

NIdays 2014

Creating scalable test architectures using LabVIEW Object
Oriented Programming and TestStand

14. lokakuuta 2014



Agenda

- Perusajatus olio-ohjelmoinnista
- Miten TestStand:ssa voidaan hyödyntää oliopohjaista LabVIEW koodia?
- Perinteinen ohjelmointitapa vs. oliopohjainen
- Demosekvenssien + koodin esittely
- Seuraavat askeleet

Esityksen Fokus

LabVIEW Object Oriented Programming & TestStand



LVOOP

LabVIEW Object Oriented Programming

- “Object-oriented programming has demonstrated its superiority over procedural programming as an architecture choice in several programming languages. It encourages cleaner interfaces between sections of the code, it is easier to debug, and it scales better for large programming teams.”

Lähde: <http://www.ni.com/white-paper/3573/en/>

LVOOP

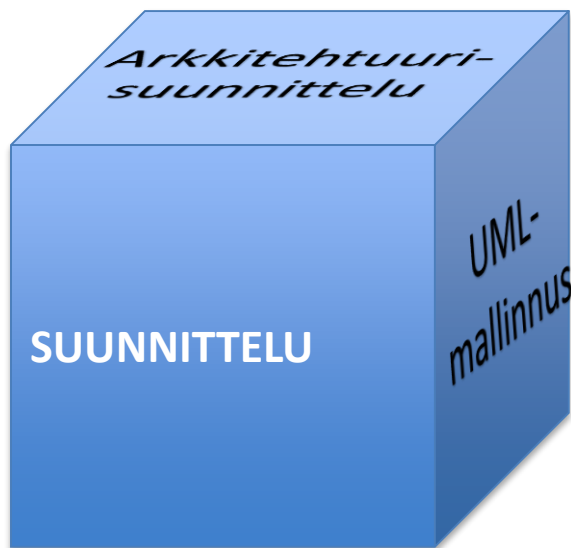
LabVIEW Object Oriented Programming

- “The idea behind object-oriented programming is that a computer program may be seen as comprising a collection of individual units, or *objects*, that act on each other, as opposed to a traditional view in which a program may be seen as a collection of functions, or simply as a list of instructions to the computer. Each object is capable of receiving messages, processing data, and sending messages to other objects. Each object can be viewed as an independent little machine or actor with a distinct role or responsibility.”

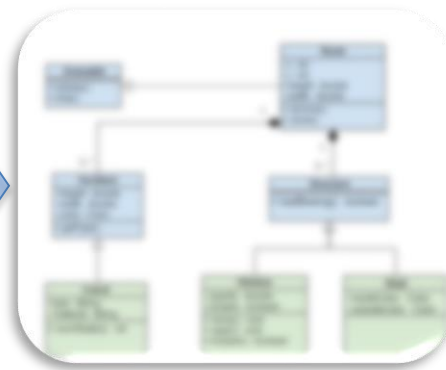
Lähde: http://en.wikipedia.org/wiki/Object-oriented_programming

LVOOP

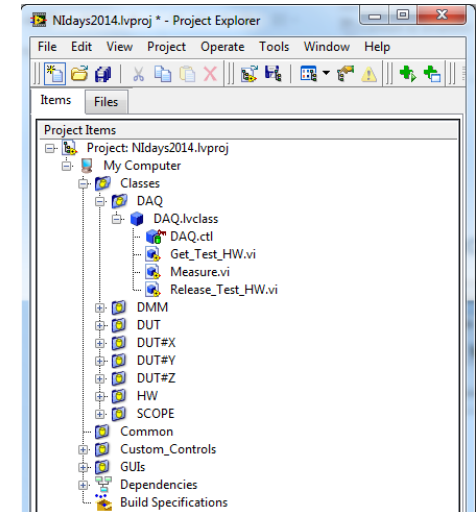
LabVIEW Object Oriented Programming & TestStand



