Gruppo Software Infrarosso – Progetto Giano

Proposta per la controllistica della nuova preslit di Giano

Firenze 2 Marzo 2015
C.Baffa, E.Giani

Arcetri thecnical Report 1/2015

Sommario

In previsione del possibile spostamento di Giano al Nashmit B, con il conseguente rifacimento della ottica di pre-slit, vengono qui esposte in breve tre differenti proposte per il software di controllo dei necessari attuatori e sensori. Vengono anche discusse le conseguenze del limitato manpower disponibile.

Le tre proposte sono:

1. *Sostituire la presente struttura con alcuni semplici driver a linea di comando, con interfaccia sincrona con IDL, eliminando il PC intermedio.*
2. *Estendere e modificare la presente struttura (demone Xill sul pc intermedio), rimane l'interfaccia asincrona IDL-Java-Balor,*
3. *Sostituire la presente struttura con un sistema di controllo a standard industriale (control framework), con interfaccia sincrona Java.*

Il presente documento vuole esporre una breve analisi, un 'executive summary', delle varie alternative allo scopo di permettere al management di Giano una scelta informata tra i vari approcci.

1 Condizioni al contorno

Le condizioni al contorno che descriveremo qui sono di due tipi: la struttura da controllare e i vincoli software/hardware del sistema di controllo e il manpower disponibile.

1.1 La struttura da controllare.

Il disegno ottico della nuova pre-slit non è ancora definitivo, quindi i dettagli delle movimentazioni non sono ancora noti. Il tipo di movimenti e sensori sarà analogo a quello delle versioni precedenti, e si baserà su movimentazioni della PI, su schede custom (Powerboard, sviluppate in casa) e su sensori della Ueye (camera). Il numero di movimentazioni sarà compreso tra 5 e 10. Tale incertezza non permette di dare valutazioni sensate sulla quantità totale di lavoro necessario. Tuttavia, quando possibile, cercheremo di dare, nelle varie eventualità, delle valutazioni parametriche. Se nel progetto entrassero anche uno o più assi veloci (ad esempio tip-tilt), la presente analisi andrebbe rieseguita.

1.2 Manpower disponibile

Il gruppo Software dell'Infrarosso (Softir) è fortemente impegnato per almeno i prossimi due anni nello sviluppo di due moduli per il Central Signal Processor del Radiotelescopio SKA. Tale impegno, nell'ambito di un progetto fortemente formalizzato come SKA, lascia poco spazio ad altre attività.
Pertanto il gruppo Softir può impegnarsi **al massimo per il 30% del tempo di una persona** (valore medio) e per **non più di sei mesi**. Inoltre, per permettere la riprogrammazione degli altri impegni, è necessario che i periodi di attività siano preannunciati con settimane di anticipo e non saranno probabili "rush finali". Le proposte seguenti sono state formulate tenendo conto di tale pesante vincolo.
Notiamo come i tempi di sviluppo ipotizzati nel seguito siano espressi in termini di settimane uomo, cioè assumendo un impegno al 100%.

2 I tre approcci proposti.

Nel seguito descriveremo tre diverse architetture che, a nostro avviso, permettono la realizzazione del sistema di controllo con i vincoli descritti sopra. Daremo anche una breve lista degli aspetti positivi e negativi di ogni scelta.

2.1 Haiku, la scelta minimalista

La ricerca di una soluzione minimalista ha prodotto la proposta di sostituire la struttura software attualmente in uso[1] con una piccola serie di programmi a linea di comando, uno per ciascun asse da controllare. In sostanza per l'ipotetico attuatore 'filtroblu', ci sono i seguenti comandi:

- inizializzazione g_filtroblu --init [-vY]
- movimento g_filtroblu --move XX [-vY]
- stato g_filtroblu --status [-vY]

Il valore XX indica la posizione da raggiungere, mentre Y indica la verbosità (quanta informazione viene mostrata a terminale). Questo tipo di interfaccia ha vari pregi. Utilizza esattamente lo stesso codice sia in laboratorio che al telescopio, ha una struttura semplice ed ogni asse è indipendente, sia come esecuzione che come sviluppo. Se vengono usati attuatori già presenti nel sistema attuale, il driver specifico può essere, almeno in parte, riutilizzato. Inoltre lo sviluppo di tale struttura limita il problema delle interazioni tra i vari driver semplificando lo sviluppo.
L'interfaccia verso la GUI, scritta in IDL, è assai semplice, sincrona, e non ha bisogno di alcun bridge aggiuntivo (ad esempio via Java). I comandi restituiscono un *codice di ritorno* con il seguente significato:

