Tratarea exceptiilor

Codul este adesea scris fara sa se ia in consideratie eventualele erori care pot aparea. Cand apar evenimente fata de care aplicatia nu se asteapta apar probleme. Atunci, in faza de depanare, trebuie sa se revizuiasca codul pentru implementarea unor capcane de erori si corectie, desi de obicei nu este suficient. Tratarea exceptiilor trebuie luata in consideratie inca din faza de inceput a dezvoltarii aplicatiei. Implementarea unei tratari a erorilor duce la un cod cu mult mai robust.

Acest capitol discuta despre erori si despre topica tartarii exceptiilor in Lab VIEW. Pentru inceput, tratarea exceptiilor va fi definita si insotita de rolul ei in aplicatii. Aceasta explicatie va clarifica si importanta tratarii exceptiilor. Dupa aceea, vor fi prezentate diferite tipuri de erori ce pot apare. Aceasta va fi urmata de descrierea utilitarelor disponibile in pachetul Lab VIEW pentru tratarea exceptiilor, ca si unele utilitare de depanare. La sfarsit, cateva moduri diferite de a combate erorile vor fi demonstrate.

6.1 Definitia tratarii exceptiilor

Eceptiile sunt evenimente neintentionate sau nedorite care apar in timpul executiei programului. O exceptie poate fi orice eveniment care in mod normal nu trebuie sa aiba loc. Asta nu inseamna ca apritia unei exceptii este neasteptata, dar nu trebuie sa apara in circumstante normale. O eroare rezulta atunci cand apare ceva ce tu nu ai vrue sa apara. Pentru aceasta, facerea mai multor pasi alternativi de executie are sens cand au loc exceptii. Cand se ivesc exceptii sau erori ea trebuie eleminata intr-o maniera potrivita.

Presupuneti ca ati scris un program in care impartiti doua variabile intregi, x la y. Rezultatul este folosit in alt scop. In unele ocazii, y poate sa fie setat pe zero. Unele programe nu capteaza unele erori de genul impartirea la zero si permite procesorului sa lanseze in executie exceptia. In Lab VIEW rezultatul aceastei exceptii este nedefinit. Lab VIEW returneaza Inf sau infinit, in acest caz. Acesta este exemplul unui rezultat neintentionat si neasteptat. Infinitul poate fi covertit cu succes intr-un cuvant intreg in Lab VIEW. Dacavaloarea este convertita de pentru alte intrebuintari, pot apare si alte erori. Acesta este exemplul unei erori simle care poate fi inlaturata folosind tratarea eceptiilor.

Este nevoie de tratari ale exceptiilor pentru a inlatura unele probleme sau erori care pot apare. Este un mecanism care permite programului sa detecteze si pe cat posibil sa-si revina in timpul excutiei erorilor. Tratarea exceptiior duce la mult cod, planificandu-se dinainte eventualele probleme. Capacitatea unei aplicatii de a raspunde la un eveniment neasteptat este critica. Implementarea unei tratari a exceptiilor duce la un cod mult mai sigur.

Puteti sa va srieti singur codul pentru a incerca sa captati cat mai multe erori, dar asta necesita si mai mult cod de implementat. La un moment dat veti avea mai mult cod inclus pentru a capta erorile decat pentru a duce la bun sfarsit o anumita intrebuintare. Codul pentru tratarea exceptiilor poate contine uneori unele erori. Tu insuti ai creeat o problema cand se capteaza erorile.

Detectia de erori si corectia de erori sunt doua activitati diferite, dar amandoua fac parte din tratarea exceptiilor. Detectia de erori este constituita din cod care gaseste erorile. Corectia de erori este procesul care capteaza si trateaza posibilitatea de aparitie a erorilor. Mai intai tu trebuie sa captati erorile atunci cand ele apar, apoi sa determinati ce actiune sa aiba loc.

Reprezentarea detectiei de erori este folositoare la depanarea codului in timpul fazei de testare si integrare. Plasarea unor verificari de erori in cod va ajuta la gasirea greselii in timpul fazei de test. Acelasi mecanism poate juca un rol dual. Mecanismul de detectie poate controla transferul la tartarea erorii cand tratarea este dezvoltata. Acesta va fi benefic daca folositi un model de dezvoltare interativ, cand se pot adauga si chestii noi in fiecare ciclu.

Tratarea exceptiilor se comporta putin diferit in diferitele limbaje de prograanare. Java utilizeaza clase de exceptii unde codul pentru tratare poate fi scris. De exemplu, o exceptie este prezenta prin intermediul unei clase “Throwable” sau de una din subclasele acesteia. Acest obiect este folosit pentru a transporta punctul unde s-a produs exceptia la tratarea care o capteaza. Programatorii pot de asemenea sa-si defineasca propriile clase de exceptii pentru aplicatiile lor.

C++ foloseste cuvinte cheie pentru tratarea exceptiilor: Try, Catch si Throw. Cuvintele cheie Try si Catch identifica blocul de cod. Comenzile Try forteaza aplicatia sa-si reaminteasca locatia curenta in stiva de apelari si sa realizeze testul pentru a detecta eroarea. Cand apare o exceptie, executia trece direct la blocul captat. Dupa ce blocul captat a fost executat, stiva de chemari va fi “rulata inapoi” la punctul program unde se afla blocul Try.

Lab VIEW furnizeaza cateva utilitare pentru detectia de erori. Dar ca si toate celelate limbaje de programare, implementarea unei tratari a exceptiilor ramane la latitudinea programatorului. Urmatoarele sectiuni va vor ghida in crearea codului pentru tratarea erorilor pentru aplicatiile dumneavoastra. Capitolul 10 acopera topici cu referire la Programarea Orientata pe Obiect, incluzand definitiile obiectelor, claselor si subclaselor, dar tratarea exceptiilor in Java si C++ nu fac capitolul acestei carti.

6.2 Tipuri de erori

Erorile care apar in programele scrise in Lab VIEW pot fi de tipul I/O sau logic. Erorile I/O sunt acelea cand ezulta la incercarea unui program de a efectua operatii cu instrumente exterioare, fisiere, sau alte aplicatii. O eroare logica este rezultatul unui defect in codul programului. Exemplul de mai inainte cu impartirea unei valori intregi la zero este o eroare logica. Aceste tipuri de erori sunt foarte delicat de gasit si corectat. Amandoua, I/O si logice, relateaza erori care vor fi discutate in sectiunile urmatoare.

6.2.1 Erorile I/O

Input/Output incorporeaza o suprafata foarte mare de activitati si Vis in Lab VIEW. Chiar daca folositi Vis pentru comunicatii (TCP, UDP, DDE, ActiveX, OLE, PPC, AppleEvent), achizitii de date, instrument I/O, sau fisier I/O, exista posibilitatea sa va loviti de erori.

Erorile I/O pot fi consecinta mai multor lucruri. Prima circumstanta care poate duce la o astfel de eroare este initializarea sau configurarea canalului de comunicatie necorespunzatoare. De exemplu, cand faceti comunicatie seriala, viteza de transmisie trebuie sa se potriveasca intre controler si dispozitivul extern. Daca aceasta initializare se face incorect rezulta erori. Pentru majoritatea dispozitivelor comanzile trebuie trimise pentru a fi puse in modul distant, ceea ce permite comunicatia cu controlerul. Cand se citeste sau se scrie intr-un fisier, fisierul trebuie mai intai sa fie deschis. Similar, cand se scrie intr-o baza de date, o coneziune trebuie relizata inainte de a

insera datele. Initializarea poate sa includa, de asemenea, punerea unui instrument sau dispozitiv intr-un stadiu cunoscut. Cateodata aceasta poate fi facuta prin resetare, dupa care dispozitivul va aduaga valorile implicite.

O a doua cauza a erorii I/O este trimiterea unor comenzi gresite sau date gresite instrumentului sau aplicatiei. Cand s-a trimis informatie invalida, va apare o eroare de scriere. Unele dispozitive pur si simplu ignora in timp ce altele returneaza o necunoscuta. Acest lucru joaca un rol important cu referire la ce tip de corectie sau tratare folositi. Cand datele sunt trimise la un dispozitiv extern, trebuie sa va asigurati ca si datele corecte si formatul corect sunt trimise. Trebuie sa ajustati informatia pe care o trimiteti pentru a se potrivi cu ce poate dispozitivul sa primeasca. Erorile tipografice pot fi clasificate tot in aceasta categorie.

O alta eroare I/O are loc cand exista o problema cu instrumentul sau aplicatia in uz. Cand lucrati cu aplicatii sau fisiere, aceasta poate apare din mai multe motive diferite. Fisierul poate ca nu exista in cale specificata. In alta ordine de idei, poate ca nu aveti permisiunile de scriere sau citire asupra fisierului. Erorile I/O ale instrumentelor de acest fel apar de obicei cand instrumentul nu este alimentat de la sursa sau cand nu functioneaza corespunzator. O problema similara apare cand intrumentul se blocheaza sau ingheata. O realimentare poata sa-l returneze la un stadiu cunoscut si sa-l faca din nou operational. Aceste tipuri de erori pot rezulta din configurarea incorecta a dispozitivului extern. Instrumentele pot face masurari necorespunzatoare cand nu sunt configurate corect.

Lipsa hardware-ului sau software-ului poate fi o sursa de erori I/O. Sunteti nevoit sa verificati daca aveti instalate corect driver-ele interfetei. Incompatibilitatea interfetei si a componetelor trebuie investigata.

6.2.2 Erori logice

Erorile logice apar cand sunt greseli in cod. Codul din diagrama Figure 6.1 ilustreaza o greasala inocenta care poate apare. In bucla While, programatorul intentioneaza sa opreasca executia cand temperatura atinge 75.0 grade sau mai mult. Bucla, asa cum este ea, va opri daca temperatura este mai mica de 75.0 grade. Acesta este exemplul unei erori usoare care poate cauza o eroare in aplicatie. Aceste tipuri de erori sunt dificil de gasit si sunt mari consumatoare de timp. Utilitarele de depanare sunt inutile in gasirea unor surse de erori.

