Software e Ingegneria

Incontro con la Commissione strutture dell'Ordine degli ingegneri di Roma
7 luglio 2006.

Intervento introduttivo

Dott. Arch. Roberto Spagnuolo
Softing srl

Software e Ingegneria

Incontro con la Commissione strutture dell'Ordine degli ingegneri di Roma
7 luglio 2006.

Dott. Arch. Roberto Spagnuolo

Introduzione

Questo incontro è stato promosso dalla Commissione Strutture e quindi ritengo sia un'ottima occasione per parlare del prodotto software dalla parte di chi lo produce, ovvero dall' esperienza di chi progetta software. E quindi fare una sorta di percorso "alla rovescia" cioè invece di farvi vedere la "facciata" del software, farvi vedere quello che vi è "dietro" perché poi meglio si capisca come certe decisioni, certe posizioni di fondo, divengano parte del prodotto software.

Per fare questo devo tratteggiare quella che, almeno a mio avviso, è la situazione dei produttori del software per l'ingegneria. Tenete presente che chi vi parla ha un'esperienza di più di trent'anni di informatica al servizio dell'ingegneria quindi ha maturato delle opinioni che possono certamente non essere condivise ma hanno il peso di una esperienza maturata sul campo.

Purtroppo il leitmotiv del rapporto software ingegneria è improntato su una scarsissima comunicazione, comprensione, collaborazione. Dico questo, sia chiaro, non come una critica, ma per dare una chiave di lettura della situazione.

Qui ho usato un'immagine scherzosa. Ma certo non si può dire che si comunichi quanto sarebbe auspicabile

Bisogna comprendere che per fare software per ingegneria si deve fare ricerca, una ricerca autonoma condotta con passione perché il vero lavoro di chi fa software dovrebbe essere quello di rendere utilizzabili dei metodi di calcolo su elaboratore. Questo purtroppo non avviene semplicemente perché i metodi di calcolo, studiati per applicazioni numeriche, non sono disponibili e devono essere oggetto di una ricerca continua da parte di chi fa software per l'ingegneria.

Coloro i quali fanno software in Italia per ingegneria sono poche persone e non sono più giovinetti. L'esperienza di tali persone sarebbe preziosa per l'Ingegneria, con la I maiuscola e, di fatto, tenerle lontano da ogni dibattito è un vero peccato, una grossa perdita, per l'ingegneria.

Cito alcuni di questi signori che considero "colleghi" e non "concorrenti". Pica, tra i decani, Aiello e Migliorini, Casciaro, Daddi e Nulli. Non me ne vengono in mente altri. Mi scuso con chi ho dimenticato. Come vedete non siamo tanti e dovreste chiedervi quando andremo in pensione chi ci sostituirà perché di passione ce ne vuole tanta mentre di soddisfazioni economiche se ne hanno davvero pochine.

Purtroppo, lo riconoscerete, nel mondo dell'ingegneria strutturale c'è molta conflittualità, pochissima capacità di coesione e questo si riflette nel mondo del software per ingegneria e tra noi (tutti ingegneri eccetto chi vi parla…) progettisti di software.

Noi siamo visti come "mercanti" che realizzano un prodotto visto più spesso come una "tassa" che come un aiuto alla professione. Non ci si sofferma affatto a chiedersi se la nostra competenza professionale non sarebbe un arricchimento per l'ingegneria e se un dialogo non sarebbe utilissimo anche per capire meglio le "specifiche" del prodotto. E in effetti fa pensare che in una Commissione Strutture non vi sia un rappresentante dei produttori di software.

In effetti oggi sono pochissimi gli strutturisti che non usano programmi di calcolo. Eppure se ne parla sempre sottovoce.

Per questo motivo quando arriva l'invito a fare una "dimostrazione" (credo che, in 25 anni che esiste la Softing, abbiamo partecipato a centinaia di tali eventi), mi sembra che l'incomprensione sia alla sua massima espressione e francamente la cosa mia addolora invece di farmi piacere. Cosa serve far "vedere" un programma quando non si può vedere cosa vi sia "dietro"? Cosa serve chiedere se il programma "fa" o non "fa" una cosa e se lo "fa" secondo la normativa? Noi progettisti di software non siamo dei marziani e conosciamo le esigenze dei nostri clienti.. Si dovrebbe chiedere COME fa quella cosa. Ed è quello che io tenterò di spiegare dal punto di vista della Softing.

