Acte d'Investidura com a Doctora *honoris causa*

de la

Universitat Politècnica de Catalunya · Barcelona**Tech**

Barbara Liskov

> ÍNDEX / TABLE OF CONTENTS


guració
urs acadèmic
-2013
Acte solemne d'investidura
Prof. Barbara Liskov
ANTONI
ANA
ABELAIRA

Acte d'Investidura de la professora Barbara Liskov com a Doctora *honoris causa* de la **Universitat Politècnica de Catalunya** · Barcelona**Tech**

28 de setembre de 2012

> ÍNDEX / TABLE OF CONTENTS

Servei de Comunicació i Promoció de la UPC, 2013 (9086)
Fotos de l'acte: Jordi Pareto

Índex

Table of Contents

Ordre de l'Acte d'Investidura

Benvinguda del rector magnífic de la **Universitat Politècnica de Catalunya** · Barcelona**Tech**.

Lectura de l'acord del Consell de Govern, per part de la secretària general.

La calma del mar (harmonització d'Enric Ribó).

Laudatio de la padrina, la professora Núria Castells.

Acte solemne d'investidura de la professora Barbara Liskov com a doctora *honoris causa* per la **Universitat Politècnica de Catalunya** · Barcelona**Tech**.

Discurs de la nova doctora, la professora Barbara Liskov.

Canticorum Iubilo (Georg Friedrich Händel).

Paraules del rector.

Gaudeamus Igitur (himne universitari, harmonització de Cornel Arany).

Interpretacions musicals a càrrec de la Coral Arquitectura i l'Orquestra de la UPC, sota la direcció de Lluís Carné i Miguélez.

Elogi dels mèrits de la professora Barbara Liskov

per la Professora Núria Castells
***Universitat Politècnica de Catalunya** · BarcelonaTech*

Distingit rector de la **Universitat Politècnica de Catalunya · BarcelonaTech**, distingits membres del Claustre Universitari i del Consell Social, i representants d'institucions i empreses, professors, estudiants, membres del personal d'administració i serveis, familiars i amics, i benvolguda doctora Barbara Liskov.

Em plau enormement ser avui aquí, en aquesta cerimònia solemne, per rebre un nou doctora *honoris causa* al Claustre Universitari de la Universitat Politècnica de Catalunya, en virtut de l'Acord de 12 de juny de 2012 del Consell de Govern, patrocinada per la Facultat d'Informàtica de Barcelona, de la qual tinc l'orgull de ser la degana. El nomenament de la professora Liskov ha rebut el suport dels departaments i les facultats de la UPC, algunes empreses importants del sector de la tecnologia de la informació, i altres organitzacions, cosa que dóna fe de la preeminència i gran reputació de la persona que rep aquesta distinció de la UPC.

La professora Barbara Liskov és *institute professor* (el títol més alt) d'Informàtica a l'Institut Tecnològic de Massachusetts, de Boston, i és cap del Grup de Metodologia de Programació. Els seus interessos en recerca inclouen sistemes distribuïts, reproducció d'algoritmes per obtenir tolerància a fallades, metodologia de programació i llenguatges de programació. Actualment els seus projectes de recerca inclouen sistemes d'emmagatzematge tolerants a fallades bizantines i sistemes d'emmagatzematge en línia que ofereixen la confidencialitat i la integritat de la informació emmagatzemada.

La investigació de Barbara Liskov l'ha conduït a fer grans descobriments en camps tan fonamentals com els sistemes operatius, els sistemes distribuïts, els llenguatges de programació i la metodologia de programació. Les seves idees han ajudat a formar la base de llenguatges de programació, com el Java, que estan dissenyats per beneficiar-se de mòduls de dades independents i d'instruccions que es poden desenvolupar una vegada i ser reutilitzats per a moltes finalitats diferents.

Avui miraré de resumir el treball extens i prolífic dut a terme per la doctora Barbara Liskov al llarg dels quaranta anys de carrera acadèmica i investigadora.

La tecnologia de la informació és una de les disciplines més noves de l'enginyeria. Les contribucions que hi ha fet al llarg dels darrers trenta anys han canviat dràsticament el món. L'eficiència i les noves oportunitats que ofereixen els ordinadors han transformat i han modificat tots els camps de l'activitat humana. Ningú no es podria imaginar un nivell de desenvolupament social tal com el coneixem avui sense la contribució dels informàtics.

El treball i la intel·ligència dels pioners de la informàtica, entre els quals hi ha Barbara Liskov, han posat els fonaments d'una ciència que és el motor del progrés que hem viscut en les últimes dècades.

El 2012 és un any especial per a la comunitat informàtica. És el centenari del naixement a Londres d'Alan Turing. El món commemora la vida de Turing i els seus èxits. Va ser extremament influent en el desenvolupament de la informàtica. Va formalitzar els conceptes d'algoritme i computació amb la màquina de Turing, que es pot considerar el model d'ordinador d'ús general. Alan Turing està considerat arreu com el pare de la informàtica.

El 2012 també és un any especial per a la Facultat d'Informàtica de Barcelona (FIB). Se celebra el seu 35è aniversari. Des de 1977 la nostra facultat ofereix una educació universitària de qualitat i excel·lència dins del camp de l'enginyeria informàtica, que respon a les necessitats de l'entorn econòmic i social. La FIB és part de la comunitat informàtica internacional. El treball que han fet els seus més de 250 investigadors i la formació que ha ofert durant aquests 35 anys han contribuït al progrés de la nostra societat.

Des d'aquest punt de vista, la comunitat científica i la UPC reconeixen el valor dels èxits assolits per la doctora Barbara Liskov.

Barbara Liskov, nascuda com a Barbara Jane Huberman el 7 de novembre del 1939 a Califòrnia, va obtenir la llicenciatura en Matemàtiques a la Universitat de Califòrnia, Berkeley, el 1961. En lloc d'anar directament a una escola de postgrau, va acceptar una feina a la Corporació Mitre, on es va implicar en el món de la programació informàtica.

A principis dels anys seixanta la programació es podia considerar un món experimental. Les característiques, el rendiment, el cost i la fiabilitat d'aquelles màquines només les feien accessibles a uns quants experts. La feina a Mitre va ser el començament de la prolífica carrera de Barbara Liskov.

Després d'un any a Mitre, es va traslladar a la Universitat de Harvard per treballar en un projecte que tenia per objectiu traduir automàticament frases de l'anglès a alguna cosa que pogués entendre l'ordinador. Tot i que el processament de llenguatge natural és un problema en què els científics informàtics encara estan treballant, aleshores la gent esperava que es resolgués en pocs anys.

De tornada a Califòrnia per fer el treball de postgrau a Stanford, va rebre suport econòmic al Laboratori de John McCarthy, en part perquè el seu treball anterior sobre traducció de llenguatge natural formava part del camp de la intel·ligència artificial. El 1968 va esdevenir una de les primeres dones dels Estats Units que va obtenir un doctorat en Informàtica. La seva tesi sobre el final de partides d'escacs va ser supervisada per John McCarthy.

Després d'obtenir el doctorat, Barbara es va casar amb Nathan Liskov i va tornar a la zona de Boston per treballar a la Corporació Mitre, a Bedford, Massachusetts, en el disseny d'ordinadors i sistemes operatius. Amb un ordinador Interdata 3 que tenia la capacitat de canviar el conjunt d'instruccions a través de microcodi, va crear l'ordinador Venus, fet a mida perquè fos compatible amb la construcció de programari complex. El sistema operatiu Venus era un petit sistema en temps compartit per a l'ordinador Venus, que s'utilitzava per experimentar sobre la manera com les arquitectures diferents ajudaven o perjudicaven aquest procés. El sistema Venus admetia 16 teletips i cada usuari estava connectat a un ordinador virtual perquè els errors importants d'un usuari no posessin en perill tot el sistema, sinó tan sols la màquina virtual d'aquell usuari.

El 1971, poc després d'acabar els experiments amb Venus i de presentar una ponència sobre el tema, una persona que assistia a la conferència va animar Liskov a acceptar un lloc de treball al MIT. Va deixar Mitre i es va incorporar a l'equip docent del MIT com a professora del Laboratori d'Informàtica. Basant-se en l'experiència en la Corporació Mitre, la seva investigació es va centrar a crear sistemes informàtics més fiables.

Al MIT va liderar el disseny i la implementació del llenguatge de programació CLU, que posava l'èmfasi en les nocions de la programació modular, l'abstracció de dades i el polimorfisme. Va ser el primer llenguatge que admetia l'abstracció de dades. Aquests conceptes són un fonament per a la programació orientada a objectes que es fa servir en llenguatges informàtics moderns com Java i C#, tot i que moltes altres funcions de la programació moderna orientada a objectes són absents en aquest llenguatge primerenc.

Totes les aplicacions de programari sofisticades són una estructura complexa de parts entrellaçades, sovint modificades al llarg del temps per un equip ampli d'enginyers. Qualsevol canvi pot tenir efectes no desitjats en altres parts del programari i fa que els programadors pràcticament hagin de reescriure el programa. Barbara Liskov va trobar maneres d'estructurar els programes en parts separades, o "mòduls d'operacions múltiples", de manera que seria menys probable que els canvis afectessin el codi fora de certs límits.

Com que explicar les seves idees als programadors era difícil, va dissenyar un llenguatge de programació que les posava directament en pràctica. "Tenia una idea molt clara de què eren programes bons i programes dolents", diu ella. "Volia fer fàcil a la gent escriure bons programes i, tot i que no es pot evitar que la gent escrigui mals programes, no volia que els resultés massa fàcil."

El seu grup de MIT també va crear el llenguatge Argus, el primer llenguatge d'alt nivell compatible amb la implementació de programes distribuïts per una xarxa, cosa que ampliava les idees de CLU, incloent-hi compatibilitat amb transaccions imbricades. Un exemple d'un programa distribuït d'aquesta manera podria ser un sistema bancari basat en la xarxa. Argus proporcionava abstraccions d'objectes anomenades "guardes" que encapsulen procediments que hi estan relacionats. Com a llenguatge experimental, l'Argus va influenciar altres desenvolupadors, però no es va adoptar mai d'una manera generalitzada ni es va utilitzar en aplicacions en xarxa.

Liskov també és a l'origen de Thor, un sistema de base de dades orientat a objectes que proporciona accés transaccional a objectes persistents altament disponibles en sistemes distribuïts a gran escala.

Posteriorment, Liskov va treballar sobretot en el camp dels sistemes distribuïts, que utilitzen diversos ordinadors connectats a una xarxa. La seva recerca va cobrir molts aspectes dels sistemes operatius i de la computació, incloent-hi un treball important sobre sistemes de base de dades orientat a objectes, recuperació de memòria, gestió de la memòria cau, persistència, recuperació, tolerància a fallades, seguretat, flux d'informació descentralitzat, actualització modular de sistemes distribuïts, encaminament geogràfic i la tolerància funcional a fallades bizantines. Molts d'aquests treballs, com la tolerància a fallades bizantines, tracten de situacions en què un sistema complex falla de maneres arbitràries. Liskov va desenvolupar mètodes per permetre un funcionament correcte fins i tot quan alguns components no són fiables.

Amb Jeannette Wing va desenvolupar una nova noció de subtipificació, coneguda com el principi de substitució de Liskov. Les seves contribucions han influït en els desenvolupaments de sistema avançat i han establert un estàndard de claredat i utilitat.

Actualment Liskov és professora d'Enginyeria al MIT, on dirigeix el Grup de Metodologia de Programació, que se centra en la recerca actual de la tolerància a fallades bizantines i computació distribuïda. El 1980 va esdevenir catedràtica del MIT. Del 2001 al 2004 hi va treballar com a directora associada d'Informàtica, i el 2007 va ser nomenada rectora associada per a l'Equitat Docent. El 2008, el MIT la va nomenar *institute professor*, el reconeixement més important que el MIT atorga als membres del professorat.

La presidenta del MIT, Susan Hockfield, va afirmar: "la comunitat del MIT revereix Barbara Liskov pel seu paper com a acadèmica, mentora i líder. La seva recerca innovadora l'ha convertida en una de les autoritats mundials en llenguatge informàtic i disseny de sistemes. A més de les seves contribucions acadèmiques fonamentals, Barbara Liskov ha servit el MIT amb molta professionalitat i bons criteris en diversos càrrecs administratius, recentment com a rectora associada per a l'Equitat Docent."

Ha escrit tres llibres científics i més de 150 articles científics en publicacions internacionals capdavanteres i conferències. Ha supervisat els programes de recerca de més de 25 estudiants de doctorat i una gran quantitat d'estudiants de màster.

La seva tasca ha tingut un impacte important en el treball de recerca dut a terme al Departament de Programació de la Facultat d'Informàtica de Barcelona (FIB), que actualment forma part del Departament de Llenguatges i Sistemes Informàtics de la UPC. El treball que s'hi va dur a terme sobre **especificació algebraica estava basat en el concepte d'abstracció de dades** de Barbara Liskov. Les contribucions metodològiques sobre disseny de programes també estaven basades en la seva tasca. A més, les propostes de Liskov s'han utilitzat durant molts anys com a base de l'ensenyament de programació a la FIB.