Erorile por apare uneori cand datele introduse de catre utilizator nu sunt validate. Daca utilizatorul nu prevede ce date permite programul, poate apare o eroare. Aplicatia trebuie sa valideze datele pentru a fi sigura ca nu se abate de la normal. De exemplu, utilizatorul poate sa specifice care numar de unitate, intre unu si zece, la care sa aplice testele. Programul trebuie sa verifice daca sunt introduse date doar din zona acceptata inaite de a incepe executia. Unitatea zero poate sa nu existe pentru teste, dar totusi codul trebuie sa verifice datele corespunzatoare. Feritiva de erori de precizie numerica si erori de conversie care sunt dificil de depistat.

Lab VIEW permite utilizatorului sa seteze regiunile acceptate pentru controlerele Numeric, Boolean si List & Ring. Aceasta poate fi facuta prin apasarea controlerului si selectarea Data Range din meniu. Programatorul are de asemenea optiunea de a constrange valoarea introdusa in asa fel incat sa fie in zona valida. Aceasta optiune este disponibila in fereastra drop-down. Optiunea de a constrange reduce nevoia de a scrie cod si de a indeplini niste sarcini.

6.3 Erorile incluse

Lab VIEW anunta utilizatorul de unele erori din timpul executiei pentru instrumente si operatii de I/O pentru fisiere prin casete de dialog. Lab VIEW nu infrunta erorile si, in general, lasa tratarea exceptiilor in seama programatorului. Cu toate acestea, Lab VIEW furnizeaza programatorului cateva utilitare pentru a vindeca in cadrul tratarii exceptiilor. Primul utilitar despre care se va discuta va fi pachetul de erori. Pachetul de erori este folosit pentru a transporta informatia de la mecanismul de detectie la tratare. Dupa pachetul de erori, va fi prezentata o descriere VISA de tratare a erorilor pe scurt. Dupa aceea, tratarea Vis a erorilor va fi luata in considerare. In particular sunt trei tipuri de erori Vis: Simple Error Handler VI, General Error Handler VI si Find First Error VI.

Sectiunea 6.4 va discuta despre implementare codului pentru tratarea erorilor.

6.3.1 Pachetul erorilor

Pachetul erorilor este un mecanism de detectie furnizat pentru programatori. Pachetul se compune dintr-un statut, cod si sursa. Fiecare dintre acestea furnizeaza informatii despre aparitia erorilor. Statutul este boolean care returneaza “true” daca a aparut o eroare. Codul care distinge erorile este pe 32 de biti. Sursa este un sir care da informatia despre originea erorii. Pachetul de erori ca un tot furnizeaza detalii de baza despre eroare care pot fi folosite in scopul tratarii erorilor.

Figura 6.2 arata pachetul de erori de intrare si erori de iesire cand apar in panoul principal. Pachetul de erori de intrare si erori de iesire pot fi accesate din subcategoria Array & Cluster in categoria “pallete”. Pachetul de erori este bazat pe conceptul de erori de I/O ale “National Instruments”. Vis care utiizeaza acest concept are indicatoare despre erorile de intrare si erorile de iesire, care sunt de obicei localizate in partea de jos a panoului principal. Informatia pachetului este trecuta succesiv prin Vis intr-o aplicatie, consecvent su “data flow programming”.

Pachetul de erori poate servi in scopuri duale in aplicatia dumneavoastra. Prin folosirea erorilor I/O, ordinea de executie a Vis-ului poate fi fortata. Aceasta elimina nevoia de structuri succesive pentru controlul ordinei de executie. Mai simplu, puneti pachetul de erori in Vis pentru detectie si ordine.

Cand pachetul este pus in VI, VI-ul verifica daca a aparut o eroare. Daca nu exista vreo eroare executia va continua. Pachetul culege informatie daca erorile apar in timpul executiei VF-ului si plaseaza aceasta informatie la urmatorul VI, care urmeaza aceleasi verificari. In cel mai simplu caz, daca erorile apar intr-un VI, VI-ul care il urmeaza nu ar trebui sa se execute. Cand preogramul se termina, erorile sunt afisate in panoul principal.

Conceptul de erori I/O si pachetul de erori sunt usor de folosit si incapsulat in aplicatii. Multe dintre VI-urile Lab VIEW-ului sunt disponibile in paletele de functii si sunt bazate pe acelasi concept: paleta Vis de comunicatie (TCP, UDP, DDE, ActiveX, HiQ), majoritatea dintre VI-urile intrumentelor I/O (VISA, GPIB, GPIB 488.2), si unele achizitii de date si VI-urile de fisiere I/O folosesc I/O. Prin folosirea acestor VI-uri si racordate la pachetul de erori, mult din munca pentru detectia de erori este gata pentru programator. Aceste VI-uri incluse furnizeaza detectia necesara in stadiul de jos a operatiei. Cand se racordeaza acest VI la diagrama codului, veti observa ca terminalul erorii de intrare este in partea din stanga jos a VI-ului, in timp ce terminalul erorii de iesire va fi in partea din dreapta jos. Aceasta este o conventie urmata de majoritatea fabricantilor de VI-uri de la National Instruments, si sunt de asemenea recomandate cand faceti drivere.

Figura 6.3 este un exemplu despre cum poate fi utilizat un pachet de erori. VI-ul foloseste GPIB Write si GPIB Read pentru paleta de instrumente I/O. Este un driver simplu pentru instrument care poate sa citeasca si sa scrie de le un instrument. Pentru a realiza o detectie, pachetele de erori de intrare si iesire si sa le racordeze corespunzator in diagrama codului. Detectia de erori este lasata in seama VI-ului instrumentului I/O. Cand un driver este necesar ca o parte dintr-o aplicatie ampla, este folosit conceptul erorilor I/O. Figura 6.4 foloseste doua drivere pentru erorie de iesire si erorile de intrare racordate. Al doile VI nu se va executa atat timp cat apare o eroare in primul VI. Ordinea executiei este fortata, cauzand primul driver sa astepte pachetul de erori de la primul. Acest mod poate fi aplicat si la aplicatiile ample.

Pachetul de erori poate fi de asemenea folosit pentru a aplica niste verificari altele decat cele facute de VI-urile disponibile in Lab VIEW. Presupuneti ca comunicati cu un dispozitiv sau aplicatie care returneaza necunoscute la trimiterea de comenzi sau date. Valoarea “OK” este returnata cand datele sunt valide si acceptate, si valoarea “NOK” cand datele sunt invalide sau comenzile sunt necunoscute. VI-ul Lab VIEW-ului nu aplica nici o verificare asupra instrumentului sau aplicatiei specifica cunoscutelor, decat la erori de comunicatie generale. Intorcandu-ne la VI-ul din exemplul anterior, putem sa implementam verificarea proprie asupra erorilor. Figura 6.5 arata cum aceasta este posibil.

Bundle by Name este folosit pentru paleta Cluster pentru indeplinirea acesteia. Daca constatarea returnata nu este “OK”, atunci informatia pachetului de erori este alterata. Boolean devine “true”, asignarea codului este 6000, si descrierea sursei este racordata. Lab VIEW isi rezerva erorile 5000 si 9999 pentru erori de utilizator. Daca constatarea se potriveste cu valoarea care se astepta, racordam pachetul de erori in cazul “true” direct la Error Out fara alteratii. Detectia de erori pentru constatarea corecta va fi aplicata de fiecare data cand driverul este apelat.

Figura 6.6, Extra Source Info.vi, arata un exemplu despre cum sa culegi mai multa informatie in scopul depanarii si tratarii erorilor. Acest VI adauga informatie suplimentara la sirul sursei din pachetul de erori. Mai intai pachetul de erori este desfacut folosind Unbundle by Name. Partile de informatie in plus care vor fi adaugate include timpul cand s-a produs eroarea si lantul de apelari. Call Chain, valabil in Application Control, returneaza tot lantul de apelari in partea de sus in format sir. Lantul de apelari este folositor pentru erorile de utilizator pentru a vedea unde s-a produs eroarea. Aceste doua bucati de informatie for fi reunite la loc impreuna cu informatia generala a sursei generata de pachetul de erori. Poti adauga orice alta informatie pe care vrei sa o returneze odata cu pachetul de erori in mod similar. Poate fi folosita pentru ai da programatorului mai multe date despre eroare care pot fi ajutatoare in compilare. Erorile pot fi puse intr-un fisier text sau baza de date pentru referinta. Punerea aceasta va fi demonstrata in sectiunea 6.4.6 in exemplul de baza.

6.3.2 Erori de cod

O lista despre posibilele erori generate de Lab VIEW sunt accesibile in Online Reference din Help Menu. Erorile sunt listate de zonele de erori de cod si de tipurile de erori. Erorile de cod pot fi valori atat pozitive cat si negative, depinzand de tipul erorii care este generata. Cand o eroare de cod de zero este returnata, ea indica ca nu a luat loc nici o eroare. Avertismentele sunt indicate de un cod care nu este zero, in timp ce statutul este “false”. Tabelul 6.1 contine o lista de erori si zonele de cod. Atentie,

daca folositi Lab VIEW 5.1, exista o lista aditionala de coduri pentru VISA furnizata de *Lab VIEW Version 5.1 Addendum Manual* care este trimis pe CD.

Tabelul 6.1 Erori de cod

Tipul erorii	Zona de cod
Erori de cod G Function	0 la 85
Erori de cod Data Acquisition VI	-10001 la -10920
Erori de cod Analisys	-20001 la -20065
Erori de cod TCP si UDP	53 la 66
Erori de cod DDE	14001 la 14020
Erori de cod PPC	-900 la -932
Erori de cod Lab VIEW Specific PPC	1 la 5
Erori de cod AppleEvent	-1700 la -1719
Erori de cod Lab VIEW Specific for AppleEvents	100.0 la 1004
Erori de cod GPIB	0 la 32
Erori de cod Instrument Driver	-1200 la -13xx
Erori de cod Serial Port	61 la 65
Erori de cod VISA	-1073807360 to -1073807202 1073676290 to 107367443

Un utilitar util pentru a cauta erorile de cod este de asemenea disponibil in meniul Help din Lab VIEW versiunea 5.0 sau mai mare. Cand este selectat Explain Errors, o fereastra noua va aparea cu pachetul de erori in partea stanga si casuta de text in partea dreapta. Erorile de cod pot fi puse in format hexazecimal au zecimal. O explicatie a erorii de cod va fi furnizata in casuta de text. Acest utilitar furnizeaza un mod rapid de a afla informatii aditionale despre o eroare in scopul compilarii.