F. Errera "La maschera e il volto".
Tratto da www.dfn.it.

Il software ha una "facciata" e un contenuto. Io vorrei parlare di contenuti e mettere in guardia dalla "facciata" che spesso nasconde contenuti insufficienti alla soluzione del problema al quale il software si dichiara dedicato.

E’ un po’ come il maquillage delle donne, che in certi casi si può chiamare, sia per le donne che per il software, più semplicemente ed emblaticamente “trucco”.

Infatti quello che ci si deve chiedere è sempre: cosa c’è sotto? Mi ricorda il titolo di un film, “Sotto il vestito niente”, titolo che non alludeva alla biancheria intima ma ai contenuti.

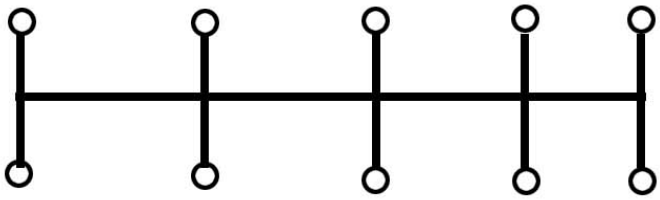
Di fronte al “trucco” o al maquillage, se non ci facciamo domande sui contenuti non abbiamo forse una visione uniformata e, in qualche modo, mistificante?

I programmi sono tutti eguali?

La trasparenza

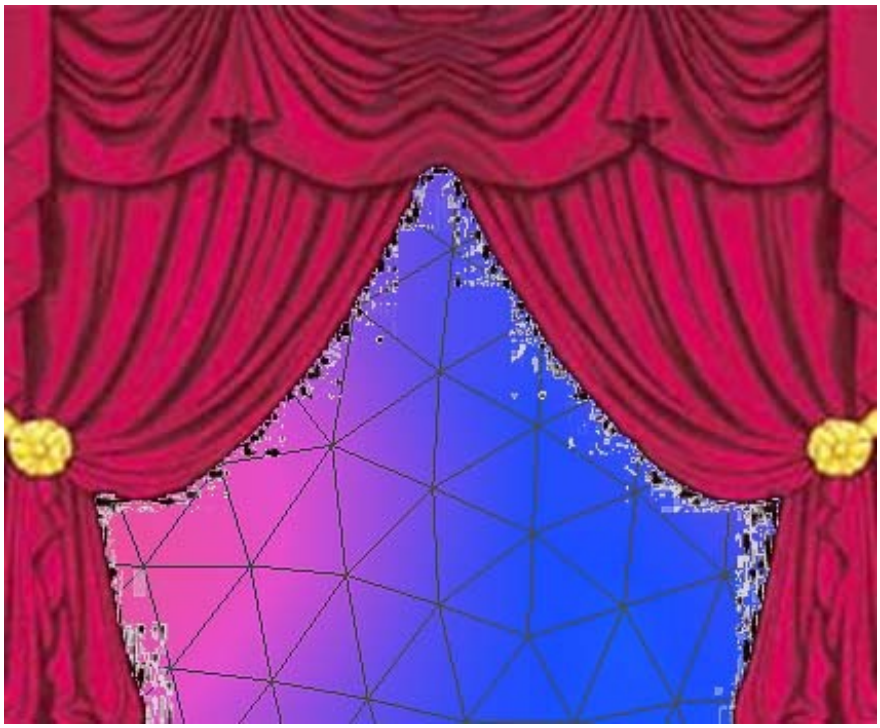
Come qualcuno avrà intuito dal colore candido dei miei capelli, ho avuto la grande fortuna di vivere l'avventura della informatica mentre arrivava, nella seconda metà degli anni settanta, sul tavolo dell'ingegnere. Una grande avventura. Vorrei ripercorrerne un attimo una tappa non come un "amarcord" ma per far meglio capire la situazione attuale.

