IBM Software Engineering for sustainability

IBM

Contents

Introduction

IBM believes in progress and in applying intelligence, reason, and science to improve the human condition, society, and business, now and in the future.

At IBM, we define sustainability as bringing prosperity to all people, communities, and the planet. Sustainability includes reducing our impact on limited resources.

As the world becomes more digital, IBM Software products have a growing impact on the environment and society at large. Software engineering for sustainability requires a diverse and empowered team. As a team, we design, develop, and deliver experiences to be easy to use, inclusive, and equitable to all people, while reducing the resources that are needed to develop and operate them. IBM follows best practices, processes, and frameworks to ensure that we take both a human-centric and planet-centric approach to software engineering.

IBM Software engineering principles

The guiding principles of IBM's engineering practices encompass conservation, portability, flexibility, measurability, and sustainability as a resource constraint.

Conservation
- Reduce energy and carbon usage both statically and dynamically.
- Reduce the footprint of computing, memory, networking, and storage that is consumed by the software.
- Simplify the topology to reduce data transportation across both hardware devices and software stacks.
- Design for efficiency by balancing resource constraints, such as recognizing static power draws and clock speeds, because idle hardware still consumes energy. Additionally, high-performance processors use more energy even when they are idle, and as CPU utilization gets close to 100%, the useful computation for the software is decreased.

Portability
- Design software to run in a data center, region, or location with fewer carbon emissions.

Flexibility
- Design software to run on older and heterogeneous hardware and platforms to reduce embedded carbon and to increase accessibility.

Sustainability as a resource constraint
- Energy consumption often increases exponentially to accomplish the last stretch of reaching software performance goals, such as getting from 99.99% to 99.999% availability of the software service. Do not over-engineer performance goals.

Measurability
- It is critical to understand what constitutes sustainability success through defined criteria and then measure it with associated systems.
- Sustainability success criteria must consider workload, resource usage, carbon footprint, and other criteria.

The influence of Enterprise Design Thinking on the engineering process cannot be overstated. Design and engineering processes are both intertwined and iterative, especially in today's agile world. The IBM Design for sustainability practices are published separately.

When IBM delivers digital experiences, our goal is to optimize performance, speed, and responsiveness for both the user and the system. We believe that Earth's resources are limited, and that conservation of resources is always preferred.

Environmental impact practices

How IBM designs software products today has a growing impact on the carbon footprint in data centers across the world. New techniques in large-scale computing and data use a significant amount of energy.

Software development

The typical software development process has changed drastically over the years, from longer delivery cycles of monolithic products to a continuous integration and continuous delivery (CICD) approach that is focused on finer-grained services, called microservices. However, the combination of these services still requires a level of architecture specification for their integrated operation.

Software architecture

Well-architected software products meet design requirements while minimizing the overall infrastructure usage that corresponds to energy and carbon footprint. The software architecture process prioritizes the tradeoffs between sustainability and optimization requirements, which can range from speed-to-market, cost of engineering, user experience, availability, social impact, and sustainability. These requirements often conflict with each other. For example, you can improve availability through redundancy, but that can negatively impact energy usage. Also, you can improve speed-to-market but sacrifice usability or sustainability. Some conflicts can be improved or resolved by increasing the cost of engineering.

Considering the carbon footprint of running the software is a crucial part of designing and architecting it for environmental sustainability. Software architecture must be guided by a few key principles that affect carbon footprint.

↑110 TW

The Cambridge Center for Alternative Finance estimated that bitcoin alone consumes around 110 terawatt hours per year, about the same energy usage as the entire country of Sweden.

Reduce carbon footprint statically and dynamically
Electricity is a scope 1 emission source, and hardware includes a significant embedded carbon footprint. Practices to reduce the footprint of CPU, memory, network, and storage that is consumed by software are important principles of software design.

Transporting data across the network also causes carbon emissions. Networks with longer distances and a complicated topology result in more data movement across both hardware devices and software stacks. Practices to reduce or eliminate the data movement between software components and from servers to clients, including the following practices, reduce the carbon footprint.

