PITZ - das grafische Nutzerinterface und jddd

Mehr als fünfzehn Jahre Arbeit an der PITZ gui - Was bleibt?

Bert Schöneich / Winfried Köhler
(abgestimmt mit Elke Sombrowski, DESY HH)
Technisches Seminar
DESY Zeuthen

HELMHOLTZ GEMEINSCHAFT

DESY

Inhalt

DESY

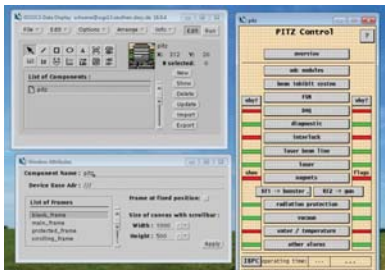
Geschichte - 1

grafisches Nutzerinterface (gui) für PITZ

– vor fünfzehn Jahren:
das erste gui Entwicklungstool

ddd

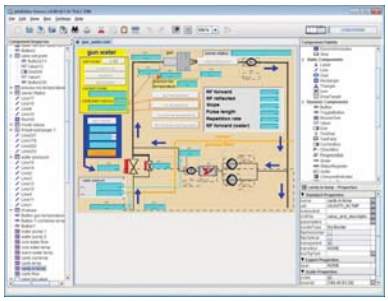
- "DOOCS Data Display"
- Ergebnis wird in CAF-Dateien gespeichert



– seit acht Jahren:
das zweite gui Entwicklungstool

jddd

- "Java DOOCS Data Display"
- Ergebnis wird in xml-Dateien gespeichert

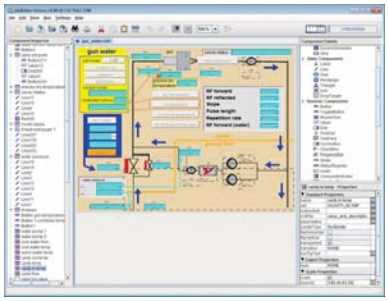


Geschichte - 2

Übergang zwischen Altem und Neuem

ddd (CAF)

- erster Schritt: teilautomatisiert, das war zu 85 % erfolgreich
- mit Tools aus DESY Hamburg und von David Melkumyan
- danach alle 400 Dateien in drei Revisionszyklen per Hand durchgearbeitet
- Dauer ca. 2 Jahre



jddd (xml)