Il calcolatore che imperversava negli studi dei progettisti nella metà del '70 non era quello della foto qui sopra. Questo modello venne invece qualche anno dopo (1978). Ho trovato però solo l'immagine di questo Olivetti BCS (Business Calculating System) 2030. Comunque la differenza non era molta. Si avevano pochi registri di memoria per cui i problemi trattabili erano molto limitati e si dovevano usare MODELLI DI CALCOLO molto semplificati.



Che vuol dire questo? Che il progettista, in genere anche programmatore dei suoi programmi, aveva una consapevolezza assoluta del MODELLO e dei limiti del modello.

In figura, il modello che allora poteva essere usato estrapolando da un telaio spaziale una trave continua con il contributo di rigidezza dei pilastri. E' ovvio che non avendo una conoscenza accurata dei limiti di questo modello, se lo si fosse usato inconsapevolmente per casi non contemplati (ad esempio forze orizzontali), i risultati sarebbero stati del tutto inattendibili.



Il software moderno offre un sipario di immagini che "nasconde" spesso i contenuti

La crescita del software ha costruito un sipario tra immagine e modello di calcolo per cui il progettista ha perso la consapevolezza di operare su modelli e questo è, a mio avviso senza esagerazione, drammatico. Infatti va chiarito che anche se il software ha ben altre potenzialità rispetto a quelle di quei tempi, opera sempre su modelli con tutti i limiti intrinseci di tale concetto.

Modello? Chi era costui?

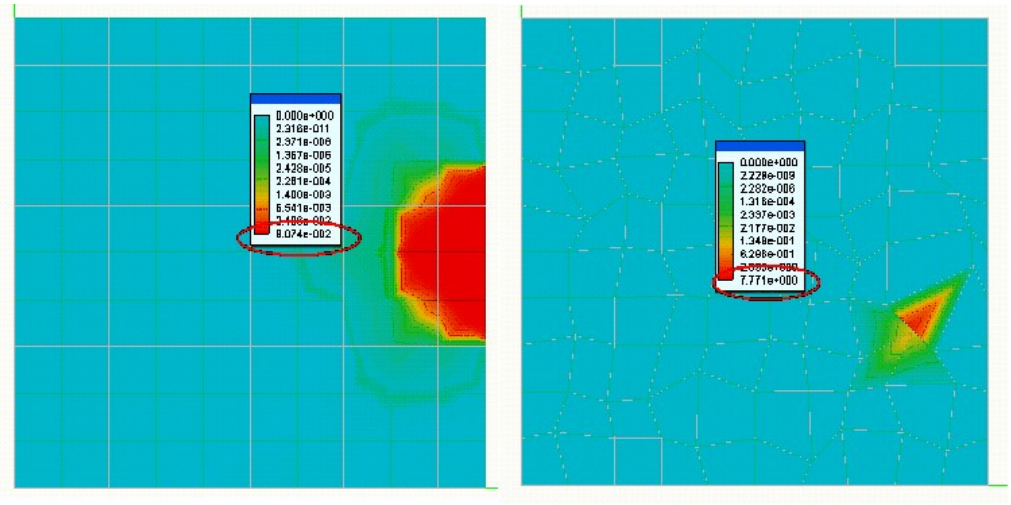
Il modello galileiano dovrebbe essere profondamente radicato nella nostra cultura invece sotto il sipario del software ce ne dimentichiamo.

Il famoso piano inclinato di Galilei in cui egli afferma consapevolmente di trascurare l'attrito. Il concetto di modello galileiano consiste, tra l'altro nell' isolamento delle caratteristiche più fondamentali dei fenomeni.

Trascurare il concetto di modello fa sì che il software per l'ingegneria possa essere visto come un elettrodomestico ma così, purtroppo o per fortuna, non è né può essere. Il concetto di modello fa sì che la mediazione del professionista non possa essere surrogata da un "ingegnere artificiale". Io dico: per fortuna.

Il software può essere usato come un elettrodomestico?

Ad esempio, ed è uno solo su una infinità di esempi possibili, anche il potente metodo degli elementi finiti è fortemente dipendente dalla mesh. Se non si conoscono queste caratteristiche del metodo e del modello di calcolo non si può avere un controllo responsabile sulla soluzione.