Barbara Liskov ha rebut diversos honors i premis. És membre de l'Acadèmia Nacional d'Enginyeria i de l'Acadèmia Nacional de Ciències, i *fellow* de l'Acadèmia Americana d'Arts i Ciències i de l'Associació per a la Maquinària Informàtica (ACM); el 1996 va rebre el Premi a la Trajectòria Professional de la Societat de Dones Enginyeres, el 2004 la Medalla John von Neuman de l'IEEE, el 2008 el Premi pels Èxits en Llenguatge de Programació del Grup Dedicat als Llenguatges de Programació (SIGPLAN) de l'ACM, i el 2009 va rebre el premi A.M. Turing de l'ACM.

Ha rebut diversos doctorats honoraris, de l'Escola Federal Politècnica de Zuric, Suïssa (2005), la Universitat de Brown (2010), la Northwestern University, de Chicago (2010) i la Universitat de Lugano, Suïssa (2011).

El 2003 la publicació *Discover Magazine* la va definir com una de les cinquanta dones més importants del món de la ciència. El 2012 va ser convidada a formar part del National Inventors Hall of Fame.

Pel que fa a la seva activitat no científica, Barbara Liskov sempre ha animat i ajudat les estudiants i, els darrers anys, ha dedicat

molts esforços a fer que la informàtica sigui un camp més obert. "Ser una dona a l'inici del desenvolupament de la ciència informàtica hauria estat difícil per a algú que prestés més atenció a aquests obstacles", diu Liskov. I malgrat que aquestes barreres no li importaven gaire, els prejudicis de la societat li van impedir reconèixer la importància de la seva carrera fins que els seus propis interessos en investigació van començar a madurar.

Com a rectora associada per a l'Equitat Docent al MIT, treballa per incorporar més dones i membres docents de minories, i ajudar-los a administrar i promoure les seves carreres.

Així mateix, per la Barbara Liskov ha estat molt important tenir un bon nombre d'aficions fora de la feina, comptant-hi la jardineria i la lectura de novel·les de misteri. És una persona molt amable i oberta.

Per acabar, cal dir que la FIB va començar la seva vida acadèmica el setembre de 1977 i que, per tant, en celebrem el 35è aniversari. Alhora celebrem el centenari d'Alan Turing. La millor manera de dotar de rellevància acadèmica aquests esdeveniments és donar el màxim reconeixement de la UPC a una dona pionera en el món de la informàtica. Felicitats, Barbara!

Discurs pronunciat per la professora Barbara Liskov

Programació de la màquina de Turing

Barbara Liskov
Institut Tecnològic de Massachusetts
Laboratori d'Informàtica i Intel·ligència Artificial
28 de setembre de 2012

Aquest any es compleix el centenari del naixement d'Alan Turing. Turing va ajudar a definir les bases teòriques de la informàtica amb la seva definició d'un ordinador de programa emmagatzemat. El model que va proporcionar fonamenta la manera com pensem sobre els programes avui en dia.

Tanmateix, és difícil fer que els ordinadors duguin a terme tasques útils. Els sistemes de programari són enormes –milions de línies de codi. I a més, fan tasques complicades. El model de Turing no ens proporciona coneixement sobre com dissenyar aquests sistemes.

Trobar tècniques per omplir aquesta llacuna de complexitat ha estat una qüestió primordial en la investigació informàtica. Una àrea en què s'ha fet un progrés colossal és la creació d'algoritmes, estructures de dades i protocols. Aquests avenços proporcionen als informàtics unes eines valuoses. Per exemple, si s'ha de fer una recerca eficient, ja no cal inventar enfocaments nous. En comptes d'això es pot triar l'estructura de dades, com ara una taula de dispersió o un arbre equilibrat, que s'adapti més bé al problema.

Tanmateix, els avenços en algoritmes, estructures de dades i protocols, per molt útils que siguin, no són un problema fonamental: quan el programador s'enfronta amb la necessitat de desenvolupar una nova aplicació o un nou sistema de programari, com elabora un disseny efectiu? Fins i tot, abans que s'utilitzin les eines, és necessari trobar aquest disseny.

Aquesta qüestió ha estat el centre de la recerca en metodologia de programació. A més, atès que la recerca en metodologia de programació ha identificat conceptes útils, s'ha desenvolupat la recerca en llenguatges de programació centrada a proporcionar característiques lingüístiques per donar suport a aquests conceptes.

En aquest discurs parlaré dels primers anys de la recerca en metodologia de programació i llenguatges de programació.

> ÍNDEX / TABLE OF CONTENTS

Vaig començar a treballar en aquest camp cap al 1970. Aleshores havia obtingut el doctorat a Stanford i treballava a Mitre Corporation. El meu doctorat a Stanford tractava d'intel·ligència artificial. A Mitre vaig passar a treballar en sistemes informàtics. A mig fer la tesi doctoral, ja sabia que volia fer aquest canvi, però vaig decidir esperar a tenir el doctorat per fer-lo.

La Corporació Mitre és una societat sense ànim de lucre que fa recerca per al govern dels Estats Units. El meu primer projecte a Mitre consistia a dissenyar un conjunt instruccions per a una nova màquina, implementar-lo utilitzant microprogramació i després mostrar-ne la utilitat fent-lo servir per implementar un sistema en temps compartit anomenat Venus **[3]**. Quan es va acabar el projecte, em van demanar que treballés en una investigació per resoldre la "crisi del software".

El problema anomenat "crisi del software" significava que la gent no sabia com construir programari d'una manera eficient perquè, al final, el programari fes el que se n'esperava. Al llarg dels anys setanta i vuitanta es parlava de projectes de programari fracassats en els quals s'havien invertit milions de dòlars i molts anys de feina per crear un sistema que finalment s'havia hagut d'abandonar perquè no funcionava bé.

El govern dels Estats Units estava preocupat per la crisi del software perquè necessitava sistemes de programari ambiciosos per a finalitats militars i d'altres tipus, i tenia problemes per obtenir el programari necessari.

Quan vaig començar a treballar en el projecte, no tenia experiència en el camp de la metodologia de la programació, així que vaig començar a llegir-ne la bibliografia. Quan vaig rebre el Premi Turing el 2009, vaig tornar a llegir aquesta bibliografia. Vaig trobar que aquests articles eren molt interessants tant per si mateixos com també des d'una perspectiva històrica. En la ponència només parlarem d'un parell d'aquests articles.

El primer és "Go to statement considered harmful **[1]**", un article famós d'Edsger Dijkstra. En aquest assaig, Dijkstra va argumentar que la gent no havia d'utilitzar l'expressió *go to* en els seus programes. Pot semblar estrany tractar d'una qüestió com aquesta en un article des de la perspectiva actual, perquè els nostres llenguatges de programació no tenen l'expressió *go to*. Però això no passava amb els llenguatges de programació que hi havia aleshores.

L'article de Dijkstra va ser molt polèmic. Alguns el van trobar ximple: estaven acostumats a fer servir expressions *go to* i no es podien imaginar que desapareixerien. D'altres es van sentir insultats. Argumentaven que, malgrat que utilitzaven l'expressió go to als seus programes, els programes estaven ben estructurats. I potser això era veritat, perquè es pot escriure un programa ben estructurat en qualsevol llenguatge de programació sempre que se sigui disciplinat a l'hora d'utilitzar-ne les característiques.

Malgrat tot, hi havia qui defensava que l'expressió *go to* era una funció imprescindible. Pensaven que sense aquesta expressió no seria possible escriure programari eficient. Aquesta preocupació s'ha d'entendre en el context dels llenguatges de programació disponibles aleshores. En aquella època, els llenguatges de programació no tenien algunes característiques que tenen avui, com ara l'expressió *case* i les excepcions. L'expressió *go to* es pot fer servir per compensar l'absència d'aquestes característiques; per exemple, acceptar etiquetes com a paràmetres ofereix una bona manera de simular un mecanisme d'excepció.

Deixant de banda la polèmica que va desvetllar, l'article de Dijkstra també és interessant pels seus arguments. Sosté que l'expressió *go to* és dolenta perquè dificulta que es raoni sobre la correcció dels programes. Assenyala que raonem sobre la correcció dels programes estàticament, mirant el text dels programes, però que raonem sobre quelcom dinàmic, concretament sobre el comportament del programa quan s'executa. I diu que, atès que raonar sobre la correcció dels programes és difícil, el fet que les estructures estàtiques i dinàmiques estiguin al més juntes possible és una bona idea. Les estructures estaran juntes si un

programa s'escriu amb mecanismes ben estructurats, com ara les expressions *if* i *while*, i crides a funcions. Però amb l'expressió *go to*, els dos tipus d'estructures poden estar molt allunyats.

Dijkstra va exemplificar aquest punt analitzant la dificultat de numerar les expressions d'un programa de manera que correspongui a l'ordre d'execució: això és difícil de fer en un programa amb l'expressió *go to*. Una manera intuïtiva de pensar en l'argument de Dijkstra és pensar en un programa depurant-se. Si s'ha descobert un error en un punt concret del programa, per esbrinar què ha succeït, s'haurà d'entendre com s'hi ha arribat, és a dir, s'han d'entendre totes les expressions que s'han executat anteriorment, en l'ordre en què s'han executat. Això és relativament fàcil de fer si el programa fa servir estructures de control ben estructurades, però pot ser extremadament difícil amb l'expressió *go to*.

El segon article és "Information distribution aspects of design methodology **[6]**", de David Parnas. Aquest article tracta de l'estructura de programes. Parnas hi diu que un programa consisteix en un conjunt de *mòduls*, tot i que no hi ha cap definició sobre què és un mòdul exactament. No obstant això, per estudiar un programa com una recopilació de mòduls, hem d'entendre les seves *connexions*, és a dir, com interactuen. A Parnas li interessa què són aquestes connexions, i a l'article diu: "Les connexions entre mòduls són les assumpcions que fan els mòduls un de l'altre." Aquí Parnas assenyala que no n'hi ha prou de definir només "les seqüències de crida i els formats de bloc de control", és a dir, la manera com es crida el mòdul, perquè sovint això no captura totes les connexions. Més endavant, l'article de Parnas tracta d'una metodologia de disseny que té en compte la importància de les connexions. Parnas va publicar un altre article sobre disseny de programes aproximadament a la mateixa època **[7]**.

En aquest punt voldria parlar de modularitat. Tal com Parnas va assenyalar, hem de considerar un programa com una recopilació de mòduls. Els mòduls són útils per diverses raons, però la més important és que ens proporcionen una manera de raonar sobre allò que fa un programa. Naturalment, és molt difícil raonar sobre la correcció d'un programa d'un milió de línies com una sola entitat. Hem de trossejar el programa en fragments i treballar-hi per separat. Aquests fragments són mòduls.

Perquè aquest mètode funcioni, hem d'imaginar que els mòduls són "caixes negres". Això significa que només en podem veure la superfície, no l'interior. L'interior d'un mòdul és el codi que comprèn. La superfície és una mena de descripció d'allò que fa, o sigui, una *especificació* del seu comportament. Aquesta especificació ha de capturar tot el comportament del mòdul que pot ser percebut per la resta del programa, és a dir, totes les seves connexions.

Si tenim èxit amb aquesta descomposició modular, podrem avaluar la correcció d'un programa enorme mòdul per mòdul. Així és com funciona.

La implementació d'un mòdul és correcta si ofereix el comportament indicat a l'especificació. Per analitzar la implementació, n'examinem el codi. Aquest codi probablement utilitza altres mòduls, però podem pensar sobre aquests usos tenint en compte les especificacions d'aquests altres mòduls: si el mòdul A utilitza el mòdul B, podem entendre el que significa mirant l'especificació del mòdul B. El més important és que no necessitem mirar el codi del mòdul B. D'aquesta manera, ho podem analitzar de manera local: només ens cal mirar el codi d'A, a més de les especificacions d'altres mòduls, però no el seu codi.

Cal destacar que la capacitat d'ignorar la implementació del mòdul B redueix molt els esforços. Normalment, una especificació és més petita que el codi, però, a més a més, si mirem el codi, probablement trobarem que utilitza més mòduls. No obstant això, com que no veiem el codi, podem ignorar l'existència d'aquests mòduls addicionals. Per exemple, la implementació del mòdul B

podria fer servir un mòdul C, però quan analitzem la correcció del mòdul A no ens cal en absolut tenir en compte el mòdul C.

La modularitat, si es fa bé, té beneficis addicionals. Facilita una manera de subdividir un programa en peces amb què diverses persones poden treballar independentment sense interferències. També facilita la *modificabilitat*. Si tot el codi d'un sistema només depèn de l'especificació d'algun mòdul, podem reimplementar aquest mòdul sense haver de preocupar-nos per l'impacte d'aquest canvi en la resta del sistema. Mentre la nova implementació compleixi l'especificació del mòdul, la resta del programa no es veurà afectada pel canvi.