jddd - 1

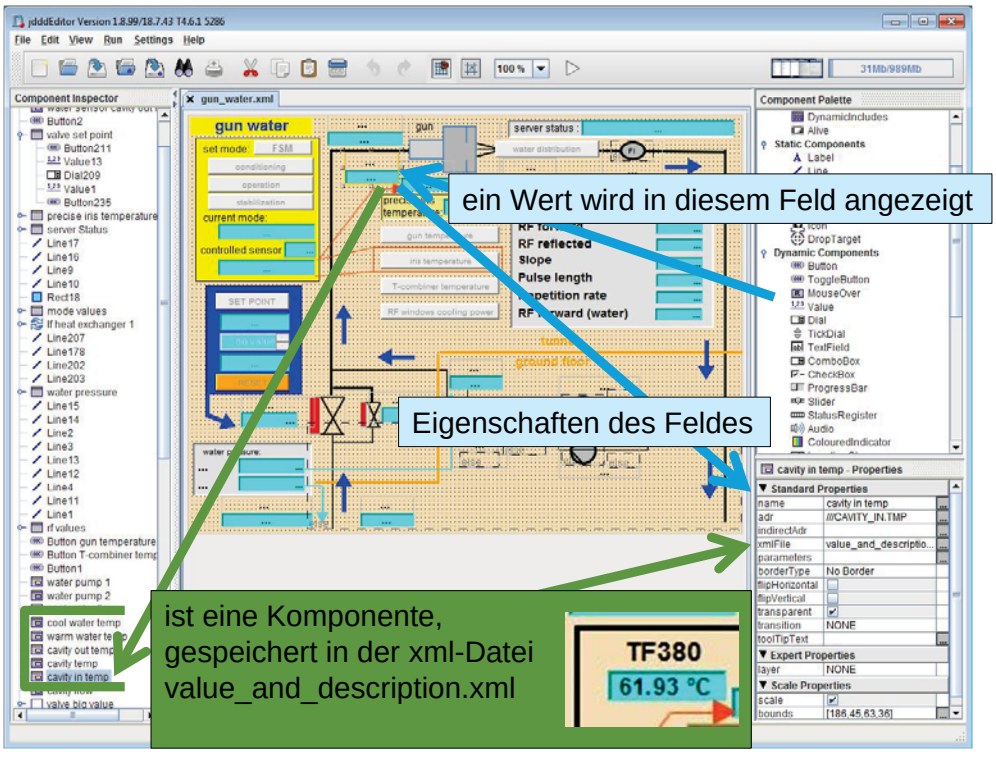
Status

(May 2017)

- PITZ gui:
  - 824 xml-Dateien
  - 145 Ordner
  - 18,9 MB
- alte CAF Dateien:
  - 372 CAF-Dateien
- PITZ gui svn:
  - 4.440 Dateien
  - 412 Ordner
  - 104 MB