6.3.3 Tratarea erorilor VISA

VISA este un standard pentru depanarea drivereleor instrumentelor si nu este specific Lab VIEW. Este o interfata de programare a aplicatiei (Application Programming Interface - API) care este folosita pentru comunicarea cu diferitele tipuri de instrumente. VISA traduce chemarile catre driverele de nivel jos, permitand sa programati interfete nesimilare cu un singur API. Vedeti capitolul 5 cu driverele instrumentelor pentru informatii despre VISA.

VISA are la baza conceptul I/O, astfel VISA Vis are amandoua pachetele erorilor de intrare si cele de iesire. Cand apare o eroare, acest Vis nu se va executa. Exista un set de erori specifice de VISA care poate fi gasit in Lab VIEW Help. VISA Status Description VI poate fi utilizat in situatiile de tratare a erorilor. Acest VI este disponibil in subpaleta VISA despre palete ale instrumentelor I/O. VISA Status Description VI ia sesiunea VISA si pachetele de erori ca date si returneaza descrierea statului erorii care a fost generate.

Cand folositi drivere ale instrumentelor care folosesc VISA, pot fi cateva erori suplimentare de care va puteti lovi. Prima cauza este aceea ca VISA nu a fost corect instalata in computerul dumneavoastra. Daca alegeti instalarea tipica, NI-VISA este selectata pentru instalare implicit. Daca ati ales instalarea selectiva, trebuie sa aveti in vedere sa bifati selectiile. Nu puteti sa folositi VISA Vis daca nu ati instalat aceasta optiune. Alta cauza ar putea fi relatata la serialul de nivel jos, driverele GPIB, sau VXI pe care VISA le apeleaza pentru a realiza comunicarea cu instrumentul. De exemplu, daca aveti o placa GPIB instalata in computerul dumneavoastra pentru a controla instrumentele, fiti siguri ca software-ul placii este de asemenea instalat corect

pentru a permite sa il foloseasca VISA Vis. Puteti folosi N1-Spy pentru a monitoriza chemarile catre driverele instalate de la National Instruments de pe sistem. N1-Spy este explicat pe scurt in sectiunea 5.5.

Cand folositi VISA in aplicatiile dumneavoastra, nu uitati sa inchideti toate sesiunile VISA sau referintele pe care la aveti deschise din timpul operatiilor I/O. Lasand sesiuni deschise poate degrada performantele sistemului. Puteti folosi Open VISA Session Monitor.vi pentru a gasi sesiunile pe care le aveti deschise, si sa le inchideti pe cele care nu sunt in folosinta. Acest VI este disponibil in urmatorul director: \LabVIEW\Vi.lib\Util-ityWisa.llb. Acest VI va poate fi de folos in timpul depanarii unei aplicatii.

6.3.4 Tratarea simpla a erorilor

O tratare a erorilor simpla poate fi gasita in paleta Time & Dialog in meniul Functions. Acest VI este uitlizat pentru raportarea eroeilor. Este utilizat impreuna cu Lab VIEW Vis care utilizeaza erorile I/O si pachetul de erori. Scopul tratarii simple erorii este de a anunta utilizatorul daca o eroare s-a produs, dar poate fi folosit si pentru alte functionalitati. El considera pachetul de erori ca un dat si determina daca o eroare a fost generata. Daca o eroare a fost generata, VI afiseaza o caseta de dialog cu codul erorii, si o descriere pe scurt a unei erori, si a locatiei erorii. Tratarea simpla a erorilor utilizeaza un tabel pentru a afisa descrierea erorii bazandu-se pe codul erorii.

Asa cum am mentionat una dintre utilizarile tratarii erorilor simple aste pentru anuntarea aparitiei erorilor. Programatorul poate sa selecteze tipul de fereastra ce va fi afisata scriind numarul intreg corespunzator sau constanta enumerata. Valoarea 1 afiseaza o casuta de dialog avand ca unic buton “Ok” pentru activare. Valoarea 2 afiseaza o fereastra cu butoanele “Continue” si “Stop”. Aceasta permite utilizatorului sa opresca executia programului. Vloarea 0 nu da nici o instintare operatorului, chiar daca o eroare a fost generata. Aceasta poate fi folosita cand tratarea exceptiilor trebuie sa fie indeplinita astfel ca si cand utilizezi eroarea?, code out, sau source out, la iesirea tratarii erorilor simple.

Trebuie sa aveti in vedere ca acest VI va opri de tot executia pana cand operatorul raspunde la caseta de dialog. Daca aveti intentia sa porniti programul si sa plecati, programul nu va continua daca apare vreo eroare. Casetele de dialog nu ar trebui folosite decat cand programul este monitorizat. Presupuneti ca folositi e-mail-ul pentru instiintari folosind pachetul SMTP ca o alternativa. Capitlul 8 de asemenea va arata cum sa includeti posibilitatea e-mail-ului folosind ActiveX.

Figura 6.7 va arata cum sa folositi tratarea simpla a erorilor. Acesta este acelasi cu cel din Figura 6.3. Aveti in vedere ca tratarea simpla a erorilor a fost adaugata doar ca ultimul VI. Valoarea 2, careia in corespunde doua butoane din caseta de dialog (Continue si Stop), este trecuta la VI. Daca o eroare a fost detectata in ambele GPIB Read si GPIB Write, caseta de dialog va apare afisand codul erorii, descrierea, si sursa erorii.

6.3.5 Tratarea generala a erorilor

Tratarea generala a erorilor face practic acelasi lucru ca si tratarea simpla a erorilor. Tratarea simpla a erorilor cateva alegeri care sa le folosim intr-o aplicatie. Tratarea simpla a erorilor este o invelis a tratarii generale a erorilor. Tratarea generala a erorilor poate fi folosita in aceleasi situatii precum tratarea simpla a erorilor, dar de cand tratarea generala a erorilor are cateva optiuni in plus, ea poate fi folosita si in alte scopuri unde se doreste si mai mult control.

Tratarea generala a erorilor permite aduagarea de cod de erori definit de utilizator si descrierea erorilor. Cand aceste caractere au fost adaugate, ale sunt adaugate la tabelul de afisare a erorilor si descrierilor acestora. Cand se iveste o eroare, sunt cautate posibilele erori definite deja in Lab VIEW, urmata apoi de erorile introduse de programator. Caseta de dialog va afisa apoi descrierea erorii si specifica unde a avut loc.

Tratarea generala a erorilor ofera de asemenea optiuni limitate de tratare a exceptiilor. Programatorul poate sa seteze statutul erorii sau ca anuleze o eroare folosind acest VI. O eroare poate fi anulata specificand codul erorii, sursa, si actiunea exceptiei. Setati actiunea exceptiei pe Cancel Error on Match. Tabelul este rasfoit cand apare o eroare. Cand sa gasit o potrivire, statutul erorii este pus pe “false”. Mai mult, descriptorul sursei este sters si codul erorii este pus pe zero in pachetul de iesire. Similar, cand statutul unei erori este “false”, el poate fi setat pe “true” prin evitarea actiunii asupra exceptiei, codului erorii si a sursei.

6.3.6 Gasirea primei erori

Find First Error VI este de asemenea gasit in paleta Time & Dialog din meniul Functions. Scopul acestui VI este de a creea un pachet de erori de iesire. Sunt necesar de urmatoarele date: Error Code Array (matricea codului erorii), Multiline Error Source (sursa erorii multilinie), and Error In Cluster (eroare in pachet). Cand statutul erorii este “false” sau nu este racordat, VI-ul testeaza sa vada daca matricea codului erorii este diferita de zero. VI-ul reuneste primul element diferit de zero, sursa si statutul valorii “true” pentru a creea pachetul de erori de iesire. Din cauza ca sursa este un sir multilinie, indexul matricii codului erorii este folosit pentru a alege ce mai convenabila sursa pentru reunire. Daca este introdus o eroare in pachet este facuta intai o verificare a statului pachetului. When statutul este “true” eroarea din pachet va fi lasata si verificare matricii nu va mai avea loc. Gasirea primei erori practic de folosit cu VI-uri Lab VIEW care nu folosesc erori I/O ci da numai valoarea codului erorii. Urmatoarele sunt cateva VI-uri care da numai codul erorii: VI I/O serial, PPC Vis, Apple Event Vis, si cateva VI-uri Analysis. Gasirea primei erori poate fi folosita pentru a converti codul erorii intr-un pachet de erori. Pachetul de erori poate fi utilizat ulterior cu alte VI-uri care utilizeaza erori I/O.

Figura 6.8 este un exemplu despre cum sa se foloseasca gasirea primei erori. Amandoua Bytes at Serial Port.vi si the Serial Port Read.vi relateaza codul erorii. O amtrice este construita din cele doua coduri de erori care sunt relatate. Un sir multilinie de la sursa este de asemenea aratat in exemplu. Sursa ne da informatia despre originea erorii. Find First Error.vi asambleaza pachetul de erori si in trimite la pachetul de erori de iesire. Daca nu a fost generata nici o eroare codul erorii de iesire va contine valoarea booleana “false”, nici un cod de eroare si sir de sursa gol. Daca a aparut vreo eroare, prima eroare care apare va fi trimisa la pachetul de erori de iesire. Pachetul de erori poate ulterior sa fis trimis la Tratarea generala a erorilor sau la Tratarea simpla a erorilor pentru a afisa vreo casuta de dialog daca este nevoie.

