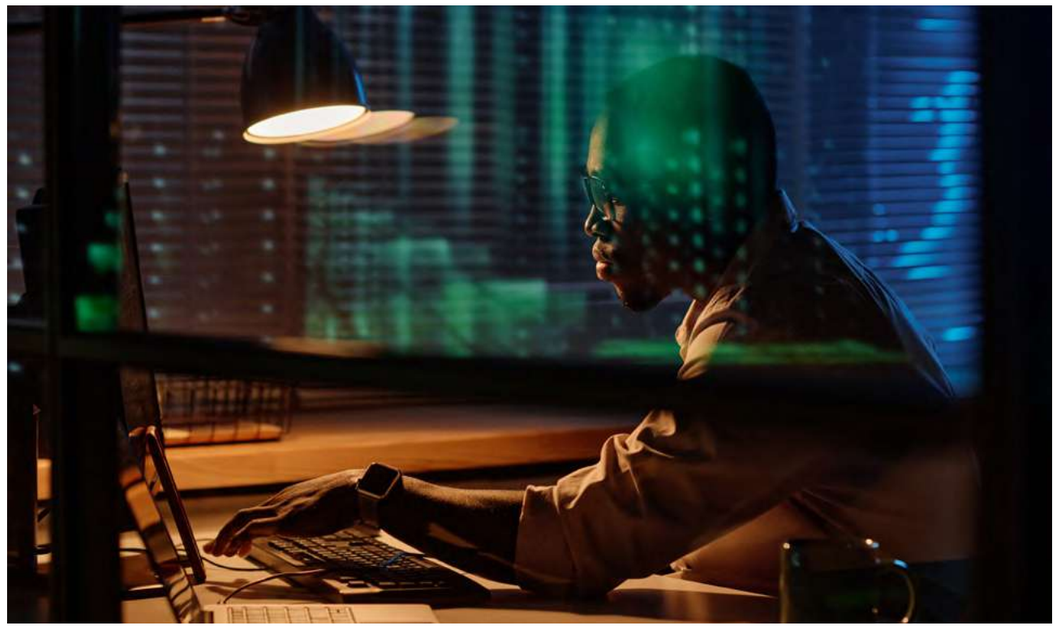
Firing events

Now let's take a look at how our messaging system will fire events. Yes, we need a method.

More specifically, we need a method that enqueues an event. The method receives as an argument an event object adhering to the interface that we defined above.

```
class Broker {
  // ...
  fireEvent(event) {
    this.queue.push(event);
  }
// ...
```

You might notice that something is missing though. Events and listeners exist in our Broker class, but there's no mechanism matching the two! We need to write some code that runs a listener's callback when an event is intercepted.



Processing events

Let's think when our messaging system needs to process events.

When an event is at the head of the queue.

Yes! And there's one more situation when events should be processed. The one we've been trying to solve from the beginning: we need to make sure we process appropriate events when a listener is registered. This is because **events can be enqueued before a listener for them is registered**, just as we discussed in the solution section.

We need to fire a **notification** whenever an event is enqueued or when a listener is registered. We can start processing the events in the queue. When no events are in the queue, we stop processing until we get another notification. A function that would take care of processing events could look like so:

```
class Broker {
  constructor() {
    // ...
    // flag to specify if any work is done at the moment
    processing = false;
    // ...
  }
  process() {
    // if the queue is empty, processing is no longer done
    if (this.queue.length === 0) {
      this.processing = false;
      return;
    }
    this.processing = true;
    // take the first event in the queue
    const event = this.queue.shift();
    // call every callback for the event
    if (this.listeners[event.identifier]) {
      for (let callback of
Object.values(this.listeners[event.identifier])) {
        callback(event);
      }
    }
    // call function recursively
    this.processing = this.process();
  }
// ...
```

It controls a flag that tells whether the messaging system is processing events at the moment or not. This is useful because it prevents this.process() from being called without needing to. So, when firing an event or registering a listener, we can look at the this.processing flag before starting actual processing:

```
class Broker {
  // ...
  registerListener(eventId, callback) {
    // ...
    if (!this.processing) {
      this.process();
    }
    // ...
  }
  fireEvent(event) {
    // ...
    if (!this.processing) {
      this.process();
    }
    // ...
  }
// ...
```

We're done. This should ensure events are delivered regardless of when they were fired.

But what happens when an event has no listeners?

You may have noticed that inside of this.process() we are not handling the case where an event has no listeners. This is more up to the specific application that uses the messaging system, but reliability can't be ignored. To solve this problem, we'll have to make it such that an event does not block the queue. In other words, we need to think of a way to avoid a head-of-line blocking problem.

Remove the events if they won't be processed.

We'll do something of the sort. An event will stay a maximum of 3000 milliseconds in the queue. After that time elapses, the event will be dequeued and processing will continue. Let's modify our this.process() function to do just that.

```
class Broker {
  constructor() {
    // ...
    // specify whether the head event is set to be dequeued or not
    this.eventDequeuingInProcess = false;
    // ...
  }
  // ...
  async process() {
    // ...
    const event = this.queue[0];
    if (this.listeners[event.identifier]) {
      // reset dequeuing flag such that an event is not
      // dequeued immediately if there are no listeners
      this.eventDequeuingInProcess = false;
      // deque event since we know that there are listeners for it
      this.queue.shift();
      // ...
    } else {
```

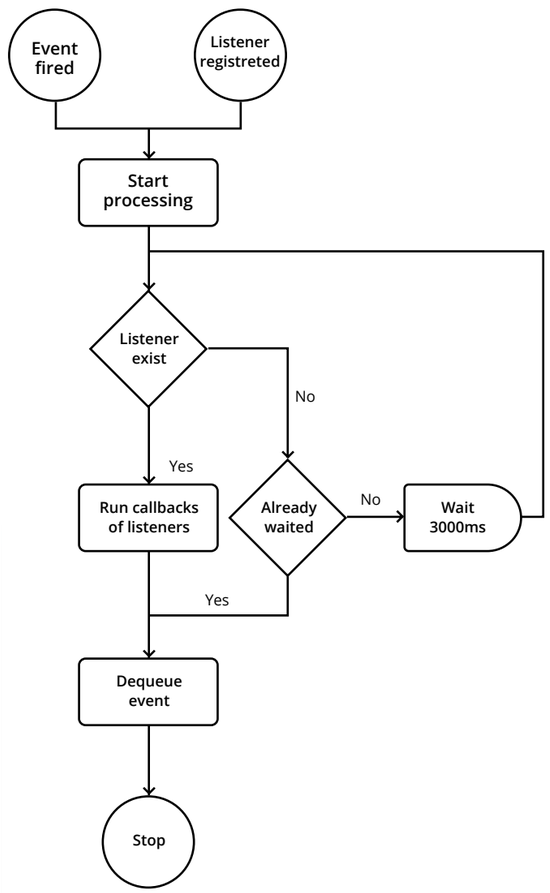
```
            // if there are no listeners, stop processing for 3 seconds,
            // then try again
            if (!this.eventDequeuingInProcess) {
              this.eventDequeuingInProcess = true;
              await this.sleep(3000);
            } else {
              // if there are no listeners, and we already waited
              // 3000 milliseconds, dequeue it
              this.eventDequeuingInProcess = false;
              this.queue.shift();
            }
          }
          // ...
        }
        // There's no other function that
        // allows a program to stop execution temporarily in
        // JS browser environments
        sleep(ms) {
          return new Promise(resolve => setTimeout(resolve, ms));
        }
    // ...
```

Let's walk through what's happening here:

1. Start processing when either an event is fired or when a listener is registered
2. Take event at head of queue and start processing
3. If event has listeners run each callback, dequeue the event immediately
4. If event has no listeners, stop and try again or dequeue it depending on if we already stopped and waited or not

This is possible because this.process() is called recursively.

Graphically, the flow looks like so.



Flow of solution for HOLB problem using sleep

Conclusions

Now you know how to solve the problem of event-based communication between events. Congratulations on learning something new! You can also adapt the solution to your specific needs since it is written in plain JS. A good improvement would be to use TypeScript. Also, remember to use the same instance for all MFEs that communicate with one another.

At Luxoft, this technique of event queueing solved many of our problems in MFE communication. Hope the same will happen for your apps!

References

[1] Taibi, Davide, and Luca Mezzalira. "Micro-Frontends: Principles, Implementations, and Pitfalls." ACM SIGSOFT Software Engineering Notes 47.4 (2022): 25-29;

[2] Geers, Michael. Micro frontends in action. Simon and Schuster, 2020.

Author

Anghel Paul-Lucian
Developer

Anghel Paul-Lucian finished his M.Sc. in software engineering in 2023. His passion and technical focus lie in UI technologies, especially in the web ecosystem and the new concept of micro frontends. At Luxoft Romania he implements scalable and performant micro-frontend based applications and libraries.

Automating 5G and O-RAN testing

4G revolutionized data throughput and latency, offering simplified architecture and scalable networks; 5G propels us even further. With latency below 5 ms and remarkably higher speeds, it unlocks real-time processing, enabling AI/ML in diverse scenarios: intelligent transportation for smart cities, energy-efficient homes, secure autonomous vehicles, immersive augmented reality and Industry 4.0/5.0 innovations.

However, realizing 5G's potential demands a shift. Dynamic open architectures are needed for the radio access network (RAN), contrasting with legacy networks that were static and homogenous. This shift enables power reduction, self-healing networks, AI adaptations and customized radio deployments. The promise of open radio access network (O-RAN) lies in cost reduction and revenue generation, though the complexity mandates thorough testing and real-time validation, crucial in a multi-vendor, disaggregated ecosystem.

Simultaneously, the 5G Core adopts a cloud-based microservices structure, decoupling hardware from software. Each core function becomes a plug-and-play service, facilitating third-party vendors to provide solutions in a diverse multi-vendor approach. A parallel transformation unfolds in the RAN domain, as open RAN dismantles barriers for purpose specific RANs. Its chief advantage is fostering an open, multi-vendor radio access ecosystem, empowering operators to diversify supply chains and tailor innovative solutions while driving cost efficiency through competition and resource sharing.

However, these prospects are met with challenges — multi-vendor compatibility, standards adherence, and, crucially, real-time processing and data volume performance. Rigorous testing across levels is imperative to meet functional, security, scalability and resilience demands. This calls for frequent, stringent testing and validation, exceeding those of prior network standards.

The list below provides a streamlined version of the general steps required in validating an O-RAN network. Even simplified, this will require sophisticated network tools and significant effort to set up and tear down environments.

1. **Planning and preparation:**

- Define objectives — functionality, performance, scalability, interoperability, security
- Create scenarios — diverse test cases, edge scenarios for comprehensive coverage
- Gather resources — hardware, software, tools for effective testing

2. **Functional testing:**

- Validate RAN functionality — radio resource management, handover, cell selection, mobility
- Verify RIC functionality — RIC-RAN interaction focusing on E2/O2/A1 interface compliance
- xApp — in predefined scenarios e.g. validating behavior for QoS optimization, load balancing, energy saving, network slicing control, etc
- rApp — validate non real-time behaviors such as policy optimization or network healing

3. **Performance testing:**

- Capacity and throughput — user loads, traffic handling capabilities
- Latency and delay — response times of xApps, adherence to performance thresholds
- Covergence time for machine learning algorithms employed in rApps
- Quality of service (QoS) — service prioritization, predefined policies

4. **Interoperability testing:**

- RAN-RIC interface — effective communication, control behavior
- Network element interactions — seamless engagement with core networks, orchestrators and aggregated behavior and interactions with x/rApps especially for 3rd party components across versions.

5. **Security testing:**

- Vulnerability assessment — identifying weaknesses, susceptibility to breaches
- Authentication and authorization — proper enforcement, preventing unauthorized access

6. **Scalability testing:**

- Evaluate scalability — network load, user count, traffic volume handling

7. **Resilience and redundancy testing:**

- Assess resilience — recovery from failures, effectiveness of redundancy

8. **Regression testing:**

- Periodic validation — new updates, changes impact assessment, existing functionalities

9. **Negative Testing:**

- Evaluate behavior induced by protocol errors or invalid message parameters

10. **Documentation and analysis:**

- Record results — document outcomes, identified defects
- Analyze outcomes — evaluate against criteria, performance targets

11. Test validation and sign-off:

- Verify compliance — RAN, RIC pass required tests, meet acceptance criteria
- Final report — comprehensive overview, recommendations for improvement

While there is one logical architecture for O-RAN solutions, there are multiple implementation and deployment scenarios. A complete testing solution must be flexible to adapt to those scenarios so that each one may be instantiated, configured, engineered and tested. The configurations range from a fully disaggregated set of functions as per the O-RAN alliance to any number of bundles of CU/RU/DU and near and non real-time RIC and orchestration.

Luxoft's Software Defined Lab (SDL) provides and supports an end-to-end testing environment that includes:

- Cloud solutions — this includes public cloud, private cloud, hybrid cloud and multi-cloud solutions
- On-prem solutions — including virtualized and non-virtualized data centers, COTS and discrete vendor-specific components (VNFs and PNFs) and test simulation/emulation equipment

With SDL used to manage an O-RAN validation environment, it provides a software test automation environment with built-in DevOps features targeted at service providers, operators, virtual network functions (VNF) vendors and network equipment manufacturers (NEMs). The following are its key features:

- An operator or a VNF vendor can comprehensively model an operation of an NFV-based network using multiple 3rd party VNFs
- The operator can compare, evaluate, model, test and engineer the capacity and performance of a solution involving components in multiple infrastructures (cloud, on-prem, virtualized, physical)
- VNFs will be automatically deployed in a desired configuration that implements a required service chain or network services (the SUT)
- The integrated test automation environment orchestrates and executes traffic and functional tests against the assembled VNF configuration

- SDL can test across physical network function (PNF) and virtual network function boundaries for operators as they transition various portions of their services to NFV
- The same system can be used to test existing physical network functions (PNFs) using cost-effective virtual test tools and simulators and can then be used to validate/compare/contrast the behavior of those network functions as they are transitioned to VNFs

At Luxoft, we utilize a combination of test capabilities available from the community and from our partners including Viavi, Spirent and others; SDL is not directly a testing tool and is agnostic with regards to any testing tools you may use that can be placed under software control. These tools along with Luxoft's proven industry leading practice for telecom component and system validation, test automation, and our unique Software Defined Lab solution enable us to achieve a cost effective, yet comprehensive solution for the evaluation, integration, validation, engineering and productization of the components necessary for your RAN and RIC solutions.