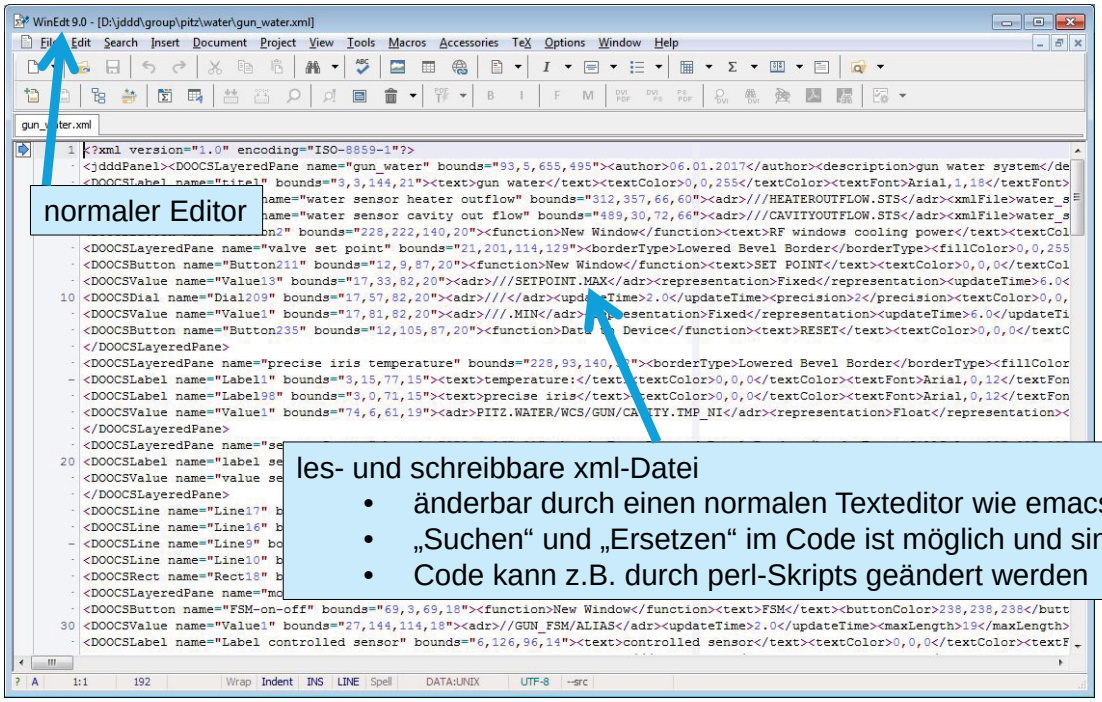
jddd - 2

Vorteile - 1



jddd - 3

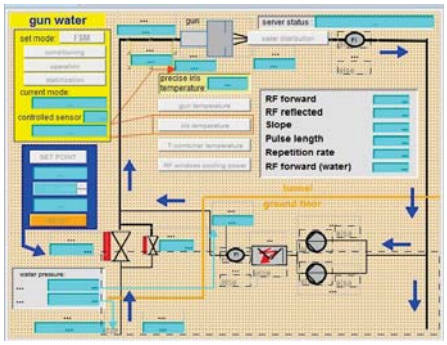
Vorteile - 2



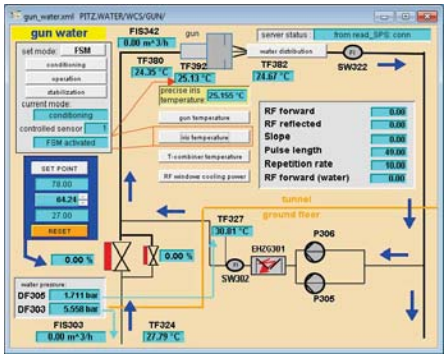
jddd - 4

grafisches Nutzerinterface (gui) für PITZ

– Fenster „gun_water“ im Entwicklungstool



– Fenster „gun_water“ in der Laufzeitumgebung



jddd - 5

Vorteile - 3

1. gute Kombination von grafischer und/oder numerischer gui Programmierung
   (Platzieren, Verschieben oder Größenänderung von Objekten)

2. objekt orientierte grafische Programmierung
   - Unterfenster, aufrufbar von vielen übergeordneten Fenstern
   - Komponenten, nutzbar in vielen verschiedenen Fenstern
     mit verschiedenen DOOCS Adressen

3. Das Ergebnis des Programmierens ist eine les- und schreibbare xml-Datei.
   - änderbar durch einen normalen Texteditor wie emacs
   - „Suchen“ und „Ersetzen“ im Code ist möglich und sinnvoll
   - Code kann z.B. durch perl-Skripts geändert werden

jddd - 6

Vorteile - 4

4. viele vorbereitete Komponenten vorhanden
   (statische, dynamische, logische, grafische, Plots)

5. Fensterteile sind gleichzeitig grafisch und als Liste sichtbar
   während des Programmierens

6. durchdachte und hilfreiche Ordnerstruktur für die xml- und andere Dateien,
   wie z.B. Images (wo speichere ich was)

7. aufrufbar von lokalen Computern und aus einem Web-Browser

jddd - 7

Nachteile

1. Das Entwickeln und Programmieren ist nur mit realen (!) Daten möglich („Operation am offenen Herzen“).

2. Während ein Gerät keine Daten liefert (z.B. bei shutdown), kann nicht an der gui für dieses Gerät gearbeitet werden.

3. einfache gui
   - kein gui interner Speicher vorhanden, auch nicht zeitweise

4. geringe Geschwindigkeit bei der Öffnung einer gui

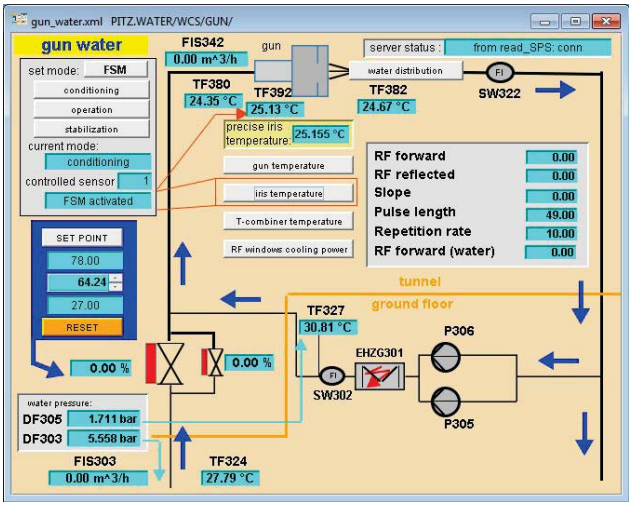
5. Keine wirkliche Dokumentation vorhanden (DESY HH veranstaltet Lehrgänge).