El 1970, tanmateix, aquests beneficis de la modularitat tot just s'estaven descobrint. A més, aleshores només hi havia un mecanisme de programació que tenia sentit com a mòdul: la subrutina o el procediment. De fet, un procediment és un bon exemple del que podem obtenir de la modularitat. Per exemple, pensem

en un procediment que ordeni una matriu. Per utilitzar aquest procediment, només s'ha de saber que, quan retorni, la matriu estarà ordenada –és el que diu l'especificació. No ens preocupa la tècnica d'ordenació que es fa servir en la implementació del procediment, i el codi utilitzat seguirà operant correctament si el procediment es reimplementa per utilitzar una tècnica diferent.

El problema és que els procediments no són prou potents per a tots els tipus de mòduls que es necessiten en els programes. Per exemple, és molt clar que un sistema de fitxers és un mòdul, però és igual de clar que no és un procediment. Per tant, en aquella època teníem el problema que ens mancaven mecanismes adients per a les idees sobre modularitat.

Un problema addicional era que els programes existents sovint no tenien una estructura modular. Per exemple, una vegada vaig ser responsable de mantenir un gran programa escrit en llenguatge d'assemblador (el fet que estigués escrit en assemblador no té importància). De tant en tant, el programa feia una crida a procediments, però sobretot era només un codi lineal, i generalment no s'hi podia aplicar un raonament modular. Això era normal en aquella època.

Finalment, fins i tot quan hi havia mòduls, aquests sovint tenien *connexions* grans. Per exemple, sovint tenien seqüències de crida complexes (tal com indica Parnas al seu article). Encara era pitjor el fet que tenien connexions addicionals a través de l'ús de variables globals. De fet, la comunicació a través de variables globals de vegades era la manera més habitual d'organitzar el codi!

Quan llegia aquests articles de què he parlat, i altres, em vaig adonar que jo havia inventat un sistema per modularitzar programes en la meva feina anterior a Mitre. Quan treballava en el disseny i la implementació del sistema operatiu Venus, volia saber si el meu grup petit de programadors podia produir un sistema de programari molt ambiciós d'una manera eficient.

Vaig entendre que, per tal d'aconseguir-ho, necessitàvem una manera d'organitzar el codi que en tingués la complexitat sota control en tot moment, perquè, així, poguéssim analitzar-ne la correcció. Per tant, vaig desenvolupar algunes normes per fer-ho. Vaig decidir que, en comptes d'organitzar el sistema en funció de variables globals a les quals tot el codi tingués accés, organitzaríem el codi amb allò que vaig anomenar *particions*. Les particions tenien dues normes:

- Cada partició tenia un subconjunt de variables globals. Només el codi d'aquella partició podia accedir (llegir o modificar) aquestes variables globals.
- Cada partició proporciona una o més operacions. Aquestes operacions podien ser utilitzades per altres particions per interactuar-hi. D'aquesta manera, altres particions podien afectar les variables globals dins la partició utilitzada, però no directament; en comptes d'això, això només es podia dur a terme fent crides a les operacions d'aquella partició.

Vaig escriure les meves idees sobre les particions a "A design methodology for reliable software systems **[2]**", publicat el 1972.

També el setembre de 1972 em vaig traslladar de Mitre a l'Institut Tecnològic de Massachusetts (MIT). Durant el meu primer semestre al MIT estava capficada amb la qüestió següent: com es podien fer més comprensibles aquestes idees sobre modularitat perquè els altres les poguessin aplicar als seus propis dissenys de sistemes. Creia que tots aquests articles contenien bones idees. Igualment, cada article explicava com es podien fer servir aquestes idees en el disseny de programes. Fins i tot de vegades contenien exemples d'un disseny que consistia en diversos mòduls. Les idees i els dissenys eren convincents; el lector devia pensar que aquesta era la manera de fer les coses bé. Però quan el lector s'havia d'enfrontar amb la feina de concebre un disseny propi, no era clar com s'hi podien aplicar les idees.

Crec que la raó per la qual això es feia tan difícil era que no hi havia res ben definit. En comptes d'això, els assaigs presentaven esbossos del que s'havia de fer, però les idees eren força vagues. Concretament, no hi havia cap definició d'una estructura de programa que tingués sentit a l'hora de definir mòduls.

D'aquesta manera, pensava de quina manera les coses es podien fer més bé. I, en algun moment de la tardor o l'hivern de 1972, vaig veure que la meva idea de particions es podia relacionar amb els tipus de dades. L'estat que s'amagava dins d'una partició corresponia a la representació d'un objecte de dades, i les operacions que la partició posava a disposició de la resta del sistema corresponien a les operacions que es podien utilitzar en l'objecte de dades.

Vaig saber de seguida que aquesta idea seria important, perquè aquests *tipus abstractes de dades* eren una extensió d'una cosa que els programadors entenien, és a dir, els tipus de dades que ja hi havia als llenguatges de programació. Els programadors estaven

acostumats a fer servir matrius cridant operacions de matriu com ara *recollir* i *emmagatzemar*. També sabien que el seu codi no podia accedir a la representació de la matriu a la memòria, ni se n'havien de preocupar. Així, la novetat principal era que el conjunt disponible de tipus de dades ja no estaria limitat al que permetés el llenguatge de programació. Al contrari, els programadors podrien definir-ne de nous. A més, podrien definir mòduls més potents que els que tenien a la seva disposició anteriorment.

El pas següent en el desenvolupament d'aquesta idea va ser esbrinar exactament què era. Per donar-li suport, vaig decidir fer-ho pensant en mecanismes de llenguatge de programació. Vaig començar treballant amb Steve Zilles en aquest projecte a la primavera de 1973. L'Steve era un estudiant de postgrau del MIT i alhora un empleat d'IBM, i també havia desenvolupat unes idees semblants [8], així que vam decidir treballar plegats.

L'Steve i jo vam treballar a la primavera i l'estiu de 1973, i cap a finals d'estiu ja havíem creat gran part del que era necessari en un llenguatge de programació que fos compatible amb els tipus abstractes de dades. Aquell estiu vam posar per escrit les nostres idees; el nostre article "Programming with abstract data types [5]" es va publicar l'any següent.

Aquest article definia un tipus abstracte de dades com un conjunt d'objectes i un conjunt d'operacions.

A més, teníem una norma: les operacions proporcionaven l'única manera d'utilitzar els objectes.

L'article també oferia un esbós del tipus de mecanismes de llenguatge que serien necessaris per permetre que es definissin i s'utilitzessin tipus abstractes de dades.

L'etapa següent va començar a la tardor de 1973. Aleshores tres estudiants de postgrau de primer any es van incorporar al meu equip: Russ Atkinson, Craig Schaffert i Alan Snyder. Aquests

tres estudiants i jo mateixa vam ser els dissenyadors d'un llenguatge de programació nou que admetia *tipus abstractes de dades*. Steve Zilles participava en les nostres reunions de disseny, però se centrava en la seva tesi doctoral, que tractava d'especificacions algebraiques de tipus abstractes de dades.

Encara que dissenyar un llenguatge de programació, lògicament, era el pas següent, jo també tenia les raons següents per fer aquest pas:

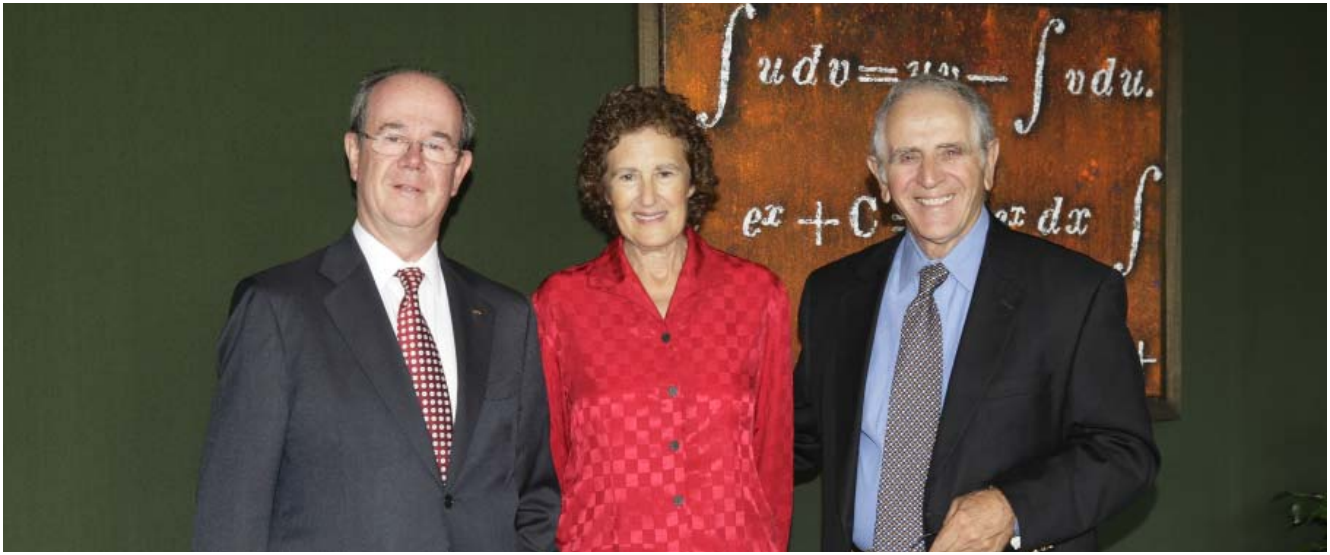
- Comunicació. Crec que els programadors entenen la programació a través de les característiques del llenguatge de programació. En conseqüència, en dissenyar i implementar un llenguatge de programació, jo permetria l'ús de tipus abstractes de dades d'una manera que no seria possible en absència del suport del llenguatge.
- Funcionalitat. En l'època en què vaig dur a terme aquesta feina, no era clar si els tipus abstractes de dades serien suficients a la pràctica per construir programes grans. En proporcionar un llenguatge de programació, permetria fer experiments que ajudarien a contestar aquesta pregunta. Els usuaris haurien d'escriure el codi utilitzant només les característiques del llenguatge de programació; no podrien fer trampa utilitzant característiques de nivells inferiors. Per tant, si els nostres mecanismes no eren adequats, ho descobriríem.

- Precisió. També creia que era important obtenir una definició precisa d'un tipus abstracte de dades. Un llenguatge de programació és un objecte matemàtic. Per tant, treballar en aquesta definició com a part d'un llenguatge de programació seria útil.
- Rendiment. No n'hi ha prou d'escriure el codi per a una aplicació –també és molt important que l'aplicació tingui un rendiment prou bo per satisfer els usuaris. Igualment, poder experimentar amb programes escrits en el llenguatge ens permetia veure com funcionava.

El llenguatge que vam dissenyar es deia CLU. El procés de disseny estava centrat en l'abstracció de dades: com permetre que es definissin i s'utilitzessin els tipus abstractes de dades, i quines característiques addicionals de llenguatge eren necessàries per facilitar les coses. Vam anar amb compte per limitar la nostra feina al problema i vam evitar entrar en altres qüestions de disseny de llenguatges que, tot i que eren interessants, no eren l'objectiu principal. Un exemple va ser la decisió, al principi, de centrar-nos en un llenguatge de programació seqüencial i ignorar la concurrència.

El CLU **[4]** va proporcionar-nos els mecanismes d'abstracció següents:

- Clústers. Es van utilitzar per implementar tipus abstractes de dades (el nom "CLU" són les tres primeres lletres de "clúster"). La capçalera d'un clúster enumerava les operacions disponibles, i aquesta era l'única manera que el codi extern al clúster pogués utilitzar els objectes. A l'interior, el codi definia com els objectes d'aquell tipus de dades es representaven i proporcionaven implementacions d'aquestes operacions.
- Polimorfisme. Molts tipus de dades tenen sentit per a diferents classes d'elements, per exemple, una pila pot contenir nombres enters o caràcters, etc. Seria preocupant per a un programador haver d'implementar un clúster per a cada tipus d'element que tingués sentit per a una pila. En lloc d'això, el



CLU admetia definicions *polimòrfiques*, cosa que funcionava en molts tipus. La part difícil de proporcionar aquest mecanisme era que sovint es va fer necessari restringir els tipus de paràmetres acceptats; per exemple, un conjunt ordenat només té sentit si el tipus de paràmetre proporciona una relació d'ordre. Per resoldre aquest problema, vam inventar clàusules *where*, que enumeraven les operacions que el tipus de paràmetre havia de tenir. D'aquesta manera, a l'hora de compilar podíem comprovar si el paràmetre concret era acceptable.