Prolong the life of hardware
A fixed amount of carbon is embedded in hardware, which is amortized over its expected life span. Practices that allow software to run on older and heterogeneous hardware and platforms prolongs the life of that hardware.

Adjust for carbon intensity
Carbon intensity is a measure of how efficient an energy source is. It is a best practive to design software so that the customer can easily move their software to run on a data center, region, or location with a lower carbon intensity score to reduce overall carbon emissions.

Adjust for energy efficiency
Due to static power draws and clock speeds, the energy efficiency of a machine is not constant. When the machine is idle, the energy consumption is still not zero. Therefore, high-performance processors use more energy even when they are idle, and as CPU utilization gets close to 100%, the useful computation for the software is decreased. It is important to design software so that it can be run most efficiently by balancing these resource constraints.

Include carbon emission as another resource constraint
Carbon emission must be treated as an additional resource constraint for the software. When carbon emission is higher due to changed carbon intensity or when the upper limit of the carbon allowance is reached, demand-shaping methods can be used to control the electricity consumption by trading off with other resource constraints, such as performance. Energy consumption often increases exponentially to accomplish the last stretch of reaching software performance goals, such as getting from 99.99% to 99.999% availability of the software service. Therefore, do not over-engineer performance or availability goals.

These architecture practices are recommended by IBM for environmental sustainability.

Evaluate microservices and containers
Evaluating both microservices and containers allows you to focus on finer-grained services that are easily adaptable to the deployment environment. This adaptability lets you select an environment based on its sustainability factors, such as a cloud provider with a data center that runs on renewable energy sources. This fine-grained approach allows optimization based on the service characteristics, instead of taking an all-or-nothing approach in a monolithic product with multiple services. In practice, IBM has found that this approach helps to avoid overprovisioning by simplifying the ability to start and stop services, as needed.

However, the drawback to using this approach is duplication within the container, specifically the operating system and other dependent services, which are bundled in each container to simplify version management and configuration. As a result, good analysis is important to find the right balance of container content that minimizes local duplication while still optimizing global resource usage.

Use dynamic scaling
Horizontal scaling of services has progressed to account for billions of requests each day. However, it is critical to also enable scaling down by using resources only when necessary to reduce wasted energy. With the use of more complex AI models and analytic services that run periodically, the ability to shut down those processes when they are not active can make a considerable difference in environmental sustainability.

Modernize software
Most software that is in use today evolved through several iterations with limitations on its architecture. Knowing the right time to modernize software architecture is an important part of the engineering process, since it is often costly. Significant resource savings can be achieved by refactoring mature products by using the microservices and container approach.

For example, the deployment of development and test environments can be sped up and automated, resulting in less overprovisioning.

Maintain balance
Too often products have requirements that are either over-engineered or under-engineered, possibly due to ambiguous priorities or to chasing the latest technology trends. Many technology choices are now available because the industry is more specialized. For example, dozens of different database technologies are available with specific design points and different resource usage characteristics. Previously, as the costs for computing, memory, and storage continued to decline, concern for the associated energy use and carbon footprint was not generally a priority. Now, the goal to reduce energy usage must be a higher priority and it is important to be specific about priorities.

For example, you might be able to accept no redundancy or continuous availability for some features, or perhaps you can accept slightly less performance by using one database technology instead of two.

Software coding

After architectural decisions are made, you must focus on actual coding practices. IBM often leverages an integrated development environment, which combines the code editor with syntax checking, debugging features, search, and reusable source libraries. Environmental factors at IBM include the overall time to complete new features or to update existing features, which means reducing the time that is needed to write the code correctly. The time spent building the code and its associated test automation is more important for resource usage.

Use continuous integration and continuous delivery (CICD) pipeline automation
Fully automated source code management systems, builds, and deployment and test environments can help to reduce energy use. Typically, automation helps to avoid overprovisioned environments that are overused. An automated CICD pipeline reduces the number of manual processes that are needed and results in shorter development and test times, while still supporting the same amount of functionality.

An automated CICD pipeline can increase the quality of the product and further reduce the energy that is used to track and fix issues. A machine that is actively working for the automation process is more efficient in energy consumption compared to machines that are handling user interactions.