Luokkakaavio



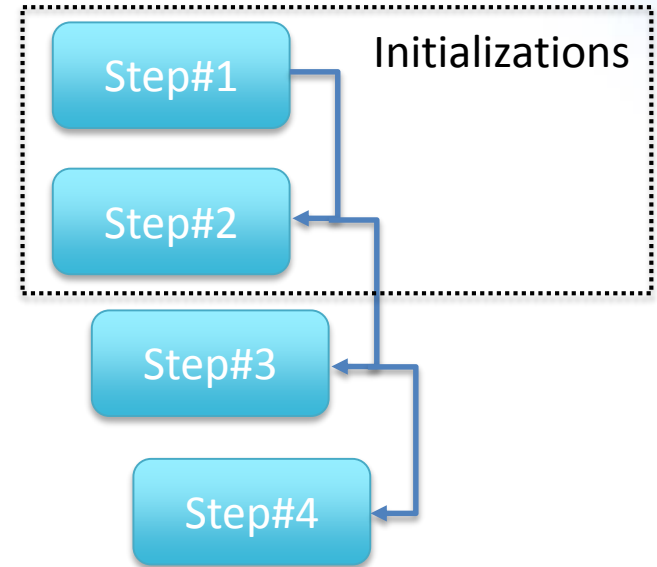
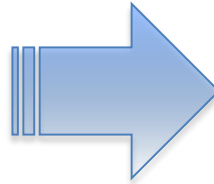
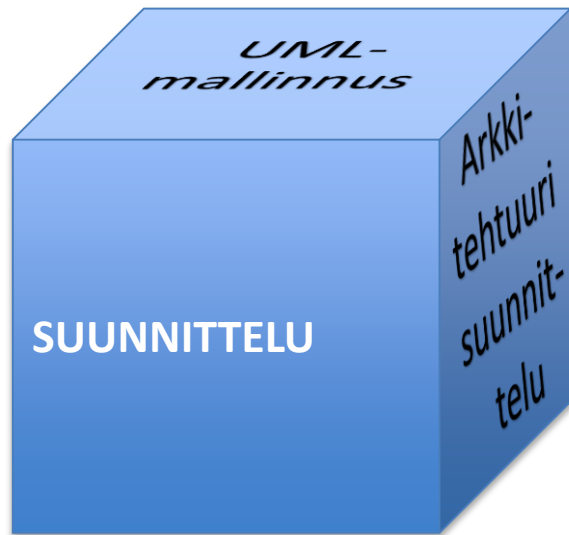
Toteutus



- Oliopohjainen ohjelmointi vaatii suunnittelua
- Code-And-Fix –pohjainen ohjelmointi ei (välttämättä) johda tarpeeksi hyvään lopputulokseen
- UML-mallinnus (Unified Modeling language): <http://fi.wikipedia.org/wiki/UML-mallinnus>

LVOOP

LabVIEW Object Oriented Programming & TestStand



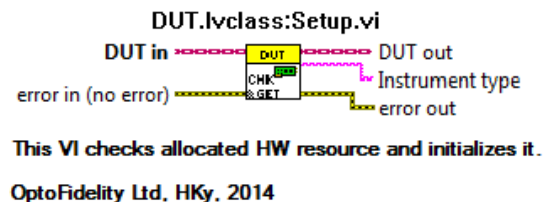
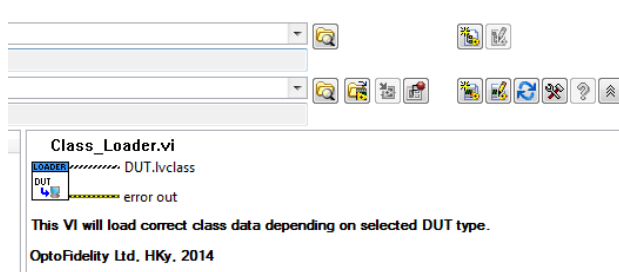
- Myös TestStand:n puolella on syytä suunnitella testirutiinin arkkitehtuuri ennen implementointia
- Syytä miettiä tarkaan mitä jätetään TestStand:n ja mitä LabVIEW:n tehtäväksi

LVOOP

LabVIEW Object Oriented Programming & TestStand

Variables			
Name	Value	Type	Co
Locals ('MainSequence')			
ResultList		Array of Result[0..empty]	
<Right click to insert Local>			
Parameters ('MainSequence')			
DUT_Class_data	Nothing	Object Reference (By reference)	
<Right click to insert Parameter>			