Qualità di due mesh valutate in Nòlian. La mesh regolare (sinistra) ha una qualità di circa 100 volte superiore a di quella di destra dove si sono volutamente impiegati elementi molto distorti

L'uso non consapevole del software e il fatto che se ne mostri (e se ne voglia vedere!) più spesso il maquillage invece che il vero volto e i contenuti, assolutamente pregevoli ma tutti da capire, conoscere e saper usare, dequalifica anche l'aspetto progettuale della professione. Dietro la scusa, perché è una scusa, di un necessario abbattimento dei costi di progetto, si nasconde spesso una scarsa dimestichezza con lo strumento software al quale si richiede di assolvere a compiti che non gli sono propri e che si traducono immancabilmente in una dequalificazione del progetto.

In effetti, se qualcuno volesse fare in qualche modo una "classifica" dei programmi per ingegneria potrebbe avere un modo di ordinarli: per trasparenza del modello. Anche non volendo dare un significato negativo alla mancanza di trasparenza, ma solo quello di un uso meno consapevole e coinvolgente, questo parametro resta un parametro validissimo che può anche guidare nell'acquisto a seconda della nostra personale maggiore o minore propensione a partecipare alle scelte non solo del progetto, ma per estensione caratteriale, della nostra vita.

Si va sempre più spesso verso quello che sembra stia diventando un vero "diritto": dover solo spingere un bottone.

Validazione del software

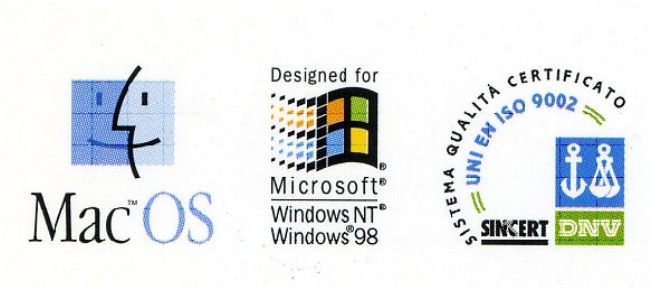
Vediamo ora se e come si può valutare la qualità intrinseca del software.

Precisiamo subito la terminologia.

Certificazione
Verifica dell'aderenza di un prodotto ad uno standard di riferimento

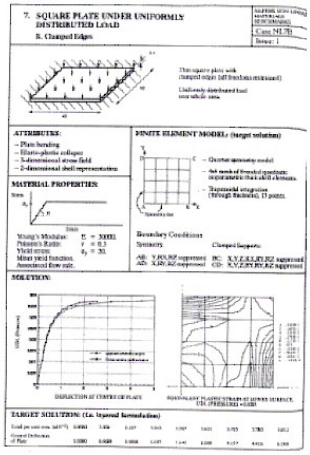
Validazione
Conferma del soddisfacimento di particolari requisiti relativi a un determinato impiego (ISO 8402).

I nostri programmi, ad esempio sono CERTIFICATI da Microsoft per la loro aderenza allo standard di Microsoft di interoperabilità, di scrittura, di aderenza alle modalità di interfacciamento con l'OS, di usabilità da parte di disabili. Cioè sono CERTIFICATI in quanto aderiscono ad uno standard e Microsoft, attraverso Veritest, ne ha certificata l'aderenza a tale standard.

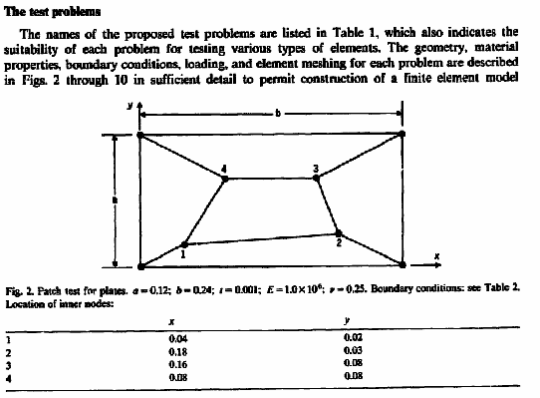


Nòlian, a quel tempo (1988) MacSap, è stato anche VALIDATO dal Politecnico di Milano in quanto si sono definiti con il Politecnico stesso dei REQUISITI (una vasta tipologia di strutture per le quali si sono predisposti dei casi-prova) e si è verificata la rispondenza di Nòlian a tali REQUISITI.