- Se positivo: la posizione raggiunta
- Se negativo: l'errore ottenuto

Il comando IDL per tutte le operazioni è del tipo[2]:

*SPAWN, '*g_filtroblu -v0 --init*', listing, EXIT_STATUS=status*

Tale comando inizializza l'asse filtroblu e restituisce l'esito dell'operazione nella variabile status.
Questo tipo di interfaccia, semplice e sincrona, ha due aspetti limitativi. Il primo è che, essendo la chiamata sincrona e *bloccante*, può essere eseguita una sola operazione alla volta. Tale difficoltà può essere aggirata o utilizzando la tecnica di backgrounding dei processi, o con l'uso della funzione di multithreading (IDL_IDLBridge[3]) descritte nel manuale IDL, oppure tramite un *wrapper,* scritto in *bash*, che esegue più comandi alla volta. La seconda limitazione è che essendo tale interfaccia priva di 'stati', non ha né memoria della configurazione, né la capacità di convertire la posizione numerica in una stringa[4]. Tali semplici funzioni possono comunque essere spostate senza difficoltà alla interfaccia IDL con un impatto limitato. Un vantaggio importante di questa scelta è che il codice è poco dipendente dai dettagli del funzionamento ottico della preslit.

1 La struttura in uso comprende: drivers software ad hoc nel demone Xill, un pc di servizio, *Xill-pc*, un'interfaccia tecnica, Titan, ed un demone di interfaccia complessiva verso la GUI, Balor
2 http://www.exelisvis.com/docs/SPAWN.html
3 http://www.exelisvis.com/docs/IDL_IDLBridge.html
4 Questi comandi potrebbero eseguire la conversione tra numeri ed etichette, ma a scapito della semplicità dell'interfaccia.

Tale struttura molto semplice funziona meno bene nel caso di motori con interfaccia USB. In questo caso occorrerebbe un calcolatore che facesse da 'ripetitore'. Tale compito molto leggero potrebbe essere svolto o dall'attuale macchina *Xill-pc*, o più semplicemente da un calcolatore della classe dei Raspberry PI. L'interfaccia verso le powerboard (accensione e spegnimento dei vari attuatori) può essere implementata o tramite un comando a sé stante o, più semplicemente, inclusa nei vari comandi.

Il tempo di sviluppo è dell'ordine di 2-4 settimane-uomo per la struttura complessiva e 1-2 settimane per ogni asse, con tempi più brevi per assi di cui disponiamo già del codice di controllo. Nel caso di interfacce USB i tempi possono essere maggiori. L'interfaccia IDL può essere sviluppata separatamente e provata in remoto (con limitati rischi di sicurezza). Questa struttura è completamente nuova, quindi c'è il rischio, anche se ridotto, di un allungamento imprevisto dei tempi di sviluppo.
Nel caso di una modifica, durante lo sviluppo in laboratorio, della struttura logica della pre-slit, l'impatto su tale soluzione è assai limitato.

Pro	Contro
Struttura molto semplice, veloce da sviluppare e semplice da manutenere.	Limiti alla flessibilità ed alla potenza del sistema
Modulare, una parte non interagisce con altre parti né le ostacola	Alcune funzioni spostate ad IDL [5]
La semplicità della struttura facilita un eventuale trasferimento della responsabilità del codice	Necessità di un wrapper o altro artificio per muovere più di un asse alla volta
Quasi totale indipendenza dai dettagli del disegno ottico.	In laboratorio non vi è display dello stato complessivo del sistema
Stesso codice usato in laboratorio ed al telescopio	Struttura completamente nuova: rischio di un allungamento imprevisto dei tempi di sviluppo.
Interfaccia tecnica pronta da subito.	Interfaccia tecnica assai spartana.
Un completo log in ASCII	Qualche complessità nella gestione delle powerboard
Protocollo di comunicazione in ASCII	

5 memoria dello stato del sistema, conversioni step-nome simbolico, scrittura dello stato della preslit su FITS.

2.2 La Gattopardata: mantenere la situazione presente

La seconda alternativa consiste nel mantenere la struttura di controllo presente. Anche se il diavolo che si conosce fa meno paura, questa alternativa presenta alcuni problemi di non ovvia soluzione.