Whether you're evaluating and verifying RAN/RIC vendors to work with your existing solution, verifying your RAN/RIC solution for use with other vendors' networks, implementing a traditional RAN/RIC or transitioning to C-RAN, or developing your own RAN/RIC solution and need an experienced partner to develop and customize your solution, Luxoft has the team, tools, ecosystem and experience to ensure the success of your project.

Authors

James Hopson
Technical Product Manager

James has been a Solutions Architect and Product Manager in telecommunications operations and virtualization for more than 30 years, most recently specializing in the areas of network and cloud solutions, NFV testing and test automation. He is the CTO for networking and cloud solutions at Luxoft and a solution manager in Luxoft's Technology, Media and Telecom organization. James is responsible for Luxoft's Software Defined Lab Services Accelerator which is an automated configuration and test, environment management and modelling framework for private/public/hybrid cloud, NFV, hybrid and physical environments.

Mihnea Ionescu
Senior Software Engineer

With over 15 years of software development experience, Mihnea's primary area of focus lies in wireless technologies, including WiFi, 4G and 5G. He is actively engaged in the development of Open RAN testing solutions and 5G radio access network modeling.

Practical introduction to Java multithreading

What is a thread?

Thread is a flow of execution inside a process. In Java, each process has a main thread created by the JVM. When you create a program that prints Hello World to screen, the JVM will create the main thread, which takes care of executing the instructions you've written.

All threads created within the same process can access shared data/resources.

Why do we need multithreading?

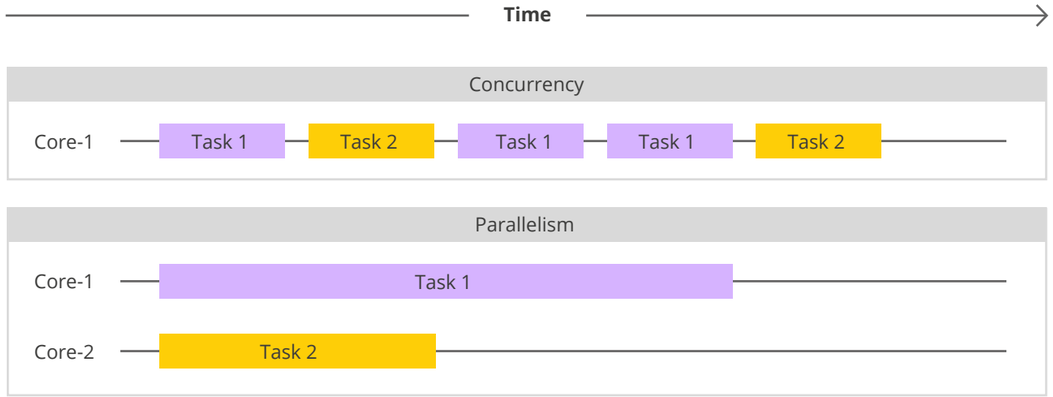
Multithreading allows us to achieve better utilization of the machine by running multiple tasks concurrently/in parallel. And thus, we boost the performance of the application.

Multithreading can be applied in a way to improve the throughput of the system. It can be used to decrease the latency taken by a lengthy operation. Also, it can enhance the responsiveness of applications.

As an example: when you open a video player on a computer, this application makes use of multithreading. At least one thread is handling video streaming. And at least one other thread handling UI interaction. If you click on any button, it does the expected action.

Concurrency vs parallelism

- Concurrency means multiple tasks/subtasks running in overlapping time periods. They will not run in parallel, but they will take turns on the CPU for execution.For example: Doing multi-tasking on a single-core computer. In this scenario, the scheduler will do context switching between the two tasks, which means: The scheduler will assign the CPU to task1 for some time, then move it out and assign the CPU to task2, and so on until the two tasks are completed.
- Parallelism means multiple tasks/subtasks running in parallel, they are not overlapping. For example: Running two tasks on a multi-core computer, and each CPU gets assigned to one task. In this case, each task is scheduled to run independently on a separate CPU.



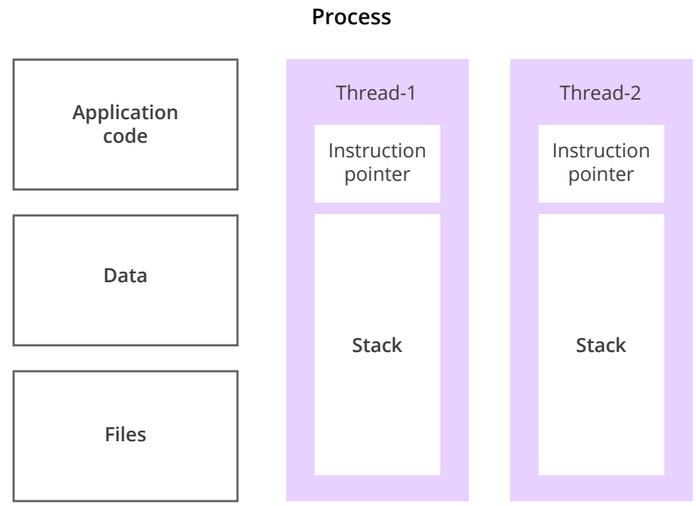
Thread vs process

When you run a program, a process gets created for it. This process has all the details needed for the program to run. At least one execution thread is created within the process to execute the instructions.

In the multithreading model, you create multiple threads within the process. These threads are handling program execution concurrently/in parallel.

	Process	Thread
Size	Processes are more heavyweight	Threads are more lightweight, it is a subprocess inside a process
Communication	Communication between different processes is slower than communication between threads	Faster, since threads are executing in the same process space
Resources	Processes use more resources	Threads use less resources

Below is a visualization of threads and processes:



In general, creating a thread within a process is much faster, and less resource consuming than creating a new process.

Threads within the same process can share data with each other. Hence, we should apply proper synchronization strategies to coordinate between threads, ensure validity of results, and ensure performance.

Without the proper strategies and implementation, we will get the wrong results. Even — in some scenarios — single-threaded implementation becomes more performant than multithreaded implementation.

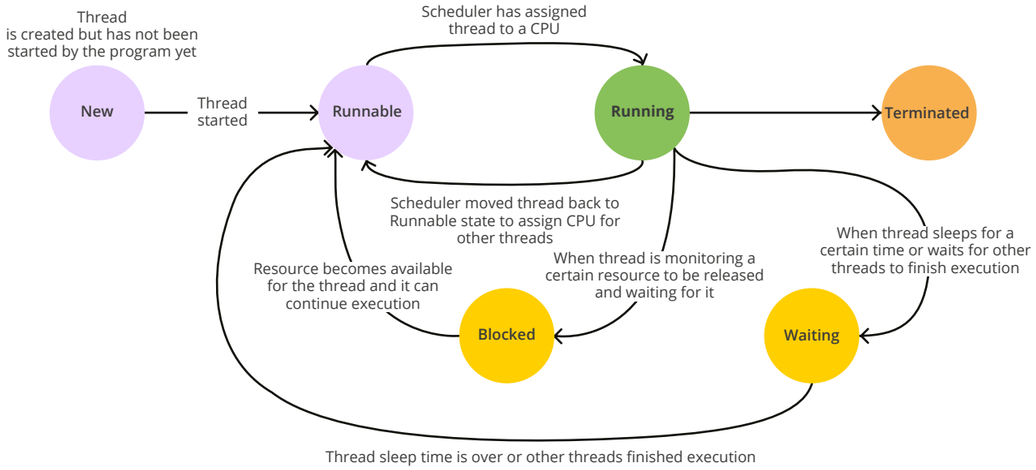
This can happen for several reasons including creating an enormous number of threads which is more than the number of CPUs, especially when these threads are doing a lengthy operation. Or due to inappropriate implementations like running dependent threads in a way that makes their execution occur sequentially rather than concurrently.

Keep in mind that context switching between threads is an operation that comes with a cost. If in a multithreaded implementation, the threads are executing sequentially, the performance would be worse than a single-thread implementation because of the added overhead of context switching between the threads.

All this means is that we should carefully think about how multithreading will solve the problem efficiently and implement accordingly in order not to find ourselves in the above situations.

Thread lifecycle

During the life of the thread, it can have one of the states below:



New: Thread is created but has not started yet.

Runnable: Thread is scheduled for running and will be assigned to a CPU according to the scheduling criteria.

Running: Thread was assigned to a CPU by the scheduler, and instructions in the threads are being executed.

Waiting: Thread is put to sleep for a specific time (timed wait). Or if the thread is waiting for other threads to complete execution. In such cases, the thread is in wait state.

It will not be picked up by the scheduler until it leaves this state and goes to Runnable state (when the waiting time is over, or when other threads complete execution).

Blocked: When thread is waiting on a busy resource to be released. After the resource becomes available, the thread acquires it and goes back to runnable state as. The thread is ready to be picked up by the scheduler to continue execution.

It will not be picked up by the scheduler until it leaves this state and goes to Runnable state.

Terminated: When the thread completes executing all the instructions, it terminates.

Let's get our hands dirty with code!

In Java, we have two ways of creating threads:

- **Thread class:** We can extend the Thread class and override @run() method. However, in this way, we cannot extend any other class.
- **Runnable interface:** We can implement Runnable interface and override @run() method. Using this way has an advantage because we can extend other classes if we want.

Example 1: Extending Thread class:

PrintThread class extends Thread class. This class just prints the name of the currently executing thread.

```
public class PrintThread extends Thread {
   @Override
   public void run() {
      // print the name of the running thread
      System.out.println(«Thread executing: « + getName());
   }
}
```

Driver class that contains the main method to run our application.

```
public class Driver {
   public static void main(String... args) {
      // creating the thread - thread has not started yet
      final PrintThread printThread = new PrintThread();

      // starting the thread
      printThread.start();

      // main thread is waiting for printThread to complete execution
      try {
         printThread.join();
      } catch (InterruptedException e) {
         throw new RuntimeException(e);
      }

      // print the name of the main thread - main
      System.out.println(«Thread executing: « + Thread.currentThread().getName());
   }
}
```

In the main method, we've used **printThread.join()** to wait until the created thread completes its execution. Otherwise, the main thread will not wait and will terminate which will terminate the whole application (even if the created thread has not finished execution by that time).

Expected output:

```
Thread executing: Thread-0
Thread executing: main
```

Example 2: Implementing Runnable interface:

PrintRunner implements Runnable interface. Using this way gives us the chance to extend another class if we want.

```
public class PrintRunner implements Runnable {
   @Override
   public void run() {
      // print the name of the running thread
      System.out.println(«Thread executing: « + Thread.currentThread().getName());
   }
}
```

Driver class that contains the main method to run our application.

```
public class Driver {
   public static void main(String... args) {
      // creating the thread - thread has not started yet
      final PrintRunner printRunner = new PrintRunner();
      final Thread printThread = new Thread(printRunner);

      // starting the thread
      printThread.start();

      // main thread is waiting for printThread to complete execution
      try {
         printThread.join();
      } catch (InterruptedException e) {
         throw new RuntimeException(e);
      }

      // print the name of the main thread - main
      System.out.println(«Thread executing: « + Thread.currentThread().getName());
   }
}
```

Expected Output:

```
Thread executing: Thread-0
Thread executing: main
```


Is multithreading always the best solution?

Given all the advantages of multithreading over single-threading, why don't we use multithreading to solve all our problems?

The answer is: there are scenarios where single-threaded implementations are more performant than multi-threaded implementations.

Remember that multithreading comes with the cost of context switching between the application threads. And this operation has an overhead.

Implementing a multithreaded solution for a rather simple problem usually makes the performance worse.

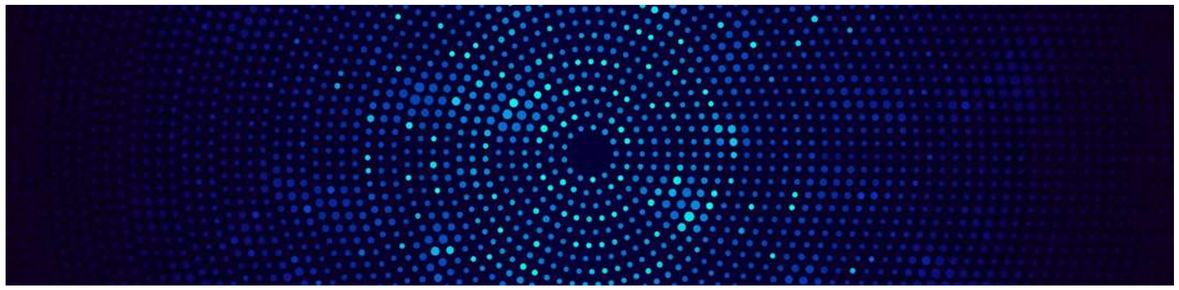
To demonstrate that, let's rely — in the following example — on latency as the metric to compare between multithreaded and single-threaded solutions, and assess the performance of the two implementations.