Va detto che la "ingegneria del software" oggi è una disciplina scientifica e quindi i metodi di validazione non possono essere improvvisati da chi non è del mestiere. Ad esempio ogni tanto ci viene proposto come "caso prova" un edificio usato in un corso di aggiornamento presso un qualche Ordine Professionale. I dati sono espressi come la carpenteria dell'edificio. Questo vuol dire non aver capito nulla della validazione dove si devono studiare dei test che "stressino" i programmi nei punti supposti critici e dove si deve ESCLUDERE TASSATIVAMENTE la perturbazione del test tramite interpretazioni del modello. Queste richieste di test da parte di inesperti in tale disciplina ancora una volta testimoniano superficialità nel capire cosa sia il software e anche sfiducia nel nostro operato. Vi risparmio di nuovo l'immagine delle tre scimmiette…



Questa è una scheda di uno dei casi prova del NAFEMS (National Agency for Finite Element Method Standard). Il NAFEMS ha istituzionalmente prodotto moltissimi e impegnativi casi prova per la validazione del software ad elementi finiti. I programmi Softing sono validati con tutti i casi prova del NAFEMS applicabili.



The test problems

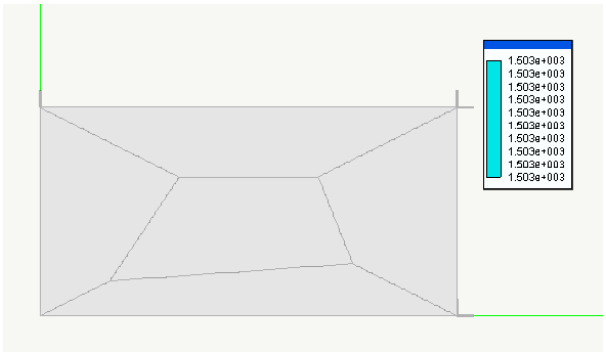
The names of the proposed test problems are listed in Table 1, which also indicates the suitability of each problem for testing various types of elements. The geometry, material properties, boundary conditions, loading, and element meshing for each problem are described in Figs. 2 through 10 in sufficient detail to permit construction of a finite element model

Fig. 2. Patch test for plates. $a = 0.12$; $b = 0.24$; $t = 0.001$; $E = 1.0 \times 10^6$; $\nu = 0.25$. Boundary conditions: see Table 2.

Location of inner nodes:

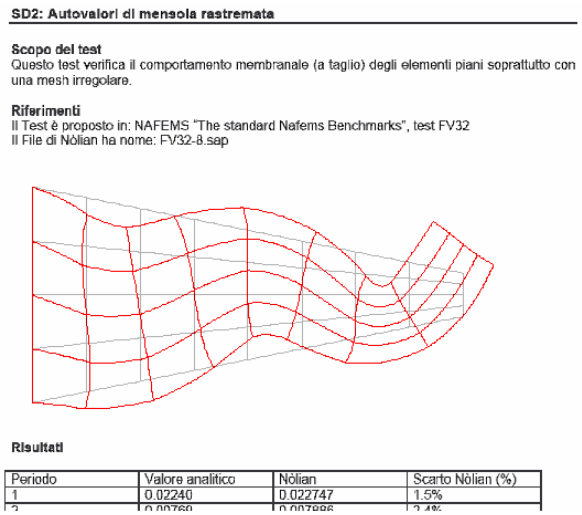
	x	y
1	0.04	0.02
2	0.18	0.03
3	0.16	0.08
4	0.08	0.08

Questa immagine invece si riferisce ad un pagina dello storico articolo (1985) di MacNeal nel quale si proponevano dei test molto significativi per i programmi ad elementi finiti. Questa pagina si riferisce ad uno dei più famosi test proposti, il “patch test”. Nei patch test ci si propone di verificare il comportamento di elementi finiti di forma distorta in un insieme. La metodologia di test più ricorrente (e ripetibile anche da un utilizzatore) è quella di sottoporre il patch ad un insieme di spostamenti nodali che comporti un campo di deformazioni costante. Se gli elementi non sono in grado di rappresentare un tale campo non sono di buona qualità e danno risultati inattendibili. E’ un test fondamentale per chi sviluppa e implementa elementi finiti.