- Procediments i iteradors. El CLU permetia que es definissin els procediments independentment dels clústers. També proporcionava un nou mecanisme d'abstracció anomenat *iteradors*, que servia per iterar els elements d'una col·lecció. Els iteradors produeixen els elements l'un darrere l'altre; són cridats per *bucles for*, i el bucle executa una iteració per a cada element. Tant els procediments com els iteradors podien ser polimòrfics i les operacions d'un tipus de dades podien ser procediments o bé iteradors.
- Excepcions. Els procediments i els iteradors podien finalitzar o bé normalment o bé amb una excepció. El CLU admetia les excepcions perquè vaig pensar que era important per a un programa ben dissenyat. Un procediment s'havia de definir per funcionar adequadament amb qualsevol argument ben creat, és a dir, la seva precondició havia de ser *cert* o tan semblant com tingués sentit, ja que això reduïa errors de programa. Tanmateix, podia ser que un procediment d'aquest tipus no actués de la mateixa manera amb tots els arguments bens creats, i era important que els casos inusuals captessin l'atenció del codi de crida d'una manera que no fos ignorable –també, per reduir errors.

Fins a finals dels setanta el meu grup es va dedicar al disseny del CLU. Va ser durant tant de temps perquè aleshores els mecanismes que vam definir eren nous. Però avui són coneguts perquè s'han traslladat als llenguatges de programació moderns, sovint en una forma molt semblant a com estaven definits en el CLU.

Quan es va enllestir el disseny del CLU, vaig començar a treballar en sistemes distribuïts, tot i que encara estava interessada en la metodologia de programació. Però no prestava gaire atenció al que havia passat amb les meves idees.

Això va canviar quan vaig rebre el Premi Turing el 2009. Llavors, vaig prendre consciència que les meves idees sobre abstracció de dades i modularitat, així com idees semblants desenvolupades per altres, s'havien difós tant que la modularitat basada en l'abstracció és la base dels programes que es dissenyen i s'implementen actualment. Avui en el meu discurs he explicat alguns antecedents del desenvolupament d'aquestes idees.

Referències

[1] DIJKSTRA, E. "Go to statement considered harmful". *Communications of the ACM*, 11 (març de 1968), 147–148.

[2] LISKOV, B. "A design methodology for reliable software systems". *Fall Joint Computer Conference*, 41, Part 1 (desembre de 1972), vol. 41, p. 191–199.

[3] LISKOV, B. "The design of the Venus operating system". *Communications of the ACM*, 15 (març de 1972).

[4] LISKOV, B.; SNYDER, A.; ATKINSON, R.; SCHAFFERT, C. "Abstraction mechanisms in CLU". *Communications of the ACM*, 20 (agost de 1976), 564–576.

[5] LISKOV, B.; ZILLES, S. "Programming with abstract data types". ACM Conference on Very High Level Languages (abril de 1974), vol. 9, *SIGPLAN Notices*, p. 50–59.

[6] PARNAS, D. L. "Information distribution aspects of design methodology". *IFIP Congress* (1972), vol. 71, North-Holland Publishing Company, p. 339–344.

[7] PARNAS, D. L. "On the criteria to be used in decomposing systems into modules". *Communications of the ACM*, 15 (desembre de 1972), 1053–1058.

[8] ZILLES, S. "Procedural abstraction: a linguistic protection mechanism". *SIGPLAN Notices*, 8 (setembre de 1973), 140–146.

Paraules del senyor Antoni Giró Roca Rector magnífic de la **Universitat Politècnica de Catalunya · BarcelonaTech**

Inauguració curs acadèmic 2012-2013

Director general d'Universitats,
President del Consell Social,
Vicerectora de Recerca,
Secretària general,
Rectors i presidents de Consell Social que ens heu precedit,
Autoritats,
Representants d'empreses i institucions,
Membres de la comunitat universitària,
Professora Barbara Liskov,
Amigues i amics tots,

Siguin les meves primeres paraules de felicitació i de benvinguda a la nova doctora *honoris causa* per la UPC, la professora Barbara Liskov.

Aquestes paraules han de ser de felicitació pel contingut del seu discurs d'acceptació de la distinció i alhora excel·lent lliçó inaugural del curs acadèmic de la nostra universitat corresponent al curs 2012-2013.

Professora Liskov moltes gràcies!

I paraules també de benvinguda, atès que malgrat que entre la professora Liskov i la nostra universitat no s'havia establert fins ara cap col·laboració física, no és menys cert que el treball desplegat per la professora Liskov en el bast camp dels llenguatges de programació i del disseny de programes ha tingut una influència notable en el treball desenvolupat a la nostra universitat.

Professora Liskov, sigueu també molt benvinguda a la nostra comunitat!

Com apuntava fa un moment, la influència de la professora Liskov s'ha fet notar força en el treball relacionat amb l'especificació algebraica desenvolupat des del Departament de Programació de la Facultat d'Informàtica de Barcelona (FIB), el qual molt probablement constitueix la línia de recerca que ha tingut més visibilitat internacional, línia que parteix del concepte d'abstracció de dades, que és precisament una de les aportacions més sobresortints que la professora Liskov ha fet a l'àmbit de la programació i que ha servit de base d'algunes de les aportacions metodològiques que sobre el disseny de programes s'han fet des del Departament de Programació de la FIB.

I encara voldria posar l'accent en una altra influència de la professora Liskov sobre la nostra universitat, ja que la seva recerca no s'ha limitat només a orientar línies d'investigació pròpies, sinó que ha servit de base en el procés d'ensenyament de la programació emprat per la Facultat d'Informàtica de Barcelona.

La distinció que acabem d'atorgar a la professora Liskov arriba en el context de l'any Turing que estem celebrant i que commemora el primer centenari del naixement d'Alan Mathison Turing, l'eminent matemàtic britànic que va destacar en els camps de la informàtica teòrica, la criptoanàlisi o la intel·ligència artificial, i que és considerat el pare de la informàtica moderna. Aquest és el motiu pel qual Alan Turing és la persona que dóna nom al prestigiós premi anual que atorga l'Association for Computer Machinery (ACM), el Premi Nobel de la informàtica. La professora Liskov va merèixer aquesta distinció l'any 2008.

Professora Liskov: és un honor, un gran honor per a la nostra comunitat universitària, poder-vos incorporar al Quadre d'Honor de la UPC. Sapigueu, a més, que sou la primera dona científica que rep el doctorat *honoris causa* de la nostra universitat.

No voldria acabar aquesta primera part d'aquest discurs sense expressar el meu reconeixement i més sincer agraïment a la Facultat d'Informàtica de Barcelona per l'encert en promoure la concessió d'aquest doctorat *honoris causa* en la persona de la professora Barbara Liskov, agraïment que he de fer extensiu a les escoles Tècnica Superior d'Enginyeria de Telecomunicació de Barcelona, Tècnica Superior d'Enginyeries Industrial i Aeronàutica de Terrassa i d'Enginyeria de Terrassa, i a la Facultat de Matemàtiques i Estadística, així com als departaments d'Arquitectura de Computadors, Llenguatges i Sistemes Informàtics, Física i Enginyeria Nuclear, Enginyeria de Serveis i Sistemes d'Informació i Estadística i Investigació Operativa, que es van sumar immediatament a la petició, així com al Consell de Govern, que va acollir-la i aprovar-la per unanimitat. Gràcies de nou a tothom!

Fa uns moments, com és habitual en l'acte d'obertura dels nostres cursos acadèmics, fèiem el reconeixement públic dels Premis Extraordinaris de Doctorat del darrer curs, així com dels millors treballs de recerca presentats per l'estudiantat de batxillerat i de cicles formatius en ocasió de la convocatòria que cada any fem per fomentar l'interès del jove estudiantat per l'arquitectura, les ciències i les tecnologies sostenibles. Al vestíbul podreu veure els treballs guardonats d'Adrià Nicolàs i de Laia Pérez Moncunill, de Judit Calveras Casanovas, de Yanick Vera Sánchez i d'Enrique Grau Ipar i Ignasi Clavera Gilaberte. Moltes felicitats!

Faig extensiva aquesta felicitació als professors i tutors que, sens dubte, són les persones que us han motivat i ajudat en el vostre treball de recerca.

He d'agrair especialment el patrocini que ens arriba per a aquests premis des de l'Institut Català de les Dones, des del Col·legi d'Enginyers Tècnics Industrials de Barcelona i des de la Càtedra UPC-Endesa Red de Valors Humans a l'Enginyeria Victoriano Muñoz Oms. Sense la col·laboració desinteressada d'aquests ens, la convocatòria dels premis no seria possible.

En aquests moments en què el context en el qual ens movem no ens posa les coses gens fàcils i els reptes i les incerteses es multipliquen dia rere dia, aquest senzill i, des del meu punt de

vista, transcendent acte de reconeixement que la Universitat fa cada any als millors doctorats i doctorades de la UPC i als millors treballs de recerca de secundària, és un exemple excel·lent dels valors que haurien de ser dominants en la societat i una mostra evident del capital humà que tenim la responsabilitat de formar. Valors i capital humà constitueixen una base de present i de futur que no hem ni podem deixar malbaratar...

M'haureu de permetre encara que tingui unes paraules per a les persones de la nostra comunitat universitària que han estat distingides amb la medalla de la UPC, persones que amb el seu treball i esforç personal han contribuït a potenciar i prestigiar la UPC:

- **Agustí Pérez Foguet**, que des del març de 2010 va estar al capdavant del Comissionat per a la Sostenibilitat i la Responsabilitat Social. De la seva tasca, cal destacar-ne especialment el lideratge del procés que va culminar amb la constitució de l'Institut Universitari de Recerca en Ciència i Tecnologies de la Sostenibilitat (IS.UPC).
- **Marisol Marqués Calvo**, que els darrers sis anys ha desenvolupat amb molt encert les seves responsabilitats com a vicerectora de Relacions Institucionals. De la feina feta sobresurt, com a més recent, el seu lideratge en el procés de creació i de posada en marxa del programa de fidelització Alumni UPC.
- I **Enrique García-Berro Montilla**, que durant els darrers anys ha estat secretari general (uns mesos), comissionat de Planificació i Avaluació i vicerector de Personal Acadèmic. De l'Enrique sobresurt la seva versatilitat i esperit de servei, així com la capacitat de diàleg i de negociació en un període de fortes restriccions pressupostàries.

No voldria tampoc en aquests moments deixar de tenir un record molt especial vers les persones de la nostra comunitat que al llarg del darrer curs ens han deixat definitivament. Elles i ells formen part per sempre més de la història de la nostra universitat.

Sabem prou bé que els vents que bufen no són els millors, ni ens són gens favorables per al conjunt de les universitats del país. Cal deixar-ho ben clar! Però precisament perquè no ens són favorables vull posar l'accent en el fet que la UPC –com a universitat de recerca i de docència que és– continua essent una universitat reconeguda arreu. Deixeu-me que en posi alguns exemples:

- La nostra universitat és capdavantera quant a nombre de patents, amb 212 de concedides en el període 2002-2010, segons que es recull en l'indicador de transferència de tecnologia que figura en l'informe corresponent elaborat per l'Observatori IUNE.
- La UPC és capdavantera en la creació d'empreses de base tecnològica amb 247 empreses, de les quals 19 són participades per la UPC.
- També ho és en transferència de resultats de la recerca, amb un volum de contractació de projectes d'R+D+I l'any 2011 de 78,2 milions d'euros.
- La UPC ha intensificat la seva projecció internacional gràcies, d'una banda, a la nostra participació en el consorci sinoespanyol (amb la Universitat Politècnica de Madrid i la xinesa de Tongji), que s'ha plasmat en la creació del primer campus universitari a la Xina en el qual són presents universitats espanyoles, el Sino-Spanish Campus; de l'altra, a través del Programa UPC Abroad, que facilita que la nostra universitat sigui present avui en 130 països.

També vull significar el fet que per primera vegada la UPC ha aparegut, enguany, en el prestigiós Rànquing de Xangai, en el qual ocupem una posició situada entre els llocs 151 i 200 en els àmbits de les enginyeries, les tecnologies i les ciències de la computació, i també en el de les ciències de la computació i de les matemàtiques.

I, ja que parlo de rànquings, afegeixo que el rànquing que elabora la publicació britànica *Times Higher Education* (THE), situa la UPC en el lloc 86 entre les cent millors universitats del món

amb menys de 50 anys d'antiguitat. I encara, en els Rankings I-UGR de Universidades Españolas según Campos y Disciplinas Científicas, la UPC ocupa el primer lloc en dos camps (enginyeria i matemàtiques) i en sis disciplines (arquitectura, automàtica i robòtica enginyeria civil, enginyeria elèctrica i electrònica, telecomunicacions i enginyeria industrial).

També en els QS World University Rankings la UPC ha pujat posicions, en la classificació general (de la 373 a la 350) i en la d'enginyeria i tecnologia (del 87 al 70), mentre que es manté entre les 100 primeres institucions a escala mundial en el Rànquing Webometrics, tot i que aquest rànquing ha introduït canvis substancials pel que fa a la metodologia que no afavoreixen les universitats politècniques.