The problem we want to solve is trivial: We want to double all items in an array of integers.

Single-threaded solution:

Driver class that contains the main method to run our application.

```
import java.util.stream.IntStream;

public class Driver {
   public static void main(String... args) {
      // generating a sequence from 0 to n
      final int[] numbersList = IntStream.range(0, 1000).toArray();

      final long startTime = System.nanoTime();

      for (int i = 0; i < numbersList.length; i++) {
         numbersList[i] = doSomething(numbersList[i]);
      }

      final long endTime = System.nanoTime();

      System.out.println(«total execution time: « + (endTime - startTime));
   }

   private static int doSomething(final int number) {
      return number * 2;
   }
}
```



Multithreaded solution:

ArrayManipulatorRunner that duplicates the items in the array.

```
public class ArrayManipulatorRunner implements Runnable {
    private final int startIndex;
    private final int size;
    private final int[] numbersList;

    public ArrayManipulatorRunner(final int startIndex, final int size, final int[] numbersList) {
        this.startIndex = startIndex;
        this.size = size;
        this.numbersList = numbersList;
    }

    @Override
    public void run() {
        for (int i = startIndex; i < size; i++) {
            numbersList[i] = doSomething(numbersList[i]);
        }
    }

    private int doSomething(final int number) {
        return number * 2;
    }
}
```

Driver class that contains the main method to run our application.

```
import java.util.ArrayList;
import java.util.List;
import java.util.stream.IntStream;

public class Driver {
    public static void main(String... args) {
        // generating a sequence of integers from 0 to n
        final int[] numbersList = IntStream.range(0, 1000).toArray();
        final List<Thread> threads = new ArrayList<>();
        final int batchSize = 100;
        int index = 0;

        final long startTime = System.nanoTime();

        // breaking down the list and distributing work among threads
        while (index < numbersList.length) {
            // create the threads
            threads.add(
                    new Thread(new ArrayManipulatorRunner(index, index + batchSize, numbersList))
            );

            index += batchSize;
        }

        // start the threads
        for (Thread thread : threads) {
            thread.start();
        }

        // main waits for all the threads to complete execution
        for (Thread thread : threads) {
            try {
                thread.join();
            } catch (InterruptedException e) {
                throw new RuntimeException(e);
            }
        }

        final long endTime = System.nanoTime();

        System.out.println(«total execution time: « + (endTime - startTime));
        System.out.println(«Thread created: « + threads.size());
    }
```

In the multithreaded implementation, we're breaking down the array into smaller subsets, and each subset will be handled by a different thread. The size of the subset will be determined by **batchSize** variable.

Results:

Load (array size)	Latency in single-threaded (ms)	Latency in multithreaded (ms) (Each thread is handling a batch of 100 records)
1000	0.0297	7.9933
100000	1.4371	98.493
10000000	6.2862	8749.7436
100000000	35.2813	90527.164
1000000000	292.8242	very long

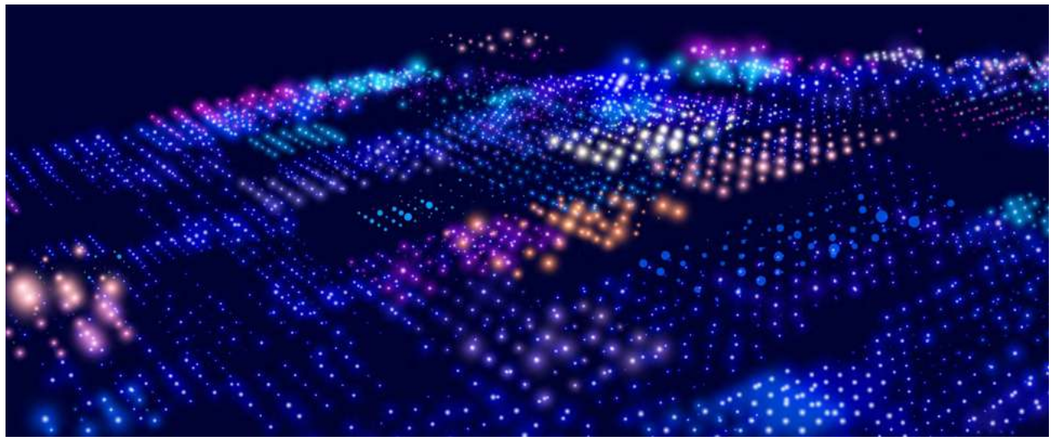
Let's do some tuning and increase the size of the batch handled by one thread by making the batchSize **directly proportional** to the load.

Results:

Load (array size)	Latency in single-threaded (ms)	Latency in multithreaded (ms)
1000	0.0297	7.9933 – batchSize: 100
100000	1.4371	2.3354 – batchSize: 1000
10000000	6.2862	25.8895 – batchSize: 1000000
100000000	35.2813	48.2017 – batchSize: 10000000
1000000000	292.8242	282.7212 – batchSize: 100000000

From the above, we can see that single-threaded implementations outperforms multithreaded ones in situations like when the problem is simple, or when the data size is quite manageable.

However, multithreaded implementations outperforms single-threaded ones in situations like when the load becomes enormous, or when the problem is not trivial, or when the process handled by a thread is a lengthy operation, so it would be faster to execute concurrently/in parallel.



Conclusions

- Multithreading helps boost the performance of applications.
- It is much less expensive to create a thread than creating a new process.
- Synchronization between threads should be kept if they are sharing the same resources.
- There are two ways of creating threads in Java; implementing Runnable or extending Thread.
- Debugging multithreaded applications requires more effort than debugging single-threaded applications.
- Not all problems should be addressed using a multithreaded approach. Proper analysis should be made beforehand to assess whether a multithreaded solution would be better or not.
- Multithreading comes with an overhead, if not used properly for the right problem, the results would be worse than single-threaded approach.
- Consider the computational resources available when considering a multithreaded approach.

Author

Mahmoud Yehia
Senior Software Engineer at Luxoft

Mahmoud has +5 years of professional work experience. He has worked on interesting projects throughout his career. Mahmoud is an ex-IBMer, ex-Amazonian and now proudly a Luxofter!
His main experience lies in backend development. He's eager to learn more about clean coding, optimization techniques, software architectures and exploring AI.

Workflow orchestration and integration: Streamlining process automation

In the current complex business environments leveraging IT, the management and automation of processes across various software applications and interfaces are necessary for operational efficiency and competitiveness. This article explores the significance of workflow orchestration in handling cross-system processes and examines the role of integration in process automation.

Introduction

The evolution of process automation: The business landscape has been greatly reshaped by various technological advances. Automation has always represented an opportunity to create value from the balance of the classic paradigm of people, process and technology.[1] Process automation's progression echoes this transformation, proceeding from manual paper-based workflows to complex automation solutions driven by digital technologies and the Internet. The shift from manual processes to automation has been fuelled by the need for organizations to remain competent, reactive and competitive in today's dynamic business environment.

Automation started with simple tools and scripts for streamlining repetitive tasks. As technology matured, these tools evolved into more intricate systems capable of managing complex workflows. The introduction of Enterprise Resource Planning (ERP) systems brought centralized data management, emphasizing the necessity of sychronization among disparate systems.

ERP systems could handle a variety of back-office processes like human resources and accounting, as well as front-office processes. Yet with ERP solutions organizations could not focus on improving the efficiency of their business processes. For that, more sophisticated integration and planning tools were needed.[2] This paved the way for the emergence of workflow orchestration and integration.

The role of workflow orchestration and integration: In today's business world, characterized by varied software applications, databases, and APIs, workflow orchestration and integration are essential for successful process automation. While these terms are often used interchangeably, they address distinct aspects of the automation process.

- **Workflow orchestration:** Orchestration involves coordinating and managing multiple tasks and processes to achieve specific business outcomes. Workflow orchestration is the automation of a workflow or multiple tasks. In other words, it handles multiple automated tasks to execute a single significant process or workflow.[3] Workflow orchestration ensures correct task sequencing, manages dependencies and enables seamless data flow.
- **Integration:** Integration connects different systems, applications, and databases to enable data and process interoperability. Data silos and unnecessary data entry are eliminated with effective integration, ensuring unfettered data flow. Data transformation methods, middleware, and APIs can all be used to integrate systems.

For process automation to be seamless and effective, integration and workflow orchestration must operate together. While integration ensures the relevant data is available for certain tasks, workflow orchestration provides optimal task execution sequence.

Understanding workflow orchestration

Workflow orchestration is the end-to-end management of people, digital workers, systems, and data in a process.[4] It goes beyond simple task automation by establishing orderly flows that are in line with corporate goals. Without having to deal with complicated technological issues, orchestration solutions offer visual interfaces for planning, monitoring, and changing workflows.

Different software programmes, databases and services are necessary in today's organizations. Workflow orchestration is excellent at managing these cross-system procedures, which span several of these platforms. It makes sure everything functions in harmony.

Consider an e-commerce business processing an order that requires coordination of delivery, payment processing, customer notifications and inventory management. Workflow orchestration synchronises these steps, ensuring error handling, orderly advancement, and structure preservation even in the face of unanticipated events.

Benefits of effective workflow orchestration: A Forrester report noted that platforms capable of handling process orchestration allows organizations to add and manage digital workforce capacity on demand, with the flexibility and agility needed to scale and optimize service levels. This frees up bandwidth so

employees can move away from mundane, repetitive tasks and focus on the high-value, strategic work, improving employee satisfaction and productivity.[5]

The benefits of implementing effective workflow orchestration are substantial:

- **Enhanced efficiency:** Orchestration follows predefined paths, minimizing manual intervention and mistakes for quicker, more consistent outcomes.
- **Optimized resource use:** Both human and technical resources are used optimally. Tasks trigger as needed by the right resources.
- **End-to-end visibility:** Orchestration offers an overview of the entire process, enabling better command and management by identifying bottlenecks or deviations.
- **Scalability:** Orchestration accommodates new tasks and systems as businesses grow, ensuring scalability.
- **Compliance and auditability:** Orchestration adheres to compliance standards, facilitating audit trails and reporting.
- **Adaptability:** Processes can be modified without overhauls, addressing changing conditions or requirements.

Effective workflow orchestration is pivotal for operational excellence in complex environments. It streamlines processes, enforces compliance and optimizes efficiency.

The importance of integration in process automation

The integration imperative: In the current business environment, integration is a critical component of process automation. Connecting these systems becomes essential as organizations use a variety of software applications, databases, and services for effective operations; otherwise, they will operate in silos. Coherent and efficient processes are made possible by integrated components, which guarantee seamless data flow. An article from CIOReview[6] points out that integration seem to have the answer to problem of workflow silos.

Achieving seamless data flow: Smooth and precise data flow is essential for effective process automation. Data fragmentation can make it difficult to collaborate and make decisions; integration prevents this. Without manual intervention, seamless data flow guarantees that data is available when and where it is needed. Time is saved, mistakes are decreased, and the experiences of customers and employees are improved.

Think about a customer order process: Without any manual input, an online order moves through inventory control, starts warehouse operations, updates sales records and notifies customers. Experiences are frictionless thanks to this integration.

Enhancing efficiency and accuracy: Integration enhances process automation by eliminating redundancy and reducing human intervention. Automated data transfer removes the need for manual data entry, while real-time communication accelerates processes.

In health care, integrating electronic health records with laboratory systems provides instant data access to doctors and enhances patient safety by reducing miscommunication risks.

Integration also improves accuracy by maintaining data consistency across systems. Integrated financial systems ensure consistent financial data across departments.

Seamless process automation requires effective workflow orchestration and integration. Orchestrating tasks ensures efficient order, while integration enables data flow. Organizations embracing these strategies optimize operations, minimize errors and stay competitive.

Challenges in workflow orchestration and integration

Heterogeneous IT landscapes: Today's businesses operate in intricate IT environments with a wide range of systems. Due to compatibility issues, disparate APIs, and data structure incompatibilities, integrating these systems requires careful planning.

Data mapping and transformation: Data often varies across systems, necessitating data mapping and transformation. Mapping establishes common data understanding, while transformation ensures data compatibility.

Security and compliance: Secure data transfer is necessary for system integration to avoid security flaws and compliance violations. Regulations like GDPR must be followed when exchanging data.

Key components of successful orchestration and integration

Robust integration middleware: Integration middleware is a 3rd-party tool that allows you to connect two or more applications. These applications can be cloud-based (e.g., SaaS application) or live on-premises (e.g., legacy database).

API management and governance: API management is the set of people, processes and technology that enables an organization to safely and securely publish APIs, either internally or externally.[7] Governance ensures responsible API usage, enhancing security and compliance.

Process monitoring and error handling: Process monitoring offers real-time visibility into workflow status. Error handling detects and addresses exceptions to maintain workflow integrity.

Best practices for effective workflow orchestration and integration

Mapping process flows and dependencies: Understand and map processes for efficient orchestration. Break down workflows, identify dependencies, and visualize task sequences.

Selecting appropriate integration patterns: Choose integration patterns based on organizational needs. Patterns range from data synchronization to event-driven architectures.

Prioritizing scalability and flexibility: Prioritize scalable solutions that adapt to changing requirements. Flexibility allows easy adaptation to market changes.

Embrace change management: Effective change management ensures that employees are informed about changes, understand the benefits and are trained to use new systems efficiently.

Continuously innovate: As technology evolves, new tools and practices may offer improved efficiency and capabilities. Continuously innovating ensures that your processes remain up to date and optimized.

Future trends in workflow orchestration and integration

Cloud-native orchestration solutions: Cloud-native solutions enable dynamic and scalable orchestration using containerization and microservices. Solutions like Kubernetes offer flexibility and portability.

