2.1 Backwards compatibility: The software is executable on x old reference system

Short description of the metric:

Backwards compatibility is a term used in the context of technology and software development to refer to the ability of a newer version of a product or technology to work with or support data or programs from an older version or and vice versa. In other words, if a product or technology is backwards compatible, it can work with files, data, or applications that were created using a previous version.

Backwards compatibility may be applied in the context of hardware usage as well. It refers to the ability of newer software to work seamlessly with older hardware. This means that the new software can run on older hardware without any issues, even though the hardware was not explicitly designed for the newer software. This type of backwards compatibility is essential because it allows older hardware to continue to function properly with newer software. It also helps to reduce the cost and waste associated with constantly upgrading hardware to keep up with the latest software updates.

Example:

The example of backwards compatibility in terms of data backward compatibility is if a user upgrades to a new version of the software, which is backwards compatible, the user should still be able to open files and use data created in the previous version without any issues. This is important because it allows users to upgrade to newer versions of software or technology without losing access to or compatibility with their existing data and programs.

For backward compatibility in the hardware section, we can take Microsoft Office Operating System as an example. To run Windows 10, a computer needs to have at least a 1 GHz processor, 1 GB of RAM for 32-bit systems or 2 GB of RAM for 64-bit systems, and 16 GB of free hard drive space. These requirements are relatively modest compared to the specifications of many modern computers, and many older computers can meet these requirements. For example, an affordable 2003-released model of Acer TravelMate 661LCi already has enough capability for Windows 10 with an Intel Pentium 4 processor, 1 GB of RAM, and a 16 GB hard drive.

Measurement and reference values related to the metric:

Backwards compatibility is a software or technology attribute and is typically measured by testing the ability of a newer version of a product or technology to work with or support data or programs from an older version. This testing may involve using a variety of methods, including *functional testing*, *performance testing*, and *compatibility testing*.

The reference for backwards compatibility is typically the previous version of the software or technology. To test for backwards compatibility, the newer version is evaluated based on its ability to work with data and programs created in the previous version without any issues. If the newer version can function with the same level of compatibility as the previous version, it is considered to be backwards compatible. As well as when software needs to run in an old reference system, it must have the ability to perform with a moderate performance as when it working on its original environment.

The standards for backwards compatibility may vary depending on the specific software or technology being evaluated, as well as the needs and expectations of the users. In some cases, it may be necessary to establish specific benchmarks or metrics for measuring backwards compatibility, such as *the ability to open and edit files* created in the previous version, or *to maintain the same level of performance and functionality as the previous version*.

Guidance on how to apply the metric in development:

To apply the backward compatibility metric in development, the following steps can be taken:

1. With proper previous versions of documentation, we can have a clear picture of stakeholders' concerns. These concerns still need to be identified and addressed in the new version.
2. Design the new product with backward compatibility in mind. This may involve incorporating standard interfaces, protocols, and formats that are widely used and accepted so that the new product can communicate and work with existing products and systems.
   Some common coding practices to ensure backward compatibility are: feature toggle/flag, decoupling services, versioning, loose-coupling services architecture design, and relational database.
3. Test the new product to ensure that it is fully compatible with existing products and systems. This may involve running compatibility tests, performing interoperability testing, and identifying and resolving any issues or conflicts that arise.

How to monitor the fulfilment of the metric after the product is ready:

After the product was developed and tested, the system developer will enter to implementation and monitoring phase. These steps can be taken in order to make sure backward compatibility:

1. Document the backward compatibility features of the new product, and communicate them clearly to customers and stakeholders. This may involve creating user manuals, technical documentation, and marketing materials that highlight the backward compatibility features of the new product.
2. Deliver the new version to the production server. Make sure to follow the correct steps for publishing the system. These 'correct' steps may vary from company to company, from system to system. But, in most cases, the system will be changed from the consumer side, database, and then the backend side.
3. Monitor and update the new product's backward compatibility features over time. This may involve tracking changes in the market and the needs of customers, and making updates and improvements to the product's compatibility features as needed.

Conditions for using the metric:

Backwards compatibility is often a key consideration in software and technology development, as it helps to ensure that users can continue to use existing data and applications even as newer versions are released, as well as, the ability of an old version of software and technology to work in a new version of the environment. However, achieving backward compatibility can sometimes be challenging, particularly when significant changes are made to the underlying architecture or functionality of a product.

Even though an application's backward compatibility may be challenging, it can be considered mandatory in systems that are used in environments where there is a need for long-term stability, interoperability with legacy systems, or continuity of service. This includes areas such as:

- Government systems: Government systems, such as those used for taxation, social security, and healthcare, often require long-term stability and continuity of service. Backward compatibility can help ensure that these systems remain operational over time and can still interact with legacy systems.
- Enterprise systems: Enterprise systems, such as those used for accounting, customer relationship management, and supply chain management, also require long-term stability and continuity of service. Backward compatibility can help ensure that these systems remain operational and can still interact with legacy systems and data.
- Infrastructure systems: Infrastructure systems, such as those used for transportation, utilities, and telecommunications, require high levels of interoperability and stability. Backward compatibility can help ensure that these systems can still function with legacy systems and can support the needs of the wider ecosystem.
- Consumer electronics: Consumer electronics, such as smartphones, laptops, and gaming consoles, are often used for several years. Backward compatibility can help ensure that these devices can still run legacy software and interact with legacy hardware.