I encara, en un altre ordre de coses, vull fer ressaltar tres èxits més, en aquest cas assolits pels nostres estudiants i estudiantes:

- El primer, de l'equip d'estudiantat de l'Escola Tècnica Superior d'Enginyeria Industrial de Barcelona, que ha dissenyat i construït només en un any el cotxe elèctric amb el qual competeixen en la prestigiosa Formula Student, i que ha aconseguit situar-se, a Itàlia, entre els deu millors cotxes elèctrics monoplaça d'Europa.
- El segon, de l'equip **(e)co** de l'Escola Tècnica Superior d'Arquitectura del Vallès, com a únic grup català que aquests dies a Madrid està competint en el concurs Solar Decathlon Europe 2012, la competició internacional més important en matèria de cases solars i sostenibles.
- I el tercer, el resultat obtingut per tres estudiants de la UPC a l'Olimpíada Internacional de Matemàtiques celebrada a la ciutat búlgara de Blagoevgrad, on Ander Lamaison ha assolit una medalla d'or, i Arnau Messegué i l'Ivan Geffner, sengles medalles de plata.

Però de tots aquests indicadors, fets i circumstàncies, i per una qüestió estrictament mediàtica, el que darrerament ens ha proporcionat una rellevància i una projecció més grans és la nostra participació en la tecnologia desenvolupada a través de l'Escola Tècnica Superior d'Enginyeria de Telecomunicació de Barcelona, que equipa un sensor del vehicle tot terreny *Curiosity*, que forma part de la missió Mars Science Laboratory de la NASA al planeta Mart.

No només en l'àmbit de la recerca i de la transferència de resultats obtenim bons resultats i posicionament. També els aconseguim en l'àmbit de la docència, com ho posa en relleu el fet que l'estudiant que enguany ha obtingut la nota d'accés més alta de tot el sistema universitari català hagi optat per cursar la doble titulació del grau en Enginyeria Física i grau en Matemàtiques, al Centre de Formació Interdisciplinària Superior (CFIS), després d'haver superat les proves d'accés corresponents.

A través de les referències que acabo de fer, observem que la UPC continua sent un referent en l'activitat i en la qualitat de recerca i docència, tant entre el conjunt de les universitats catalanes com entre les de la resta de l'Estat. Aquesta és una prova més que des de les universitats públiques, des dels nostres centres i instituts de recerca, on es desenvolupa una gran part de l'activitat de recerca al servei de la societat i del país, es treballa de manera eficient al servei de la societat, una societat que de nosaltres també espera que siguem motor de canvi econòmic i social, i que actuem com a agents necessaris que som per a la sortida de la crisi.

La Universitat ha estat i continua essent punt de trobada, de debat i de reflexió. I des d'aquest rol d'agent social és que reivindiquem prendre part activa en la cerca de les solucions que ens han de permetre trobar sortides de la crisi. La universitat pública no forma part dels problemes que es deriven de la crisi, sinó de les solucions, pel fet que correspon a la universitat, d'una banda, formar les persones que hauran de liderar el progrés del país d'aquí uns anys, i, de l'altra, aprofundir en la transmissió del coneixement i dels resultats de la recerca a la societat en general i molt particularment a les empreses, que són un altre dels motors necessaris per al creixement econòmic i per al progrés social.

No em cansaré d'insistir-hi. La universitat pública en general i la nostra en particular no han arribat fins on ara som, ni hem obtingut els resultats que presentem, per atzar o per casualitat. Hi hem arribat per l'esforç i la dedicació personal i col·lectiva de moltes persones. Persones anònimes la gran majoria. I és que darrere l'activitat de la nostra universitat hi ha sempre persones i equips que amb el seu treball constant, eficient i eficaç possibiliten l'obtenció dels resultats que amb orgull podem presentar davant la societat.

Són persones que més enllà de la conjuntura de cada moment, reflexionen i treballen amb un objectiu comú: que el projecte que la UPC defensa progressi i que a través seu contribueixi al reforçament del sistema universitari català.

Assistim els darrers anys a una dràstica reducció dels ingressos públics, sense que l'Estat adopti mesures compensatòries que evitin que els perjudicats per aquesta disminució d'ingressos acabin essent sempre els mateixos, en forma de menys prestació de serveis en les àrees que més tenen a veure amb l'essència mateixa de l'estat del benestar i de la cohesió social.

Els serveis públics són molt afectats pels efectes d'aquesta disminució d'ingressos, i entre aquests serveis públics i d'una ma-

nera molt especial, els que tenen a veure amb la salut i amb l'educació, i per extensió amb la formació universitària.

Des de les universitats públiques hem de recordar que a hores d'ara ens veiem obligades a treballar en unes condicions que considerem injustes i que afecten tots i cadascun dels seus col·lectius. D'una banda, el PAS, que, com la resta de personal de l'Administració pública, a més de veure com es redueixen les seves retribucions, pateix canvis significatius en les condicions laborals fins ara vigents. De l'altra, els darrers decrets del govern afegeixen incerteses que planen sobre el PDI i en condicionen la motivació. Així, professorat en formació, després d'un intens treball durant 8 anys, veu ara com de sobte els són modificades les regles de joc i deixen de convocar-se noves places.

Quant a l'estudiantat, s'ha trobat amb un notable increment dels preus de matrícula per cursar estudis oficials de grau i màster. En aquest cas, però, és oportú destacar l'esforç de la Generalitat perquè el procés fos al més equitatiu possible i que, a hores d'ara, estem pendents de concloure una anàlisi amb la finalitat de millorar l'eficiència de les beques d'equitat. Pel que fa a l'augment del preu de les matrícules, se'ns va assegurar que aquest fet no significava un canvi de model i que aquesta mesura no representava un punt de partida, sinó un punt d'arribada. Davant aquesta realitat continuo afirmant que és del tot inacceptable que el Ministeri endurís les condicions dels becaris i becàries sense que ni tan sols acceptés l'ajornament de l'entrada en vigència d'aquestes noves condicions acadèmiques com a mínim un any.

Per no allargar-me, no repetiré arguments que he exposat a bastament en altres ocasions. Però, no obstant això, vull insistir en un concepte central i cabdal: recerca i educació són certament apostes costoses des d'un punt de vista estrictament econòmic, però no és menys cert que en cap cas es tracta de despeses oneroses des d'un punt de vista democràtic i sobretot de progrés social, ja que formació i recerca formen part del capítol de les

inversions estratègiques de futur que un país i una societat han de preservar per garantir el seu futur. En unes altres paraules: recerca i formació es troben en el centre mateix del ser o no ser com a universitat i com a societat avançada.

Contra el que hauria de ser, la universitat pública catalana en general, i la nostra en particular, cada any disposem paradoxalment d'un finançament inferior, i malgrat aquestes dificultats hem pogut evitar que la universitat deixés d'avançar. Els sacrificis que hi ha al darrere perquè això fos així són més que notables, i els costos no precisament econòmics, incalculables.

En qualsevol cas hem arribat ara a un punt en què des de les universitats, des de la UPC, no podem fer res més nosaltres mateixos per continuar garantint la qualitat de la recerca i la docència que la societat ens reclama.

Durant el darrer curs, la UPC s'ha adaptat a la nova situació que la dinàmica de la crisi ha comportat. Ho ha fet –ho està fent, repeteixo– a canvi d'un notable exercici de responsabilitat i de sacrifici dels col·lectius que constitueixen la comunitat universitària. Ens hem compromès a gestionar els recursos dels quals disposem d'una manera més eficient. El mapa territorial de la UPC en el qual estem treballant ens ha de permetre ordenar

centres i titulacions i potenciar recursos. I ens hem compromès a reduir el dèficit que arrosseguem, un dèficit una part important del qual no és atribuïble a la UPC, sinó a l'incompliment dels compromisos contrets per les administracions i a les retallades sobtades més recents i sense avís previ. No obstant això, previsiblement acabarem l'any sense que augmentem el dèficit en relació amb l'exercici anterior.

Sabem que la millor manera de lluitar contra el dèficit és cercant noves fonts de finançament fonamentades en la valorització de la recerca i de la interacció de la universitat amb la societat i fent-nos més competitius. I aquesta és la nostra responsabilitat i en aquest procés estem obtenint bons resultats, tal com podeu comprovar als indicadors que conté el resum del curs que us ha estat lliurat.

Davant tot això, algú podria preguntar-nos: "I vostès, estan en crisi?" I la resposta inequívoca seria: "Sí, hi estem, en una crisi molt profunda, però tenim una gran potencialitat i disposem d'oportunitats per sortir-ne, que hem d'aprofitar!" Perquè això sigui possible cal, a més del que he apuntat fins ara, que sense cap dilació es clarifiquin les coses que permetin encarar les reformes en el sistema universitari català que calgui. Cal disposar d'un sistema de finançament públic clar, estable i transparent que tingui com a característica bàsica la no penalització, com passa ara, de les universitats que més bons resultats obtenen, un sistema de finançament que tingui en compte que la iniciativa d'aquestes reformes correspon, òbviament, al Govern de la Generalitat de Catalunya, que ha de mantenir un diàleg franc i obert amb les universitats. Compartim els criteris del Govern que cal racionalitzar el mapa de titulacions. Els compartim i els apliquem, com ho posa en relleu que hàgim començat la dinàmica que ens ha de conduir a agrupar titulacions. En cap cas, però, seria bo que a les universitats se'ns acabessin imposant mesures en les quals prèviament no s'hagués pensat ni reflexionat d'una manera conjunta.

La nostra universitat assumirà les responsabilitats que li pertoquen. Que ningú no ho dubti. Ho està fent i ho continuarà fent. I això malgrat les dificultats internes que comporta haver de reduir, per força, una part de l'activitat que portàvem a terme. Som conscients que per millorar hem d'incidir en l'estructura, en la forma de gestió. També en la governança. Incidir en l'estructura vol dir que hem d'acabar amb situacions que res o ben poc ens aporten. Hem de propiciar un canvi a partir del debat intel·ligent al si de la comunitat universitària. I això no és fàcil si no trobem incentius que afavoreixin aquest canvi i transformació, uns incentius que necessàriament han de ser, de moment, més intel·lectuals que no pas econòmics.

Declaro solemnement inaugurat el curs acadèmic 2012-2013 de la **Universitat Politècnica de Catalunya** • Barcelona**Tech**.

Order of Investiture Act

Welcome from the rector of the **Universitat Politècnica de Catalunya** · Barcelona**Tech**.

Reading of the Governing Council's agreement by the general secretary.

La calma del mar (arranged by Enric Ribó).

Oration by the sponsor, Professor Núria Castells.

Conferral of the honorary doctorate on Professor Barbara Liskov by the **Universitat Politècnica de Catalunya** · Barcelona**Tech**.

Honorary Degree Speech by Professor Barbara Liskov.

Canticorum Iubilo (Georg Friedrich Händel).

Rector's speech.

Gaudeamus Igitur (universitary anthem, arranged by Cornel Arany).

The music will be performed by the School of Architecture Choir and the UPC Orchestra and conducted by Lluís Carné i Miguélez.

Oration for Professor Barbara Liskov

by Professor Nuria Castells
***Universitat Politècnica de Catalunya** · BarcelonaTech*

Distinguished rector of the **Universitat Politècnica de Catalunya** • BarcelonaTech, distinguished members of the University Senate and Board of Trustees, authorities and representatives of institutions and companies, lecturers, students, administrative staff, family and friends, and dear Dr. Barbara Liskov.

I am very pleased to be here today at this solemn ceremony to welcome to the University Senate the recipient of an honorary doctorate from the Universitat Politècnica de Catalunya that is pursuant to the Governing Council agreement of 12 June 2012 and sponsored by the Barcelona School of Informatics, of which I am proud to be the dean. Professor Liskov's nomination was supported by UPC departments and schools, important companies in the IT sector, and other organizations, attesting to the prominence and high repute of the person receiving this distinction.

Professor Barbara Liskov is an institute professor (the highest title) of computer science and head of the Programming Methodology Group at the Massachusetts Institute of Technology in Boston. Her research interests include distributed systems, replication algorithms for fault tolerance, programming methodology, and programming languages. Her current research projects include Byzantine-fault-tolerant storage systems and online storage systems that provide confidentiality and integrity for the stored information.

Barbara Liskov's research has led to major breakthroughs in such fundamental areas as operating systems, distributed systems, programming languages, and programming methodology. Her ideas helped to form the foundation for modern programming languages such as Java, which are designed to make use of self-contained modules of data and instructions that can be developed once and reused to many different ends.

Today, I will attempt to summarize the extensive and fruitful work carried out by Dr. Barbara Liskov in the course of her 40-year academic and research career.

Information technology is one of the youngest disciplines in engineering. Its contributions over the last 30 years have drastically changed our world. All fields of human activity are modified and transformed by the efficiency and new opportunities that computers offer. One cannot conceive of the level of social progress we know today without the contribution of computer scientists.

