



General approach to automation of FLASH subsystems









- Motivation
- Nature of the problem and underlying information
- Required capabilities
- Basic formalization of problem domain
- Anatomy of the solution





Motivation



Necessity of uniform approach to design and implementation of automation software for high energy experiments.

Ultimate role of the automation software:

- Maximization of lasers availability.
 - Automation of routine activities as startup, shutdown ...
 - Continuous monitoring of hardwares condition.
 - Human error minimalization.
 - Autorecovery from specific trips.
- Improvement of breakdown cause location.

Important software engineering challenges:

- Standardization of design, implementation and documentation of automation software for laser subsystem.
- Development of formal verification and testing procedures, thereby improvement of reliability, predictability and safety of the software.







- Online measurements and status signals form the hardware and their meaning
- Definitions of operation modes of the subsystem
- Definitions of known fault patterns and remedial measures
- Primitive procedures and actions that are primary tools of the automation





Basic formalization



• Observables:

Input DOOCS signals of types int, bool, symbolic.

Quantized condition:

Symbolic variable with limited domain. Its value is based on values of the observables

- State of the subsystem: Valuation of all quantized conditions
- Operation mode:

Valuation of set of quantized conditions

• Procedure:

Procedure + valuations that its execution entails

• Long run sequence:

Sequence of procedures bringing the subsystem from current to the desired operation mode.

• Fault pattern:

Valuation of set of observables

• Exception:

Classified fault pattern with ascribed remedial procedure.





Automation capabilities



Required capabilities:

- To automate achieve and maintain pre-selected operation mode
- To adapt to unpredictably changing conditions of the hardware
- To recover the subsystem from known faults.
- To persist in above processes to reasonable extent





Anatomy of the solution - planner



Ensemble of:

- State estimator ullet
- Transition relation (state space) •
- Planning (path finding) algorithm ullet
- **Execution engine** ullet



est st(S1) planseq(S1,S2) exec(p1) est st(S1) planseq(S3,S2) exec(p2)



Planner execution engine



Bogusław Kosęda



Planner's algorithms



 $T=O(b^d)$

 $S=O(b^d)$



Best first search



f(n) = wi + h(n)h(n) = w1 + w2 + ...

 $T=O(b^d)$ $S=O(b^d)$ $T=O(d^b)$ $S=O(d^b)$

Now used:

Breadth first search

- Depth first search with iterative deepening T=O(d),S=O(b^d)
- Cycles prevention
- Hill climbing



DESY

Anatomy of the solution – exception handler



Role of exception handler execution engine:

- Discovering the fault patterns
- Conflict resolution based on faults categorization
- Execution of remedy procedure for encountered faults







Pattern matching + conflict resolution



Conflict resolution:

- Fault arbitration based on
 - Categorization
 - Order of definition in specification file

Classes of faults:

- Warnings
- Recoverable
- Not recoverable

Exception handler's policy:





Cooperation scenarios



Scenario 1

- SE cannot estimate state
- indicate "incomplete"
- ask if EH can solve problem
- not
- remain in "incomplete"
- SE estimated state
- go back to automatic operation

Scenario 2

- exceptions detected
- disable planner
- planner is suspended
- choose exception with the highest priority
- execute remedial procedure
- no exceptions
- awake planner
- planner back in operation













Implementation - planner









Implementation – exception handler









Twofold deployment



📲 Planning and execution 📃 🗆 🗙					
Planning and execution Planned sequence					
Execute step					
Refresh					
Simulation / testing					
Simulation					
Testing					

			Symb	olic tester	· for	automation			
	Symbolic :	state conditi	ons.	<u>P</u> lanning a	ind e	xecution.	eration	modes.	<u>E</u> xit.
Symb.	olic propert	ies			×.	Opera	ation m	odes	
MODULATOR STATUS:	<u>O</u> N	OEF	<u>R</u> ECOV	ERY		WORKING HI	lode		
<u>H</u> V LVL: <u>R</u> F GATE STAT:		<u>T</u> ECH <u>C</u> LOSED	LOW			WORKING LOW NOT WORKING MACHINE READY			

🔲 bogus@mskbkoseda: ~/grammar 💷 🛛
<u>P</u> lik <u>E</u> dycja <u>W</u> idok <u>T</u> erminal Zakła <u>d</u> ki Pomo <u>c</u>
bogus@mskbkoseda: ~/grammar 🛛 🗙 doocsadm@mskbkoseda: ~/doocs/L 🗙
<pre>% operators.pl compiled 0.00 sec, 3,400 bytes % statespace.pl compiled 0.00 sec, 4,340 bytes % pathfinder.pl compiled 0.00 sec, 3,892 bytes % exec_scheme.pl compiled 0.00 sec, 6,700 bytes Warning: (/home/bogus/grammar/gui.pl:129): Singleton variables: [DlgRef, MenuRef] Warning: (/home/bogus/grammar/gui.pl:130): Singleton variables: [Cond] Warning: (/home/bogus/grammar/gui.pl:134): Redefined static procedure pretty_print/1 % gui.pl compiled 0.00 sec, 11,568 bytes % loader pl compiled 0.00 sec, 208 844 bytes</pre>
Yes ?- estimate_state(X).
X = [eq('MODULATOR STATUS', 'ON'), eq('HV LVL', 'TECH'), eq ('RF GATE STAT', 'CLOSED')] □







To be continued ...



Thank You