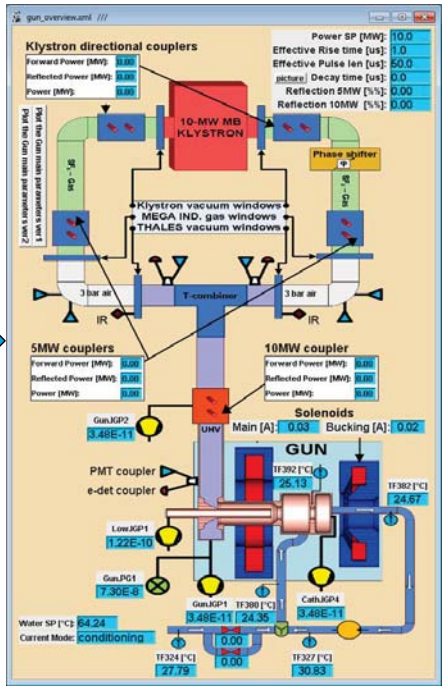
6. Manchmal ist es schwierig, eine Linie/Objekt mit der Maus zu fangen..

7. Vorsicht bei der Zusammenfassung von Komponenten in „Gruppen“.
   Eine Gruppe überdeckt unsichtbar hinter ihr liegende Komponenten.
   Zum Beispiel ist eine sichtbare Taste nicht mehr bedienbar, wenn sie hinter einer Gruppe liegt.