6.4 Executarea tratarii exceptiilor

Tratarea erorilor contine si detectia de erori si tratamentul aplicat erorilor daca acestea sunt gasite. Sectiunea anterioara prezinta cateva tipuri de erori care pot apare, ca si functiile incluse in Lab VIEW pentru tratarea exceptiilor. Sectiunea ilustreaza caile care sunt utilie pentru managmentul erorilor. Eficacitatea unei tratari de erori poate fi imbunatatita prin includerea ei in aplicatie inca din stadiile primare ale

dezvoltarii. Ea va suporta claritatea si mentenabilitatea codului dumneavoastra, ca si reutilizarea codului. Cand tratarea erorilor nu este luata in considerare cand creeati aplicatia, cadul de tratari va fi constituit din petice ale fiecarei exceptii.

Va trebui sa va ouneti cateva intrebari despre implementarea tratarii erorilor in loc sa faceti tratarea in mod eficient si eficace. Cand este nevoie de detectie de erori, raportare si tratare? Ce sa faca aplicatia cand este detectata o exceptie? Unde si cum sa fie implementata? Urmatoarele subsectiuni va relata unde, cum si de ce tratarea exceptiilor este utila in aplicatiile dumneavoastra.

6.4.1 Cand?

Intrebarea cand sa se implementeze este un pic ciudata si depinde de siyuatiile sau aplicatiile specifice pentru care este dezvoltata. Aceast poate diferi de obiectivele aplicatiei, de timpul disponibil, de intentiile programatorului si de alti cativa factori. Unele zone din apalicatie care au nevoie de taratare sunt mai usor de indentificat decat altele. Poti fi capabil sa identifici zone unde taratrile nu sunt tolerate, sau unde erorile pot apare. Acestea sunt tinte definite pentru detectia de erori, raportare si tratare.

Pentru a raspunde la aceasta intrebare cat se poate de complet, trebuie sa va uitati la o aplicatie in interior dupa niste instante specifice pentru a determina ce altfel de scenariu mai poate fi prevazut ca si posibilele consecinte. Pentru a ilustra aceasta, considerati un exemplu in care trebuie sa cititi si sa scrieti intr-un fisier folosind operatii I/O. Pantru a raspunde, daca este nevoie de cod pentru exceptii, si poate si ce este nevoie, considerati urmatoarele consecinte si scenarii. Ce se va intampla daca fisierul in care vreti sa scrieti nu se deschide? Ce se intampla daca operatia de citire sau scriere esueaza? Ce se intampla daca fisierul nu poate fi inchis? Raspunsurile la aceste intrebari va va ajuta sa intelegeti perspectiva tratarii aplicatiei. Va va ajuta de asemenea sa priviti aplicatia si sa determinati cand este nevoie de tratarea exceptiilor este utila punandu-va intrebari similare. Tratarea erorilor va fi neaparat necesara in operatiile I/O cu fisiere, si daca si alte parti ale aceleiasi aplicatii suntdependente de zona aceasta.

6.4.2 Exceptiile – Tratare la nivel mediu

Raspunsul la intrebarea “cand”, tratarea exceptiilor trebuie condusa la nivel principal sau la nivel de excutare a testelor. Nivelul principal controleaza si disteaza decurgerea aplicatiei. Prin efectuarea tratarii exceptiilor la nivel principal, executia programului si controlul pot fi tinute la nivelul de sus. Acest lucru este important pentru ca tratarea exceptiilor poate altera decurgerea normala a programului daca este detectata o eroare. Poate ca o sa vreti ca codul sa efectueze mai multe actiuni diferite daca apare vreo eroare. Cand taratrea exceptiilor este efectuata la nivelul de jos controlul programului trebuie trecut de asemenea la nivelul de jos. Acesta este un motiv puternic de ce trebuie luata in consideratie tratarea exceptiilor inca din faza de constructie a aplicatiei. Structura aplicatiilor si procesele dezvoltarii aplicatiilor sunt sidcutate in capitolul 4. Citind capitolul 4 va va ajuta sa obtineti perspective bune pentru modul in care sa te apropii de dezvoltarea unei aplicatii si alte topici ce trebuie luate in considerare inainte de a incepe.

Efectuarea taratrii exceptiilor la nivel principal elimina nevoia de a dubla codul in unele subVI-uri. Acest lucru permite tratarii exceptiilor sa fie localizata intr-un singur loc. Separarea unui cod pentru tratarea erorilor de restul de cod reduce confuzia si creste lizibilitatea si mentenabilitatea. Cursul logic al programului va fi

pierdut in dezordinea in care taratrea erorilor este efectuata laolalta cu restul programului. Acest lucru este explicat mai incolo in sectiunea 6.4.6 la taratarea exceptiilor cu masinile de statut.

Stilul sugerat este similar altor limbaje de programare unde informatia eronata este trimisa unei parti de cod separate in scopul tratarii. Cum am mentionat mai inainte, atat Java cat si C++ au sectii diferite pentru efectuarea tratarii exceptiilor dupa ce evaluarea unei erori este completa. Nu exista un asemenea mecanism inerent in Lab VIEW.

6.4.3 Programatorul – erori precizate

Definirea erorilor a fost discutata pe scurt in sectiunea 6.3.1 laolalta cu pachetul de erori. Abilitatea de a defini erori este importanta deoarece Lab VIEW lasa taratarea erorilor specifice aplicatiei in seama programatorului. Cum am mentionat mai devreme, codul erorilor 5000-9999 sunt dedicate uzului programatorului. Programatorul trebuie sa efectueze verificari in circumstante unde greselile nu sunt tolerate, cum a fost aratat in figura 6.5. Codul erorii trebuie supus verificarii de erori la fel ca sirul sursei pentru a gasi punctul de plecare.

Cand se implementeaza o eroare definita de programator intr-un subVI, trebuie sa va asigurati ca nu ati strecurat vreo eroare. Desfaceti pachetul de erori si verificati valoarea booleana a statutului. Daca s-a strecurat vreo eroare, dar nu ati verificat statutul, puteti suprascrie pachetul de erori cu o noua informatie despre erori. Este aproape imposibil de gasit radacina acestei probleme in timpul compilarii. Trebuie, de asemenea, sa folositi registre de transfer cand folositi pachete de erori incadrate in structura buclei pentru a trece datele de la o iteratie la urmatoarea. Daca nu sunt folositi registrii de transfer, datele vor pierdute la fiecare iteratie.

Inregistrarile pot eleimana codurile erorilor care au fost aignate de utilizator. Poate fi creat un tabel care sa contina toate codurile erorilor si sursele asignate. Acesta poate fi utilizat cu tratarea generala a erorilor sau cu p lata procedura de tratare a erorilor. Poate fi o manevra buna de a mentine baze de date sau foi de calcul tabelar pentru codurile definite de utilizator. O baza de date faciliteaza managementul atunci cand numarul codurilor creste ca valoare.

Cand asignati cod de erori, puteti grupa erori similare in diferite regiuni. Acest lucru este folositor cand trebuie decisa calea de actiune cand apare o eroare. De exemplu puteti sa asignati codurile erorilor 6000-6999 pentru raspunsuri incorecte de la intrumentele I/O. Erorile generate de Lab VIEW sunt grupate in cateva moduri pentru a facilita managmentul si identificarea lor.

Avertismentele definite de utilizator poate de asemenea sa fie asignate codurilor pentru a indica ca un eveniment nedorit s-a intalplat. Puteti folosi aceasta cand semnalul pe care l-au luat datele nu va fi tot valid datorita acuratetei unora dintre evenimente in timpul executiei programului. Utilizatorul poate investiga sursa avertismentului pentru a verifica dupa aceea daca datele sunt valide. Pot fi raportate erori multiple si tratate dupa desfacerea pachetului de erori si aduagarea de noi informatii.

6.4.4 Managmentul erorilor

Odata ce aveti lista cu erorile pe care vreti sa le eliminati si care pot detectate, trebuie sa va decideti ce sa faceti cu ele cand apar. Cand apare o eroare, ea trebuie sa fie trimisa la codul de tratare a erorilor. Codul tratarii exceptiilor paote anfrunta

erorile in mai multe feluri. Largind ideea de a grupa erorile similare, codul poate verifica in ce zona se incadreaza eroarea pentru a determina actiunea ce trebuie luata. Figura 6.9, Error Range Example.vi, este un exemplu de grupari ale zonelor de cod de eroare in scopul tratarii. Cand un set de exceptii sunt considerate ca ar fi relatate logic, este mai bine sa le organizezi intr-o familie de exceptii.

Cel mai usor mod de a infrunta este de a afisa o caseta de dialog pentru a anunta utilizatorul ca s-a produs o eroare. Aceasta caseta de dialog poate fi atat de simpla ca si cea afisata de tratarea generala a exceptiilor. Puteti sa va creeati un VI nou pentru a afisa o caseta de dialog care include si mai multa informatie, incluzand ce poate utilizatorul sa faca pentru a da la o parte eroarea. Aceste rezultate obisnuite opresc executia programului.

Puteti sa va implicati si mai mult prin incercarea de a corecta erorile din codul pentru tratarea exceptiilor. In acest caz, tehnicile de verificare cat mai generale nu sunt suficiente deoarece acelasi cod este folosit determinand bum sa-l corecteze. De asemenea este nevoie de cunoasterea in detaliu a erorii si cum se poate corecta. Presupuneti ca aveti a eroare specifica care va spune ca dispozitivul supus testului nu raspunde la comenzi. De asemenea stiti ca acest lucru se intampla cand dispozitivul nu este alimentat sau nu a fost initializat corect. Puteti sa incercati sa corectati eroarea prin alimentarea dispozitivului si reinitializarea lui. Atunci puteti incerca sa comunicati cu el din nou si sa rulati aplicatia daca totul a decurs bine.

Figura 6.10 ilustreaza tehnicile de infruntare a unor coduri de erori specifice ca o alternativa la metoda de verificare generala a zonei. Aceasta metoda poate fi folosita in Lab VIEW 4.1 sau mai mare. Lab VIEW 5.0 are un proces initial care va permite sa legati codul direct la terminalul selectat a structurii procesului. Puteti folosi meniul care se deschide pentru selectarea procesului initial, care este normal procesul 0. Acest proces se va executa pentru coduri de erori pentru care nici un proces nu a fost definit.