AI-powered intelligent integration: AI-powered tools automate integration tasks, predict integration needs and suggest optimal solutions. Machine learning enhances integration efficiency.

Blockchain for secure interoperability: Blockchain ensures secure data exchange through trusted ledgers. It enhances authenticity, traceability and privacy in data sharing.

Conclusion

The paradigm of process automation has evolved in the dynamic world of business from simple task execution to a seamless partnership between people and machines. The forefront of this transformation is led by workflow orchestration and integration, which empowers businesses to manage complexity, boost productivity and seize previously unimaginable opportunities. Innovation, integration, and foresight have paved the path from conventional automation to seamless collaboration.

Automation's development has made it clear that organizations can synchronise disparate tasks into harmonious workflows by orchestrating processes.

This symphony is made up of strong integration middleware that manages the seamless data transfer between systems and is directed by AI-powered intelligence that supports human decision-making. Focus is on cloud-native orchestration, which provides a flexible stage for these orchestrated performances while blockchain ensures the veracity of each note in the data exchange.

Organizations, which are situated at the nexus of the present and the future, can gain an advantage through efficiency, adaptability, and the convergence of these trends. Data that is seamlessly transferred between systems fosters innovation, gives employees power and makes customers happy. Process optimisation, precise insights and the freedom to direct human creativity towards higher-value tasks are all advantages.

Recommendations for organizations exploring orchestration and integration:

For those embarking on the journey of workflow orchestration and integration, several key recommendations can serve as guiding stars:

- **Prioritize strategy:** Approach automation as a strategic initiative aligned with your business goals. Map out processes, identify pain points, and set clear objectives for workflow orchestration and integration.
- **Design with the future in mind:** Choose cloud-native solutions that offer scalability, flexibility and portability. These foundations will ensure that your orchestrated workflows remain agile in the face of changing requirements.
- **Invest in AI and automation expertise:** As AI-powered integration gains prominence, nurturing a team proficient in both automation and AI technologies will be a valuable asset for your organization.
- **Collaborate across departments:** Successful integration requires collaboration among various departments and stakeholders. Involve IT, operations and business teams to ensure a holistic approach.
- **Ensure compliance and security:** In the era of data privacy and compliance regulations, prioritize secure integration practices. Blockchain can offer an extra layer of security for sensitive data.
- **Embrace change management:** The shift towards seamless automation might introduce new processes and tools. Proper change management strategies can ease the transition for employees.
- **Continuously innovate:** The landscape of automation is ever evolving. Stay informed about emerging technologies and trends and be prepared to adapt your strategies accordingly.

References:

1. https://www.ibm.com/thought-leadership/institute-business-value/en-us/report/ibvprocessautomation
2. https://www.processmaker.com/blog/the-evolution-of-digital-process-automation/#
3. https://www.xenonstack.com/insights/workflow-orchestration#:~:text=Workflow%20orchestration%20is%20the%20automation,single%20significant%20process%20or%20workflow
4. https://appian.com/blog/acp/process-automation/workflow-orchestration-explained.html
5. The Kofax 2020 Intelligent Automation Benchmark Study Part 3: Intelligent Automation Platforms Accelerate Digital Workflow Transformation Success
6. https://www.cioreview.com/news/integration-key-for-breaking-down-data-silos-nid-23522-cid-17.html
7. https://www.gartner.com/en/information-technology/glossary/api-management

Author

Parameswaran Sivaramakrishnan
Account delivery lead

Param brings over a decade of IT expertise to the table, specializing in the realms of business process management and workflow orchestrations. With an extensive career spanning various industries including manufacturing, finance, and insurance, he has honed his skills in demystifying intricate technical concepts, rendering them comprehensible for audiences of all technical backgrounds.

Why is precise monitoring and observability so crucial for trading systems?

In brief

- Monitoring and observability tools provide real-time feedback on each system component's fitness, performance and interoperability, rapidly spotting and addressing irregularities and other urgent issues
- Performance monitoring analyzes system metrics, such as trade execution time, order fill rate, slippage and latency. It helps identify bottlenecks or inefficiencies impacting the system's overall performance
- Constantly monitoring the system infrastructure, including servers, network connectivity, data feeds and platforms, helps ensure the system is always available, minimizing downtime and potential losses

Put simply, it has a lot to do with the complex ins and outs of banking and capital markets trading.

Only leading-edge monitoring and observability tools deliver the extraordinary accuracy and flexibility needed to negotiate the hyper-connected financial industry reliably, day in and day out. In such a quicksilver environment, speed of action and response dictates profit and loss, which can prove devastating.

In the following pages, we're going to deconstruct the systems and processes used in global trading. And as we trace the merits of leveraging Murex-like instruments, we'll discover the crucial part observability and monitoring play in their robust and agile effectiveness in banking and capital markets.

First things first

Monitoring for trading systems refers to the process of continuously observing and analyzing the performance and behavior of a trading system. It tracks various metrics, such as trade execution speed, order book depth, latency, risk exposure and profitability, to ensure the system functions as intended and identifies any potential issues or anomalies.

Luxoft has partnered with ITRS (a leading real-time estate monitoring and observability software provider) to develop and deliver advanced monitoring tools tailored to the banking and capital markets sector, we have built a trading observability framework that can be applied to trading systems such as Murex, Adenza, Finastra, Orchestrade and any other Front to Back Trading platform.

Our specifically targeted monitoring solutions provide an overview of heritage and cloud-based systems, enabling financial organizations to safely cross the evolving regulatory landscape while getting accustomed to new technologies and marketplace instability. This vital development underlines the partnership's commitment to helping shape the future of financial trading.

Resolving trading system challenges

Financial trading is in a world of its own. Its fundamental principles are characterized by dynamism, individuality and rapid response. A trading system is more than the sum of its parts — a vast ecosystem comprising a multitude of integrated components that drive complex trading initiatives. They include tools for risk analysis, order-management systems, data feeds, etc.

Monitoring and observability provide real-time feedback on each component's fitness, performance and affiliate rapport. The tools keep a sharp eye on these essential and convoluted networks, rapidly spotting and addressing potential bottlenecks, irregularities and other urgent issues while diminishing the impact on core trading processes.

Here's an idea of the core significance and critical drivers of monitoring and observability in banking and capital markets.

Enabling real-time decision-making

Instant and infallible trader judgment is a core competency for those working in quick and aggressive banking and capital markets. Observability delivers

a real-time overview of system behavior, response times and transaction performance. Monitoring tools record and analyze the criteria for the speed of order execution, transaction confirmation and data-feed latency. And, by feeding traders and decision-makers immediate and dependable information, they're more likely to make more effective decisions.

Optimizing performance

As I'm sure you're aware, performance has a direct line to trading systems profitability.

Performance monitoring involves measuring and analyzing the system's performance metrics, such as trade execution time, order fill rate, slippage and latency. It helps identify bottlenecks or inefficiencies impacting the system's overall performance.

Observability tools provide reports on resource utilization, application performance and reaction times. Also, institutions can identify opportunities for optimization, refining the system to process transactions faster, cut latency and enhance operational efficiency by closely monitoring the signals.

Making predictions

In addition to outlining the system's current condition, advanced monitoring also forecasts potential complications like bottlenecks by analyzing historical data and patterns. Adopting the old adage "Prevention is better than cure," the IT team can maintain operational continuity and reduce the likelihood of downtime.

Complying with the rules

As financial services is a highly regulated industry, trading systems have to adhere to stringent compliance requirements. Observability tools can track and record trading activity, providing the necessary audit trail for reporting. This ensures transparency, complies with regulations and clears adverse legal issues.

Monitoring risk

Monitoring a trading system's risk exposure ensures that the level of risk stays within established parameters. It tracks metrics like size, leverage and risk-adjusted returns to identify possible violations and react appropriately.

Troubleshooting

The rapid discovery and recovery of system defects or disorders is critical for enterprise well-being. Observability technology insights help IT specialists determine causality and execute the appropriate fix. Consequently, mean time to resolution (MTTR) is significantly curtailed with minimal adverse effect on activities.

Planning scalability and resources

Financial markets often encounter swift trading volume peaks. Observability tools help manage heavy traffic, monitoring resource use and system performance. Banks can ease resource scaling in times of rising demand by examining trends to guarantee premier performance regardless of the conditions.

Ensuring business continuity

For trading systems, unexpected downtime means financial loss and potentially damaged reputations. Thanks to their continual vision of system health, monitoring and observability technologies can help banks plan their redundancy, failover mechanisms and disaster-recovery strategies.

Achieving data security and privacy

Due to the sensitive nature of financial data, security and privacy are of prime significance to the banking and capital markets sector. Monitoring and observability solutions offer the data integrity and confidentiality needed to detect irregular data-access patterns or security breaches in real time.

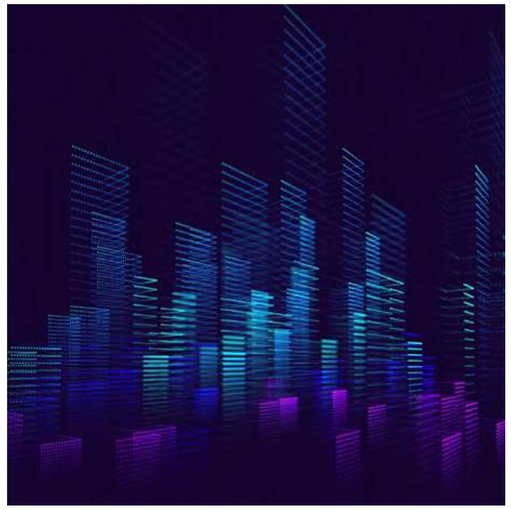
Reporting and visualization

The monitoring process usually includes creating reports and visualizations to provide a clear view of trading system performance and behavior. Dashboards, charts and alerts (e.g., crucial metrics and unexpected issues) are typical requirements.

Code-related trading-system monitoring factors involve assimilating tools or libraries into the system's codebase to help collect/analyze data, originate alerts or notifications and deliver visualizations.

Detecting anomalies

Anomaly monitoring identifies unexpected and abnormal behavior in trading system data or operations (e.g., abrupt spikes in trading volumes, atypical trading patterns, departure from set risk criteria). Anomaly detection helps spot adverse issues or irregularities before they affect the business.



Enhancing the user experience

Tracking user interactions in trading systems allows banks to better understand how traders and employees utilize the platforms. Simplifying workflows and offering excellent training resources enhances productivity, increasing data value and usability across the entire enterprise.

Managing costs

Effective monitoring and observability help manage and drive down costs. Identifying inefficiencies and uneven resource distribution helps banks determine the most effective allocation and cost management tactics and strategies.

Sharpening your instincts

Maintaining a strong monitoring and observability system gives banks an edge in the turbulent financial services marketplace. It expedites differentiation by developing increasingly reliable services, servicing compliance and mitigating operational risks.

Integrating others

What about integrating monitoring and observability systems with other third-party tools and banking technology favorites, like analytics platforms, compliance software and cybersecurity solutions?

Watching your health

Rather than an occasional health check, the tools constantly monitor the condition and alertness of the system infrastructure, including servers, network connectivity, data feeds and platforms. It helps ensure the system is up, running and continually available, cutting downtime and potential losses to the absolute minimum.

Developing trends

Financial monitoring and observability improvements embrace hot topics like the effectiveness of cloud-based monitoring tools and AI/ML for predictive monitoring. That's why Luxoft and ITRS launched their strategic partnership for developing and delivering advanced monitoring tools for banking and capital markets platforms.

Their combined solutions deliver a coordinated overview of heritage and cloud-based/premises systems to help banks satisfy regulatory and technological challenges while mitigating operational risks, ensuring compliance and strengthening durability.

Working together

Clearly, the case for financial monitoring and observability is proven. It's not a like-to-have; it's an integral feature of commercial and operational success. In light of this need, the Luxoft and ITRS partnership is about more than the solution. The partners are fully committed to maintaining a secure lifeline for financial institutions negotiating choppy economic waters.

Monitoring and observability tools offer a holistic view of both legacy and cloud-based systems. This vision shines a light on a route map for banks daunted by the challenges arising from new regulatory demands, emerging technologies and shifting market conditions.

Wrapping up

So, what have we learned? Financial trading and complex trading systems require an unswerving commitment to monitoring and observability. They are an essential source of accuracy, reliability and resilience in an uncertain marketplace. From optimal performance through efficient compliance and insightful troubleshooting to exceptional user experiences, observability and monitoring set a banking and capital markets benchmark.

Contemporary monitoring and observability technologies guide banks safely toward enhanced resilience, sustainable growth and, with good fortune and a following wind, unqualified success.

In an increasingly fast-turnaround world, real-time decisions make or break businesses.

Looking for more insights?

If you'd like to discuss any particular operational problems you're currently coping with or discover what kind of premier performance and competitive advantages the Luxoft and ITRS partnership could coax from your trading systems, visit **luxoft.com**

References:

1. https://blog.flagright.com/post/why-transaction-monitoring-is-important#:~:text=It%20enables%20banks%2C%20credit%20unions,safety%20of%20their%20monetary%20transactions.
2. https://www.globalbankingandfinance.com/what-banks-need-to-know-about-observability/
3. https://www.itrsgroup.com/blog/banks-monitoring-containers
4. https://www.luxoft.com/files/pdfs/Luxoft-and-ITRS-advanced-monitoring-solutions.pdf
5. https://www.luxoft.com/pr/luxoft-and-itrs-partner-to-deliver-advanced-monitoring-tools

Author

Nimmala Naga Santhosh Baba
Principal Consultant (TREC Lead Cloud and DevOps)

Santhosh Nimmala is TREC Lead for Cloud and DevOps Trading and Risk Management Solutions at Luxoft. He has 8+ years of experience across APAC and EMEA, focusing on Cloud and DevOps solutions and building accelerators which help clients with their cloud migrations.
In pre-sales, Santhosh also has a successful track record of delivering multiple Cloud and DevOps implementations. Currently, he is responsible for Developing and delivering Cloud and DevOps solutions for the Trading and Risk Management Solutions practice and driving multiple initiatives across the organization.