Modern CICD pipelines that use a containerized platform, such as a pipeline on Kubernetes, an open-source orchestration tool for containers, increase the consistency of testing and development environments, reduce the time and energy that is needed to create and maintain them, and save resources when the workload is run. A test environment can be instantiated in minutes, and then be removed when it is no longer needed, to reduce overall resource requirements for automated development and testing.

Develop and test by using container images
Container images can be used to quickly onboard a new engineer by providing a complete test environment that includes prerequisites for the component. These container images can eliminate many steps that an engineer typically needs to complete before they can concentrate on development and debugging, such as setting up a database with test data or configuring an external system. The system can be instantiated quickly, which drastically reduces the unnecessary consumption of energy in the engineering process. IBM has embraced the use of Red Hat OpenShift and its opinionated approach to managing container-based services.

Use Red Hat OpenShift Kubernetes Operator
Kubernetes Operator is a model that encourages automation by making it a built-in feature of the product. It is a method of sharing automation knowledge from product engineering to the information technology administrators.

Kubernetes Operator handles the installation, configuration, upgrades, automatic scaling, and automatically runs the product when the operator maturity increases. Kubernetes Operator reduces user errors, increases the resource usage efficiency, and takes full advantage of the container platform to adjust the required resources based on the workload. Kubernetes Operator can also be leveraged to build sustainability consideration factors into the automation, such as reallocating the containers into data centers with less carbon intensity. Moving through the Operator Capabilities maturity model produced immediate benefits at IBM.

Reuse components
Leveraging API economy is another way to modernize and reduce energy use. Within IBM, we use common services and common components to speed up engineering and reduce development costs by reusing and using InnerSource components and frameworks. IBM contributes to and uses open source software that helps the overall sustainability of the software business. For more information, see IBM's approach to open technology.

Software operations

Software service operation causes a large environmental impact and can lead to a great reduction in energy use and carbon footprint, and therefore is where the greatest savings can occur. Continuous evaluation of software operations is an excellent opportunity for continuous improvement in sustainability

Operations monitoring and actionable insights

Operations monitoring starts with the defined workload and capacity benchmarks, and then focuses on monitoring services and continuously improving their efficiency through actionable insights.

Define service-level objectives (SLO)
To decide what to track and how to optimize, each service must have clearly defined objectives. These objectives must be defined in terms of their business value, including the environmental goals for the organization. Defining sustainability objectives leads to an understanding of which indicators are required to monitor the system.

For example, to track carbon footprint, the amount of computing resources that are used by services must be reported, linked to different hardware energy usage rates, and then linked to the carbon intensity of the energy provider for the data center location. Initially, you might be able to track only the computing usage and then add the links and data over time. The SLO definition must clearly describe goals so that a roadmap can be defined to achieve those goals.

Define workload parameters
A workload definition is a parameter set that specifies the inputs to the product services. These parameters are often expressed as a rate, current count, or maximum count, such as the number of transactions per hour, total assets, or number of anticipated concurrent users. Each service must determine which parameters determine its scalability, much like a time-complexity analysis of algorithms, and these parameters can be used for capacity planning, service-level objectives, and value-based pricing models.

IBM uses these definitions for simulations, analysis, predictions, and cost models, and connects them to environmental factors to determine how to reduce our carbon footprint.

Use a capacity planning baseline
To reduce carbon footprint, you must first understand the baseline for normal usage. Then, you need to monitor and detect changes in that baseline to alert the operator when they might need to act. As specific improvements are made, this monitoring can help measure impact. The defined workload parameters can drive simulations that help to establish this baseline and optimize energy use. For many years, IBM has created publicly available Capacity Planning Guide and Reference materials that can assist with capacity planning.

Use monitoring and automation
For basic production operations, it is important to monitor services for health, availability, performance, and security. Monitoring is also important to continuously track the environmental impact of the services. As operations mature, you can automate actions based on this monitoring, such as scaling up and down dynamically, or adjusting the location of services to more energy-efficient environments.