La presente soluzione si basa su di un demone, Xill, che gira su un PC dedicato (*Xill-pc*). All'interno del demone è presente una struttura multi-tasking, dove ogni thread controlla un asse, in modo indipendente dalle altre operazioni. Questa struttura è potente e permette l'estensione ad un numero elevato di attuatori, che possono agire senza difficoltà in modo concorrente. Tale demone produce un log ASCII, ed è controllato tramite il protocollo proprietario IRLab (Binario) dal demone centrale di controllo di Giano (Balor)[6].

Il controllo in laboratorio avviene tramite la GUI tecnica Titan, che è stata sviluppata per permettere la gestione di ogni aspetto del sistema di acquisizione di Giano, ed è quindi complessa. Inoltre, essendo un'interfaccia custom, occorre riadattarla ad ogni modifica dello hardware, in modo da rifletterne il funzionamento. Titan, così come la GUI del telescopio, controlla il demone Xill attraverso il demone Balor.

Il controllo al telescopio avviene tramite il demone centrale di controllo di Giano (Balor) utilizzando il il protocollo proprietario IRLab (Binario). Per accedere a questo canale IDL utilizza un'interfaccia, scritta in Java. Il protocollo IRLab è intrinsecamente asincrono, quindi è stato necessario un lavoro di adattamento, che prevede anche un registro esterno (file su disco) per gli eventi asincroni e gli errori. Siccome questo approccio correttamente prevede che la struttura del software rifletta la struttura dello hardware da controllare, la modifica del numero e delle funzioni dei vari assi deve riflettersi anche nei dettagli del protocollo tra Balor e la GUI, con ovvie ricadute negative sui tempi di sviluppo.

Il lavoro di adattamento alla nuova pre-slit richiede l'adattamento di differenti sezioni del codice. Uno sviluppo tortuoso dell'ottica di pre-slit si rifletterebbe in un continuo lavoro di modifica della struttura software dei controlli. Comune a tutte le proposte è lo sviluppo dei driver per i nuovi dispositivi. Il cambiamento di struttura e di funzionamento della pre-slit deve riflettersi anche nella struttura del

6 Tale approccio è diventato meno efficace per il vincolo (aggiunto in un secondo momento) di un controllo rigorosamente sincrono a monte. Questa struttura è pensata per una situazione complessa, ma stabile e diventa di laboriosa gestione di fronte a cambiamenti della struttura fisica della pre-slit.

programma di laboratorio (Titan) che richiede quindi un lavoro di affinamento più o meno continuo. Da ultimo anche l'adattamento della comunicazione tra Balor e la GUI deve, almeno parzialmente, riflettere la struttura fisica della pre-slit.

Il tempo di sviluppo è dell'ordine di 4-6 settimane-uomo per riorganizzare la struttura complessiva, e 1-2 settimane per ogni asse nuovo. A questo va aggiunto circa 1-2 settimane per l'adattamento della struttura interna e il colloquio con la GUI. L'interfaccia IDL può essere sviluppata solo parzialmente in remoto. Tale processo di riorganizzazione va ripetuto per ogni riorganizzazione profonda della pre-slit, mentre modifiche minori hanno bisogno di tempi più ridotti.

Pro	Contro
Struttura già stabilita e provata. Potente e flessibile	Complessivo, occorre attenzione all'interazione tra i vari moduli
Interfaccia di laboratorio che offre un quadro sintetico dello stato del sistema	Complesso da manutenere e difficile da trasferirne la responsabilità manutentiva.
Parte del codice di controllo viene usato sia in laboratorio ed al telescopio	La struttura interna intrinsecamente asincrona si adatta laboriosamente alla struttura IDL della GUI
Protocollo proprietario molto solido ed affidabile.	Non tutta la catena di controllo è comune al telescopio ed in laboratorio
Un completo log in ASCII	Protocollo di comunicazione in binario
La struttura del codice riflete la struttura dello hardware e quindi il codice riconosce correttamente ogni evento, anche agli errori.	Ogni modifica dello hardware richiede il cambio di parti importanti del codice.