The work and intelligence of computer pioneers, among them Barbara Liskov, has laid the foundations of a science that is the driving force of the developments we have experienced in recent decades.

2012 is a special year for the informatics community. It's the centenary of Alan Turing's birth in London. The world is honoring Turing's life and achievements. He was highly influential in the development of computer science and he formalized the concepts of "algorithm" and "computation" with the Turing machine, which can be considered a model for a general purpose computer. Alan Turing is widely considered to be the father of computer science.

2012 is also a special year for the Barcelona School of Informatics (FIB). It is the celebration of its 35th anniversary. Our school has been offering top-quality university education and excellence in computer engineering that responds to the needs of society and the economy since 1977. The FIB is part of the international computer community. The work done by over 250 researchers and the education we have been providing for 35 years have contributed to the progress of our society.

From this standpoint, the scientific community and the UPC recognize the value of the achievements of Dr. Barbara Liskov.

Barbara Liskov, born Barbara Jane Huberman on 7 November 1939 in California, earned her BA in mathematics at the University of California, Berkeley in 1961. Rather than going directly to a graduate school, she took a job at the Mitre Corporation, where she was involved in computer programming.

In the early 1960s, computer programming could be considered an experimental world. The features, performance, cost and reliability of those machines made them accessible only to a few experts. The job at Mitre was the beginning of a prolific career for Barbara Liskov.

After a year at Mitre, she moved to Harvard University to work on a project that sought to automatically translate English sentences into something a computer could understand. Though natural-language processing is a problem that computer scientists are still working on, back then people expected it to be solved in a few years.

Returning to California to do graduate work at Stanford, she was given financial support in John McCarthy's lab partly because her earlier work on natural language translation was in the general area of artificial intelligence. In 1968, she became one of the first women in the United States to be awarded a PhD in computer science. Her thesis on chess end-games was supervised by John McCarthy.

After receiving her PhD, Barbara married Nathan Liskov and moved back to the Boston area to work at the Mitre Corporation in Bedford, MA on computer design and operating systems. Using an Interdata 3 computer that had the ability to change the instruction set via microcode, she created the Venus Computer, which was tailored to support the construction of complex software. The Venus operating system was a small timesharing system for the Venus machine that was used to experiment with how different architectures helped or hindered this process. The Venus system supported 16 teletypes and each user was connected to a virtual machine so that major errors would not compromise the entire system, just the virtual machine for that user.

In 1971, shortly after finishing her experiments with Venus and presenting a conference paper on the topic, Liskov was urged by another attendee to consider a position at MIT. She left Mitre and joined the MIT faculty as a professor in the Laboratory for Computer Science. Building on her experience at Mitre, her research has focused on creating more reliable computer systems.

At MIT, she led the design and implementation of the CLU programming language, which emphasized the notions of modular programming, data abstraction, and polymorphism. It was the first language to support data abstraction. These concepts are a foundation of the object-oriented programming used in modern computer languages such as Java and C#, although many other features of modern object-oriented programming are missing from this early language.

Any sophisticated software application is a complex structure of interlocking parts, often modified over time by a large team of engineers. Any change can have unintended effects on other parts of the software, requiring programmers to essentially rewrite the program. Barbara Liskov came up with ways to structure programs in discrete chunks, or "multi-operation modules," so that changes would be less likely to affect code outside certain boundaries.

Because it was hard to illustrate her ideas to programmers, she designed a programming language that put them directly into practice. “I had a very strong idea about what were good programs and what were bad programs,” she says. “I wanted to make it easy for people to write good programs, and while you can't prevent people from writing bad programs, I didn't want to make it too easy for them.”

Her MIT group also created the Argus language, the first high-level language to support implementation of distributed programs over a network, which extended the ideas of CLU, including support for nested transactions. An example of such a distributed program might be a network-based banking system. Argus provided object abstractions called “guardians” that encapsulate related procedures. As an experimental language, Ar-

gus influenced others developers but was never widely adopted or used for deployed networked applications.

She is also at the origin of the Thor object-oriented database system, which provides transactional access to persistent, highly-available objects in wide-scale distributed environments.

Liskov's subsequent work has mainly been in the area of distributed systems, which use several computers connected by a network. Her research has covered many aspects of operating systems and computation, including important work on object-oriented database systems, garbage collection, caching, persistence, recovery, fault tolerance, security, decentralized information flow, modular upgrading of distributed systems, geographic routing, and practical Byzantine fault tolerance. Many of these, like Byzantine fault tolerance, deal with situations where a complex system fails in arbitrary ways. Liskov developed methods to allow correct operation even when some components are unreliable.

With Jeannette Wing, she developed a new notion of subtyping, known as the Liskov substitution principle. Her contributions have influenced advanced system developments and set a standard for clarity and usefulness.

Liskov is currently the Ford Professor of Engineering at MIT. She leads the Programming Methodology Group at MIT, with a current research focus on Byzantine fault tolerance and distributed computing. She became a full professor at MIT in 1980. She served as the associate head of Computer Science from 2001 to 2004, and in 2007 was appointed associate provost for faculty equity. In 2008, MIT named her an institute professor, the highest honor awarded to a MIT faculty member.

MIT president Susan Hockfield said, "Barbara is revered in the MIT community for her role as scholar, mentor and leader. Her pioneering research has made her one of the world's leading authorities on computer language and system design. In addition to her seminal scholarly contributions, Barbara has served MIT with great wisdom and judgment in several administrative roles, most recently as associate provost for faculty equity."

She has written three science books and more than 150 scientific articles published in leading international journal and conferences. She has supervised the research programs of more than twenty-five PhD students and large numbers of MSc students.

Her work had a significant impact on the research carried out at the FIB's Department of Programming, which is at present part of the UPC's Department of Software. The work done on algebraic specification was based on Barbara Liskov's concept of data abstraction. The methodological contributions on program design were also based on Liskov's work. Moreover, her proposals have been used for many years as the basis for teaching programming at the FIB.

Just to mention some honors and awards: Barbara Liskov is a member of the National Academy of Engineering and the National Academy of Sciences, and a fellow of the American Academy of Arts and Sciences, and of the Association for Computer Machinery. She received the Society of Women Engineers' Lifetime Achievement Award in 1996, the IEEE von Neumann medal in 2004, the ACM SIGPLAN Programming Language Achievement Award in 2008, and the ACM's A.M. Turing Award in 2009.

She has received several honorary doctorates: from ETH Zurich, Switzerland (2005), Brown University (2010), Northwestern University, USA (2011), and the University of Lugano, Switzerland (2011).

In 2003, she was named one of the 50 most important women in science by Discover Magazine. In 2012, she was inducted into the National Inventors Hall of Fame.

Talking about non-scientific aspects, Barbara Liskov has always encouraged and helped female students and in recent years has devoted considerable attention to making computer science a more inclusive field.

Being a woman in the early days of computer science would have been difficult had she paid attention to such obstacles, Liskov says. And even though those barriers didn't much bother her, societal expectations prevented her from acknowledging the importance of her career until her own research interests began to mature.

As associate provost for faculty equity at MIT, she works to recruit more women and minority faculty members and to help them to manage and advance their careers.

In addition, Barbara Liskov made it a priority to cultivate a rich set of interests outside the workplace, including gardening and reading mystery novels. She is a very kind and friendly person.

To conclude: the FIB started its academic life in September 1977, so we are celebrating our 35th anniversary. We are also celebrating the centenary of Alan Turing's birth. The best way to give academic relevance to those facts is to give the UPC's highest honor to a pioneering woman in informatics. Congratulations, Barbara!

Honorary Degree Speech by Professor Barbara Liskov

Programming the Turing Machine

Barbara Liskov, Massachusetts Institute of Technology Computer Science and Artificial Intelligence Laboratory.
Setember 28, 2012

This is the centenary of the birth of Alan Turing. Turing helped to define the theoretical foundations of computer science with his definition of a stored program computer. The model he provided underlies the way we reason about programs today.

However, getting computers to do useful things is difficult. Software systems are huge – millions of lines of code. Also they do complicated things. Turing's model does not provide any insight into how to design these systems.

Inventing techniques to address this complexity gap has been a major theme in computer science research. One area where tremendous progress has been made is in the invention of algorithms, data structures, and protocols. These advances provide computer scientists with a tool box. For example, if you need to do efficient search, it is no longer necessary to invent new approaches; instead you can choose the data structure, such as a hash table or a balanced tree, that is best matched to your problem.

However, advances in algorithms, data structures, and protocols, useful as they are, do not address a fundamental problem: when faced with the need to develop a new software application or system, how does the programmer come up with an effective design? Before the tools can even be used, such a design is needed.

This question has been the focus of research in programming methodology. In addition, as programming methodology research has identified useful concepts, research in programming languages focused on providing language features to support the concepts has followed.

In my talk today I'm going to discuss early research in programming methodology and programming languages.

I started work in this area in about 1970. At that point I had received my PhD, from Stanford, and was working at the Mitre corporation. My doctoral work at Stanford was in artificial intelligence. At Mitre I switched to working in computer systems. I knew part way through my doctoral work that I wanted to make this switch, but I decided to wait to do this until after I had my PhD.

Mitre is a non-profit corporation that does research for the United States government. My first project at Mitre involved designing a new machine instruction set, implementing it using microprogramming, and then showing the usefulness of the instruction set by using it to implement a time-sharing system called Venus **[3]**. When that project was complete I was asked to work on research concerned with solving the "software crisis."

The problem being identified by the phrase "software crises" was that people didn't know how to build software in an effective way so that at the end the software did what it was supposed to do. All through the seventies and eighties there were stories about failed software projects in which millions of dollars and many man-years were spent on a system that in the end had to be abandoned because it didn't work properly.

The US government was concerned about the software crisis since they needed ambitious software systems for military and other purposes and they were having problems getting the needed software.

When I began work on the project I had no background in the area of programming methodology, so I started by reading the

literature. When I received the Turing award in 2009, I went back and re-read this literature. I found these papers to be interesting both in their own right and also from an historical perspective. In my talk today I'll tell you about just a couple of them.

The first is "Go To Statement Considered Harmful [1]," a famous paper by Edsger Dijkstra. In this paper Dijkstra argued that people should not use go to statements in their programs. This may seem an odd thing to write a paper about from today's perspective since our modern programming languages do not have go to statements. However this was not true of programming languages that existed at the time.

Dijkstra's paper was very controversial. Some people thought it was silly: they were accustomed to using go to statements and couldn't imagine that they would go away. Others were insulted. They argued that although they used go to statements in their programs, the programs were nevertheless well structured. And this may have been true because you can write a nicely structured program in any programming language provided you are disciplined in the way you use its features.

Still others were concerned that go to statements were a necessary feature; they thought that without them it wouldn't be possible to write efficient software. This concern needs to be understood in the context of the programming languages that were available at the time. In those days programming languages did not have certain features that we count on today, such as case statements and exceptions. Go to statements can be used to compensate for the absence of such features; for example, passing labels as parameters provides a good way to simulate an exception mechanism.

Dijkstra's paper is interesting for more than just the controversy it aroused; his arguments are also interesting. He argues that go to statements are bad because they make it hard to reason about program correctness. He points out that we reason about program correctness statically by looking at the program text, but we are reasoning about something dynamic, namely the behavior of the program at run time. And he says that since reasoning about program correctness is hard, it is a good idea for the static and dynamic structures to be as close together as possi-ble. The structures will be close if a program is written using well-structured mechanisms, such as if and while statements and function calls. But with go to statements, the two structures can be far apart.

Dijkstra illustrated this point by discussing the difficulty of numbering the statements in a program in a way that would correspond to the execution order: this is hard to do for a program with go to statements. An intuitive way to think about Dijkstra's point is to consider program debugging. If you have discovered an error at a particular point in your program, to figure out what happened you have to understand how you got there; in other words you have to understand all the statements that have been executed in the past, in the order they were executed. This is relatively easy to do if the program uses well-structured control structures, but can be extremely difficult with go to statements.

The second paper is "Information Distribution Aspects of Design Methodology [6]," by David Parnas. This paper is about program structure. In it Parnas says that a program consists of a set of *modules* although there isn't any definition of exactly what a module is. Nevertheless, in order to reason about a program as a collection of modules we need to understand their *connections*, i.e., how they interact. Parnas is interested in what these connections are, and the paper states: "The connections between modules are the assumptions which the modules make about each other." Here Parnas is making the point that it isn't sufficient to define just the "calling sequences and control block formats," i.e., how to call the module, because this often does not capture all the connections. Parnas's paper then goes on to discuss a design methodology that considers the importance of connections; Parnas published another paper on program design at about the same time [7].

At this point I want to discuss modularity. As Parnas pointed out, we want to consider a program to be a collection of modules. Modules are useful for a number of reasons, but most important is that they provide a way of reasoning about what a program does. It's clearly very difficult to reason about the correctness of a million-line program as a single entity. Instead we need to break the program up into pieces and deal with each one separately. These pieces are modules.