Arten des Programmierens mit jddd - 1

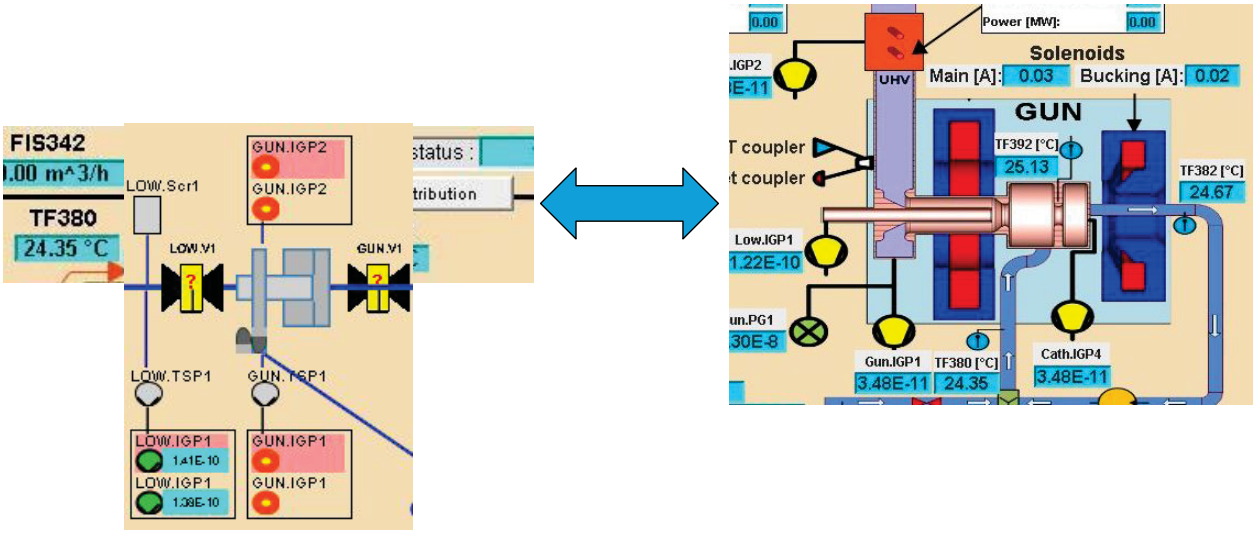
symbolisch gegen realistisch





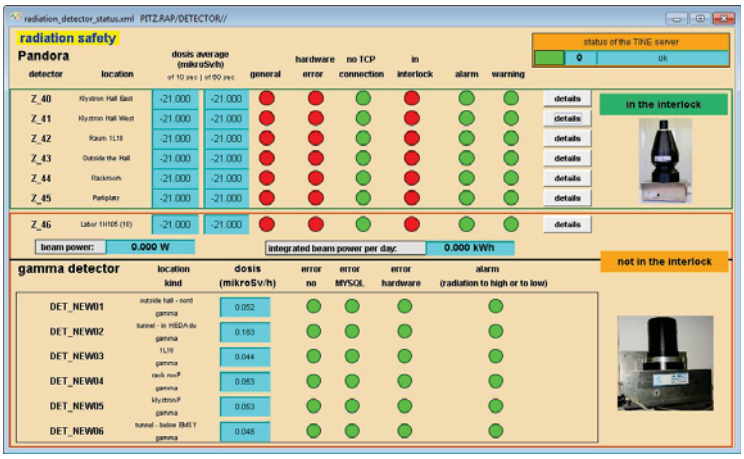
Arten des Programmierens mit jddd - 2

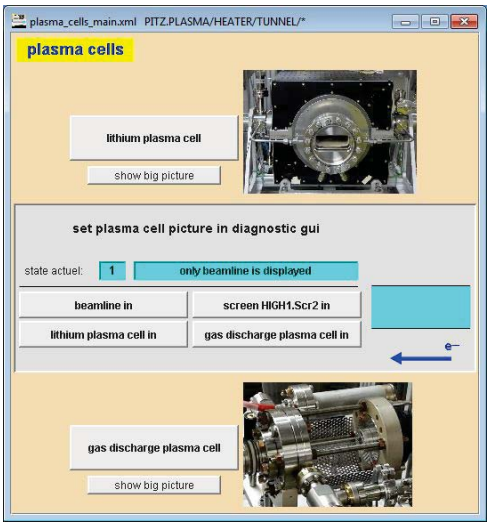
symbolisch gegen realistisch



Arten des Programmierens mit jddd - 3

- realistisch, wenn Realität notwendig ist
  (dann auch nur so realistisch, wie nötig)
- realistisch, wenn Realität nett ist und nicht stört





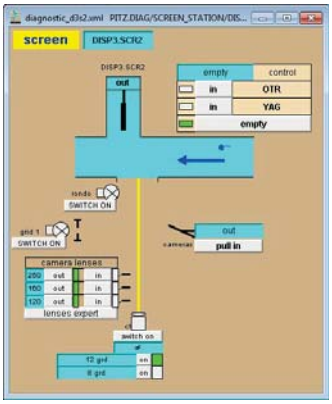
Arten des Programmierens mit jddd - 4

Standardisierung (mit Bagrat und Jörg)

gut für den Operator:
- leichter erkennbar für den Operator
- weniger Bedienungsfehler

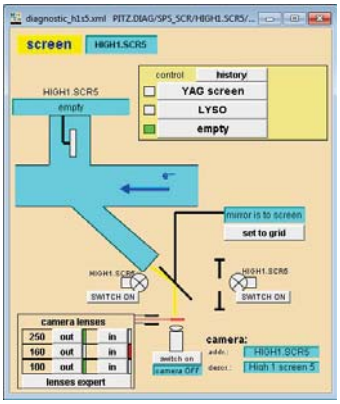
gut für den Entwickler:
- schnellere Entwicklung
- weniger Entwicklungsfehler

aber:
- Standardisierung beginnt bei der hardware(!)
- Alle Zwischenschritte von der Hardware zur gui müssen standardisiert werden.
- Nur die wirklich notwendigen Dinge dem Operator anzeigen.
- „einfach“ ist erfolgreich



