CERN Accelerator Controls GUI Strategy: Status & Plans

Stephane Deghaye, CERN

Purpose of the GUI Strategy

- We need a GUI Strategy in order to manage 2 risks that could impact beam time and future evolutions of the accelerators
  1. Breaking applications
  2. Difficult (if not impossible) to maintain & extend applications

Types of applications & Target users

- **Equipment experts** need to configure, maintain, control, tune, visualize, diagnose, and monitor their equipment

➔Need to develop specific applications

- **Operations crews** need to operate, optimise and supervise accelerators, and the complex systems they are made of, and automate repetitive tasks

➔Need to develop operation-use-case-driven applications

Types of applications & Target users

- **Operations (OP)** also need to diagnose, and monitor the equipment

➔Applications not developed for them end up in the control room & this can (and does) create issues and the usability is sub-optimal

What triggers changes?

- For all groups
  - Hardware changes (evolution & renovation)
  - Expert needs (developments & operation)
  - OP needs (to increase efficiency & new operational requirements)
  - Bug fixes
  - Evolution of Controls solutions & frameworks
  - Evolution of underlying technologies (software & hardware)

All the time

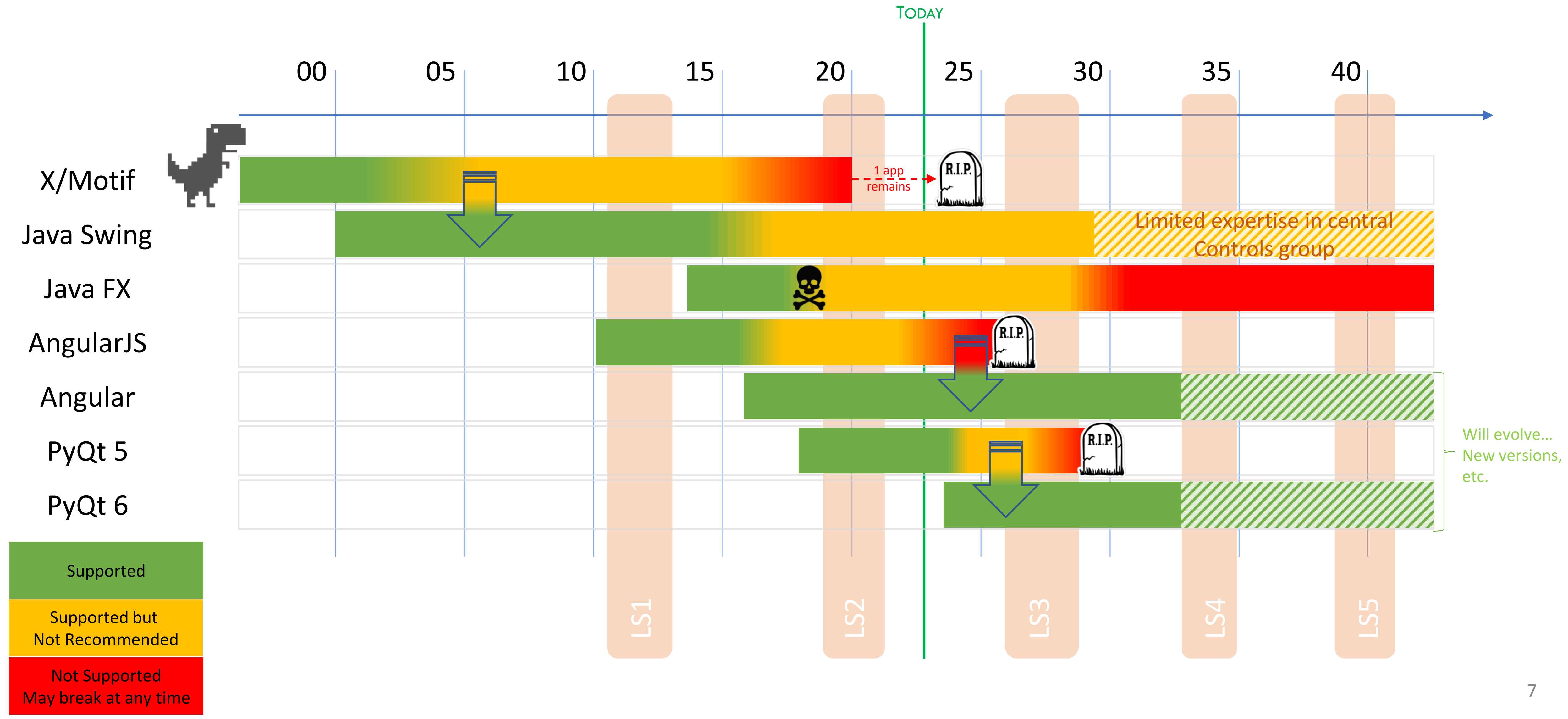
Year-end stops
&
long shutdowns

Purpose of the GUI Strategy - Reminder

- We need a GUI Strategy in order to manage 2 risks that could impact beam time and future evolutions of the accelerators
  1. Breaking applications
  2. Difficult (if not impossible) to maintain & extend applications

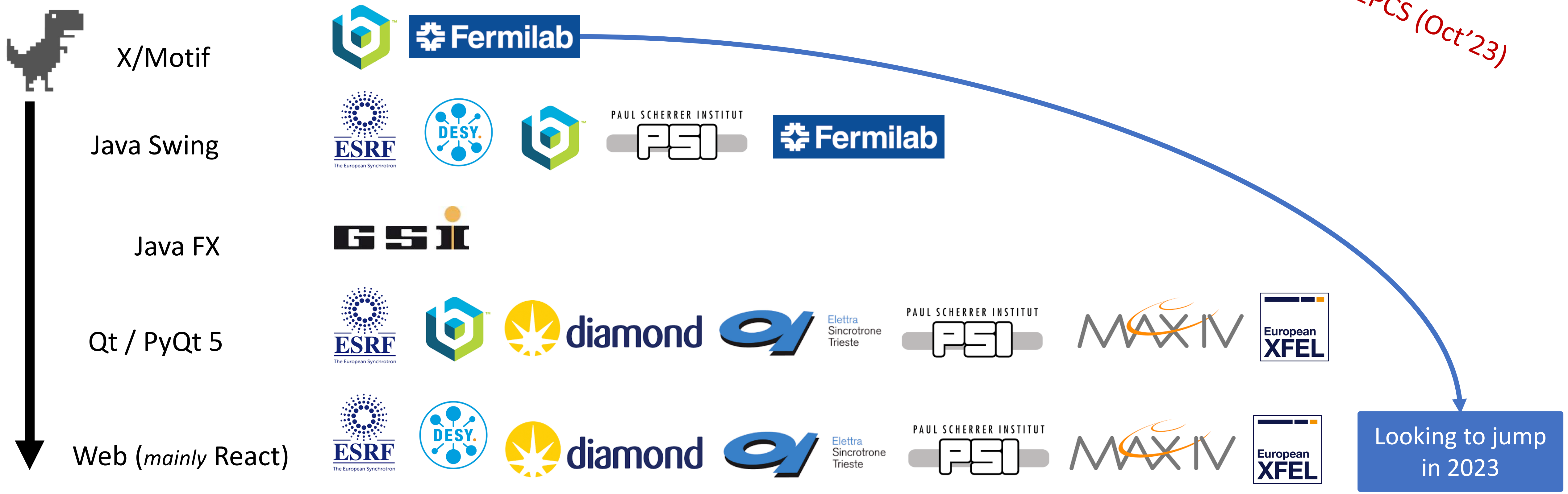
➔ **Controlled follow-up of the technology evolution**

GUI technologies



GUI technologies @ Other Labs

To be updated today !
GUI Workshop in ICALEPCS (Oct'23)