- Objekti välitetään TestStand:ssa “Object Reference” tietotyyppin avulla

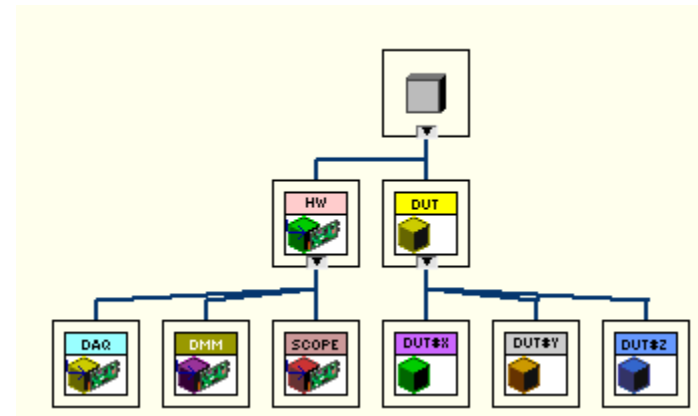
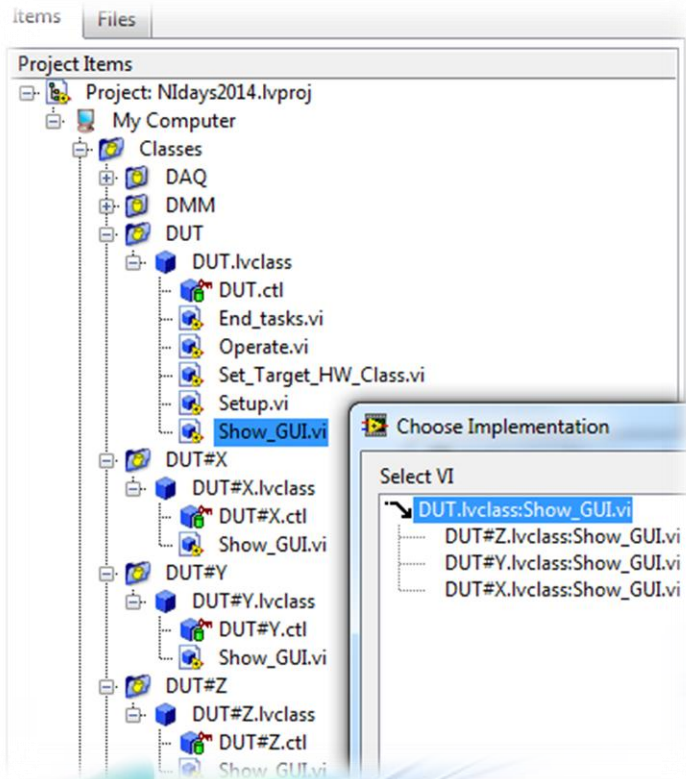


- “Langan” malli indikoi onko luokka ladattu muistiin

Dynamic Dispatching

LabVIEW Object Oriented Programming & TestStand

- **HUOM!** Sinulla tulee olla vähintään TestStand 2012 käytössäsi jotta voit käyttää tätä ominaisuutta

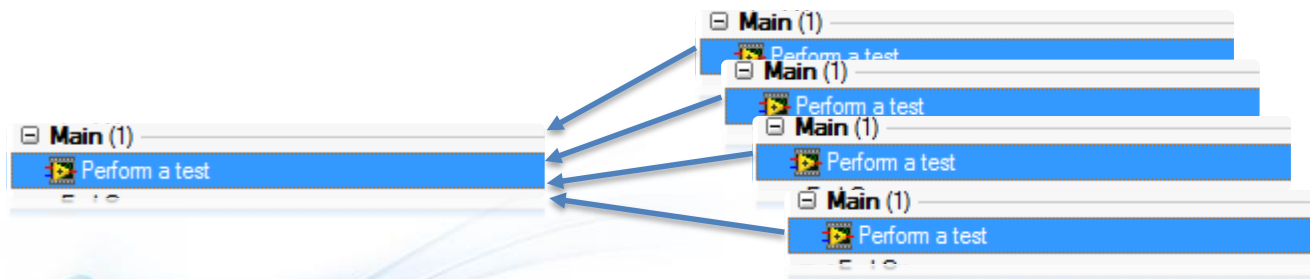


Dynamic Dispatching

LabVIEW Object Oriented Programming & TestStand

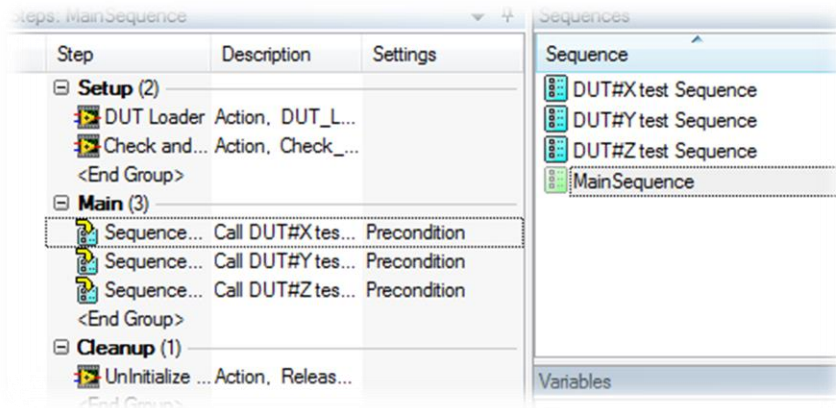
“Dynamic dispatch subVIs can call any one of a set of VIs in a LabVIEW class hierarchy. LabVIEW determines which implementation of the subVI to call at run time, depending on the class data type flowing into the dynamic dispatch terminal.”