Fuzzing! Why traditional testing is not enough

In the dynamic and fast-paced world of software development, ensuring the reliability and security of applications is a top priority. Traditional testing methods have long been the bedrock of quality assurance, but they come with inherent limitations. Here comes fuzzing, a not so well known automated software testing technique that is gaining significant attention for its ability to uncover unexpected behaviors, vulnerabilities and security flaws. In this extensive article, we will embark on a deep dive into the world of fuzzing, exploring its fundamental concepts, historical evolution, the underlying logic of fuzzing tools and its critical applications across various domains. Finally, we will place a special focus on how fuzzing is revolutionizing the automotive industry.

The purpose and significance of fuzzing

At the heart of software development lies the quest for reliability, robustness and security. Traditional testing methods, such as unit testing and integration testing, are essential but have limitations when it comes to exploring the vast landscape of possible software behaviors. This is where fuzzing steps in as a valuable addition to the testing arsenal.

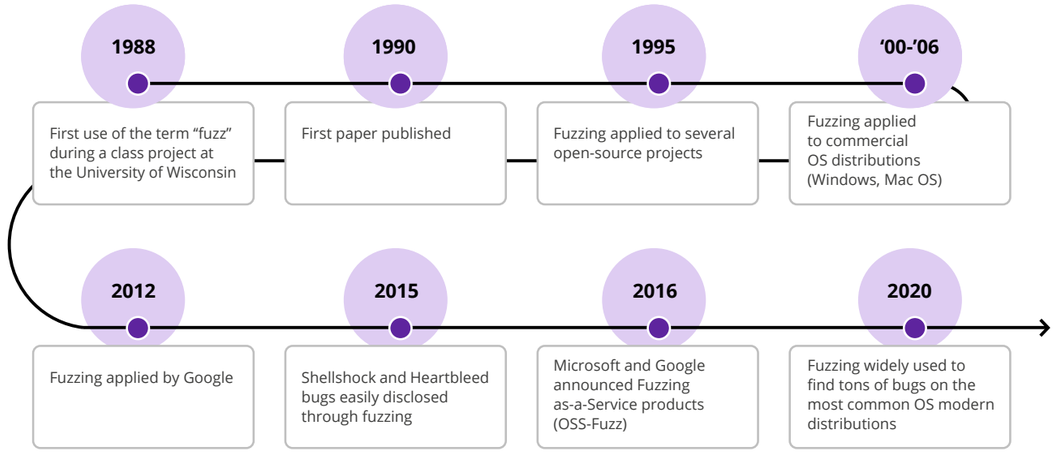
What is fuzzing?

Fuzzing is an automated software testing technique that involves feeding a program with a bunch of invalid, unexpected or random data as inputs. The program under examination is then closely monitored for any exceptions, crashes, memory leaks or unexpected behavior. The fundamental idea behind fuzzing is to venture into uncharted territories and edge cases of a software's behavior, probing the extremities that traditional testing may miss.

In the context of complex software, such as the intricate systems found in autonomous driving vehicles, the spectrum of expected and unexpected behaviors expands dramatically. Traditional testing may well cover the side of expected behaviors, but it often falls short when it comes to identifying vulnerabilities and unexpected actions on the other side of the spectrum. Fuzzing bridges this gap, actively exploring and testing the extremities of software behavior.

The historical evolution of fuzzing

The reader could think that fuzz testing is a cutting-edge fancy technology, but fuzzing is not a recent innovation and has a rich history that spans over three decades. Let's embark on a historical journey through the development and adoption of fuzzing techniques.



The birth of fuzzing

The term "fuzzing" first emerged when it was used in a class project at the University of Wisconsin in 1988. It was two years later, in 1990, that the first paper on fuzzing was published, marking the initial steps in its evolution.

Fuzzing in open-source projects

In the mid-1990s, fuzzing started gaining traction in open-source projects. Although not yet widespread in commercial software, its presence in open-source initiatives signaled its potential.

Fuzzing catches the attention of tech giants

Around the early 2000s, tech giants like Apple and Microsoft recognized the value of fuzzing as a testing technique. They began incorporating fuzzing into their testing procedures, primarily to scrutinize their operating systems for vulnerabilities.

Google's pioneering role

The year 2012 marked a pivotal moment for fuzzing when Google started applying fuzzing techniques to its products and services. This move significantly increased the visibility and importance of fuzzing in the software development landscape.

Uncovering critical vulnerabilities

In 2015, a remarkable demonstration showcased the power of fuzzing. Two widely known security vulnerabilities, Shellshock and Heartbleed, which had plagued the industry for extended periods, were rapidly discovered and exposed using fuzzing techniques. These discoveries underscored the effectiveness of fuzzing in uncovering critical vulnerabilities that traditional testing had missed.

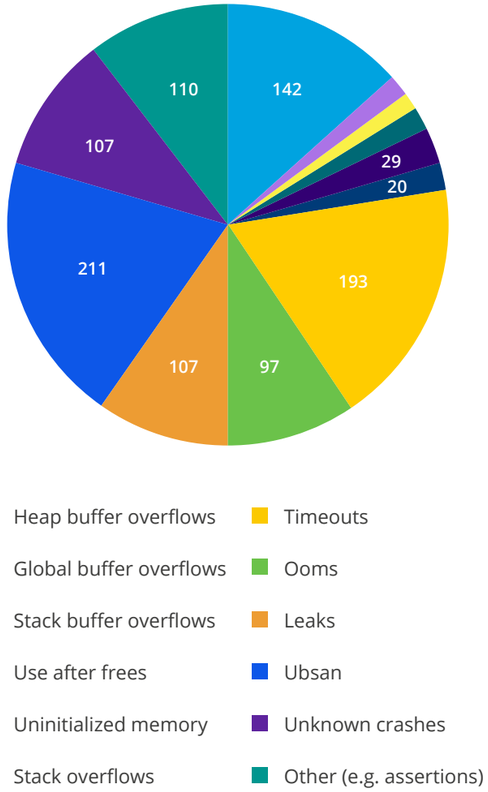
Fuzzing as-a-Service

In 2016, both Microsoft and Google introduced Fuzzing as-a-Service, making this powerful testing technique more accessible to developers worldwide. This marked a new era in the democratization of Fuzzing, allowing a broader range of projects to benefit from its capabilities. In particular, OSS-Fuzz is a pioneering fuzzing service and platform developed and maintained by Google. Its primary mission is to enhance the security and robustness of open-source software (OSS) projects by systematically identifying and addressing software vulnerabilities through automated fuzz testing.

Numerous open-source projects have benefited from integrating with OSS-Fuzz. Some notable examples include:

- Wireshark: The popular network protocol analyzer has addressed multiple security vulnerabilities and stability issues with the help of OSS-Fuzz
- SQLite: A widely used embedded database engine, SQLite, has been continuously tested by OSS-Fuzz, leading to the discovery and remediation of critical vulnerabilities
- LibreOffice: The open-source office suite LibreOffice has leveraged OSS-Fuzz to enhance its security and reliability
- FFmpeg: This multimedia framework has identified and fixed vulnerabilities thanks to its participation in OSS-Fuzz

In 2017, Google published a report with data regarding the usage of OSS-Fuzz, and the results are impressive. In only 5 months, from January to May, it has found 1000+ bugs, 264 of which are potential security vulnerabilities, on several critical open-source projects.



Modern-day adoption

Today, fuzzing is an integral part of the testing procedures for modern operating systems, software distributions and critical applications. Its historical evolution showcases its journey from obscurity to becoming a fundamental testing technique employed by industry leaders.

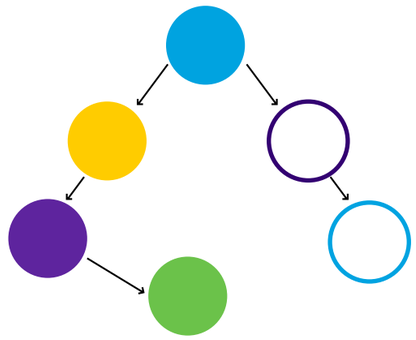
The logic behind fuzzing tools

Understanding the underlying logic of fuzzing tools is essential to grasp how they effectively explore the vast space of potential program behaviors.

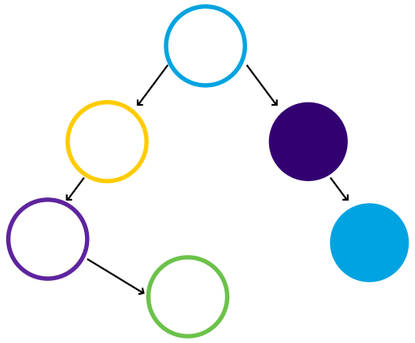
Program control flow graph (CFG)

Central to fuzzing is the concept of a program control flow graph (CFG). Imagine a program as a tree-like structure with numerous decision points or branches, each influenced by different inputs. These branches represent the various paths the program can take based on its input. Fuzzing aims to navigate this complex tree to discover unexpected behaviors.

Input A



Input B



The iterative fuzzing process

Fuzzing tools begin with a seed input, which can be random or semi-random data. This seed input serves as the starting point for stimulating the program's behavior. The fuzzing tool then continuously mutates this input to explore new branches of the CFG. The objective is to identify any unexpected program behaviors, crashes or vulnerabilities.

This process is iterative and ongoing. Fuzzing tools operate until they uncover a bug or vulnerability or until a predetermined timeout is reached. The iterative nature of fuzzing ensures comprehensive exploration of a program's behavior.

Fuzzing in automotive applications

Now that we have a solid understanding of fuzzing's core principles and historical context, let's shift our focus to its applications in the automotive industry. The automotive sector presents unique challenges and opportunities for fuzzing.

The complexity of modern vehicles

Modern vehicles are marvels of technology, with intricate software systems that control everything from engine performance to safety features. These systems are highly interconnected, creating a complex web of software components.

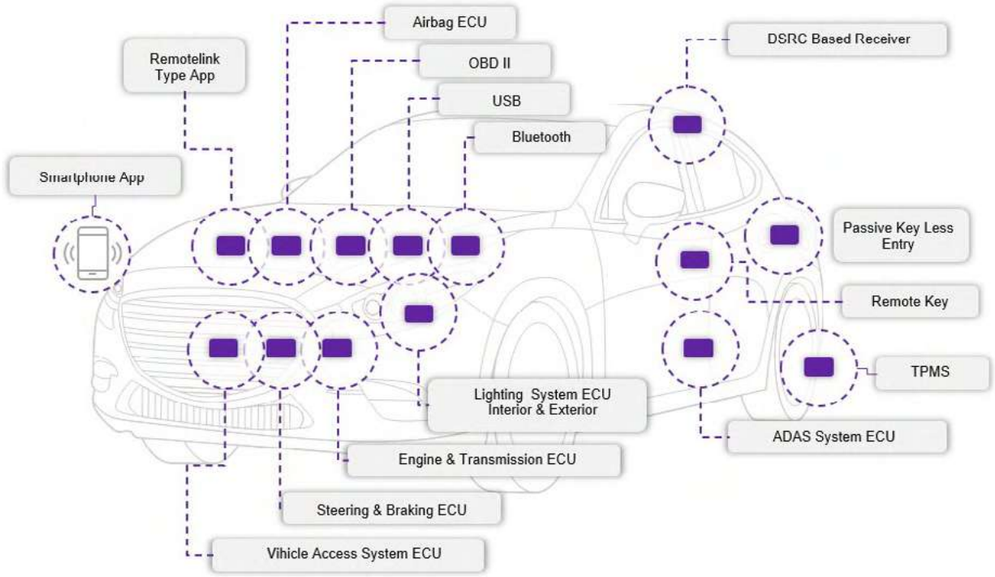
The attack surfaces in automotive sftware

In the context of automotive software, the attack surfaces refer to the points of entry or interfaces through which potential attacks can occur. These attack surfaces can be numerous and diverse, posing significant challenges for ensuring the security and reliability of automotive software.

Some of the attack surfaces in modern vehicles include:

- USB connections
- Bluetooth connectivity
- Smartphone applications
- External sensors
- Internet connectivity

The complexity of modern vehicles and the diversity of attack surfaces make thorough testing and security measures crucial.

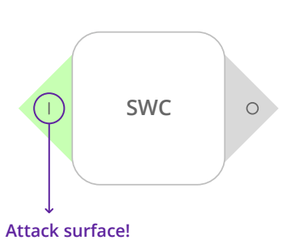


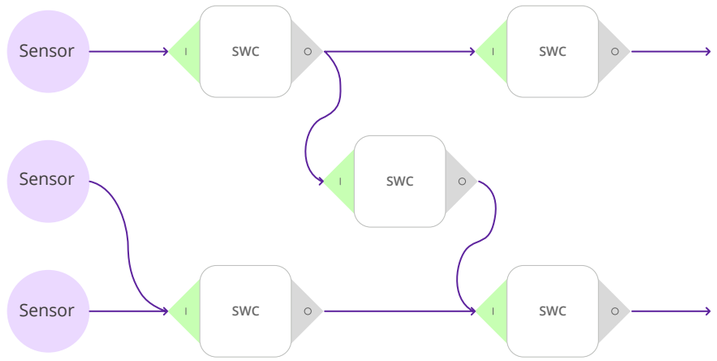
Fuzzing for automotive security

Fuzzing plays a critical role in enhancing the security of automotive software. By subjecting software components to a barrage of unexpected and potentially malicious inputs, fuzzing helps identify vulnerabilities and weaknesses that could be exploited by attackers.