A good example of monitoring and automation is the IBM Weather services, which are delivered through IBM websites and mobile applications. During major weather events, such as hurricanes or tornadoes, the number of requests increase dramatically during the event and then drop back into a normal range. These services scale up dynamically, as necessary, avoiding costly overprovisioning when they are not needed.

Use a workload analysis tool
A workload analysis tool can provide several benefits, starting with a basic understanding of the workload and the interactions between services in actual scenarios. Understanding actual service use and corresponding recommendations enables you to optimize your infrastructure.

For example, one IBM data center used the IBM Turbonomic analysis tool to avoid purchasing hundreds of thousands of dollars in new equipment by rebalancing and optimizing the current workloads to make room for new services.

Develop workload prediction models
Using the data from the workload definition and analysis tools along with regularly monitored time series data, you can develop workload prediction models that help anticipate and dynamically adjust infrastructure. Workload prediction models are especially important in products with variability due to seasonal or event-based loads, such as large-scale weather events, retail specials, or demand-generation activities. These insights correspond to the Red Hat OpenShift Kubernetes Operator maturity model Level 4 and 5, in which the service makes operational adjustments on its own.

IBM's experience is that we can conservatively improve overall consumption by up to 50% by using time series models such as SARIMAX and generalized linear models.

Workload and data center factors

As your operations become more sophisticated, you can adjust your workload schedule and placement based on environmental factors. The goal is to spread the mix of workloads to minimize resource usage while still meeting service-level objectives.

This optimization requires more environmental data at a faster rate, such as the current carbon intensity score of the energy provider where the service is operating. When the score is high, the energy consumption is creating more carbon emissions, which means that the service should be moved to a data center with a lower carbon intensity score.

Identify high-consumption services
To identify high-consumption services, first measure and identify the services that use the greatest amount of energy, usually in the form of compute units or cores. Determine whether these services can be dynamically scheduled in different data center locations.

Examples of high-consumption services that might be candidates to move to a low carbon intensity score schedule are bitcoin mining, AI model training, large data aggregation, and highly complex algorithms that process extensive data. This schedule might include the time of day that the service runs in addition to the location.

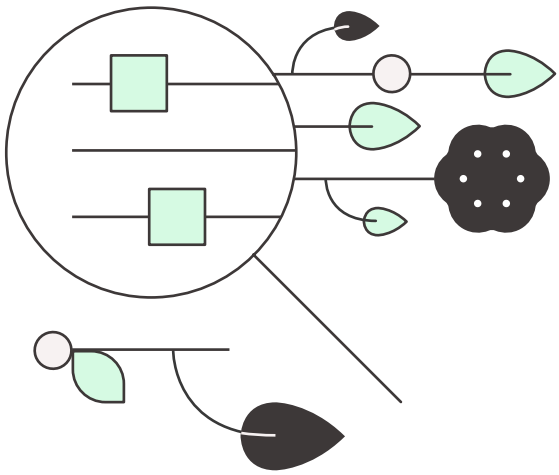
Obtain carbon intensity and water usage metrics
For on-premises deployments, carbon intensity and water usage metrics might be available from the cloud provider or directly from energy or water providers. While energy usage is always a focus, water availability is a global issue, and data centers built in arid climates are accelerating the problem. Data centers that operate based on hydroelectric excess capacity are good targets for high-consumption services.

Push for more frequent metrics
If metrics are available each hour instead of each day or week, decisions can be made faster, based on the demand at that moment. This demand-shifting allows for continuous optimization and greater carbon footprint reductions.