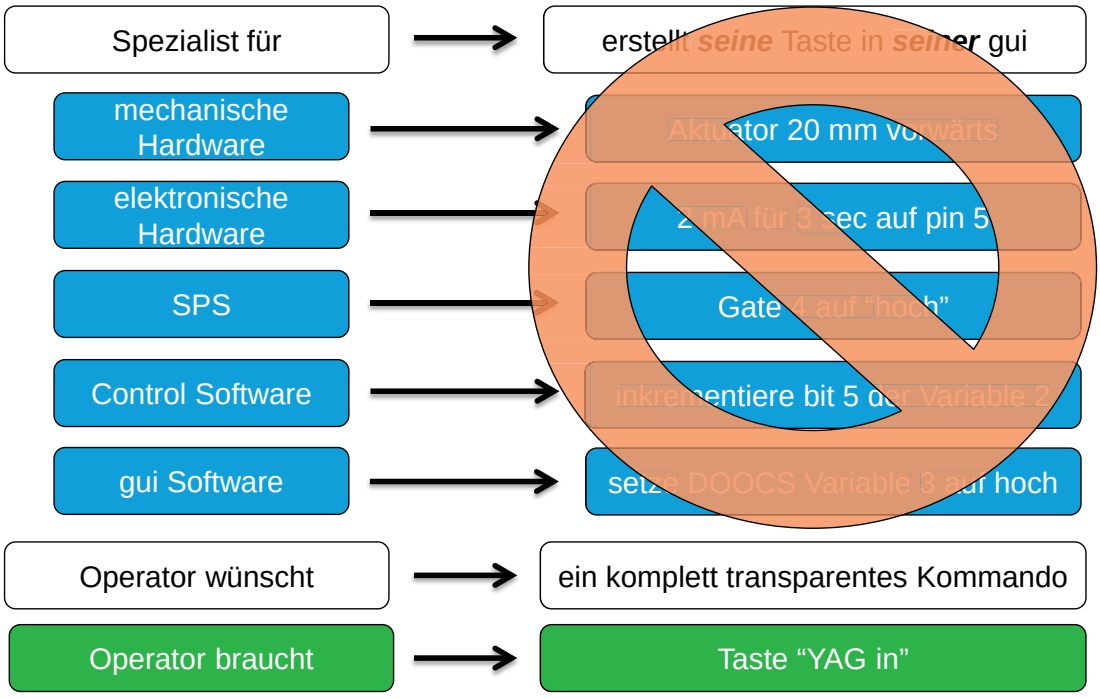
standardisiert (neu)

speziell (alt)



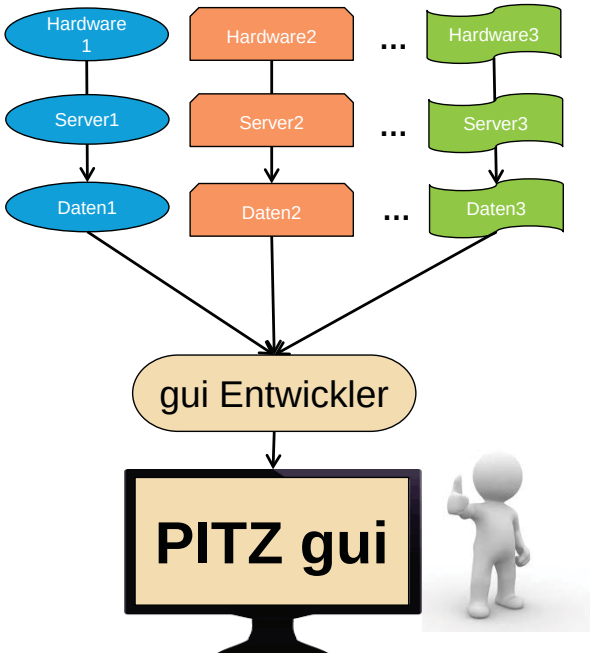
Wer programmiert die gui? - 1

Spezialist Komponente versus Spezialist jddd : “YAG-Schirm in den Strahl”