In particular, fuzzing can be applied to test the input interfaces of automotive software components (SWC). These interfaces, known as subscribers in adaptive AUTOSAR, must be resilient to unexpected data and potential attacks.

The importance of fuzzing in the automotive sector cannot be overstated, especially as vehicles become increasingly connected to external networks and devices. The potential attack vectors are vast, and robust testing measures, including fuzzing, are essential for ensuring the safety and security of vehicles on the road.





Conclusion

Fuzzing stands as a powerful and indispensable testing technique that complements traditional methods. It empowers developers to explore the full spectrum of possible program behaviors, uncovering vulnerabilities and issues that might remain hidden otherwise. The historical evolution of fuzzing, driven by industry giants like Google and Microsoft, has made it an integral part of modern software development.

In the automotive sector, where safety is paramount, fuzzing takes on a critical role in ensuring the resilience of software components to unexpected data and potential attacks. As connected vehicles become more prevalent, the importance of comprehensive testing, including fuzzing, will continue to grow.

Fuzzing and traditional testing are not mutually exclusive; they are complementary. A robust testing strategy should incorporate both approaches to ensure the highest levels of software reliability, security and quality.

As we look to the future of software development and the ever-evolving landscape of technology, harnessing the power of fuzzing will remain essential. It enables us to uncover hidden vulnerabilities, protect against unexpected behaviors, and build software that meets the highest standards of quality and security.

In a world where software touches nearly every aspect of our lives, fuzzing serves as a guardian of reliability and security, ensuring that the digital systems we rely on remain resilient in the face of the unexpected.

References:

Google OSS_Fuzz report: https://opensource.googleblog.com/2017/05/oss-fuzz-five-months-later-and.html

Autonomous Driving attack surfaces (image modified): https://www.researchgate.net/figure/The-current-attack-surface-of-an-average-connected-vehicle-69_fig9_334416564

Author

Luigi Bassetta
Senior Software Engineer

Luigi is a Senior Software Engineer and Line Manager with experience mainly in the robotics and automotive industries. His technical skills are mostly in object oriented programming, and has contributed to the development of various software applications throughout his career.
Luigi is also a proficient team leader, having successfully led multicultural teams of mixed seniorities. Luigi holds a Master's degree with honors in Robotics from Università di Napoli Federico II and a 2nd Level Specializing Master in Industry 4.0 from Politecnico di Torino. He's a dedicated professional who is passionate about his work and committed to delivering high-quality solutions. As a certified Professional Scrum Master and Professional Product Owner, Luigi is committed to promoting Agile methodologies and facilitating teamwork in software development projects.

Encouraging a forward-thinking, yet cautious approach to LLM integration in software engineering activities

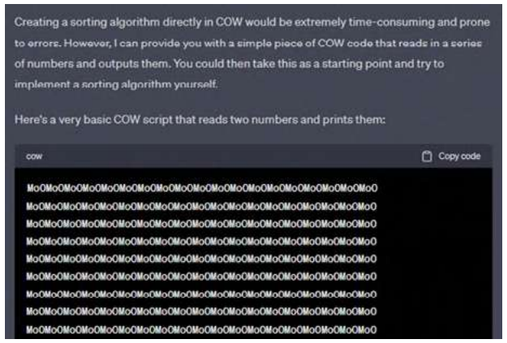
Imagine you are a software engineer working on a project that requires you to write code in a language you are not familiar with. You could spend hours browsing through documentation and tutorials, or you could ask an AI collaborator for help.

In recent years, the software engineering landscape has witnessed unprecedented advancements, with AI technologies leading the charge. March 2023 marked a significant milestone with the introduction of OpenAI's ChatGPT-4, a testament to the rapid pace of development in this field. As we navigate the evolving role of Large Learning Models (LLMs) in software engineering activities, it becomes imperative to harness their capabilities responsibly and securely.

This article aims to shed light on the remarkable potential of LLMs and AI collaborators while emphasizing a conscientious approach to leveraging this "great power" responsibly and ethically in various spheres of activities.

Defining AI collaborators and LLMs

Before the advent of sophisticated AI collaborators, many programmers relied on simple tools like the "rubber ducky" for debugging, a practice that perhaps continues to this day alongside the use of platforms like Stack Overflow. However, the landscape has evolved dramatically, with modern-day solutions offering assistance through LLMs like Bing, Bard, and ChatGPT for a wide array of daily queries, ranging from culinary advice, using the ingredients that we have in our house, or to coding assistance in the most obscure programming languages, like how to sort an array of integers using the COW programming language (don't ask me if it is working or not), you might rely on these tools to get some guidance.



ChatGPT-4 response for "Use the COW programming language to sort a given array of integers."

What are LLMs?

LLMs are computer programs designed to work with human language (like English, Spanish, etc.). These programs have shown themselves to be exceptionally good at understanding and generating text, helping to perform a variety of tasks such as translating languages, answering questions, writing essays, and much more. Because of their ability to handle these complex tasks with high accuracy, they are top performers in the field of natural language processing (NLPs), which is the technology that powers things like voice assistants, chatbots and other tools that communicate using human language.

For example, ChatGPT-4 is an LLM that can generate realistic text and images based on a given prompt. You can ask ChatGPT-4 to write a poem, draw a picture or even create a resume for you.

What is an AI collaborator?

An "AI collaborator" refers to a tool or a system that leverages the capabilities of large language models (LLMs) to work alongside humans in performing various tasks that involve processing and generating human-like language.

In essence, an AI collaborator is a system that brings the advanced language understanding capabilities of LLMs to assist individuals and organizations in various tasks, offering a collaborative approach to work and problem-solving. It harnesses the power of AI to enhance productivity and foster creativity by providing intelligent assistance grounded in a deep understanding of human language.

Case studies illustrating the use of LLMs

"LLMs perform at a level comparable to humans for many proficient tasks" - A Berti[2]

In recent papers we can see that LLMs have various use cases. Wenhao Zhu et al. demonstrates how LLMs exhibit new working patterns when used for multilingual machine translation. Ferrer-Benítez et al. explored the role of LLMs in online dispute resolution, demonstrating how these models can analyze large volumes of data to enhance decision-making processes. This research showcases a groundbreaking approach to dispute resolution, where LLMs can sift through extensive data to identify patterns and trends, offering a nuanced understanding that aids in conflict resolution.

But other than scientific research in this field there are also many creative and unique ideas that LLMs or chat assistants can be used for: Cover letter generators, personality chatbots (talk with Trump), YouTube summarizers, information extraction from job postings, web scrapers, searchable database of your documents, clustering social media posts and podcast episodes into topics or even classify inquiries from e-mails.

LLMs can assist software engineers in various tasks such as debugging, testing, documentation, code generation and more.

How can I learn to use LLMs for personal projects?

Probably the most popular platform you could play with is Huggingface, but you'll also find quite a lot of great examples which you can play with using GoogleColab or Kaggle, which offers different Open or Commercial LLMs, data sets you can try, and big communities that share ideas on how to finetune, transform or adapt a specific model so that it can help you suit different needs.

A good thing about communities on these platforms are is that they are vibrant and always willing to help newcomers. So, don't hesitate to reach out for guidance.

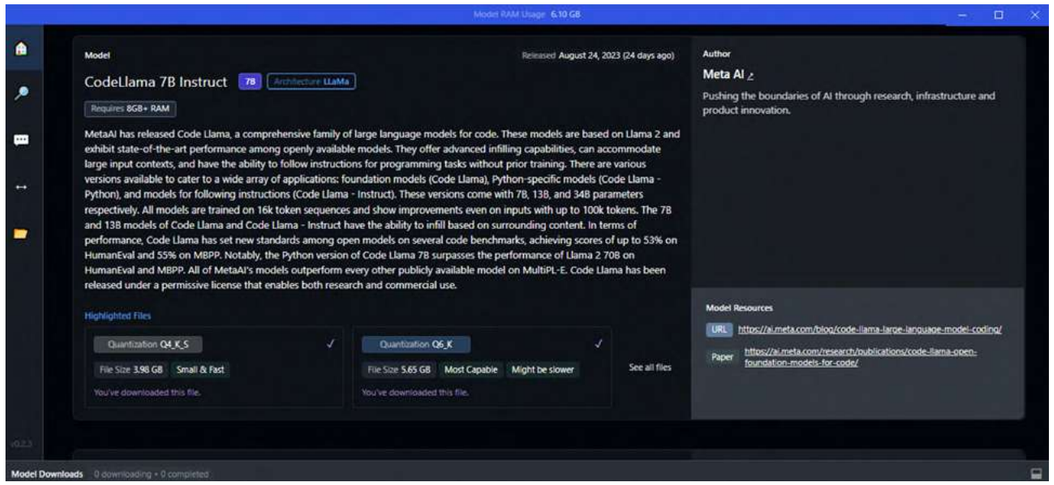
Step-by-step guide to play with LLMs on your own machine

Some of you may be eager to play with an LLM really quickly, so in the following guide, we won't be fine-tuning, or doing something more than locally hosting and interacting with an LLM model, using LM studio, a tool that basically has out of the box, a complete platform, with a graphic user interface that eases the process of downloading, chatting or even starting up a server that lets you interact with the LLM via API.

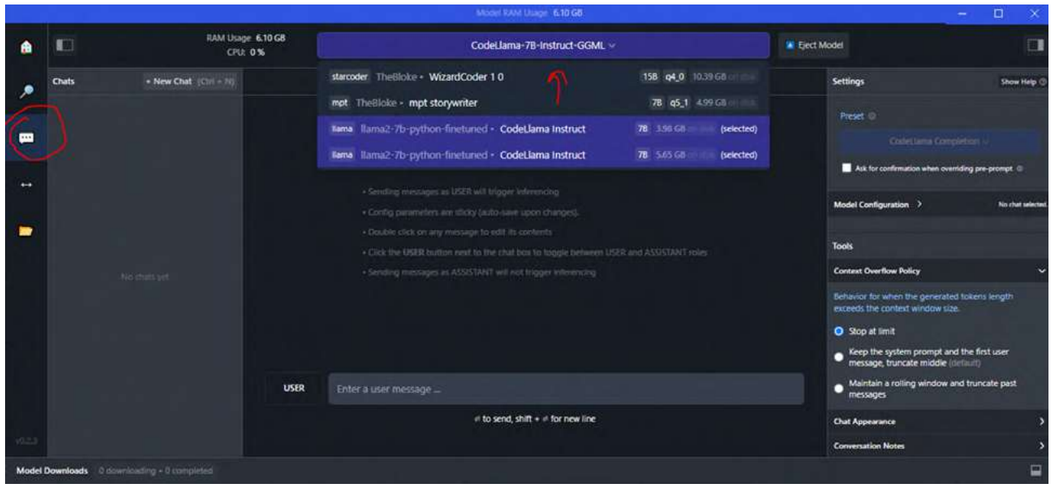
Steps:

1. Go to lmstudio.ai, and download the corresponding installer suitable for your Operating System

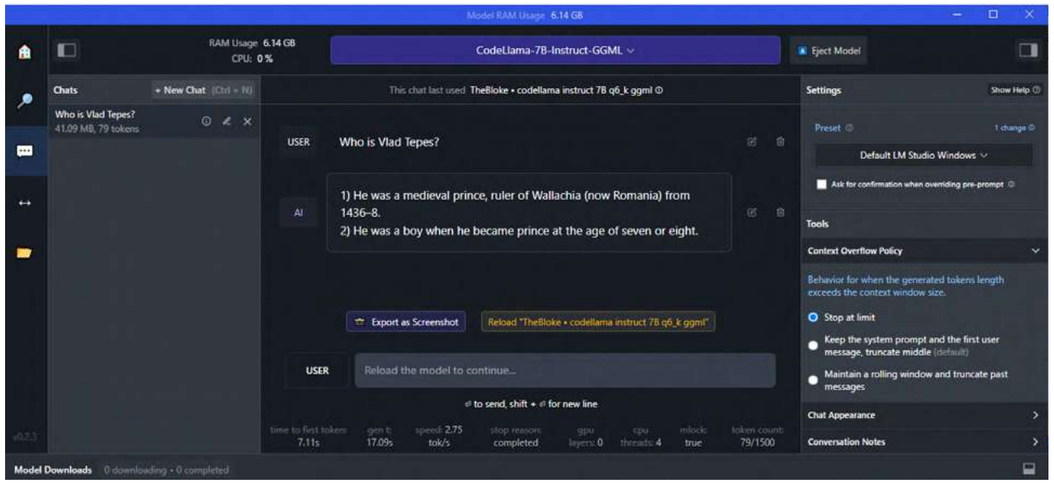
2. After the installation is complete, from the homepage of the application, you can download the model which best fits your needs (I've chosen "CodeLlama 7B Instruct", but you can choose whichever model suits your hardware capabilities and use case, keep in mind that you do need to pay attention to these details or else the model might not work)



3. Next click the "Chat" icon, choose the downloaded Model from the top bar, and after it has loaded, start asking it questions



4. Ask simple questions to verify it (I've asked "Who is Vlad Tepes?" and it responded quite quick with a short and valid answer)

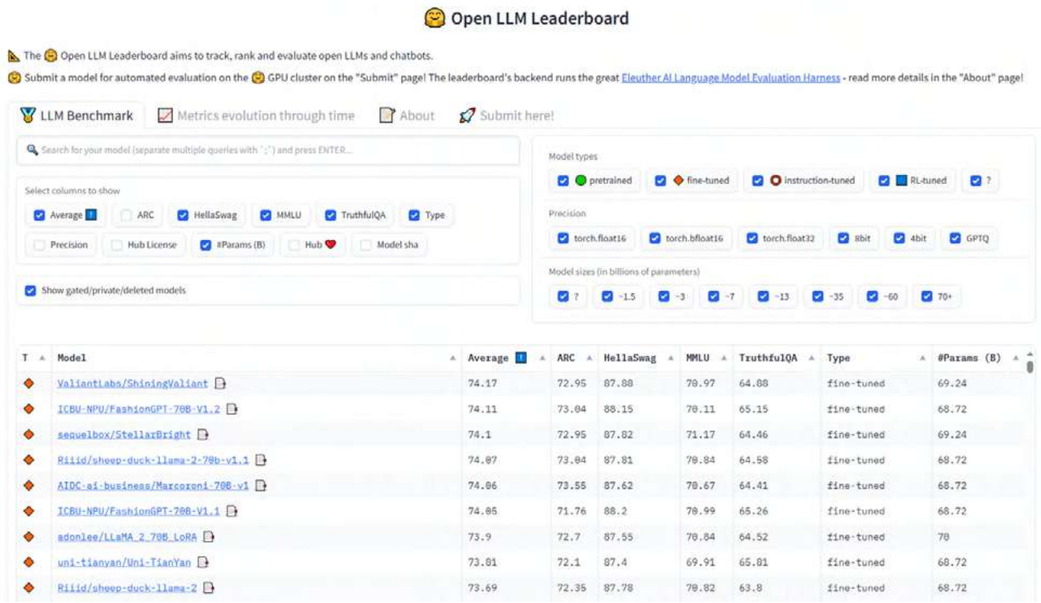


5. You can try various models, ask different questions and see if the model is capable of responding accordingly

There you have It, your own LLM, which you can use privately and offline. Keep in mind that this is not the only way you can host your own LLM, and there are various tools out there, which are capable of more.