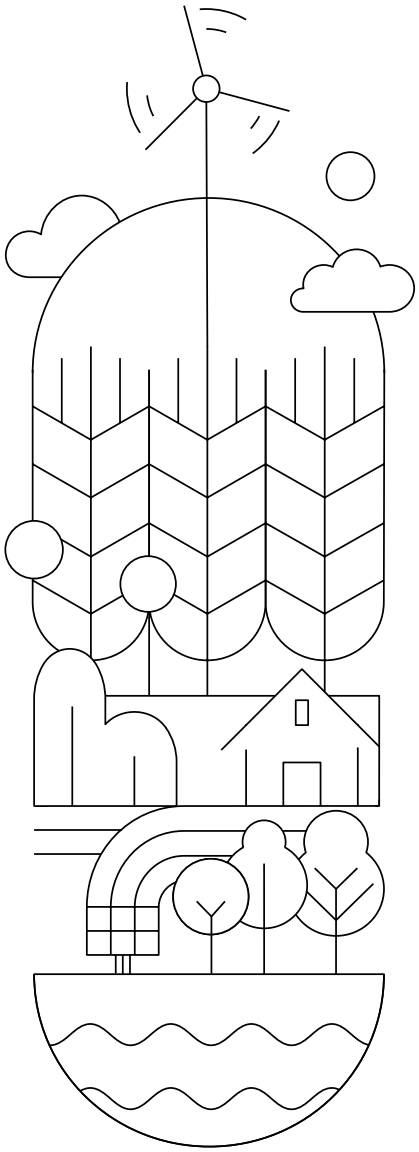
Frequent metrics are a good addition to service-level optimization strategies.

Develop a feedback loop
All of the gathered data must be fed back into the software engineering processes so that high usage components can be designed for better isolation and potential changes in their approach to sustainability.

Deprovision unused services
Services are often kept active even when they are rarely or never used. It is important to create automation that deprovisions services based on certain rules or predictions and then reprovisions them based on future demand.



Equitable impact practices



Equitable impact practices include ensuring that software produces fair results, is accessible, and includes support for different national languages. For more information on equitable impact practices, see IBM Equitable Impact.

Software results

To be equitable, software must produce fair results. The software industry has undergone fundamental changes in the last decade. The recent advances in machine learning and mathematical techniques have created enormous potential for productivity, but also have enormous potential for unintended consequences. The results that are produced by algorithms, workflows, rule engines, and AI models can have drastic, unintended consequences on a global scale. These consequences have existed on a smaller scale for years, such as in loan processing rules, financial calculations, and insurance risk assessments.

Software accessibility

IBM is a leader in making software accessible. Software engineering processes must use multiple tools and techniques to account for diverse user abilities and user interaction styles, and to ensure that the software is accessible to all users. Accessible experiences increase ease of use, task efficiency, system responsiveness, and user satisfaction. Simplified experiences are more efficient and inclusive because the content works for the largest possible number of people.

IBM also provides an Equal Access Toolkit that provides tools and guidance to help others build inclusive and accessible experiences.

Software language support

As the software industry has matured to support different languages across the world, software engineering has evolved user interface and programming language support for different character strings that are pulled from a single stored location. The set of stored character strings can then be translated into the necessary languages, which increases the availability of technology to more people. Ensuring wide access to technology is even more important when building services, such as government services, with a social benefit for disadvantaged groups.

Ethical impact practices

The impact of sustainable software engineering on governance is summarized by the IBM Business Conduct Guidelines, which is the IBM code of business conduct and ethics. All IBM employees are trained and certified on these practices annually and these guidelines are continually emphasized so they are an intrinsic part of the IBM culture. Software privacy is another key practice with ethical impact.

Software business conduct guidelines

The IBM Business Conduct Guidelines have been part of the IBM culture for decades and remind every employee about IBM's commitment to integrity, compliance, and ethical behavior in global business dealings. Also, these guidelines describe how IBM protects employees, assets, and intellectual property. IBM respects every individual, intellectual property, legal obligations, and commits to being honest, accurate, and complete. Additionally, IBM employees separate personal interests from business responsibilities.