Metoda afisata este similara unui tabelului descris mai devreme. O matrice care contine toate codurile erorilor este folosita cu Search ID Array VI (VI-ul de cuatare a ID-ului in matrice). Codul eroarei este trimis si incepe sa caute dupa indexul codului erorii. Indexul conduce la comandarea procesului, care reia cursul corect al actiunii pentru codul erorii. Daca nu exista vreo potrivire pentru codul erorii, Search ID Array returneaza -1. Adaugand 1 la rezultat, procesul 0 (Case 0) este selectat din structura. Acest proces este selectat ca initial daca nu este nici o potrivire. In exemplul aratat apare o casuta de dialog care ne arata ca nu a fost definita nici un cod de eroare.

O alta alternativa disponibila in Lab VIEW 5.0 este folosirea sirurilor pentru a conduce la procese ale structurilor. Puteti implementa exemplul anterior prin desfacerea pachetului pentru a gasi informatia despre sursa. Acest sir poate fi folosit pentru a determina cursul actiunii prin scrierea in terminalul procesului de selectie.

6.4.5 Aparatul de stare pentru tratarea exceptiilor

Folosirea aparatului de stare ofera cateva avantaje pentru codul tratarii erorilor. Unul dintre ele este acela ca codul tratarii exceptiilor este localizat intr-un singur loc. Acesta este gata prin folosirea statutului erorii. Statutul erorii este responsabil pentru toate tratarile exceptiilor din aplicatie. Aceasta elemina nevoia de a plasa codul tratarii exceptiilor in mai multe locuri. Mentinerea codului devine mai usoara cand codul este rezident intr-un singur loc. Folosirea unui aparat de stare faciliteaza managmentul tratarii exceptiilor la nivelul de test executiv si principal. Starea erorii face parte din nivelul principal, asa ca controlul este mentinut la nivele superioare.

Alt avantaj este ca dupicarea codului erorii este redusa cand codul este plasat intr-un singur loc. Erorile similare pot fi generate in zone diferite din cod. Daca nu efectuati tratarea

erorilor intr-un sinfur loc, sunteti nevoit sa scrieti cod in mai multe locuri pentru acelasi tip de eroare.

Executia conditionala a codului poate fi implementata fara sa creati o tratare complexa a erorilor in timpul folosirii aparatului de stare. Codul tratarii exceptiilor determina exceutia programului bazata pe severitatea erorii care a fost generate. Puteti sa lasti sa sara peste executia partii din cod care este afectat de eroare, si sa continuati executia restului din program. De exemplu, presupuneti ca aveti o serie te zece teste diferite pe care vreti sa le efectuati pe un dispozitiv sub analiza. Daca apare o eroare in testul 1 si aceeasi eroare va afecta testul 5, 6 si 7, puteti inca sa mai executati testele 2, 3 si 4. In acest caz, folfosind un rand de aparate de stare va simplifica procedura pentru a efectua aceasta sarcina. Starea erorii poate sa analizeze statuturile care corespund testelor 5, 6 si 7 din lista statuturilor de executie. In caz ca eroarea poate fi corectata, programul trebuie sa-si aminteasca unde a fost intrerupta executia ca sa poata sa continue din acelasi loc. Utilizarea aparatelor de stare faciliteaza implementarea acestei posibilitati in codul tratarii exceptiilor. Pentru diagnosticarea statutului informatia trebuie retinuta pentru a face aceast lucru posibil. In plus, trebuie adaugate rutine de salvare si inregistrare trebuie incorporate pentru a fi sigur ca nu se pierd date.

Executia conditionata poate fi aplicata de asemenea testelor care esueaza. Puteti proiecta aplicatia sa execute un test ghidandu-se dupa datele primite de la alt test. Daca testul 1 a dat gres puteti sa sariti peste testele 2, 3 dar sa continuati cu testele ramase. Din nou, puteti sa analizati testele care nu ar trebui sa fie executate.

6.4.6 Inregistrarea erorilor

Inregistrarea erorilor este folositoare pentru pastrarea inregistrarilor de greseli care apar de-a lungul executiei programului. Jurnalul erorilor trebuie sa raporteze codul, originea, o descriere pe scurt si cand a aparut eroarea. Pana la aparitia erorii, fisierul furnal este deschis, scris in el si inchis. In tratari ale exceptiilor suplimentare exista cod, eroarea poate fi infruntata intr-un mod corespunzator.

Inregistrarea erorilor este benefica in cazurile in care codul tratarii exceptiilor a fost deja implementat si cand nu exista vreo tratare a exceptiilor in aplicatie. Cand a fost implemntata o tratare a exceptiilor, inregistrarea erorilor da programatorului intuitia la ce tipuri de erori sa se astepte si daca codul le va trata corect. Jurnalul poate fi folosit ca un mecanism de reactie pentru determinarea zonelor din codul tratarii exceptiilor care este nesatisfacator. Aceste zone pot fi folosite dupa aceea pentru constructia unei plicatii mult mai robuste.

In unele instante unde codul pentru tratarea exceptiilor nu a fost inca implementat, jurnalul de erori poate fi folosit in mod similar. Jurnalul poate servi ca fundament pentru dezvoltarea codului de tratare a exceptiilor. Erorile care apar mai frecvent pot fi adresate mai intai. Aceasta metoda vrea sa gaseasca o legatura cu timpul mare consumat la dezvoltarea tratarii exceptiilor. Aici, conceptul este de a fi mai benefic atacand cele mai comune erori.

Figura 6.11 este un exemplu de VI care inregistreaza erorile. Mai, intai este verificat statutul din pachetul de erori pentru a determina daca a aparut vreo eroare. Daca a fost generata o eroare data, timpul, codul erorii si sursa sunt scrise intr-un fisier care serveste drept jurnal de erori. VI-ul “Write Characters to File” este folosit pentru a efectua inregistrarea. Acest VI poate fi folosit in mult locuri unde este dorita inregistrarea, sau in locatiile centrale laolalta cu alte coduri tratari de exceptii. De cand informatia eronata a fost convertita intr-un det de instructiuni delimitate prin spatii, poate fi importate in Excel pentru a folosi drept baza de date.

6.4.7 Tratarea externa a erorilor

O tratare a exceptiilor care este externa aplicatiei poate fi scrisa sa tarteze exceptiile care pot apare in timpul executiei programului. Aplicatia trebuie sa faca o chemare la tratarea externa a rorilor. Acest lucru poate fi benefic cand folositi HI Test Executive. VI-ul pentru trayarea erorilor este incarcat cand axista o referinta spre el in aplicatie. VI-ul pentru tratarea erorilor poate fi scris sa efectueze toate sarcinile semnificative, similar scoaterii tratarii afara din aplicatie.

Daca tratarea erorilor este scrisa pentru a gazdui exceptiile generale, poate fi chemat din atatea aplicatii de cate avem nevoie. Figura 6.12, Load External Handler.vi, ne arata cum poate fi incarcat un VI si executat din aplicatie. Mai intai trebuie deschisa o referinta catre VI folosind Open VI Reference. Acest VI poate fi accesat din paleta Application Control. Trebuie sa specificati calea si directorul unde se afla VI-ul. Setati VI Server Class pe “Virual Instrument” prin localizarea VI Refnum. Nodul “Invoke” este folosit pentru executia VI-ului extern. Nodul “Invoke” este de asemenea accesibil din paleta Application Control. Cand referinta catre VI este trecuta la nodul “Invoke”, VI Server Class se va schimba automat in “Virtual Instrument”(Instrument Virtual). Apoi, prin localizarea “Methods”(metode) puteti selecta matoda Run VI(executa VI) din meniu. Prin folosirea si selectarea metodei Set Control Value(setati valoarea de control) nodului Invoke pot fi trecute date catre VI-ul de tratare a erorilor.

Exemplu:

Un exemplu despre cum este implementata o tratare externa a exceptiilor este aratat in figura 6.13. Aceasta diagrama de cod arata pasii implicati in folosirea tratarii externe: deschiderea referintei catre un VI, trimiterea datelor initiale, executia VI-ului extern, si inchiderea referintei. Deschiderea unei referinte si executia VI-ului extern au fost deja explicate. In acest exemplu, pachetul de erori este trimis catre taratarea externa a exceptiilor, fiind cea care determina cursul actiunii.

Mai intai, este deschisa o referinta catre VI prin intermediul External Handler.vi asa cum este aratat in calea VI. Apoi pachetul de erori este trimis catre External Handler.vi folosind metoda Set Control Value in nodul Invoke. Aceasta metoda solicita programatorului sa specifice numele controlului (Control Name), descriptorul tipului (Type Descriptor) si datele aplatizate (Flettened Data). Pachetul de erori este trecut la aceasta metoda prin aplatizare folosind Flatten to String din subpaleta Data Manipulation in paleta Advanced. Sirul de date aplatizat si descriptorul tipului sunt apoi legate direct da la Flatten to String la metoda Set Control Value. The Control Name (numele controlului) este un sir care trebuie sa se potriveasca identic cu numele controlului panoul frontal al VI-ului catre care sunt trimise datele. Numele specificat in diagrama de cod este *Error In (No Error)*, asa cum apare pe panoul frontal al External Handler.vi. VI-ul este executat folosind metoda Run VI si, in final este inchisa referinta.

Figura 6.14 ilustreaza diagrama de cod a External Handler.vi. Acest VI este similar cu un VI de tratare a exceptiilor aratat mai inaite. El ia informatie despre pachetul de erori si decide cursul actiunii bazat pe codul erorii. Informatia despre eroare este inregistrata folosind Error Log.vi si structura procesului este ghidata dupa codul erorii. Procesul 0 (Case 0) este folosit ca valoare implicita in codurile de erori pentru care nu exista tratare de erori.

In acest exemplu pachetul de erori a fost trimis catre un VI extern. Similar, datele pot fi primite de la controloare si indicatoare ale VI-ului daca se doreste. Metoda Get All Control Values (ia toate valorile de control) poate fi folosita la efectuarea acestei actiuni. Aceasta metoda ia toate valorile controloarelor si indicatoarelor de la VI-ul extern. Datele sunt returnate ca o matrice de pachete, cate un element pentru fiecare panou frontal al controlorului sau indicatorului. Pachetul contine numele controlorului sau indicatorului, descrierea tipului si datele aplatizate, similar cu datele care sunt trimise catre VI-ul de tratare externa a erorilor External Handler VI in exemplu.

