1. **What criteria should be used in choosing an appropriate requirements engineering tool?**

Carrillo de Gea et al. (2021) provided an alternative list of 17 items to be used in the evaluation of RE tool capabilities:

* Organization of requirements with metadata, attributes, and reuse
* Reports, database queries, and open interface language
* Internal checks, that is, consistency, dependencies, and history
* Traceability support, that is, drag and drop (horizontal and vertical)
* Providing support for reuse
* Remote working, cloud only
* Multiple views of requirements
* Performance
* Collaboration, workflow management
* Easily adapted and integrated into business processes
* Federation and notification with ALM/PLM tools
* Export/import with standard formats
* Macros for repeated commands
* Training and learning curve effort
* Agile, CI/CD, and DevOps
* Intelligent support
* Scalability

This checklist (or an adapted version) and an appropriate consensus management approach (e.g., Wideband Delphi, AHP) can be used to select the right tool for a team or enterprise.

1. **Are there any drawbacks to using certain tools in requirements engineering activities?**

(Tùng)

* Hard:
* Technology dependency: Some tools require the use of specific technology or depend on a specific database management system. This can cause limitations when integrating or moving data to another environment.
* Difficulty in customization: some tools are not flexible in customization. So when you encounter a problem, you may have to completely destroy it and rebuild it to solve it.
* Complicated and difficult to use: Some tools have complex interfaces that require employee training or customer training, causing discomfort for users, slowing progress, and increasing costs.
* High cost: Many tools have very high costs and are not suitable for small businesses with little capital.
* Incompatible with existing processes
* Limited scalability: Some tools have limited scalability as the project grows or the organization expands. This can create limitations for future expansion and growth.
* Lack of quality and support: Some tools may not meet quality standards or have good technical support from the vendor. This can make it difficult to resolve technical issues or problems.

(Trí)

Several studies of commercial tools have been conducted using the ISO/IEC TR 24766 framework (e.g., Carrillo de Gea et al. 2011, 2015; Daud et al. 2014). These studies have generally found that the tool market is rapidly changing and that tools are becoming increasingly complex and difficult to use. The complexity of the expensive commercial tools then creates opportunities for inexpensive tools to emerge but don’t offer sophisticated features. Furthermore, these studies have indicated that validation functionalities such as consistency, correctness, and completeness are still lacking in most of the tools.

1. **When selecting an open-source tool, what characteristics should you look for?**

(Tùng)

* Popularity: High popularity means a good and successful project should be used a lot.
* Is there strong maintenance? Because it is an open-source, it requires the owner of the project and key members to have high productivity and motivation to maintain it to avoid errors. desire to damage the system when used.
* License: need to comply with regulations when using open source code.
* Documentation: is the document complete and easy to understand?
* Is it easy to expand and use, is it flexible?

(Trí)

Sud and Arthur (2003) evaluated a number of requirements management tools using the following dimensions:

* Requirements traceability mechanism
* Requirements analysis mechanism
* Security and accessibility mechanism
* Portability and backend compatibility
* Configuration management approach
* Communication and collaboration mechanism
* Change management support
* Online publishing support
* Usability features such as word processor compatibility
* SRS documentation format

While the findings of their study are now somewhat obsolete, these evaluation dimensions can be used by engineers to compare various commercial and open-source requirements management tools prior to adoption. For example, a simple checklist could be created using these dimensions, or a Likert scale–based evaluation could be conducted.