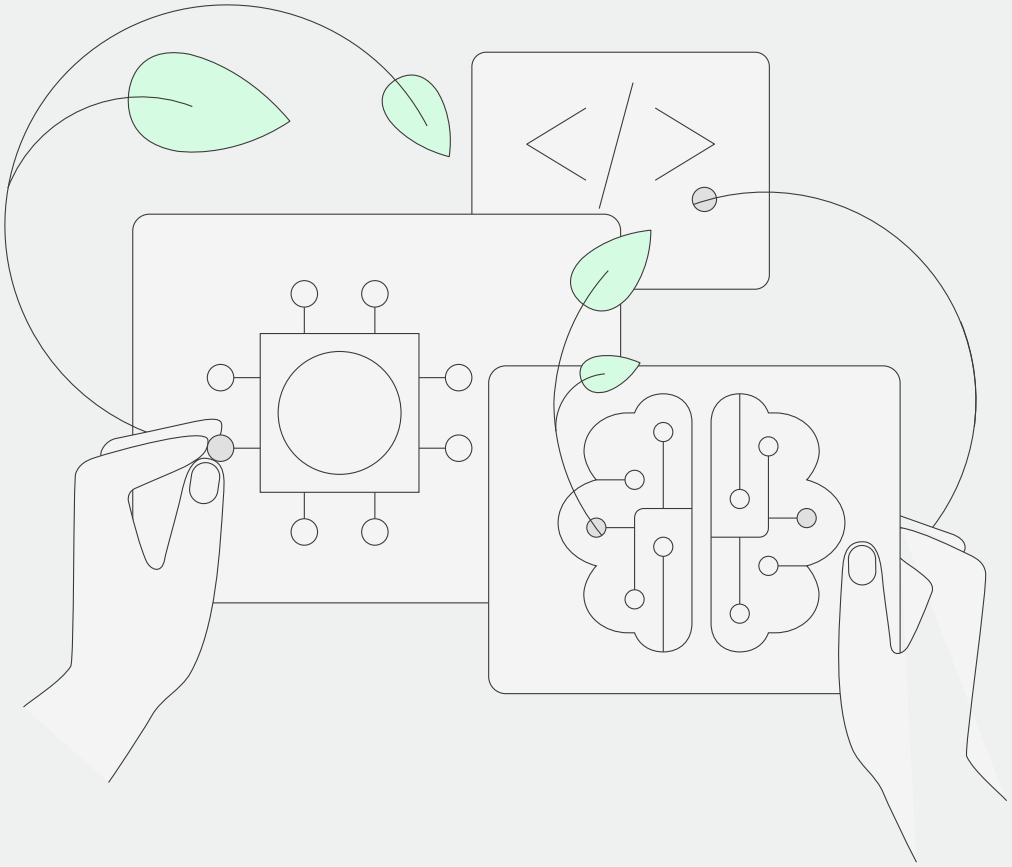
Software privacy

The IBM Privacy Statement describes how IBM collects, uses, and shares personal information. This statement details information management, such as security, retention, and customer rights. IBM values customer privacy and is committed to protecting and processing personal information responsibly.

Role of hybrid cloud in sustainability

Demand-shifting is only possible at a large scale when there is flexibility about where the services can operate. Hybrid cloud and technologies such as Red Hat OpenShift containerization make that flexibility possible.

If services are containerized and portable, choose run locations across different cloud providers or internal data centers. When demand-shifting is increased, more specialized sustainability providers are available, much like data centers that specialize in the Federal Risk and Authorization Management Program (FedRAMP®) or Criminal Justice Information Services (CJIS).



Role of AI in sustainability

AI has a role in sustainability. The recent advances in machine learning and mathematical techniques have created enormous potential for productivity but also have enormous potential for unintended consequences.

AI model training

The most important aspect of model training is the size, quality, and sample relevance of the data content. Shifting the demand of model training provides possible energy savings by training data size, data quality, and data bias.

Consider the size of training data
Generally, the larger the data sample size, the more accurate the model can become. The larger the size, the more expensive the training becomes in both cost and environmental terms. Therefore, understanding how much data is "enough" must be determined for each service. At IBM, we continuously look at models that can improve a business process with a minimal amount of data. This method is important for enterprise services where industry aggregation, which is done for industries such as retail advertising, does not occur.

Ensure the quality of training data
Due to integration and quality issues, it was traditionally estimated that 60–70% of data science project resources are spent on getting the data into a useful condition. To create better results, you must ensure that you verify the quality of the data content for accuracy and consistency.