➔ Reassuring that we are all (almost) going in a similar direction

Source: International Controls GUI Workshop 2022

GUI development offering

- The GUI strategy comes together with a GUI Development Offering
- Key aims:
  - Keep under control the potentially huge expense due to obsolete technologies that would require rewrite of the applications
  - Reduce the cost of GUI developments in the long run, for both creation and maintenance of applications
- Several coherent solutions:
  - 2 no-code application platforms: **WRAP** & **NavPy**
  - 2 application frameworks: Accsoft-Commons-Web (**ACW**) with Angular & **PyUI** with PyQt

GUI development solutions - Platforms

- From the outset, the platforms aim to build on the success of an old Java-Swing-based platform (aka **Inspector**)
  - Reduce the number of full-fledged specific applications
  - Facilitate the development thanks to no-code/low-code approach (domain focus)
  - Reduce the exposure to technology changes
- **WRAP**
  - Web-based, available from anywhere (on CERN's network)
  - Integrated with high-level services (setting management, archiving, etc.)
  - High-level of customisation
- **NavPy**
  - Desktop application, low-level access, available on 1st day of RT software development
  - Low-level tools for experts and diagnostics (cycle comparison, copy, etc.)
  - Flexible layout with many viewers

GUI development solutions - Frameworks

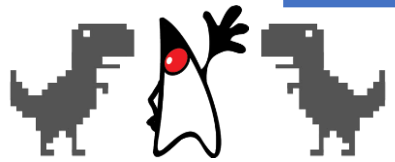
- The frameworks aim to facilitate the development thanks to:
  - Pre-selection and integration of technologies (Angular, Java, Spring Boot)
  - Integration with the Control System (role-based access, Timing, etc.) & libraries of widgets
  - Simplified software-engineering processes
- **PyUI** based on PyQt
  - Desktop applications
  - Library of controls-aware widgets
  - Based on in-depth user interviews over the summer $\rightarrow$ clear plans for the next year
- **ACW** based on Angular
  - Web-based applications
  - Angular (web front-end) & Java with Spring Boot (back-end)
  - Many generic components already available
  - Used in all recent Controls web apps

	Java Swing	Java FX
SY-ABT	✓	✓
SY-BI	✓	✓
BE-CEM	✓	🚫
SY-EPC	🚫	🚫
TE-MPE	✓	✓
BE-OP	✓	✓
SY-RF	✓	🚫