6.4.8 Proceduri corespunzatoare de iesire

In situatii in care apar rori fatale sau nerecuperabile cel mai bine este sa se opreasca executia programului. Acest lucru este de asemenea adevarat cand nu este rezonabil sa se continue executia programului cand apar unele erori specifice. Cu toate acestea, terminarea anormala a programului poate sa cauzeze probleme. Cand va decideti sa opriti executia unui program datorita erorilor, trebuie sa va asigurati ca programul exista intr-un mod corespunztor.

Toate instrumentele I/O, fisierele si canalele de comunicatie trebuie inchise inainte ca apicatia sa se termine. Efectuarea acestei sarcini inainte de iesirea din program minimizeaza problemele de mai sus. Presupuneti, de exemplu, ca un fisier a ramas deschis cand o aplicatie s-a terminat. Acest lucru poate cauza erori cand altii vor sa scrie in fisierul respectiv pentru ca privilegiile de scriere sunt interzise.

Pana la aparitia unei erori controlul este trecut pe seama tratarii erorii. Din cauza asta, tine de responsabilitatea tratarii erorilor ca toate tratarile, fisierele si cananlele de comunicatie sa fie inchise. Cel mai usor de implementat acest lucru este de a face tratarea erorilor de a identifica mai intai erorile. Daca eroarea care a fost generata necesita terminarea programului codul din tratare poate efectua aceasta sarcina. Figura 6.15, Close Handles.vi, este un exemplu de VI care este folosit singur pentru a inchide canalele de comunicatie deschise. O sesiune VISA, fisier refnum, conexiune TCP, si refnum automat sunt trecute in seama acestui VI, care trece la inchiderea referintelor.

Un program trebuie scris pentru a avea un singur punct de iesire unde toate sarcinile necesare sunt executate. Cel mai bun mod de a implementa aceasta este de a utiliza un aparat de stare. Prin folosirea unui aparat de stare este nevoie doar de un punct de iesire si va folosi drept stare de inchidere (Close State). Corespunzator, exista doar un singur loc unde unde toate tratarile exceptiilor sunt efectuate: starea erorilor (Error State). Cand o eroare este identificata ca fatala, starea erorilor va forta aparatul de stare sa inchida toate starile. Starea de inchidere (Close State) va fi responsabila pentru inchiderea programului intro maniera potrivita. Toate tratarile, fisierele si canalele de comunicatie vor fi inchise in aceasta stare. Din moment ce este nevoie de un singur punct de inchidere vafi si ultima stare care se va executa atunci cand nu s-au depistat erori. Acest stil face codul mai lizibil si mai usor de intretinut.

6.4.9 Exemplu de tratare a exceptiilor

In aceasta sectiune au fost relatate cateva metode de tratare a erorilor. Un exemplu de inchidere care utilizeaza unele topici care au fost discutate este prezentat in figura 6.16. Exemplul utilizeaza structura aparatului de stare laolalta cu starea erorilor pentru tratarea acestora.

Scopul Next State.vi (starea urmatoare) este simplu de determinat care stare va fi executata dupa aceea. Next State.vi este responsabil pentru verificarea daca s-a produs vreo eroare dupa terminarea fiecarei stari. Cand a aparut o eroare, urmatoarea stare care va fi executata va fi starea erorilor. Starea erorilor inregistreaza mai intai eroarea folosind Error Lod.vi. Codul erorii este verificat pentru a determina daca exista intr-o zona certa care corespunde cu erorile driverul instrumentului. Daca codul erorii se afla in zona aceea, eroarea este considerata ca fatala sau nerecuperabila in acest exemplu. Cand o eroare fatala a aparut, starea de inchidere (Close State) este legata direct la Next State.vi (starea urmatoare) pentru a executa procedura potriita de iesire.

Daca codul erorii nu se incadreaza in acea zona specificata, codul este verificat din nou si comparat cu codurile de erori definite de utilizator. Aceasta conduce la structura procesului, care va actiona in functie de eroarea care a fost generata. Cand nu se gaseste nici o potrivire, procesul 0 (Case 0) este executat ca in figura 6.17.

Cand nu rezulta nici o potrivire pentru procesul 1 (Case 1), Remove State.vi va inlatura procesele care nu pot fi executate din cauza unei erori care a fost generata. Dupa aceea programul va continua ca procesele care pot fi executate conform elementelor din matricea de proces. Acest lucru este aratat in figura 6.18.

Figura 6.19 ne arata procesul de inchidere a aparatului de stare. Acest proces este executat la terminarea normala a programului si cand se determina ca s-a ivit o eroare fatala. Cum este arata in figura 6.16, procesul de eroare va forta procesul de inchidere sa se execute cand este depistata o eroare fatala. Singura sarcina a Close Handles.vi (inchiderea tratarilor) este de a inchide toate referintele si canalele de comunicatie care sunt deschise. Acest lucru va minimiza problemele cand aplicatia este rulata din nou.

Exemplul demonstreaza ideile prezente in aceasta sectiune. Mai intai, tratarea exceptiilor s-a produs la nivelul principal pentru ca aplicatia sa nu se treaca la nivelul de jos. In al doilea rand, codul de tratare a exceptiilor este separat de restul de cod pentru a fi mai lizibil. Nu numai ca reduce confuzia, dar reduce si nevoia de a dubla codul in unele locuri. Dupa aceea, aparatul de stare permite introducerea codului pentru tratarea exceptiilor intr-un sindur loc pentru a creste mentenabilitatea si analizarea testelor conditionale. Inregistrarea erorilor este efectuata pentru a tine evidenta erorilor care au aparut. In final, a fost implementata o procedura potrivita de iesire din aplicatie. Practicarea multa in crearea tratarilor de exceptii va duce la un cod mai stabil si sigur.

6.5 Depanarea codului

Tehnicile prezentate in sectiunile anterioare pentru tratarea exceptiilor pot fi utilizate si in depanarea codului in Lab VIEW. Detectia de erori este foarte pretioasa in timpul testarii codului. Detectia este asistata de gasirea locului si de s-a produs eroarea. Un defect este o greseala in cod care trebuie eliminata. Cu cat sunt gasite aceste defecte mai devreme cu atat sunt mai usor de reparat. Aceasta sectiune acopera unele utilitare care care faciliteaza procesul de depanare a VI-urilor. Mai intai vor fi discutate VI-urile stricate si lista de erori. Apoi va urma cum puteti activa executia pas cu pas cu ajutorul butoanelor de executie pas cu pas. Apoi utilitarul de proba care foloseste puncte de intrerupere si suspendarea executie vor fi descrise. Inregistrarea datelor si NI Spy vor fi prezentate. In final, vor fi aratate niste idei despre cum sa folositi aceste utilitare in depanarea programelor.

6.5.1 Lista de erori

Un buton de executie sters (Run) indica ca VI-ul nu poate fi executat. Un VI nu poate fi executat cand una sau mai multe erori exista in cod. Erorile pot rezulta dintr-o varietate de evenimente cum ar fi legarea gresita sau terminale nelegate la diagrama codului. Puteti de asemenea sa vedeti un buton de executie sters (Run) cand editati diagrama codului. Cu toate acestea, cand terminati de acris codul, butonul de executie (Run) nu ar mai trebui sa fie sters. Daca butonul de executie este sters, puteti afla mai multe informatii despre erorile care impiedica codul sa se excute prin apasarea pe butonul *Run*. Figura 6.20 arata cum apare fereastra cu lista de erori.

In partea de sus a ferestrei cu lista de erori exista o casuta care listeaza toate VI-urile care contin erori. O casuta care listeaza toate erorile din fiecare VI este exact sub aceasta casuta. Amandoi, erorile din panoul frontal si din diagrama blocurilor vor fi listate. Lista descrie natura erorilor. Cand este selectata o eroare casuta de langa relateaza mai multe informatii despre eroare si cum poate fi acesata inlaturata. Butonul *Find* (gaseste) va gasi si activa cauza erorii care este selectata. De asemenea exista o casuta de bifare, Display Warnings (afiseaza avertismentele), care listeaza toate avertismentele din VI-ul respectiv. Avertismentele nu impiedica executia VI-ului, dar sunt recomandari pentru programare. Puteti sa selectati afisarea avertismentelor totdeauna prin selectarea casutei de bifare din meniul Preference in submeniul Edit.

Folosirea listei de erori puteti efectiv rezolva toate erorile care pot impiedica VI-ul sa se execute. Odata ce ati infruntat toate erorile, butonul de executie (Run) nu va mai fi sters. Lista de erori realizeaza un mod usor de rezolvare a erorilor din cod si determina cursul actiunii pentru a le elimina.

6.5.2 Execution Highlighting

Lista de erori prezentata mai sus va ajuta sa rezolvati erorile care impiedica aplicatia sa se execute. Dar nu asista la identificarea de defecte care duc la producerea de erori. Execution Highlighting (activarea executiei) este un utilitar care depisteaza defectele din program. Execution Highlighting va permite sa vizualizati cursul datelor de la un obiect la urmatorul in timp ce se executa VI-ul. Datele care sunt reprezentate ca niste balonase care se misca de-a lungul firelor, pot fi vazute cum se misca “cu incetinitorul” prin noduri. *G Reference Manual* numeste acest lucru “animatie”. Acesta este un utilitar foarte eficace a celor de la National Instruments care a fost incorporat in Lab VIEW pentru depanarea VI-urilor. Din moment ce Lab VIEW este un limbaj de programare vizual, are sens incorporarea de uitlitare vizuale pentru ai ajuta pe programatori.

Daca nu vedeti balonase de date, poate ca setarile din meniul Preference nu a activat aceasta optiune. Initial aceasta otiune este activata. Selectati Preference din meniul Edit si alegeti Debugging. Fiti siguri ca optiunea este activata pentru a vedea balonase in timpul Execution Highlighting.