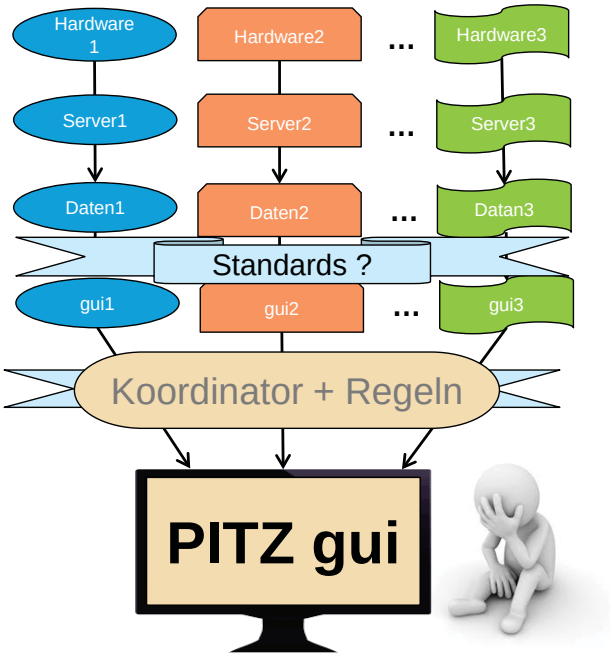


Wer programmiert die gui? - 2

ein Entwickler



oder viele Entwickler

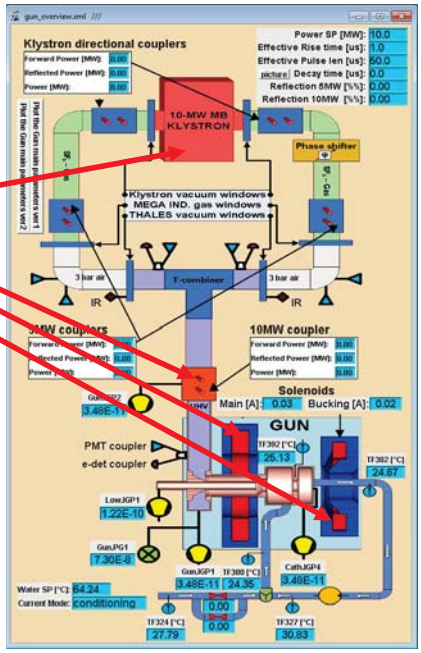


Wer programmiert die gui? - 3

als Beispiel eine erste Regel:

Kein **rot**, außer für Fehler oder Gefahr.