1. **How can tools enable distributed, global requirements engineering activities? What are the drawbacks in this regard?**

(Tùng)

* **Supporting:**
* Collaboration Platforms: Tools like Jira, Trello, and GitHub project help enable real-time collaboration between team members regardless of their geographic location. These platforms allow teams to manage, track, and update tickets, ensuring everyone has the same information.
* Meeting: Many apps support you calling directly with video rendering, like Microsoft Teams, Discord, Google Meets., .... It also allows you to send messages and file in real time.
* Cloud-Based Requirement Management Tools: Easy sharing files between many people, easy access to the last version of documents.
* Version Control Systems: Help you manage the version of documents, source code, ensuring that the entire team is aware of any modifications or updates made by other team members. This helps maintain a coherent and up-to-date understanding of the project's requirements. Easy back to old version if meet any problems with current version.
* **Drawbacks:**
* Technical challenge: Need to learn how to use the tools before using it
* Communication Barriers: Despite communication tools, cultural and language differences can still pose challenges in understanding requirements, leading to misunderstandings and misinterpretations.
* Lack of Personal Interaction
* Dependency on Technology: Relying too heavily on tools can create dependency issues. If there are technical glitches or system failures, it may disrupt the entire requirements engineering process, leading to delays and misunderstandings.

(Trí)

* **How tools enable distributed, global requirements engineering activities:**

Various mobile technologies such as cell phones and personal digital assistants can be used to capture requirements information in place. For example, while physicians are working with patients, they can transmit information about the activities they are conducting directly to the requirements engineer, without the latter having to be on site. Using mobile devices is particularly useful because they enable instantaneous recording of ideas and discoveries. Such an approach can support brainstorming, scenario generation, surveys, and many other standard requirements elicitation techniques even when the customer is not easily accessible (such as in offshore software development situations).

For example, Lutz et al. (2012) developed an application called **CREWSpace** that allowed users to role-play through Android-enabled devices such as smartphones and tablets and, in doing so, to interact with a representation model displayed on a shared screen. By keeping track of the role-playing states, the software was able to create CRC cards, which are a brainstorming tool used in object-oriented software systems. The CRC cards could then be incorporated into the requirements document.

A second example is the **Mobile Scenario Presenter (MSP)** that has been developed by the Johannes Kepler University of Linz, Austria, and the City University London, UK. The application allows for both mobile analysts and future system users to acquire requirements systematically and in situ using structured scenarios.

**Arena-M** (Anytime, Anyplace Requirements Negotiation Aids - Mobile) is the third example. It is a mobile application that provides support for distributed requirements negotiations.

* **Drawbacks** (personal opinions, not in book):

Distributed collaboration can also face challenges like:

* + Time zone differences affect meeting schedules and real-time collaboration.
  + Cultural and language barriers in distributed communication.
  + Reliance on network connectivity and infrastructure across locations.
  + Difficulty in resolving conflicts, misunderstandings remotely without face-to-face interaction.
  + Information silos due to non-integration of tools across dispersed teams.
  + Learning multiple tools instead of a unified interface.

1. **If an environment does not currently engage in solid requirements engineering practices, should tools be introduced?**

(Tùng)

* Introducing tools in an environment that does not currently engage in solid requirements engineering practices can be a complex decision, their introduction in an environment with inadequate practices should be accompanied by a comprehensive understanding of the current state, appropriate training, organizational readiness, a sound change management strategy, and a focus on continuous improvement.
* Conducting a thorough evaluation of the existing requirements engineering practices to identify challenges and gaps, facilitating an effective tool introduction strategy.
* Training and Education: Without a solid foundation in requirements engineering practices, the introduction of tools alone may not yield the desired improvements.
* Evaluating the organization's willingness to adopt new tools, ensuring a smooth transition without encountering resistance from team members.
* Developing a comprehensive change management plan that addresses concerns, provides support, and highlights the advantages of the new tools to encourage their adoption.
* Pilot Testing: Before full-scale implementation, consider conducting pilot tests to evaluate the effectiveness of the tools in the existing environment.
* Viewing the introduction of tools as an integral part of a continuous improvement initiative, fostering a culture of ongoing enhancement within the organization.