Java Swing’s Future @ CERN

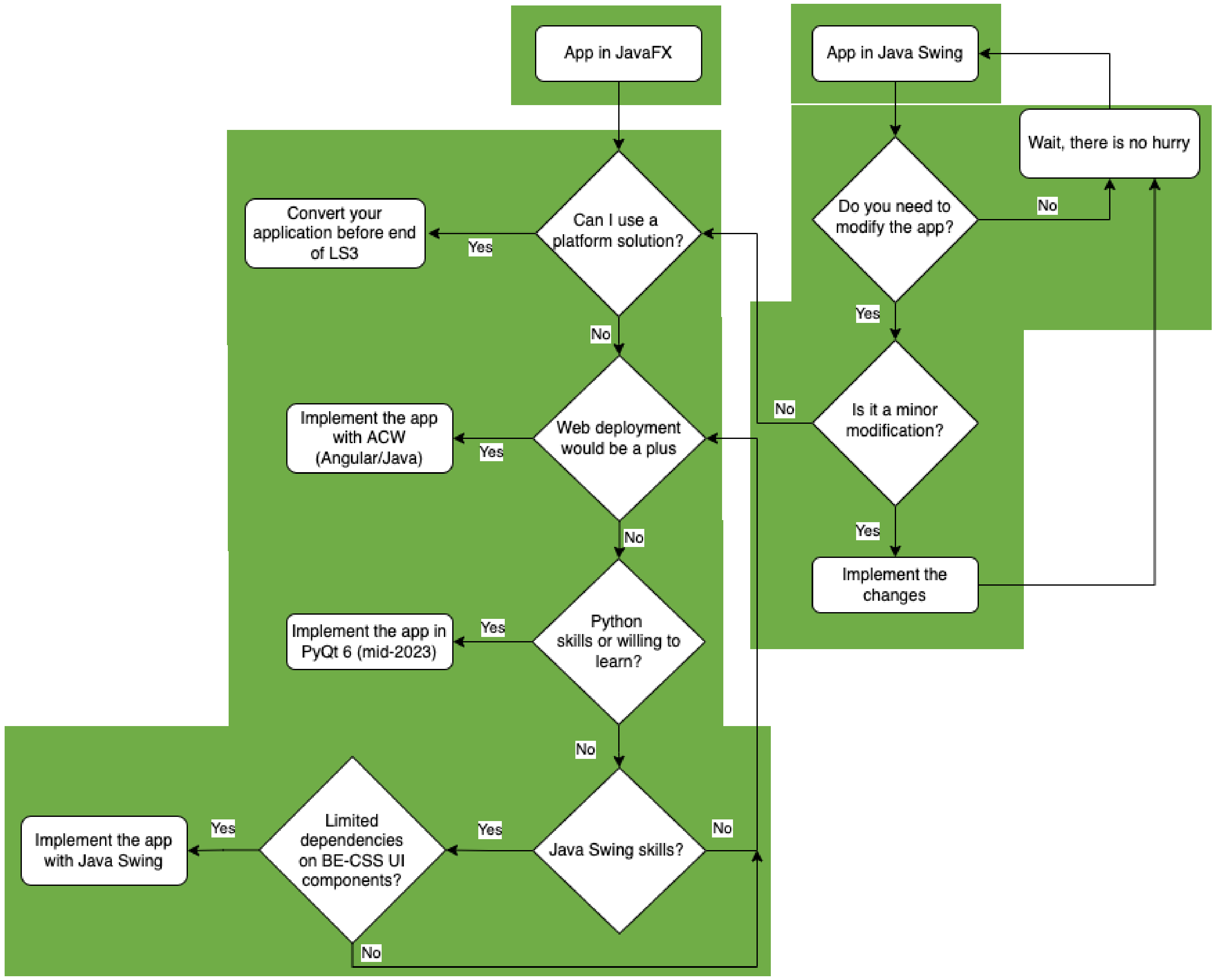
- Swing is part of the core of Java

  →Unlikely that Oracle manages to get rid of it
- Swing was released in the late 90; that’s ~25 years ago



- At the end of the decade, very little knowledge of Swing will remain in the central controls group (Java expertise remains)
- It’s unrealistic to plan to train 1 or 2 young engineers for the **evolutive** maintenance of Swing components.
- However, maintaining the Swing code base as-is must (and can) be done until the end of LHC
- Therefore, **even though Swing will be supported, it is <u>not</u> recommended to use it for any new developments**

Which solution to choose for Java GUIs?


App in JavaFX
App in Java Swing
Wait, there is no hurry
Can I use a platform solution?
Convert your application before end of LS3
Yes
No
Do you need to modify the app?
No
Yes
Is it a minor modification?
No
Yes
Implement the changes
Web deployment would be a plus
Implement the app with ACW (Angular/Java)
Yes
No
Python skills or willing to learn?
Implement the app in PyQt 6 (mid-2023)
Yes
No
Java Swing skills?
No
Yes
Limited dependencies on BE-CSS UI components?
Yes
Implement the app with Java Swing
No

Conclusions

- The strategy for GUIs was reviewed in a recent workshop at CERN
- Most of the clients support it!

➔Overall, they would like that we deliver the new solutions faster

- For some communities, it’s hard to see Swing not being recommended

➔With tens (hundreds) applications in Swing, we foresee to maintain it for at least the next 15 years (lifetime of the LHC)