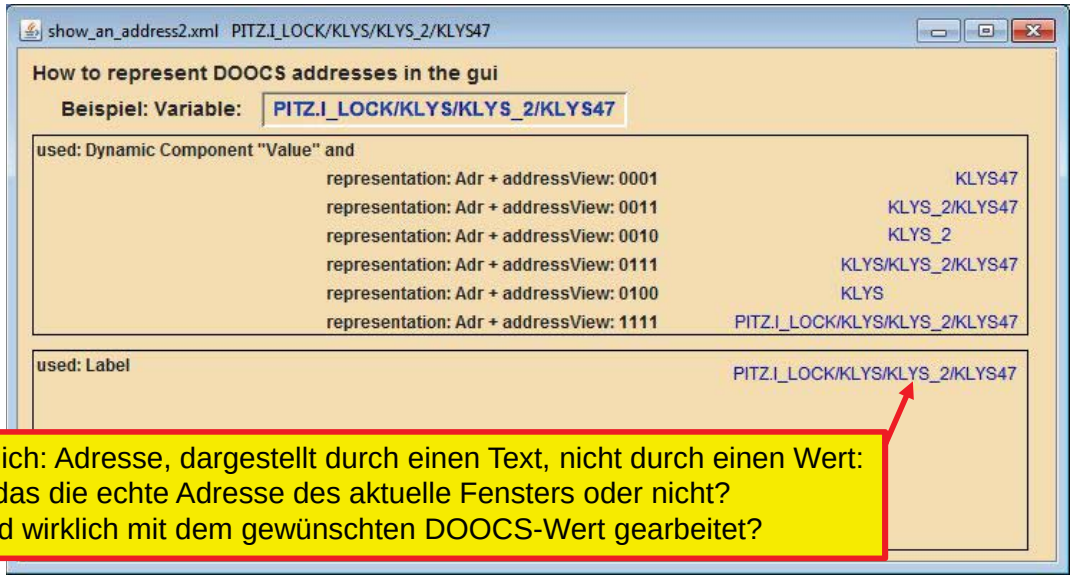
aber



Wer programmiert die gui? - 4

eine zweite Regel:

Wenn ein DOOCS-Adresse angezeigt werden soll, dann als „ausgelesener Wert“, nicht als „fest geschriebener Text“.



gefährlich: Adresse, dargestellt durch einen Text, nicht durch einen Wert:
- Ist das die echte Adresse des aktuelle Fensters oder nicht?
- Wird wirklich mit dem gewünschten DOOCS-Wert gearbeitet?

Wer programmiert die gui? - 5

eine letzte Regel:

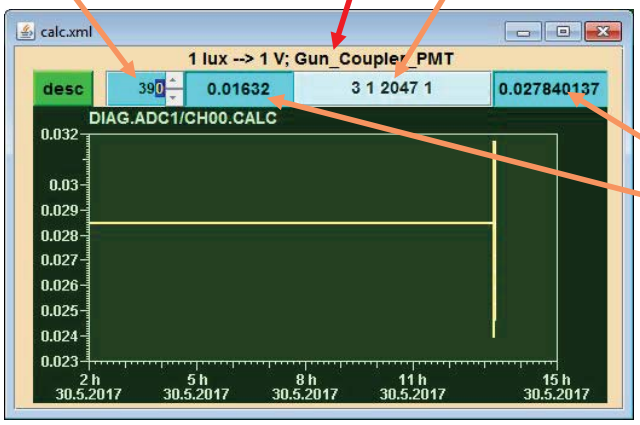
Die Art der dargestellten Variable muss immer auch grafisch erkennbar sein.

Diese Regel muss in der gesamten gui gelten.

änderbarer Wert

ausgelesener Wert! (so nicht erkennbar)

editierbarer Textbereich



ausgelesene Werte

offene Dinge

offen

- durchgehende Standardisierung, z.B. Diagnoseschirme (Viele von denen sind es, aber noch nicht alle.)
- Dokumentation (ist die notwendig?)

jddd - Wiki am DESY Zeuthen - 1

das Wiki

1. https://wiki-zeuthen.desy.de/JDDD
2. erstellt von Winfried Köhler
3. gefüllt von Bert Schöneich und Winfried Köhler
4. Inhalt
   - generelle Bemerkungen zur Nutzung von JDDD
   - Tipps und Tricks
   - bekannte JDDD Fehler und ihre Umgehung
5. Unterstützung für den Entwickler
   - praktische Tipps
   - Erfahrungen
   - „Standards“ für Zeuthen
   - Techniken
   - Fehler und ihre Umgehung / Vermeidung

jddd - Wiki am DESY Zeuthen - 2

andere hilfreiche Webseiten

1. https://jddd.desy.de/
   - Einführung
   - spezielle Möglichkeiten
   - Bildschirmabzüge
   - Beispielfenster
   - Hilfen
   - Frage und Antwort
2. http://tesla.desy.de/doocs/doocs.html
3. https://www-zeuthen.desy.de/pitz/apps/
   - PITZ java Anwendungen
   - Starthilfen
4. http://pitz.desy.de/

Ende

1. März 2018

Wer macht diese Arbeit weiter?

(Die Antwort ist nicht 42.)