Avoid training data bias
One big area of concern is data that does not accurately reflect the intended results, contains bias, or codifies oppressive historical results into the model. The confirmation of fair training data must become part of the software engineering data science process, just like any standard test process. A recent example of these concerns is facial recognition, which can achieve amazing results, but when it is used without context it has caused negative societal implications and ethical concerns. Until these models can be shown to be accurate and unbiased, they must be avoided.

AI models for software development

AI models can be used for workload optimization and prediction, which can help reduce energy use. AI models are also useful in other areas of software engineering, including the use of AI to scan for security vulnerabilities and diagnose problems.

Use AI to scan for security vulnerabilities
The search for security vulnerabilities continues to evolve and products typically undergo multiple scan types, each with strengths in certain areas. IBM uses AI models help to identify duplicate vulnerabilities from these scanners and reduce analysis time.

Use AI for problem determination and diagnosis
As problems are reported by users, AI models can identify known problems and their possible solutions, which help to avoid duplicate issues and reduce the time that is needed to resolve issues.

AI model ethics and transparency

Typically, the results from these training models are measured only against the training data for accuracy, starting with the assumption that the historical results are fair and can be optimized for profit. Often, the models provide little explanation about how the results were obtained. IBM follows AI ethics principles and practices that are now being evaluated across the industry, including these guiding values that distinguish IBM's approach to AI ethics.

- The purpose of AI is to augment human intelligence.
- Data and insights belong to their creator.
- Technology must be transparent and explainable.

IBM uses several techniques to ensure transparency, such as including an explanation directly in the user interface (UI) and including details on how the results are determined as the feature is used. In other cases, IBM describes the AI model in the product documentation or in a data science notebook environment such as IBM Watson Studio. The best location for this information is based on where the user typically interacts.

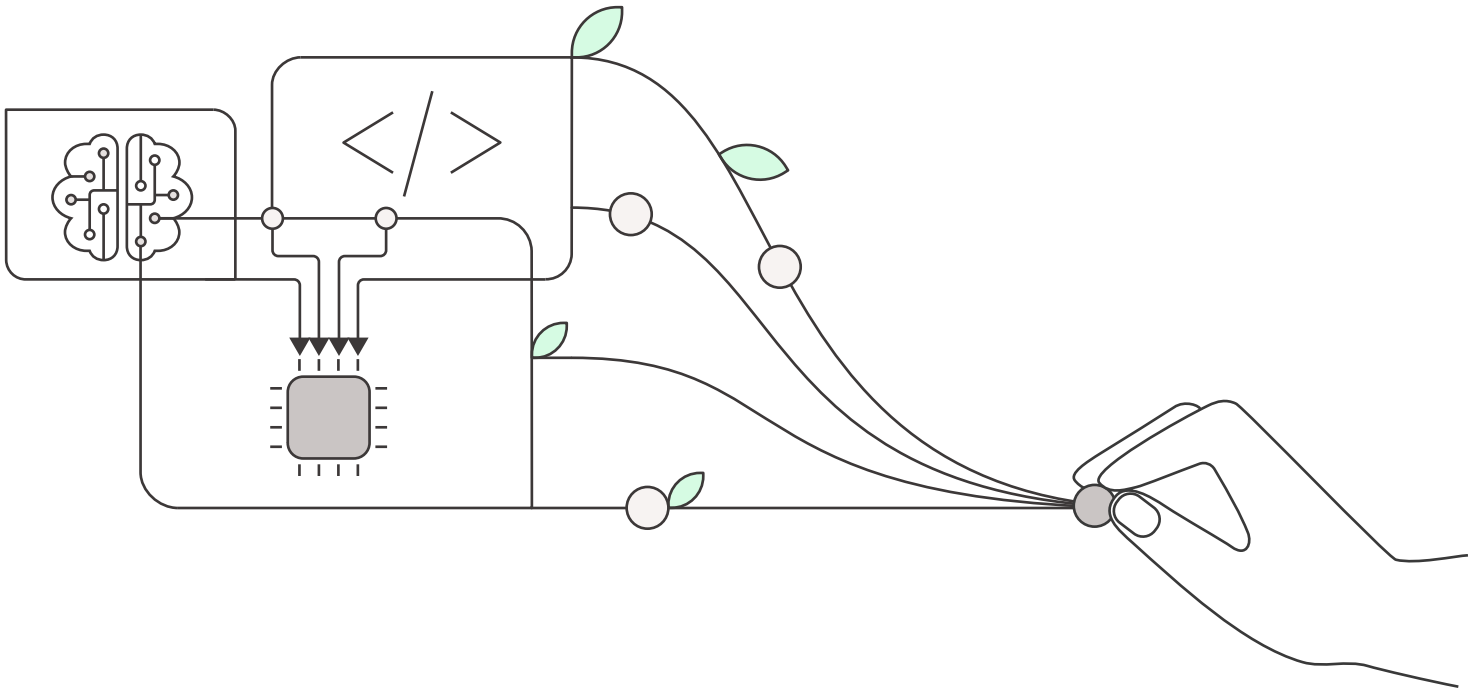
IBM has also decided not to use techniques that are shown to be biased, such as facial recognition technology.