(Trí)

Whatever requirements engineering tool(s) you use, it is appropriate to use the tool judiciously and follow certain best practices. An excellent set of such practices is offered by Cleland-Huang et al. (2007):

* Trace for a Purpose: That is, determine which linkages are truly important; otherwise, a large number of extraneous links will be generated.
* Define a Suitable Trace Granularity: For example, linkages should be placed at the appropriate package, class, or method level.
* Support In-Place Traceability: Provide traceability between elements as they reside in their native environments.
* Use a Well-Defined Project Glossary: Create the glossary during initial discovery meetings with stakeholders and use it consistently throughout the requirements engineering process.
* Write Quality Requirements: Make sure to follow generally accepted best practices such as IEEE 29148, which are particularly important for traceability.
* Construct a Meaningful Hierarchy: Experimental results show that hierarchically organized requirements are more susceptible to intelligent linking software.
* Bridge the Intradomain Semantic Gap: For example, avoid overloaded terminology, that is, words that mean completely different things in two different contexts.
* Create Rich Content: Incorporate rationales and domain knowledge in each requirement.
* Finally, be sure to use a process improvement plan for improving the requirements engineering process.

Following disciplined practices can result in better results from tool usage and a framework from which processes can be improved. Every project plan should include a description of the tools to be used and how they will be used.

1. **What sort of problems might you find through a traceability matrix that you might not see without one?**

Requirements Source Traceability Matrix

Yet another kind of traceability matrix links requirements to their sources. Aside from those coming directly from users, many requirements are derived from governmental regulations and from standards. Linking the requirements to these sources can be very helpful when the sources change. Table 9.7 shows the typical format for such a traceability matrix.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Requirement ID | Federal Regulation #1 | Federal Regulation #2 | State Regulation #1 | State Regulation #1 | International Standard #1 |
| 3.1.1.3 | X |  |  |  |  |
| 3.1.2.9 | X | X |  |  |  |
| 3.2.1.8 |  |  | X | X |  |
| 3.2.2.5 |  |  | X |  |  |
| 3.2.2.6 |  |  |  |  | X |
| 3.3.1 |  | X |  |  |  |
| 3.3.2 |  | X |  |  |  |
| 3.4.1 |  | X |  |  |  |
| 3.4.3 |  | X |  |  |  |
| 3.4.4 |  | X |  |  |  |
| 3.6.5.1 |  |  |  | X |  |
| 3.6.6.4 |  |  |  |  | X |

*Table 9.7 Traceability Matrix Showing Requirements and Their Sources*

This kind of traceability matrix is especially useful for tracking nonfunctional requirements. Failure to trace nonfunctional requirements throughout the project life cycle can be a significant problem (Kassab et al. 2008).

1. **How is AI being proposed for knowledge acquisition and representation in requirements specifications?**

In the past few years, the thread of work on Artificial Intelligence (AI) for RE has made strides in rigorously investigating how general-purpose AI tools can be tailored best for RE tasks (Dalpiaz and Niu 2020). For example, recent research studies are investigating how human intervention in the requirement gathering processes can be reduced by using “Speech Understanding Methodology” techniques with the capability to “listen in” on a conversation and suitably collect stakeholders’ declarations into a distinct vision (Spoletini and Ferrari 2017; Sharma and Pandey 2014).

Speech understanding methodology can be combined then with “Automatic Keywords Mapping,” another AI technique being investigated that can enhance the requirements elicitations. Many requirements issues are related to not having the stakeholders able to depict their requirements properly, or the ignorance of the domain experts and developers to “observable” words that lead essentially to system requirements. These issues can be eliminated by automatically mapping every keyword spoken by each stakeholder. The earlier studies released a keyword mapping technique for designers so that they can recognize the keywords used by stakeholders to assist them in making ideal requirements (Sharma and Pandey 2013).

Case-based reasoning is also being investigated for requirement elicitation, which can reduce the problem of natural language understanding as well as save the time of the requirement expert. There has also been recent research on the use of machine learning algorithms to identify user preferences based on their sentiment (Li et al. 2018).