In general, any system that is expected to have a long lifespan and interact with legacy systems or data may benefit from backward compatibility.

Anticipated environmental effects of the metric:

The anticipated environmental effects of the backward compatibility metric in product design are related to the potential reduction of waste and environmental impact associated with the disposal and replacement of products. By designing products with backward compatibility in mind, products can be used for longer periods and can be integrated with existing systems and infrastructure, reducing the need for frequent replacement and disposal.

In the general, one potential effect of backward compatibility is that it may encourage users to continue using older software or technology for longer periods.

This can result in a slower rate of adoption of newer, more energy-efficient technologies, which could have a negative impact on energy consumption and greenhouse gas emissions. For example, if users are able to continue using older technology without having to replace it, this may result in a reduction in e-waste and the associated environmental impacts of electronics manufacturing and disposal.

In which kind of situations the metric is irrelevant:

There are some situations in which backward compatibility may be less relevant or not relevant at all. These may include:

1. Brand new products or systems: If a product or system is completely new and has no legacy systems or data to interact with, then backward compatibility may not be relevant.
2. Systems with short expected lifespans: If a product or system is expected to have a short lifespan, backward compatibility may not be a priority, as it may not be necessary to integrate with legacy systems or data.
3. Systems with no legacy data: If a system does not need to interact with legacy data, then backward compatibility may not be relevant.
4. Systems with no external interfaces: If a system does not need to interface with external systems or hardware, then backward compatibility may not be relevant.

However, it is worth noting that even in these situations, there may still be some benefits to designing products with backward compatibilities in mind, such as potential cost savings and reduced environmental impact associated with the disposal and replacement of products. Additionally, as technology evolves and new systems and data become available, there may be opportunities to introduce backward compatibility to improve the longevity and usefulness of products and systems.

Estimate of the price of implementing the metric or the price impact of implementing the metric:

The price of implementing backward compatibility can vary widely depending on the product, system, and level of compatibility required. In general, implementing backward compatibility may require additional design, development, and testing resources, as well as potential hardware or software updates or modifications.

For example, implementing backward compatibility in a software product may require additional testing to ensure that the product is compatible with older operating systems, hardware configurations, or third-party software libraries. It may also require additional development time to create workarounds or compatibility layers for older systems or software.

In terms of cost impact, implementing backward compatibility may increase the cost of production, as it may require additional design and development resources, as well as potential hardware or software modifications.

Anyway, even though implementing backward compatibility may increase the upfront cost of production, it may also provide long-term benefits such as reduced support and maintenance costs, improved customer satisfaction, and potential cost savings associated with extending the lifespan of legacy systems and reducing the need for frequent upgrades.

Other related metrics you could imagine (if something is required to get this metric usable or this enables some other metric):

Backward compatibility can be considered a metric that measures the ability of a newer version of software or technology to work with data and programs from an older version. In order to make this metric usable, it may be necessary to establish specific benchmarks or performance criteria for backward compatibilities, such as the ability to open and edit files created in the previous version or to maintain the same level of functionality and performance.

Backwards compatibility can enable or contribute to other metrics on the list, such as:

Anticipation of the future (2.3): Backwards compatibility can help ensure that software data and data formats are transferable to other software, which is an important aspect of anticipating the future. By allowing older software and data to be seamlessly integrated into new systems, backwards compatibility can help ensure that data can be used and accessed even as technology evolves and changes.

Transparency (2.4): Backwards compatibility can contribute to transparency in software, as it helps ensure that older versions of the software can be accessed and studied. This can be important for ensuring that the software is operating as intended and that any issues or bugs can be identified and addressed.

Technical optimization (2.5): Backwards compatibility can also help minimize resource consumption, as it allows users to continue using older systems rather than having to upgrade to newer, more resource-intensive systems.

Minimized technical debt (2.7): By maintaining backwards compatibility, software developers can avoid accumulating technical debt by ensuring that older software and systems can continue to be used and maintained. This can help ensure that the software remains in good working order and that any technical issues are resolved in a timely manner.

However, backward compatibility may not directly impact metric 3.1 (CO2 emissions of the company) as it is primarily a software or technology attribute and is not directly related to the company's emissions. Nevertheless, it can contribute to reducing e-waste and thereby reducing the environmental impact of electronics manufacturing and disposal, which can help to reduce the company's overall carbon footprint.

Process support (3.2): Backwards compatibility can be integrated into the software development process as a quality requirement metric, which can help ensure that the software is developed sustainably and efficiently.

Awareness and knowledge of the developers(3.3): Backward compatibility can also help increase the awareness and knowledge of developers, as it may require them to understand the underlying architecture and functionality of older versions of the software or technology. This can lead to a deeper understanding of the product and its development process.

Certification of the developers or the company(3.4): Backward compatibility can also be a factor in the certification of developers or the company. If a company can maintain backward compatibility with older versions, this may be seen as a sign of their technical expertise and commitment to quality.

Overall, backwards compatibility is an important metric that can enable and contribute to other metrics related to software quality and sustainable development. By ensuring that software is able to work with older systems and data, it can help promote longevity and reduce the environmental impact of software development.

Additional information and sources:

Some countries, such as the European Union, have implemented specific regulations that require manufacturers to ensure backward compatibility. For example, the EU's Eco-design Directive sets standards for energy efficiency, but also includes requirements for products to be designed with backward compatibility in mind.