For this approach to work, we want the modules to be "black boxes." This means that we can only see their surface, not their interior. The interior of a module is the code that comprises it. The surface is some kind of description of what it does, i.e., a *specification* of its behavior. This specification has to capture all behavior of the module that can be observed by the rest of the program, i.e., all its connections.

If we are successful with this modular decomposition, we will be able to reason about the correctness of a huge program module by module. Here's how this works.

The implementation of a module is correct if it provides the behavior indicated in the specification. To reason about the implementation we examine its code. This code probably uses other modules, but we can reason about these uses by considering the specifications of those other modules: if module A uses module B we can understand what this means by looking at the specification of module B. Importantly we do not need to look at the code of module B. And thus we can reason locally: we only need to look at the code of A, plus the specifications of other modules, but not their code.

An important point here is that being able to ignore module B's implementation greatly reduces the effort. Not only is a specification typically smaller than the code; additionally if we look at the code we are likely to discover that it uses still further modules. However, since we don't see the code, we can ignore the existence of these additional modules. For example, module B's implementation might make use of a module C, but when we are reasoning about the correctness of module A we don't need to consider module C at all.

Modularity, if done properly, has additional benefits. It provides a way to subdivide a program into pieces that can be worked on independently, by different people, without interference. It also provides *modifiability*. If all the code in a system depends only on the specification of some module, we can reimplement that module without having to worry about the impact of this change on the rest of the system. As long as the new implementation meets that module's specification, the rest of the program will be unaffected by the change.

In 1970, however, these benefits of modularity were just being discovered. Furthermore there was only one programming me-

chanism at the time that made sense as a module: the subroutine or procedure. In fact, a procedure is a good example of what we might get from modularity. For example, consider a procedure that sorts an array. To use this procedure you only need to know that when it returns the array will be sorted – this is what its specification says. You are not concerned with the sorting technique being used in the the procedure's implementation, and the using code will continue to run correctly if the procedure is reimplemented to use a different technique.

The problem is that procedures aren't powerful enough to be satisfactory for all kinds of modules we need in programs. For example, a file system is clearly a module, but equally clearly it is not a procedure. So one problem at the time was that we lacked mechanisms that matched the ideas about modularity.

An additional problem was that existing programs often did not have a modular structure. For example, at one point I was responsible for maintaining a large program written in assembler (the fact that it was written in assembler does not matter). The program made an occasional procedure call, but mostly it was just straight line code, and modular reasoning could not be used for it to any large extent. And this was typical at the time.

A final point is that even when there were modules they often had large "connections." For example they often had complex call sequences (as indicated by Parnas in his paper). Even more serious was that they had additional connections through their use of global variables. In fact, communication through global variables was in some cases a preferred way to organize the code!

As I was reading the papers described above and a number of others, I realized that I had invented a way of modularizing programs in my earlier work at Mitre. When I was working on the design and implementation of the Venus operating system, I was concerned about whether my small team of programmers could produce what was a very ambitious software system in an effective manner. I understood that in order to accomplish this we needed a way to organize our code that kept its complexity under control so that we could reason about its correctness. So I developed some rules about how we would do this. I decided that rather than organizing the system around global variables that all the code could access, we would instead organize the code into what I came to call partitions. There were two rules about partitions:

- Each partition owned a subset of the global variables. Only code in that partition could access (read or modify) those global variables.
- Each partition provided one or more operations. These operations could be used by other partitions to interact with it. Thus other partitions could affect the global variables inside the used partition, but not directly; instead this could be done only by calling that partition's operations.

I wrote up my ideas about partitions in "A Design Methodology for Reliable Software Systems [2]," published in 1972.

Also in September of 1972 I moved from Mitre to MIT.

During my first semester at MIT I was consumed with the following question: how could these ideas about modularity be made more understandable so that others could apply them in their own system designs. I felt that all of these papers contained good ideas. Also, each paper explained how the ideas could be used in program design; sometimes they even gave examples of a design consisting of a number of modules. The ideas and the designs were compelling; the reader would think that this was clearly the right way to go about things. But when the reader was faced with coming up with a design of his or her own, it wasn't clear how the ideas could be applied to it.

I think the reason this was so difficult was that nothing was well-defined. Instead the papers presented sketches of what ought to be done, but the ideas were quite vague. And in particular there was no definition of a program structure that made sense for defining modules.

So I was trying to think of how things could be made better. And at some point during the fall or winter of 1972, I saw that my idea of partitions could be related to data types. The state that was hidden inside a partition corresponded to the representation of a data object, and the operations the partition made available to the rest of the system corresponded to the operations that could be used on the data object.

I knew right away that this idea was going to be important because these *abstract data types* were an extension of something programmers understood, namely the data types already in programming languages. Programmers were accustomed to using arrays by calling array operations such as fetch and store. Also they knew that the representation of the array in storage was not accessible to their code, nor did they need to be concerned about it. So the main new thing was that the available set of data types would no longer be limited to what the programming language provided. Instead programmers would be able to define new ones. Yet by doing so they would be defining more powerful modules than had been available previously.

The next step in developing this idea was to figure out exactly what it was. I decided to do this by thinking about programming language mechanisms to support it. I started working with Steve Zilles on this project in the spring of 1973. Steve was a graduate student at MIT and also an employee of IBM, and he had developed similar ideas on his own **[8]**, so we decided to work together.

Steve and I worked over the spring and summer of 1973 and by the end of the summer we had figured out much of what was needed in a programming language that supported abstract data types. We wrote up our ideas that summer; our paper

"Programming with Abstract Data Types [5]" was published the following year.

This paper defined an abstract data type as ajj set of objects. and a set of operations. Plus we had a rule: the operations provided the only way to use the objects.

The paper also provided a sketch of what kind of language mechanisms would be needed to allow abstract data types to be defined and used.

The next step began in the fall of 1973. At that point three first year graduate students joined my group: Russ Atkinson, Craig Schaffert, and Alan Snyder. These three students, along with myself, became the designers of a new programming language that supported abstract data types. Steve Zilles was a participant in our design meetings, but he was focused on his PhD thesis, which was about algebraic specifications of abstract data types. Although designing a programming language was the obvious next step, I also had the following rationale for taking this step:

- Communication. I believe that programmers understand programming through programming language features. Therefore by designing and implementing a programming language, I would enable the use of abstract data types in a way that would not be possible in the absence of language support.
- Practicality. At the time I did this work it was unclear whether abstract data types would be sufficient in practice for building large programs. By providing a programming language, I could enable experiments that would help to answer this question. Users would have to write code using just the programming language features; they could not cheat by using lower level features. So if our mechanisms weren't adequate, we would find this out.
- Precision. I also felt that it was important to get a precise definition of an abstract data type. A programming language is a mathematical object, so working on this definition as part of a programming language would be helpful.
- Performance. It isn't enough that you can write the code for your application – it's also important that the application perform well enough to satisfy its users. Again being able to experiment with programs written in the language allowed us to see how this worked out.

The language we designed was called CLU. The design process was focused on data abstraction: how to allow abstract data types to be defined and used, and what additional language features were needed to make things convenient. We were careful to limit our work to just this problem and avoided looking at other issues in language design that, while interesting, were not in the main path. An example was an early decision to focus on a sequential programming language and ignore concurrency.

CLU [4] provided the following abstraction mechanisms.

- Clusters. These were used to implement abstract data types (the name "CLU" is the first three letters of "cluster"). The header of a cluster listed the available operations, and this was the only way code outside the cluster could use the objects. Inside the code defined how the objects of that data type were represented and provided implementations of the operations.
- Polymorphism. Many data types make sense for different kinds of elements, e.g., a stack can hold integers, or characters, and so on. It would be annoying if a programmer had to implement a cluster for each type of element that made sense for a stack. Instead CLU supported **polymorphic** definitions, which worked over many types. The hard part of providing this mechanism was that often it was necessary to constrain what parameter types were allowed; for example a sorted-set makes sense only if the parameter type provided an ordering relation. To solve this problem we invented *where clauses* which listed the operations the parameter type had to have. This way we could check at compile time whether the actual parameter was acceptable.
- Procedures and Iterators. CLU allowed procedures to be defined independently of clusters. It also provided a new abs-

traction mechanism called *iterators* that made it convenient to iterate over the elements of a collection. Iterators produce the elements one after another; they are called from **for** loops and the loop runs an iteration for each element. Both procedures and iterators could be polymorphic and the operations of a data type could be either procedures or iterators.

- Exceptions. Procedures and iterators could terminate either normally or with an exception. CLU supported exceptions because I thought this was important for well-designed program. A procedure ought to be defined to work for any legal arguments, i.e., its precondition ought to be "true" or as close to it as made sense, since this reduced program errors. However, such a procedure might not be able to behave the same way for all legal arguments, and it was important to bring the unusual cases to the attention of the calling code in a way that wasn't ignorable –again to reduce errors.

The design of CLU occupied my group until the late 1970s. It took this long because the mechanisms we defined were new at the time. They are familiar today, however, because they have moved into modern programming languages, often in a form quite similar to the way they were defined in CLU.

When the design of CLU was finished, I started working on distributed systems, although I continued to be interested in programming methodology. But I wasn't paying a lot of attention to what had happened to my ideas.

This changed when I received the Turing award in 2009, Then I became aware that my ideas about data abstraction and modularity, and similar ideas developed by others, had spread so that today modularity based on abstraction is the way programs are designed and implemented. In my talk today I have provided some background about how these ideas developed.

References

[1] DIJKSTRA, E. Go to statement considered harmful. *Communications of the ACM* 11 (March 1968), 147148.

[2] LISKOV, B. A design methodology for reliable software systems. In *Fall Joint Computer Conference* 41, Part 1 (December 1972), vol. 41, pp. 191-199.

[3] LISKOV, B. The design of the Venus operating system. *Communications of the ACM* 15 (March 1972).

[4] LISKOV, B., SNYDER, A., ATKINSON, R., AND SCHAFFERT, C. Abstraction mechanisms in CLU. *Communications of the ACM* 20 (August 1976), 564576.

[5] LISKOV, B., AND ZILLES, S. Programming with abstract data types. In ACM Conference on Very High Level Languages (April 1974), vol. 9, *SIGPLAN Notices*, pp. 50-59.

[6] PARNAS, D. L. Information distribution aspects of design methodology. In *IFIP Congress* (1972), vol. 71, North-Holland Publishing Company, pp. 339344.

[7] PARNAS, D. L. On the criteria to be used in decomposing systems into modules. *Communications of the ACM* 15 (December 1972), 1053-1058.

[8] ZILLES, S. Procedural abstraction: a linguistic protection mechanism. *SIGPLAN Notices* 8 (September 1973), 140-146.

Brief Speech by Antoni Giró Roca, Rector of **Universitat Politècnica de Catalunya** · BarcelonaTech

Inauguration of the 2012-2013 academic year

Director general of Universities,
Chair of the Board of Trustees,
Vice-Rector for Research,
General secretary of the Interuniversity Council of Catalonia,
Former rectors and chairs of the Board of Trustees,
Distinguished academics,
Representatives of companies and institutions,
Members of the university community,
ProfessorBarbara Liskov,
Friends one and all,

I would like to start by welcoming and offering my congratulations to the newly invested doctor honoris causaof the UPC, Professor Barbara Liskov.

These opening words have two aims. Firstly, I must thank Professor Liskov for her acceptance speech and for her magnificent inaugural lecture that marks the beginning of the 2012-2013 academic yearat our university.

Professor Liskov, many thanks!

Secondly, I must extend a formal welcome,since although Professor Liskov's work on programming languages and program design has had a considerable impact on the work carried out at our university, this is the first time that she has visited us in person.

Professor Liskov, welcome to our community!

As I remarked, Professor Liskov has had a strong influence on work related to algebraic specification at the Department of Pro-

gramming of the Barcelona School of Informatics (FIB), which is probably the most internationally recognised line of research at our university. This research area is built on the concept of data abstraction, which is one of the most outstanding contributions that Professor Liskov has made to the field of programming and forms the basis of program design methodologies developed at the FIB's Department of Programming.

I would also like to highlight another area in which Professor Liskov's work has influenced our university, since her research has not only served as a guide for our own lines of research but is also at the root of the teaching process for programming courses at the Barcelona School of Informatics.

The distinction that we have today conferred on Professor Liskovcomes against the backdrop of ongoing celebrations ofthe Alan Turing Year, which marks the centenary of the birth of Alan Mathison Turing, the eminent British mathematician who excelled in the fields of theoretical computer science, cryptanalysis and artificial intelligence, and who is widely considered the father of modern computer science.It is for this reason that his name is given to the prestigious annual A.M. Turing Award of the Association for Computer Machinery (ACM), the "Nobel Prize" of the computing world, which Professor Liskovwon in 2008.