Role of open source in sustainability

Potentially the single largest opportunity for advancing sustainability in software engineering centers around the sharing of software practices, information, and code. Throughout the technology industry, from corporate IT organizations to software product companies to governments and independent developers, a tremendous amount of duplication and inefficiency exists. This duplication includes human effort, intellectual property, algorithms, build processes, security vulnerabilities, and more.

The open source movement has helped to remove duplication in areas such as operating systems, databases, network stacks, infrastructure-as-code, operations, and more. Similar projects inside an organization, sometimes called InnerSource, can lead to the same efficiency gains.

Every organization must decide when software is a competitive advantage versus a duplicate expense that might be shared to reduce environmental impacts and free human capital for better social benefit. While the actual practice and rules for this sharing falls under governance, stronger leadership and collaboration are needed in this area to effect more significant improvements.

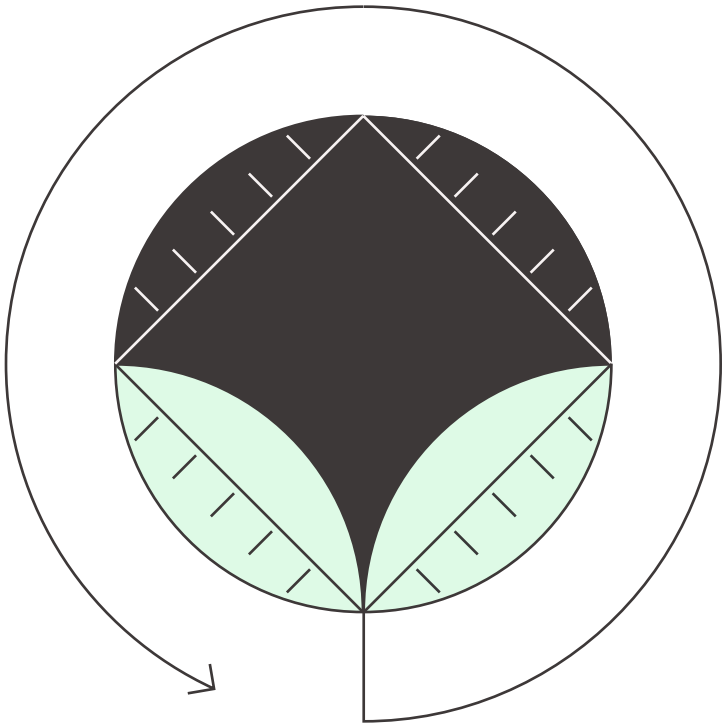


Measuring software sustainability

As with any initiative, understanding what constitutes success and then measuring it is critical to achieving that success. For each sustainable software engineering practice, we have defined those success criteria in this document, and we encourage you to develop your own measurement systems.

As the standards for Environmental Social Governance (ESG) reporting evolve, building consistent units of measure and time periods is important for each of the practices that were covered in this document.

At IBM, we provide the tools to help with ESG reporting, as well as the services that can provide source data to measure those workloads, resource usage, carbon footprint, and more.



IBM Corporation
New Orchard Road
Armonk, NY 10504

Produced in the United States of America
March 2023