Apasand butonul cu un simbol luminos in forma de balon, localizat in bara de utilitare a diagramei de cod, va activa Execution Highlighting. Cand se porneste VI-ul incepe si animatia. Execution Highlighting poate oprit sau pornit in timpul executiei VI-ului. Execution Highlighting devine mult mai valoros cand este folosit in modul pas cu pas. Viteza de executie a programului este redusa semnificativ ca sa puteti vizualiza animatia si sa folositi si alte utilitare de depanare in timpul executiei.

6.5.3 Pas cu pas

Modul pas cu pas poate fi activat prin apasarea butonului *Pause* (pauza). Acest mod va permite sa folositi butoanele de pasi pentru executia unui nod odata din diagrama de cod. In plus, cand Execution Highlighting este activat, puteti vizualiza curgerea datelor si animatia pentru cod in timp ce se executa un nod odata. Butonul *Pause* poate fi apasat sau neapasat in timpul executiei sau chiar pana cand sa inceapa executia. Puteti de asemenea sa apasati butonul pentru executia pas cu pas localizat langa butonul *Execution Highlighting* pentru a intra in modul pas cu pas. Butonul *Pause* va deveni automat activ cand sunt utilizate acestea.

Cand un VI este in modul pas cu pas, sunt trei butoane de pe bara de utilitare a diagramei de cod pentru controlul executiei programului. Depinzand de diagrama de cod, butoanele de pasi vor efectua actiuni diferite. Folositi Sinple Help (simplu ajutor) ce va face fiecare buton la un nod specificat din diagrama de cod. Simple Help poate fi accesat din meniul Help. Cand cursorul se afla deasupra unui buton de pasi va aparea descriere a functiei acestuia. Figura 6.21 arata Error Log.vi in modul pas cu pas cu Execution Highlighting activat. Pot fi vazute de asemenea si butoanele de executie pas cu pas.

Primul buton de pasi din bara de utilitare este folosit pentru intrarea intr-o structura particulara a unui VI. Structura sau subVI-ul poate de asemenea sa fie in modul pas cu pas. Trebuie sa folositi butoanele de pasi pentru a termina structura sau subVI-ul. Urmatorul buton este folosit pentru saltul peste obiecte, structuri si subVI-uri. Daca acest buton este apasat structura sau subVI0ul se va executa si va va permite sa reluati executia pas cu pas dupa incheierea acesteia. Al treilea buton va va permite sa terminati executia la sfarsitul diagramei de cod. Odata apasat, codul ramas se va executa si nu va va permite sa treceti la un obiect dacat daca este apsat din nou butonul *Pause*.

6.5.4 Utilitarul de proba (Probe Tool)

Probe Tool poate fi accesat din paleta Tools sau apsand pe un fir din diagrama de cod. Probr Tool este folosit pentru a examina valorile datelor prin firele din diagrama de cod. Cand un fir este probat, datele vor fi afisate intr-o fereastra care are ca titlu numele valorii. Probele si firele sunt numarate pentru a va ajuta sa le urmariti cand sunt folosite mai multe odata. Puteti proba orice tip de data sau format pentru a vedea valorile care sunt trecute de-a lungul firelor. De exemplu, daca un grup de fire sunt probate, o fereastra cu numele grupului apare afisand valorile grupului. Valorile vor fi afisate imediat dupa ce datele trec de punctul de pe fir unde este plasata proba in timp ce VI-ul este executat.

Probbe Tool este foarte valoros cand depaneze VI-uri pentru ca va permite sa exminati valorile care sunt trecute prin fire. Daca se ivesc rezultate sau erori neasteptate, puteti verifica valorile pentru a fi siguri ca sunt corecte. Acest utilitar va va permite sa aflati radacina problemei. Figura 6.22 ilustreaza o proba la pachetul de erori intre VISA Close.vi si File Close.vi. Firul este marcat cu un numar in timp ce fereastra afiseaza valorile pachetului.

Implicit, probarea automata este activa in modul Execution Highlighting. Acest lucru cauzeaza ca Lab VIEW sa valorile datelor la noduri in timpul Execution Highlighting. Cu toate acestea, datele complete nu pot fi intotdeauna vazute in acest fel si este folositor doar in scopul verificarii. Probr Tool va fi inca necesar pentru tipuri de date ca matrici si grupuri. Probarea automata poate fi de asemenea activata

sau dezactivata din aceeasi fereastra Preference cum am discutat mai devreme la balonase de date.

6.5.5 Utilitarul pentru puncte de oprire (Breakpoint Tool)

Breakpoint Tool este alt utilitar disponibil in paleta Tools. Cu sugereaza si numele, Breakpoint Tool ne permite sa inseram un punct de oprire in diagrama codului. Punctele de oprire pot fi puse pe obiecte, VI-uri, structuri si fire. Un cadru rosu in jurul unui obiect sau structuri ne arata ca este un punct de oprire. In timp ce un punct rosu indica un punct de oprire pe fir. Punctele de oprire cauzeaza opirea executiei in locatia in care a fost amplasat. Daca este un fir, datele vor trece de punctul de oprire inainte sa se opreasca executia. Un punct de oprire poate fi eliminat folosind acelasi utilitar ca si cum l-ai pune.

Punctele de oprire valoroase pentru ca permite utilizatorului sa opreasca executia la o locatie specificata in cod. Programul va functiona in mod normal si va grabi executia inaitea punctului de oprire, unde se va opri. Codul care este suspect poate fi apoi depanat folosind modul pas cu pas, Execution HighLighting sau utilitarul de proba.

Odata ce a fost pus un punct de oprire, programul se va opri la acea locatie de fiecare data cand este executat. Trebuie sa va amintiti sa stergeti punctul de oprire daca nu mai vrei ca programul sa se opreasca inainte de urmatoarea iteratie sau executie. Daca salvati VI-ul cu punctul de oprire setat, VI-ul se va salva si cu punctul de oprire. Urmatoarea data cand incarcati VI-ul si il executati, executia se va opri la punctul de oprire. Puteti folosi functia Find (gaseste) pantru a localuza orice puncte de oprire care au fost setate.

6.5.6 Suspendarea executiei

Puteti forta un subVI sa suspende executia, in scupul depanarii, cand este chemat. Acest lucru poate fi facut folosind una din cele trei metode. Prima metoda este de a selecta Suspend when Called din meniul Operate. Cea de-a doua metoda este sa apasati pe subVI-ul dib diagrama de cod si sa selectati SubVI Node Setup. Apoi, selectati casuta Suspend when Called. Puteti sa apasati pe iconita cand subVI-ul este deschis si sa selectati VI Setup. Apoi casuta de bifare Suspend When Called.

Cand un subVI a cauzt suspendarea executiei, panoul lui central va fi afisat cand este chemat. SubVI-ul intra intr-un mod special de executie cand este suspendat. Butonul *Run* incepe executia subVI-ului. Cand un subVI este suspendat, poate fi executat in mod repetat folosind butonul *Run*. La dreapta de butonul *Run* este butonul *Return to Caller*. Odata suspendat, puteti folosi Execution Hightlightning, Singl-Stepping (Pas su Pas) si utilitarul de proba pentru depanarea subVI-ului. Cand folositi executia pas cu pas cand subVI-ul este suspendat, puteti sari peste inceput si sa executati subVI-ul de cate ori este nevoie.

6.5.7 Data Logging (Inregistrarea datelor)

Data Logging este un alt utilitar inclus in Lab VIEW care poate fi folosit in scopul depanarii. Datele din panoul frontal pot fi inregistrate automat activand Log at Completion din meniul Operate. Cand VI se executa pentru prima data, va aprea o caseta de dialog, cerandu-I utilizatorului un nume de fisier in care sa depoziteze. Un fisier jurnal poate fi selectat inainte de a executa VI-ul, prin selectarea Log din

submeniul Data Logging din meniul Operate. Cand numele fisierului este selectat inaintea executiei VI-ului, casuta de dialog nu va aparea. Datele din panoul frontal sunt salvate in jurnal dupa ce se executa VI-ul.

Data Logging este o metoda pentru salvarea datelor din teste, similar unei baze de date. Lab VIEW introduce o marca de timp si data, laolalta cu datele din indicatoare si controloare din panoul frontal. Datele pot ulterior fi vizionate selectand Retrieve din submeniul Data Logging. Figura 6.23 ilustreaza cum apar datele cand sunt salvate si receptionate folosind aceasta caracteristica. Acesta este un panou frontal simplu cu doua indicatoare si doua controloare. Rezultatele multiplicate si aditionale ale celor doua controloare intregi sunt afisate in indicatoare. Acesta arata cum sunt afisate datele atunci cand sunt receptionate. Marca cu timpul si data apare in partea de sus, laolalta cu controloare pentru rularea prin inregistrari si stergerea din ele.

Data Logging este util pentru salvarea valorilor datelor din timpul testelor si depanarii VI-urilor. Serveste drept mecanism pentru salvarea rapida a datelor din VI-urile specificate care sunt depanate. Jurnalul de date salvat poate fi vizualizat pentru valori suspecte. Jurnalul de inregistrari de date este util pentru monitorizarea problemelor intermitente ale VI-urilor. Datele din panoul frontal pot fi apoi salvate, primite si curatate dupa necesitati.

6.5.8 NI Spy/GPIB Spy

Aceste doua utilitare sunt foarte similare si amandoua sunt folosite ca utilitare de depanare pentru sistemele de operare Windows 95/98/NT. NI Spy monitorizeaza chemarile care sunt facute de aplicatie catre driverele NI-488.2, NI-VISA, IVI si NI-VXI. In mod similar, GPIB Spy urmareste toate chemarile spre driverul GPIB. Sunt utile pentru determinarea sursei de erori de comunicatie, chiar daca sunt relatate in probleme generale de comunicatie sau specifice aplicatiei. Va ajuta sa verificati ca comunicatia cu un instrument este corecta. Cand vor rula amandoua in acelasi timp vor degrada viteza aplicatiei voastre. Folositile doar cand depanati un program pentru a elibera din resursele sistemului de operare, in special cand timpul de executie este un considerent important.