Professor Liskov, it is an honour, a great honour for our university community to welcome you onto the Roll of Honorary Doctors of the UPC. You should also know that you are the first female scientist to receive the title of doctor *honoris causa* from our university.

I do not wish to conclude the first part of this address without expressing my acknowledgement and most sincere gratitude to the Barcelona School of Informatics for its wise decision in nominating Professor Barbara Liskov for the *honoris causa* distinction, gratitude that I must extend to the Barcelona School of Telecommunications Engineering, the Terrassa School of Industrial and Aeronautical Engineering, and the School of Mathematics and Statistics, as well as to the departments of Computer Architecture, Software, Physics and Nuclear Engineering, Service and Information System Engineering, and Statistics and Operations Research, which were quick to lend their support to the proposal. Finally, I am grateful to the Governing Council of the UPC, which admitted and approved the nomination by unanimous decision. My thanks again to all of you!

A few moments ago, as is customary during the inauguration ceremony, we publicly congratulated the winners of the special doctoral awards for the previous academic year and the prizewinnersfor the best research assignments by upper secondary students and higher vocational training students, which are organised every year to stimulate interest in sustainable architecture, science and technology among younger learners. In the entrance hall you will find a display of the winning assignments, by Adrià Nicolàs, Laia Pérez Moncunill, Judit Calveras Casanovas, Yanick Vera Sánchez, Enrique Grau Ipar and Ignasi Clavera Gilaberte. Many congratulations!

I also offer my congratulations to your teachers and tutors, who no doubt provided motivation and assistance while you were carrying out your research assignments.

I must offer particular thanks for the sponsorship received from the Catalan Women's Institute, theAssociation of Technical Industrial Engineers of Barcelona, and the Victoriano Muñoz Oms UPC-Endesa Red Chair for Human Values in Engineering. Without their generous contributions, these prizes could simply not be awarded.

At a time when the situation around us complicates our work and the challenges and uncertainties multiply by the day,this simple yet –in my view– significant act of recognition that the University dedicates each year to its best doctoral students and to the authors of the best research assignments in secondary education and vocational training is an excellent example of the values that must be placed at the forefront of society and a showcase of the human capital for whose education we are responsible.Values and human capital are assets on which the present and the future can be built and which we cannot and must not squander...

With your permission, I would also like to dedicate a few words to members of our university community who have been awarded the UPC Medal of Honour, people whose work and effort have contributed to the potential and prestige of the University:

- **Agustí Pérez Foguet**, who in March 2010 became thecommissioner for Sustainability and Social Responsibility. Of his many achievements, I would highlight his leadership of the process leading to the creation of the University Research Institute for Sustainability Science and Technology (IS.UPC).
- **Marisol Marqués Calvo**, who for the last six years has performed her duties as vice-rector for Institutional Relations in exemplary fashion. Notable in her recent work is the leadership of a process to design and implement the UPC Alumni loyalty programme.
- **Enrique García-Berro Montilla**, who has filled a number of important roles in recent years, including those of general secretary (for a few months), head of the Planning and Assessment Committee, and vice-rector for Academic Staff. Enrique is known particularly for his versatility and commitment to service, and for his skilled handling of discussion and negotiations during a period of severe budgetary restrictions.

It would also be remiss of me not to make a special mention of those members of our community who are no longer with us. They will always bepart of the history of our University.

We are well aware that the current climate is far from favourable, for ourselves and for the other universities in our country. This must be made absolutely clear. But it is precisely because conditions are so adverse that I want to underline the fact that the UPC, both as a teaching institution and a research univer-

sity, continues to win widespread recognition. Allow me to give you a few examples:

- The UPC is the leading Spanish university in successful patent applications, having obtained 212 domestic patents over the period 2002–2010, as reflected by the corresponding indicator in the report compiled by the IUNE Observatory.
- The UPCis also the most active creator of technology-based spin-offs. It has overseen the emergence of 247 new companies, 19 of which are still part-funded by the University.
- The UPC is the leading university in the transfer of research results, having generated 78.2 million euros in 2011 through R&D contracts.
- The UPC has stepped up its efforts to internationalise, largely through two activities: the first is our involvement in the Sino-Spanish University Consortium (together with the Technical University of Madrid and Tongji University), which has led to the creation of the first university campus in China on which Spanish universities are represented, the Sino-Spanish Campus; the second is the UPC Abroad programme, thanks to which our university is now present in over 130 countries.

I would also like to highlight the fact that the UPC has this year been included in the prestigious Shanghai Ranking for the first time. We are in the group of universities ranked in positions 151-200 for the field of Engineering, Technology and Computer Science and for the subjects Computer Science and Mathematics.

Since I have brought up the subject of rankings, I should add that the World University Rankings complied by the British publication *Times Higher Education* places the UPC 86th in its list of the top 100 universities under 50 years old. In the I-UGR Rankings of Spanish Universities, the UPC is the highest placed university in two fields (Engineering and Mathematics) and in six scientific disciplines (Architecture, Automatic Control and Robotics, Civil Engineering, Electrical and Electronic Engineering, Telecommunications, and Industrial Engineering).

The UPC has also climbed several positions in the QS World University Rankings, both in the general classification (from 373 to 350) and in the Engineering and Technology subject ranking (from 87 to 70), as well as retaining its place among the top 100 universities worldwide in the Ranking Web of Universities world classification, despite significant changes to the methodology behind the latest edition of this classification that do little to help the cause of technical universities.

At this point I would like to shift the focus to three other notable successes of the past year, achieved, in this case by our students:

- First, a team of students from the Barcelona School of Industrial Engineering designed and constructed in the space of just one year an electric car that they use to compete in the prestigious Formula Student championship, managing to rank among the ten best single-seater racing cars in Europe in their class.
- Second, the **(e)coteam** from the Vallès School of Architecture were the only Catalan representatives to take part in Solar Decathlon Europe 2012 in Madrid – last year's edition of the foremost international competition for research in the development of solar sustainable housing.
- Third, three UPC students achieved excellent results at the International Mathematical Olympiad held in the Bulgarian city of Blagoevgrad: Ander Lamaison won a gold medal, and Arnau Messegué and Ivan Geffner were both silver medal winners.

However, of these many indicators, achievements and distinctions, the work that has won us the greatest plaudits over the last few months –thanks to its coverage in the media– is the development of technology at the Barcelona School of Telecommunications Engineering that forms the basis of a sensor fitted on the *Curiosity*, the all-terrain vehicle used as part of NASA's Mars Science Laboratory mission.

However, we do not only achieve excellent results and ranking positions through research and transfer activities. We are also successful as a teaching institution, which is reflected particularly well in the fact that the student with the highest grade in the whole of Catalonia for last year's public university entrance examinations opted to study at the UPC, gaining admission to the double bachelor's degree in Engineering Physics and Mathematics offered by the Interdisciplinary Higher Education Centre (CFIS) after passing the CFIS admission tests.

The achievements I have mentioned show that the UPC remains a leading centre for teaching and research and for quality across both of these areas,not only among Catalan universities but among its counterparts in the rest of Spain as well. This adds to the existing evidence that public universities –through our research centres and institutes, which are behind much of the research conducted for the benefit of society and for the country as a whole– work efficiently in the service of society, a society that looks to us to drive economic and social change and to live up to our role as key agents in emerging from the financial crisis.

The University has always been and will continue to be a centre for interaction, debate and reflection, and it is from our position as a social agent that we seek to play an active role in finding the solutions that will help us out of the crisis. The public university system is not one of the problems that stem from the economic crisis, it is one of the solutions. Not only are universities responsible for training those individuals who will taking a leading role in the progress of the country in the years to come, they are also charged with transferring knowledge and research results into general society and into the knowledge bases of companies, which are another of the groups needed to drive economic growth and social advancement.

I shall not grow tired of saying it: public universities, in particular our own, have not reached where we are today or obtained the results we can boast of by chance or good fortune. We have

achieved this thanks to the effort and the individual and collective dedication of many people, the vast majority of whom go unnamed. There are always people behind the scenes whose constant, efficient and effective work makes it possible for us to achieve the results we proudly offer society.

These are people who, whatever the circumstances, continue to reflect on and work towards a common goal, ensuring that the project to which the UPC is committed continues to progress, to contribute to the consolidation of the Catalan university system as a whole.

Over the last few years we have witnessed drastic cuts in public spending but no attempts by the State to compensate by introducing measures that would prevent this drop in income affecting the same people every time. By failing to do so, the government has reduced services in those areas that have the greatest bearing on the very essence of the welfare state and social cohesion.

Public services are hit particularly hard by the effects of this dropin funding, especially those related to health and learning and thus, by extension, to university education.

Those of us at public universities must not lose sight of the fact that the current situation obliges us to work in conditions which we consider unjust and that affect each and every one of the groups we represent. On the one handare the administrative and service staff, who, like all other public employees, have not only seen their salaries reduced but have also suffered substantial changes to their working conditions. On the other, the most recent decrees passed by the government have added to the uncertainties hanging over teaching and research staff and place a strain on their motivation. As such, after eight years of hard work, trainee academic staff are suddenly seeing the rules changed before their eyes and permanent positions are no longer being made available.

As for students, they have found themselves facing significant increases in tuition fees for official bachelor's degree and master's degree courses, although it should be stressed that the Catalan government has taken steps to make the enrolment process as equitable as possible and that we are currently completing a study of the situation to increase the efficiency of the newequity grants. With regard to the rise in tuition fees, we were assured that this was not the starting point in an overhaul of the university model but rather the culmination of a process. As such, I continue to believe that it is altogether unacceptable for the Ministry of Education to toughen the conditions offered to potential grantholders without accepting to defer the introduction of theseconditions for at least one academic year.

I will not extend my address unnecessarily by reiterating points that I have made on other occasions. However, I would like to stress one concept that is absolutely vital: while it is undoubtedly true that research and education are costly undertakings in a strictly financial sense, these costs cannot be considered a burden if we view them through the lens of democratic freedom and, in particular, social progress, since they represent strategic investments that any country or society must strive to maintain if it is to safeguard its future. Put another way, research and education are at the very core of what we are and what we are not, both as a university and as an advanced society.

Contrary to what would seem logical, Catalan public universities, including our own, actually receive less funding every year, yet despite these difficulties we have managed to prevent progress from stalling. The sacrifices behind this achievement have been considerable to say the least, and those costs that cannot be measured in strictly economic terms have been incalculable.

In any case, we have reached a point at which the UPC and the Catalan universities cannot do any more to continue guaranteeing the quality of research and teaching that society asks of us.

Over the last academic year the UPC has adapted to the situation brought by the financial crisis. This has been and shall continue to be achieved thanks to the strong sense of responsibility and the sacrifices of the groups that make up our university community. We are committed to managing the resources available to us more efficiently. The geographical organisation of the UPC that we are currently working on must enable us to manage our schools and harness our resources effectively. We have also undertaken to reduce the deficit by which we are undeniably burdened, much of which is due not to the actions of the UPC but to government failures to meet stated commitments and to the most recent round of cuts, which were imposed without prior warning. Despite this, we are confident of reaching the end of the year without seeing a rise in the deficit with respect to the previous tax year.

We know that the best way to fight against the deficit is to seek new sources of finance built on research valorisation and greater interaction between the University and society, and to make our institution more competitive. This is our responsibility, and it is a process that is producing good results, as you can see in the indicators presented in the summary of the academic year that you have been given.

Given the situation in which we find ourselves, we might be asked: "And you? Are you in crisis?" To this there is only one answer: "Yes, we are. A very deep crisis. But we have the potential and the opportunities to find a way out, and we must make the most of them!" For this to be possible, we must, in addition to the steps we have taken so far, act without delay to clarify the issues on which the necessary reforms to the Catalan university system hang. We must have a system of public funding that is clear, stable and transparent, founded on the basic premise that we should not –as is the case at the moment– penalise those universities that obtain the best results. We need a funding system which acknowledges that the primary responsibility for bringing about these reforms lies with the executive of the Catalan

talan government, which must hold free and open discussions with the universities. We share the government's view that the list of available degree courses needs to be redimensioned. We agree and are acting accordingly, which is evident in the work we have already done towards the eventual merging of certain degree courses. That said, it would not be a good thing for universities to find themselves subjected to new measures to which joint consideration had not been given.

Our university will accept the responsibilities placed on it. Let there be no doubt about this. It is doing so now and will continue to do so, despite the internal difficulties brought by the need to substantially reduce certain areas of our activity. We are aware that, to improve, we must make adjustments to our management structures and practices and to our governance model. Structural adjustments must seek to eliminate those areas that make little or no contribution to our institution. We must bring change through intelligent debate at the heart of the university community, but this is not easy without the incentives to catalyse change and transformation –incentives that, for the present time, will necessarily be of an intellectual rather than an economic nature.

I declare open the 2012-2013 academic year at the **Universitat Politècnica de Catalunya** • Barcelona**Tech**.