2.3 Utopia, il sistema di controllo come andrebbe fatto

La terza proposta di realizzazione del sistema di controllo della pre-slit di Giano si basa su di un ripensamento complessivo della struttura in uso. Un sistema nuovo dovrebbe basarsi su una soluzione moderna, utilizzando quanto l'ambiente del software di controllo può offrire. Questa proposta, assai attraente ma forse non facilmente realizzabile, per la scarsità di risorse umane disponibili, si basa su di un 'frameworrk' standard, Tango[7]. In realtà all'inizio dello sviluppo di Giano il gruppo Softir aveva

7 http://www.tango-controls.org/

ipotizzato l'uso di un framework simile, all'epoca si era valutato Epics[8], ma lo spostamento ad altro gruppo della responsabilità della controllistica, aveva fatto accantonare il progetto. Quando poi tale responsabilità è tornata ad Arcetri, i vincoli di tempo e risorse avevano impedito questo approccio.

Questa proposta si basa sullo sviluppo, su di un PC dedicato (ad esempio *Xill-pc*) di un device server, che sarebbe responsabile del controllo effettivo dei vari devices, e di un'applicazione 'client' sul calcolatore di controllo, dove risiederebbe anche il database di configurazione del sistema.

Per il controllo in laboratorio dei devices vi sono delle applicazioni che permettono di controllare un device generico, in base alle sue proprietà registrate nel database di configurazione, inoltre esistono strumenti per lo sviluppo rapido di pannelli di controllo tecnico. Il controllo al telescopio può avvenire attraverso una interfaccia per Java (Tango Java Client API[9]).

Esiste una discreta libreria di device driver già pronti, e per quanto non già disponibile, vi sono dei generatori di device driver classes semi-automatici sia per C++ che per Java e Python (ad esempio Pogo[10]). In tale libreria sono ben rappresentati anche sensori di immagini (come quello del cosiddetto 'fotometro') e sensori di temperatura, pressione ed un'interfaccia con il Modbus, permettendo, in un futuro, l'estensione di tale struttura anche alla telemetria tecnica (ora gestita da Lillend).

8 http://www.aps.anl.gov/epics/
9 http://www.esrf.eu/computing/cs/tango/tango_doc/kernel_doc/tango_java_api/index.html
10 http://www.esrf.eu/computing/cs/tango/tango_doc/tools_doc/pogo_doc/index.html

Questa struttura, completamente nuova, richiede uno sforzo iniziale di sviluppo maggiore delle altre alternative, ha un rischio intrinseco maggiore, ma offre la possibilità di un ambiente coerente, stabile, robusto ed estendibile. Inoltre, essendo uno standard affermato nel campo della strumentazione scientifica, è possibile trovare personale già addestrato all'uso di tali strumenti, o almeno desiderosi di diventare esperti nel campo.

La valutazione del tempo di sviluppo è più incerta che nei casi precedenti. Il tempo di sviluppo iniziale è valutato essere dell'ordine di 8-10 settimane-uomo per riorganizzare la struttura complessiva, e 1-2 settimane per ogni nuovo attuatore; gli assi per cui esiste già un driver o per cui si può utilizzare quello corrente richiedono tempi inferiori. A questo vanno aggiunte circa 2-4 settimane per l'adattamento della struttura interna e lo sviluppo di una GUI tecnica. L'interfaccia IDL/Java può essere sviluppata in remoto, ma ha bisogno di test in laboratorio prima dell'installazione al telescopio. Parte di tale lavoro può essere svolto in remoto.
Il processo di riorganizzazione non va però ripetuto per ogni riorganizzazione della pre-slit, con evidenti vantaggi in caso di uno sviluppo complesso.

Pro	Contro
Framework diffuso e largamente provato. Potente e flessibile	Elevato investimento iniziale

Pro	Contro
La struttura interna intrinsecamente asincrona si interfaccia alla IDL della GUI tramite API-Java, che invece è sincrona.	Ampia documentazione standard per la manutenzione e il trasferimento della responsabilità manutentiva.
Interfaccia di laboratorio con un quadro sintetico dello stato del sistema facile da sviluppare	Prodotto nuovo per lo strumento, maggiore rischio tecnologico
Il codice di controllo viene usato sia in laboratorio ed al telescopio	Solo pochi attuatori hanno un driver già disponibile
Protocollo proprietario molto solido ed affidabile.	Protocollo di comunicazione in binario
Un completo log in ASCII	
Database storico delle operazioni e delle posizioni	
Riconfigurabile al volo tramite tool grafico	

3 Conclusioni

Lo sforzo di offrire una struttura di controllo più facile da sviluppare e da manutenere ha prodotto la presentazione di due alternative rispetto alla semplice estensione della struttura presente.
Il presente documento vuole esporre una breve analisi, un 'executive summay', delle varie alternative allo scopo di permettere al management di Giano una scelta informata tra i vari approcci.