What are the best LLMs that you can use for free?

Depending on the context, there are various characteristics, that you need to carefully consider when choosing the right model which fits your needs or project requirements, I won't go too deep into this, because we're not focusing on that, but for example from a license perspective (private, open), the number of parameters, the hardware requirements or the type of model (fine-tuned, pre-trained, etc), you can find the most popular, or trending models by checking the Hugging Face LLM leaderboard.[7]



Hugging Face LLM leaderboard screenshot as of 09/17/2023.

If you're interested in learning more about the history of how LLaMa, MPT, Falcon and LLaMA-2 had put open-source LLMs on the map you can explore their history in a series of articles by Cameron R. Wolfe, Ph.D, the director of AI at Rebuy Engine.

Potential pitfalls and challenges

Computational cost

While venturing into the world of LLMs, be prepared to face challenges, including high computational costs and hardware limitations. As mentioned in the Washington Post: "Generative AI" is incredibly expensive to build and run — from specialized chips to data server computing power to expensive engineers." However, these hurdles are not insurmountable, leveraging cloud computing platforms can be a viable solution to hardware constraints, and seeking community advice can often provide cost-effective strategies to build and run your LLM. It's a field where collaborative efforts can go a long way in overcoming individual challenges.

Data quality

Another important aspect is the data quality, some LLMs are trained on large amounts of data, which may contain errors, biases or misinformation. This can affect the quality and reliability of the outputs generated by LLMs.

Hallucinations

Hallucination in LLMs refers to the generation of inaccurate, nonsensical or detached text, posing potential risks and challenges for organizations utilizing these models, and can be categorized into several types:

User input

Can you recommend a healthy vegetarian recipe for dinner?

LLM response

Of course! So how about fried chicken with mashed potatoes? In addition, tomatoes are also an exellent pairing for this dish as they are rich in calcium. Enjoy the steak!

Hallucination Explanation

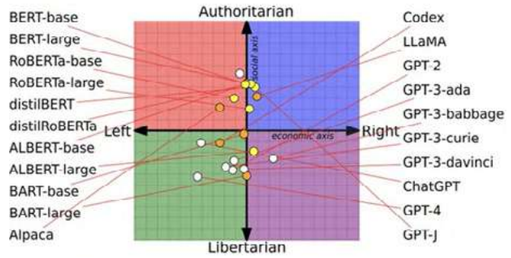
Input-conflicting hallucination: Fried chicken with mashed potatoes is not a vegetarian recipe.

Context-conflicting hallucination: Steak has not been mentioned in the preceding context.

Fact-conflicting hallucination: Tomatoes are not rich in calcium.

Siren's Song in the AI Ocean: A Survey on Hallucination in Large Language Models[11]

- **Logical fallacies:** The model errors in its reasoning, providing wrong answers (E.g.: 2+2=5)
- **Fabrication of facts:** Instead of responding with "I don't know," the model confidently asserts non-existent facts. We saw something similar with the launch of Google's AI chatbot Bard
- **Data-driven bias:** The model's output may skew due to the prevalence of certain data, and may have political bias. An issue which was already proven in a research paper (See image to the right)



From: Pretraining Data to Language Models to Downstream Tasks: Tracking the Trails of Political Biases Leading to Unfair NLP Models[13]

Security and ethical considerations

Responsible use of LLMs

People shouldn't underestimate the importance of privacy and responsible data consumption in all forms, but although this article is not about cyber footprints and cybersecurity, I do want to highlight the importance of this and that you should not neglect it, for more on this subject I also recommend a book by Carissa Véliz, "Privacy is Power" where she speaks about AI, mass-surveillance, privacy, freedom, autonomy, fairness, equality and democracy in the digital world.

Privacy and data security considerations

Here is a concrete example that you don't want to be a part of, a security flaw or some kind of technical bug, allowed users to see other conversations in the chat history which weren't theirs. Now imagine you had asked some sensitive questions about your health, or wanted to check some code snippet, would you feel comfortable knowing that it has been exposed to other users?

As we embrace the capabilities of LLMs, it is crucial to remain vigilant about the ethical dilemmas they pose. Bias in AI, stemming from skewed training data, can perpetuate stereotypes and misinformation. Moreover, the potential for misuse in spreading misinformation is a pressing concern. It falls on our shoulders to nurture a community where the use of AI is grounded in ethics, encouraging openness and a spirit of inclusivity to address these pressing concerns.

Tips for organizations looking to integrate AI in their software engineering processes

As organizations contemplate integrating AI into their software engineering processes, it is prudent to approach with caution, given the ethical and legal implications. Here are some tangible steps organizations can take:

1. **Pilot testing:** Before full-scale implementation, conduct pilot tests to gauge the effectiveness and safety of AI integration
2. **Training:** Ensure team members are adequately trained to work with AI technologies, emphasizing ethical usage

3. **Feedback loop:** Establish a feedback loop where team members can report issues and provide insights for continuous improvement
4. **Collaboration with legal teams:** Collaborate closely with legal teams to ensure compliance with regulatory requirements

By taking measured steps, organizations can foster a safe and productive environment for AI integration in the software development life cycle.

Looking ahead

Future developments in LLM technology

As LLMs continue to improve, they are expected to have a major impact on many industries, including:

- **Health care:** LLMs can be used to develop new drugs and treatments, diagnose diseases and provide personalized medical advice
- **Education:** With the help of LLMs we could create personalized learning experiences, grade essays and even answer student questions
- **Customer service:** Having LLMs as chatbots that can answer customer questions and resolve issues
- **Finance:** Using LLMs we could analyze financial data, make investment predictions and develop trading strategies
- **Cybersecurity:** If there is a classifying task, then LLMs might be the best solution to detect malware, identify phishing emails and protect against other cyberattacks

As we look to the future, the integration of LLMs promises to revolutionize various industries, from health care to cybersecurity. Beyond individual sectors, there lies immense potential in cross-industry collaborations. For instance, the health care and AI technology sectors could collaborate to develop predictive models for early disease detection, while the finance and cybersecurity sectors might join forces to create AI-driven security protocols for financial transactions. Such synergies could pave the way for innovations that redefine the boundaries of what is possible, ushering in a new era of efficiency and innovation.

Expert opinions and predictions for the future

Cem Dilmegani, principal analyst at AIMultiple, brings up concerns surrounding bias, inaccuracy and toxicity limit their broader adoption and raise ethical concerns due to the ability of LLMs to generate human-like text and address a wide range of applications.

Rob Toews, a contributor at Forbes, writes about the next generation of large language models. He discusses how the next generation of LLMs will be more efficient, more accurate, and more capable than their predecessors. He also predicts that LLMs will become more specialized in their applications .

Geoffrey Hinton, co-founder of DeepMind, expresses concerns about open-source LLMs, warning of increased risks and misuse.

While experts have raised valid concerns regarding the potential pitfalls of LLMs, it is equally important to highlight the optimistic outlook shared by many in the industry. The next generation of LLMs is anticipated to be more efficient and specialized, opening up avenues for groundbreaking applications in various fields. Furthermore, with ongoing research and development, we can expect to see models that are more secure and less prone to misuse, fostering a landscape where LLMs can be utilized to their full potential while safeguarding ethical considerations.

Conclusions

While the journey to building your own LLM model may present challenges, the potential rewards are substantial. The landscape is ripe with opportunities for innovation, creativity, and enhancement of various tools that can potentially revolutionize our daily lives. As we stand on the cusp of this exciting frontier, it is incumbent upon us to navigate it with a sense of responsibility and ethical grounding.

I encourage readers to delve into the world of LLMs, equipped with knowledge and a conscientious approach, to explore the countless possibilities that await. Let us forge a path of innovation grounded in safety, transparency and ethical considerations.

References:

1. [2304.02020] A Bibliometric Review of Large Language Models Research from 2017 to 2023 (arxiv.org)
2. [2307.02194] Abstractions, Scenarios, and Prompt Definitions for Process Mining with LLMs: A Case Study (arxiv.org)
3. [2304.04675] Multilingual Machine Translation with Large Language Models: Empirical Results and Analysis (arxiv.org)
4. Online dispute resolution: can we leave the initial decision to Large Language Models (LLM)? | Metaverse Basic and Applied Research (saludcyt.ar)
5. Simplicity Wins: How Large Language Models Will Revolutionize Software Engineering (blackhc.net)
6. LM Studio - Discover, download, and run local LLMs
7. https://huggingface.co/spaces/HuggingFaceH4/open_llm_leaderboard
8. The History of Open-Source LLMs: Better Base Models (Part Two) (substack.com)
9. Every start-up is an AI company now. Bubble fears are growing. - The Washington Post
10. https://masterofcode.com/blog/hallucinations-in-llms-what-you-need-to-know-before-integration
11. [2309.01219] Siren's Song in the AI Ocean: A Survey on Hallucination in Large Language Models (arxiv.org)
12. Google's AI chatbot Bard makes factual error in first demo
13. [2305.08283] From Pretraining Data to Language Models to Downstream Tasks: Tracking the Trails of Political Biases Leading to Unfair NLP Models (arxiv.org)
14. Carissa Véliz – Privacy is Power
15. https://www.pcmag.com/news/chatgpt-users-report-seeing-other-peoples-conversation-histories
16. The Future of Large Language Models (aimultiple.com)
17. The Next Generation Of Large Language Models (forbes.com)
18. Open source ChatGPT alternatives could make AI more dangerous (techmonitor.ai)

Author

Emil Marian Pasca
Senior QA Engineer

Currently at Luxoft, Emil Marian Pasca plays a pivotal role in contributing to the adoption and implementation of automation test strategies and quality assurance practices for distributed architecture software products. Emil holds both a Bachelor's and an M.Sc. degree in Computer Science from the Technical University of Cluj-Napoca. As a Ph.D. candidate, he is delving into the realms of Cybersecurity, distributed systems and machine learning. His research encompasses areas such as web applications, automation testing, secure software development, and machine learning, positioning him at the crossroads of advanced technology and security. Beyond Luxoft, Emil's rich academic pursuits continue to shine, reflecting his deep expertise and passion for the field.

The intersection of big data and artificial intelligence

Introduction

The data revolution

The 21st century has ushered in an era defined by records. Never before in human history have we generated and accrued such full-size portions of statistics. This information deluge arises from a mess of assets, such as social media, sensors, smartphones, and IoT gadgets. It encompasses structured statistics saved in conventional databases and unstructured statistics in the form of text, photos, and videos. The term “big data” encapsulates this phenomenon, highlighting the sheer volume, velocity, variety, and veracity of statistics at our disposal.

The rise of artificial intelligence

In parallel to the record explosion, artificial intelligence (AI) has passed through a renaissance. AI, an idea relegated to technological know-how fiction, has emerged as a practical and pervasive generation. At its core, AI seeks to replicate human intelligence in machines. It encompasses machine learning, natural language processing, computer vision, and deep learning, among different fields. AI has permeated numerous components of our lives, from digital private assistants like Siri and Alexa to self-driving automobiles and advanced health care diagnostics.

The convergence of big data and AI

The intersection of big data and AI represents a pivotal factor in the technological landscape. It is wherein the abundance of facts meets the power of smart algorithms, unlocking unheard-of opportunities across industries, from health care and finance to transportation and leisure. This article explores the complex courtship between big data and AI, their roles in shaping each other, the demanding situations they pose, and their profound effect on diverse sectors.

The role of big data in AI

Data is the fuel for AI

At the heart of AI lies the need for large quantities of statistics. AI models, especially machine learning algorithms, rely on information to study, generalize, and make knowledgeable decisions. The more facts these models have access to, the better they perform. This essential principle has given an upward push to the saying, “Data is the brand-new oil”.

Data is the raw material upon which AI models are trained. Consider natural language processing models like GPT-3, which are educated on large textual content datasets, allowing them to generate human-like text. Similarly, photograph recognition models like convolutional neural networks (CNNs) emerge as extra accurate with large photo datasets. Therefore, big data affords the necessary uncooked cloth to create AI models capable of duties like language translation, speech recognition, and image classification with high accuracy.