Lähde: http://zone.ni.com/reference/en-XX/help/371361K-01/lvhowto/viewing_imp_dyn_dispatch/

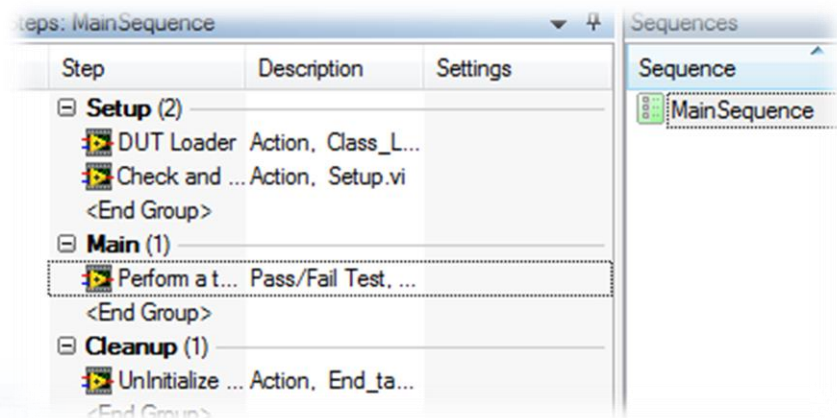


Ohjelmointitavat

Perinteinen vs. Oliopohjainen



VS.



Ohjelmointitavat – Perinteinen tapa

Perinteinen vs. Oliopohjainen

- Perinteinen TestStandin ”ohjelmointitapa” johtaa mahdollisesti tulokseen jossa:
 - Testirutiini on sekava ja monista eri osista koostuva
 - Skaalautuvuus heikkoa
 - Modulaarisuus heikkoa
 - Yleensäkin ylläpidettävyyys voi olla hankalaa, jos muutostaajuus on suuri

Ohjelmointitavat – oliopohjainen tapa

Perinteinen vs. Oliopohjainen

- Oliopohjainen ”ohjelmointitapa” johtaa mahdollisesti tulokseen jossa:
 - Skaalautuvuus hyvää
 - Ylläpito helppoa
 - Uuden suunnittelu aina hieman työläämpää

LVOOP

LVOOP

LVOOP

DEMO(t)

Käyttöesimerkki perinteinen ohjelmointitapa

DEMO

DEMO(t)

Käyttöesimerkki oliopohjainen ohjelmointitapa #1

DEMO

DEMO(t)

Käyttöesimerkki oliopohjainen ohjelmointitapa #2

- Dynamic Dispatching

DEMO

Mahdollisia ongelmaita varten

LabVIEW Object Oriented Programming & TestStand

- **Why Do I Receive Error -18002 When Running a Dynamically Deployed VI in TestStand?**
<http://digital.ni.com/public.nsf/allkb/74798BFEB7A53C7486257C54006BC5A1>
- **Calling LabVIEW Class Member VIs from TestStand**
<http://zone.ni.com/reference/en-XX/help/370052M-01/tslabview/infotopics/class/>
- **TestStand Deployment Fails When Deploying Dynamic Dispatch VI**
<http://digital.ni.com/public.nsf/allkb/4C708D2BD07FC49B862573BD006AC416>

Seuraavat askeleet

Mistä lisää tietoa tai apua?

- **OptoFidelity Oy**
 - Vankkaa LabVIEW osaamista (8 CLD tasoista kehittäjää)
 - Hyvää TestStand osaamista (avaimetkäteen-, osaprojekti-, ja konsultaatiota)
- **National Instruments**
 - Object-Oriented Design and Programming in LabVIEW**
<http://sine.ni.com/tacs/app/overview/p/ap/of/lang/fi/oc/fi/pg/1/sn/n24:14963/id/1587/>
- **Self-Packed**
 - Introduction to LabVIEW Object Oriented Programming**
<http://www.ni.com/webcast/2703/en/>
 - LabVIEW Object-Oriented Programming FAQ**
<http://www.ni.com/white-paper/3573/en/>

Kiitos ajastanne!