NI-Spy afiseaza numarul de index asignat chemarii, o descriere a operatiei si parametrii si timpul cand au aparut. Utilitarul afiseaza chemarile asa cum sunt facute in timpul executiei programului. Erorile sunt imediat scoase in evidenta pentru a arata esecurile. NI Spy va permite de asemenea sa inregistrati activitatile pentru a le vizualiza mai tarziu.

GPIB Spy monitorizeaza chemarile catre driverul de Windows GPIB sun Win32, si le afiseaza in timp ce aplicatia se executa. Toate erorile si esecurile sunt scoase in evidenta pentru o identificare rapida. Puteti vedea fiecare chemare asa cum este ea facuta si sa vedeti rezultatele incluzand si eventualele timeout-uri. Acest utilitar poate fi folosit pentru a verifica daca aplicatia trimite chenarile corecte catre driverul de Windows GPIB. GPIB Spy listeaza numerul de index al chemarii, numele chemarii catre GPIB, scotand cuvantul de stare *ibsta* dupa chemare, *iberr* in caz de eroare, variabila de numarare *ibcntl* si timpul fiecarei chemari. Toate din acestea contin informatii despre performanta aplicatiei. Puteti vizualiza informatii mai pe larg folosind butonul Properties din bara de stare.

Familiarizarea cu protocoalele de comunicatie GPIB, NI-488.2 si ANSI/IEEE488.2 ar fi necesare pentru intelegerea completa si utilizarea capacitatii de

depanare pentru GPIB Spy cat si pentru NI Spy. O discutie despre IEEE 488.2 nu face subiectul acestei carti.

Tabelul 6.2 Utilitare de depanare

Utilitarul	Aplicatia	Accesarea
Lista de erori	Folosita la listarea, localizarea si rezolvarea erorilor care previne VI-ul sa se execute	Apasati pe butonul sters *Run*
Execution Highlighting	Folosit la animatia si vizualizarea curgerii datelor de-a lungul firelor in diagrama de cod	Apasati butonul luminos ca un balonas
Modul pas cu pas	Permite executia unui singur nod odata	Apasati butonul *Pause*
Utilitarul de proba	Afiseaza valorile datelor care trec prin fire	Disponibil in paleta Tools
Utilitarul de puncte de oprire	Opreste executia programului la o locatie specificata	Disponibil in paleta Tools
Suspendarea executiei	Suspenda subVI-uri pentru a nu se executa in mod repetat in timpul depanarii	Folositi meniul Operate, apasati pe iconita de setare a subVI-ului (SubVI Node Setup sau VI Setup) in timp ce VI-ul este deschis
Inregistrarea datelor	Activeaza inregistrarea datelor din panoul frontal in fisiere	Folositi meniul Operate si submeniul Data Logging
GPIB Spy/NI Spy	Monitorizeaza chemarile catre driverii Windows-ului	Porniti aplicatia

6.5.9 Utilizarea utilitarelor de depanare

Lista de erori, Execution Highlightning, modul pas cu pas, utilitarul de proba, uitlitarul de puncte de oprire si suspendarea executiei au fost discutate in sectiunile anterioare. Aceste posibilitati incluse in Lab VIEW sunt foarte eficiente in depanarea codului cand sunt folosite odata cu ajutorul on-line (On-Line Help). Fiecare este o arma pe care programatorul o poate folosi sa urmareasca si sa rezolve problemele. Aceste utilitare sunt prezentate pe scurt in tabelul 6.2. Tabelul 6.2 listeaza utilitarul, aplicarea sau uzul acestuia si cum sa-l accesezi sau sa-l accesezi.

Software-ul proceseaza modelul care ruleaza determinand cand incepe faza de testare sau depanare a codului. Intr-un model iterativ depanarea este implicata in fiecare ciclu al procesului. In modelul cascadat, depanarea este facuta doar intr-un ciclu. In fiecare caz prima actiune este de a elimina erorile care impiedica VI-ul sa se execute. Lista de erori va asista in eleiminarea acestor erori ca sa se poata executa VI-ul. Aceasta parte din depanare ar trebui efectuata cand aplicatia este in curs de dezvoltare, fara sa priviti cu atentie modelul folosit. Eliminarea erorilor care

impiedica aplicatia sa se execute poate fi considerata ca parte din faza de codare in Lab VIEW. Acesta este analog erorilor de sintaxa in limbajele traditionale care ii reiese programatorului in fata in timpul codarii. Lista de erori face acest lucru usor inca si pentru nivicii programarii. Il ghideaza pe programator in a rezolva rapid erorile.

Daca este posibil incercati sa testati cate un VI, pe rand. Testati driverele individual si subVI-urile individual, inainte de executie. Veti mult mai protejat daca veti incerca sa depanati un program mare cu multe subVI-uri. Nu numai ca este mai usor sa va concentrati asupra partilor mai mici din program, dar puteti reduce erorile care pot apare la interactiile intre subVI-uri unul cu altul. Un proiect modular apropiat VI-ului si care este explicat simplifica testele. Interactiunea cu cursul datelor poate face sa para ca exista maimulte erori. Puteti fi in stare sa creati un simulator pentro I/O sau pentru alte protiuni din cod care nu au fost inca pregatite. Din nou, acest lucru ajuta la izolarea problemelor fara a va confrunta cu erori I/O.

Odata ce VI-ul poate fi executat, urmatorul pas este sa-l executati cu Exception Highlighting activat. Animatia te ajuta sa vezi cursul datelor prin diagrama de cod. Execution Hightlighting va va ajuta sa gasiti defecte cauzate de conectarea incorecta a obiectelor. In timp ce VI-ul este rulat, fiti siguri ca codul se executa in ordinea intentionata de dumneavoastra, care nu poate fi identificata cu scoaterea in evidenta.

Poate ca vreti sa probati unele fire cu Execution Highlighting si sa fiti siguri ca valorile sunt corecte folosind Probe Tool. Probarea pachetului de erori intre doua obicte sau VI-uri puteti identifica unde este generata eroarea. Veti vedea valoarea utilitarului de proba odata ce incepeti sa-l folositi. Probe Tool si Execution Highlighting pot fi folosite in modul pas cu pas. Modul pas cu pas va lasa sa va uitati la o sectiune de cod in mai mult detaliu pentru gasirea problemelor existente.

Daca problemele persista, iata cateva sugestii pe care ar trebui sa le luati in consideratie. Acestea pot parea de baza, dar sunt cele care se omit foarte usor. Mai intai, fiti siguri ca valorile relatate de controloarele de utilizator sunt corecte. Utilitarul de proba poate fi folosit pentru efectuarea acestei verificari in diagrama de cod. Cand aceste valori date sunt in afara limitei acceptate, codul nu se va executa cum se intentiona.

Daca efectuati comunicatia cum un dispozitiv extern. Fisier sau aplicatie, verificati comenzile si datele care se trimit. Dispozitivul poate nu va raspunde la comenzi neasteptate. In timpul acestui proces verificati numele corect al fisierului, tratarilor si adreselor. Examinati dispozitivul extern pentru a vedeadaca functioneaza corect, si efectuati manual actiunile pe care incercati sa le automatizati. Presupuneti ca folositi pauze in program daca dispozitivul nu raspunde repede. Investigati ordinea de executie di cod pentru a vedea daca are loc secventa corecta de evenimente. Pot rezulta conditii concurente daca codul nu se executa cum se intentioneaza.

Inspectati matricile pentru folosirea corecta a indicilor. Matricile, listele, sunetele si tipurile enumerate, toate pornesc de la zero si pot cauza probleme serioase daca nu sunt luate in consideratie. In timpul acestei inspectari, verificati structurile proceselor care sunt coordonate de aceste valori sa vedeti daca corespund. De asemenea fiti sigur ca aveti procesul implicit incorporat pentru a fi sigur ca se executa codul corect. Puteti de asemenea sa examinati structurile bucla pentru a vedea daca sunt folositi corect registrii de transfer ca sa nu se piarda informatie. Acest lucru include si initializarea corecta a registrilor de deplasare.

Setati limite de timp proprii pentru cat timp vei incerca sa determini unde exista eroare in cod. Poate deveni foarte frustant sa incercati sa depanati o sectiune de

cod ore in sir. Cand limita de timp expira o a doua opinie este adusa in fata. Aceasta a doua perspectiva va vedea problema programarii diferit si poate cu succes sa propuna o solutie la intrebarile puse care te poate conduce la o solutie.

6.6 Sumar

Cand depanati o aplicatie ar putea fi mai usor doar sa omiti din codul pentru efectuarea tratarii sau detectiei de erori pentru ca necesita munca in plus. Cu toate astea, tratarea exceptiilor este necesara pentru controlarea problemelor care pot apare in timpul executiei. O exceptie este ceva ce poate sa se iveasca in timpul executiei programului. Aceste evenimente neasteptate sau exceptii, trebuie sa le infruntati intr-o manierapotrivita. Daca exceptiile sunt lasate in pace, puteti pierde controlul programului, care poate duce la mai multe probleme.

O tratre de exceptii permite utilizatorului sa infrunte unele situatii care pot apare in timpul rularii aplicatiei. Este un mecanism care poate detecta si eventual corecta erorile. Lab VIEW prezinta cateva utilitare incluse care il ajuta pe programator in detectia de erori, dar este responsabilitatea programatorului sa implementeze codul de tratare a exceptiilor. Cateva metode pentru confruntarea su erorile au fost descrise in acest capitol. Topica discutata va asista programatorul in a scrie cod cat mai robust pentru implementarea tratarii erorilor.

Tratarea erorilor poate fi considerata ca o faza primara in dezvoltarea unei aplicatii. Este potrivit sa luat in seama tratarea erorilor si dupa aceea sa decidem structura si arhitectura aplicatiei. Aplicatii mai bune se pot depana cand tratarea esceptiilor este primul primara, nu secundara. Tratarea exceptiilor, cand este inclusa in aplicatie, va conduce la un cod mai solid si mai sigur.

BIBLIOGRAFIE

G Programming Reference, National Instruments
Professional G Developers Tools Reference Manual, National Instruments
LabVlEWFunction and VIReference Manual, National Instruments LabVIEW On-line Reference, National Instruments