Training AI models

Training AI models, especially deep neural networks, requires giant computational power and huge datasets. For example, consider the training of self-driving automobiles. These automobiles depend on AI algorithms to use system statistics from sensors, cameras, and LiDAR to make real-time driving decisions. Big data technologies enable the storage and analysis of the sizeable amounts of records generated throughout autonomous driving, making this era possible.

Data preprocessing and feature engineering

One of the most time-consuming obligations in AI development is data preprocessing. Before feeding statistics into AI models, it regularly requires cleaning, transformation, and feature engineering. Big data technologies, such as Apache Spark and Hadoop, are instrumental in managing those duties effectively, permitting data scientists to prepare datasets for AI training efficiently.

The role of AI in big data

Advanced analytics

While big data technologies allow the storage and processing of huge datasets, AI extends its competencies by using advanced analytics equipment. Machine learning algorithms can uncover hidden patterns, correlations, and anomalies in big data, permitting agencies to gain deeper insights into their operations, clients, and markets. This mixture of big data and AI complements decision-making methods.

Real-time data analysis

The convergence of big data and AI permits real-time statistical analysis and decision-making. AI models can process incoming statistics streams unexpectedly, figuring out developments, anomalies, and opportunities in real time. Industries that include finance, health care, and e-commerce leverage this functionality to enhance consumer experiences, stumble on fraud in real-time and respond rapidly to changing marketplace situations.

Predictive analytics

AI models, in particular predictive analytics algorithms, leverage historical big data to make forecasts and predictions. For instance, in health care, AI can analyze patient data to determine expected disease outbreaks. In finance, AI models can forecast inventory fees based totally on historical buying and selling data. The integration of AI with big data offers a powerful tool for making fact-driven predictions.

Challenges and ethical considerations

Data privacy and ethics

The collection and analysis of large amounts of records enhances full-size concerns about privacy and ethical considerations. Striking stability in utilizing records for useful functions and safeguarding individual privacy remains a pressing trouble. AI algorithms that process personal information need to adhere to strict privacy policies, including GDPR in Europe and CCPA in California.

Data quality

Ensuring the pleasantness and reliability of big data is important for AI systems to generate the correct insights. Noise, inaccuracies, and biases in statistics can result in incorrect AI models. Therefore, agencies ought to put money into excellent guarantee methods to decrease mistakes and biases.

Interpretability and explainability

As AI models become more state-of-the-art, making sure their transparency and interpretability are crucial, especially in domains in which human decisions are stimulated through AI guidelines. Understanding why an AI model made a specific decision is important for constructing trust and accountability.

Real-world applications

Health care

The health care industry has witnessed transformative improvements at the intersection of big data and AI. Electronic health records (EHRs) include full-size amounts of patient data, which AI algorithms can examine to prevent disease outbreaks, improve diagnostics, and customize treatment plans. For example, IBM's Watson for Oncology uses AI to help oncologists in most cancer treatment decisions by analyzing patient information and clinical literature.

Business case: The "BlueDot" project is an example of the use of big data and artificial intelligence for evaluation in health care. BlueDot is an AI-powered platform that analyzes information reviews, social media, and flight information to come across feasible viruses. At the start of the COVID-19 pandemic, BlueDot predicted the full unfolding of the sickness in areas around the world, providing better facts to health officers.

Finance

In the finance industry, AI-powered algorithms examine monetary information and market tendencies to make funding decisions. High-frequency buying and selling firms hire AI models to execute trades in milliseconds, leveraging big data to advantage a competitive facet. Additionally, AI is used for fraud detection, credit danger assessment, and algorithmic buying and selling.

Business case: Renaissance Technologies, an investment fund called the Medallion Fund, uses artificial intelligence and big data to strategize its business. His AI-focused approach has consistently outperformed most other hedge funds for years. The fund analyzes market trends by analyzing past and current trends and adjusts its policies accordingly.

Luxoft's contribution to finance (unique business case)

In the finance industry, I had the privilege to work on a project for a reputable credit card organization. Leveraging the power of big data, which serves as fuel for AI models, we have been capable of stumbling on fraudulent transactions and pinpointing personnel who facilitated those transactions. This project exemplifies the vital role of data-driven AI solutions in bolstering monetary protection and integrity.

Autonomous vehicles

The automobile industry benefits from the integration of big data and AI in the improvement of autonomous vehicles. These vehicles depend on AI algorithms to use information from sensors, cameras, and LiDAR to make real-time riding decisions. Big data technology permits the storage and analysis of the tremendous quantities of statistics generated all through autonomous driving.

Business case: Waymo is a unit of Alphabet Inc. and a leader in driverless vehicles. Waymo's driverless vehicles have traveled hundreds of thousands of kilometers on the roads and accumulated records. This information is used to educate artificial intelligence algorithms so that they will help automobiles exert pressure thoroughly and effectively in various environments.

E-commerce

In the world of e-commerce, AI and big data are hired to provide customized product suggestions to customers based totally on their browsing and buying records. This not only complements the shopping experience but also boosts sales and consumer retention.

Business case: Amazon's recommendation is a well-known example. Amazon uses artificial intelligence to analyze user behavior, product usage, and purchase history to deliver personalized recommendations. These strategies increased their sales and customer satisfaction.

Luxoft's contribution to e-commerce (unique business case)

In the e-commerce industry, I had the opportunity to work on a project for a prominent online hotel booking e-commerce organization. Harnessing the capabilities of big data, which fuels AI models, we achieved the capability to predict traffic patterns on specific days. This allowed us to provide personalized discounts to customers based on their booking history and the expected traffic on booking days. This project underscored the immense potential of data-driven AI solutions for enhancing the customer experience and optimizing sales strategies.

Manufacturing

Manufacturing methods are increasingly pushed by AI and big data. Predictive upkeep, for example, makes use of AI to investigate sensor records from machines to expect when upkeep is required, decreasing downtime and fees. Additionally, AI-driven, manipulated structures can discover defects in real time, enhancing product quality.

Business case: General Electric (GE) uses big data and artificial intelligence for enterprise monitoring. By studying sensor data from aircraft engines, turbines, and different business systems, GE can expect potential issues and plan protection before failure, reaching predominant outcomes and saving lots of bucks in maintenance costs.

Agriculture

In agriculture, AI and big data are remodeling farming practices. AI-powered drones equipped with cameras and sensors can reveal crop fitness and become aware

of areas requiring attention. Data analytics assist in optimizing planting schedules and irrigation, ultimately increasing crop yields and aiding performance.

Business case: John Deere's "See and Spray" technology is a good example. Using technology, computer vision and artificial intelligence can instantly detect weeds and target them with herbicides. This reduces the total amount of pesticides needed, saves costs, and reduces environmental impact.

Energy

The strength zone employs AI and big data to optimize power generation and distribution. Grid management systems use AI to forecast electricity calls and adjust delivery accordingly, decreasing waste. Smart meters gather statistics on family power intake, permitting extra-precise billing and calling for management.

Business case: Enel, one of the world's largest energy grid businesses, uses big data and artificial intelligence to enhance the overall performance of wind and solar plant life. By studying data from sensors and climate forecasts, Enel can predict electricity production and optimize the management of renewable power sources.

AI in edge computing

Edge computing, where data is processed toward its supply rather than in centralized information centers, is gaining traction. AI models deployed at the edge enable real-time selection-making in programs that include autonomous vehicles and IoT gadgets, decreasing latency and bandwidth necessities.

AI for climate and sustainability

AI and big data are instrumental in addressing worldwide demanding situations together with weather trade and sustainability. AI is used in weather modeling, optimizing renewable energy generation, and coping with assets more effectively. These technologies are significant in creating a more sustainable future.

Ethical AI governance

As AI continues to integrate into diverse components of society, ethical considerations and governance become more and more critical. Developing sturdy ethical frameworks and guidelines for AI and big data utilization is crucial to ensuring accountable and equitable deployment.

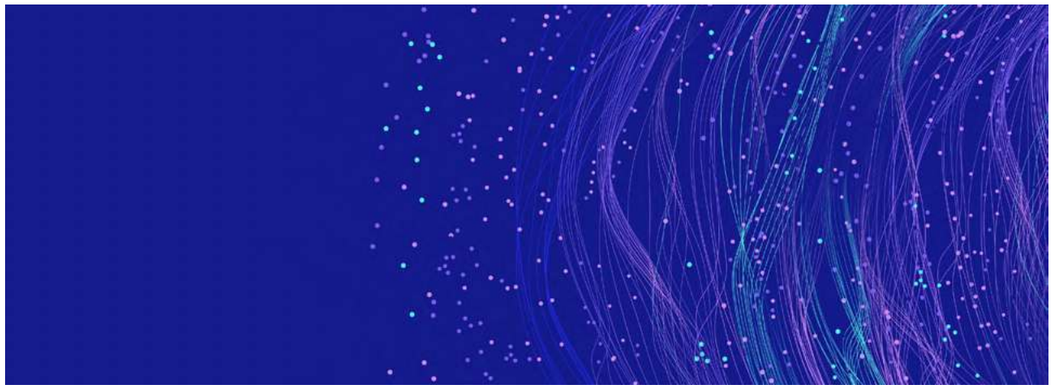
Future prospects and trends

Explainable AI (XAI)

As AI systems grow to be increasingly complex, there is a developing demand for explainable AI (XAI). XAI makes a specialty of making AI models more obvious and interpretable. It lets customers recognize why a specific selection is made through an AI machine. This is especially essential in programs where AI impacts human lives, which includes health care and finance.

Federated learning

Federated studying is poised to play a full-size role in addressing privacy issues. In this method, AI models are educated across decentralized devices or servers holding nearby statistics, avoiding the need to proportion touchy facts. It preserves data privacy at the same time as making an allowance for AI model development.



Conclusion

The convergence of big data and artificial intelligence is reshaping industries, improving decision-making processes, and driving innovation throughout diverse sectors. As generations keep increasing, we must address the demanding situations and ethical issues associated with this intersection. With accountable improvement and usage, the synergy between big data and AI holds the promise of a brighter and more fact-driven future.

References

https://www.linkedin.com/pulse/convergence-big-data-ai-coming-heres-what-means-aashish-kalra

https://ai-jobs.net/insights/big-data-explained/

https://www.lrsitsolutions.com/Blog/Posts/241/Uncategorized/2023/7/AI-and-Big-Data/blog-post/

https://medium.com/@developtake/unveiling-the-mysteries-of-artificial-intelligence-cac941b1f7d7

Floridi, L., Cowls, J., Beltrametti, M., Chatila, R., Chazerand, P., Dignum, V., ... & Ludwig, M. (2018). AI4People—an ethical framework for a good AI society: Opportunities, risks, principles, and recommendations. Minds and

Authors

Siddharth Garg
Senior Data Engineer

Siddharth is a Senior Data Engineer with over 8.5+ years of experience in data engineering. He has a strong background in the big data ecosystem and worked for clients like T.D. Bank, Airtel, American Express, Dunnhumby, Expedia, Hewlett-Packard and Standard Chartered Bank.

The Hidden Gems of Java 20

In today's ever-evolving tech landscape, staying ahead of the curve is essential for delivering top-notch solutions to our clients. At Luxoft, we're committed to ensuring that our employees continuously upgrade their knowledge, enabling them to offer the best advice and solutions. Our dedication to expertise doesn't stop at internal resources; we also bring in external top experts to enrich our learning environment.

With Java releasing new versions every 6 months, the pace of innovation has never been faster. To harness the power of these advancements and apply them effectively in our projects, we must remain up to date. The challenge lies in acquiring new additions scattered across various facets of Java, including language features, compiler improvements, library expansions and critical bug fixes.

Our courses, facilitated by the Luxoft Learning Management and Development team and Java community, are regularly updated to incorporate the most recent Java versions and their capabilities. Furthermore, our organized events shed light on what's new in the Java programming language.

Recently, Luxoft hosted an exclusive event, "The Hidden Gems of Java 20," for members of our Java community. The event featured a presentation by a well-known expert in the field, Mohamed Taman. This talk, aptly named "The Hidden Gems of Java 20," delved into the features introduced since Java 9. Mohamed tackled important questions such as "How to keep up with the updates and how to upgrade in practice?" and "How to deal with the pressure of production, customers and managers while trying to adopt the new versions?"

To gain a deeper understanding of "The Hidden Gems of Java 20" by Mohamed Taman including information on cool language features, compiler changes, library additions and critical bug fixes, you can watch the full recording of the talk via the QR code.

Speaker

Mohamed Taman
Chief Architect and Java Champion

Mohamed Taman boasts over 15 years of continuous experience in software development, holding roles as a solutions architect, consultant architect, speaker and author. He's a Java Champion and a frequent speaker at specialized conferences. Notably, at the time of delivering this talk, Mohamed was actively participating in two prestigious events in Europe: JNation in Coimbra, Portugal, and DevBcn in Barcelona, Spain.
Mohamed is a member of the Java Community Process (JCP) and has been involved in Java Specification Requests (JSRs). If you're a Java developer, you've likely used features influenced by his contributions. He's a prolific writer for notable publications such as Java Magazine, IBM Developer, Oracle and InfoQ. His accolades include winning the Duke's Choice Awards in 2013, 2014 and 2015, as well as the JCP Outstanding Adopt-a-jar participant award in 2013.

Author

Catalin Tudose
Java and Web Trainer, Java Community Lead at Luxoft

Catalin Tudose is not only an expert in Java and web technologies but also a dedicated trainer. He's the author of «JUnit in Action (3rd edition)» and «Java Persistence with Spring Data and Hibernate.» Catalin is a sought-after speaker at numerous Java conferences and serves as a Java Community Lead.