In questa immagine, la rappresentazione grafica del campo di tensioni in Nòlian che mostra la splendida risposta degli elementi finiti di Nòlian al patch test di MacNeal.

Per inciso, mi preme fare un'osservazione. L'utilizzatore è più colpito dal "bug" (malfunzionamento) che non dalle carenze di modello. Si tratta di un effetto psicologico. Il malfunzionamento interferisce con il nostro spazio di controllo, la carenza del modello, essendo subdola, non ha lo stesso impatto emotivo. Va notato che il bug si elimina, spesso facilmente, e non inficia la qualità del risultato. Forse, solo momentaneamente, diminuisce la produttività mentre la carenza del modello influisce in modo permanente sulla qualità dei risultati. Per questo è molto importante parlare di contenuti e di modelli impiegati.



SD2: Autovalori di mensola rastremata

Scopo del test
Questo test verifica il comportamento membranale (a taglio) degli elementi piani soprattutto con una mesh irregolare.

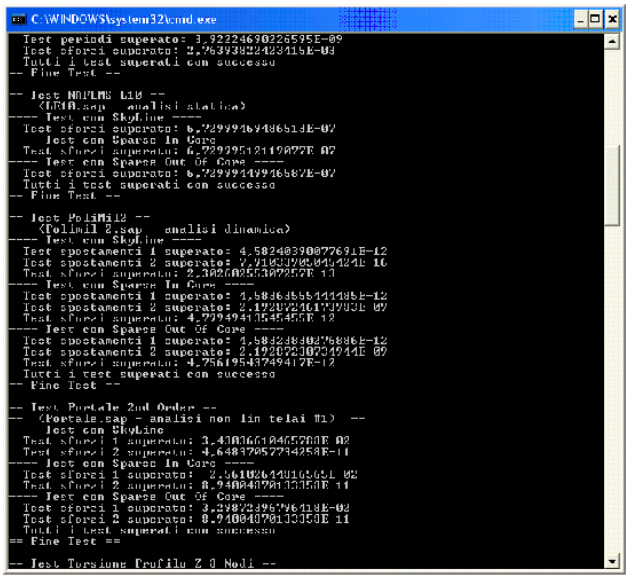
Riferimenti
Il Test è proposto in: NAFEMS "The standard Nafems Benchmarks", test FV32
Il File di Nòlian ha nome: FV32-8.sap

Risultati

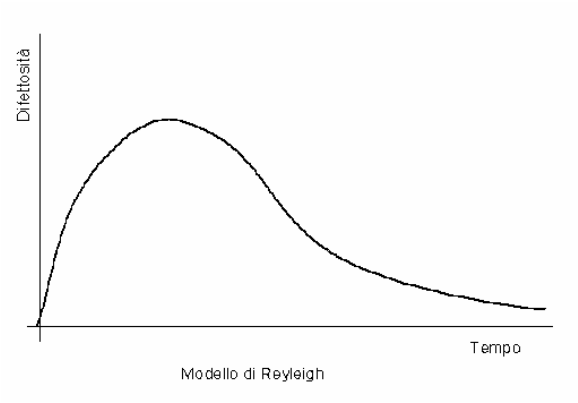
Periodo	Valore analitico	Nòlian	Scarto Nòlian (%)
1	0.02240	0.022747	1.5%
2	0.00760	0.007886	3.4%

In questa immagine una pagina di uno dei manuali di Validazione di Nòlian. In questo caso un benchmark del NAFEMS.

Inoltre Nòlian è sottoposto, tramite un sistema automatico, all'esecuzione di circa 80 casi prova ogni volta che viene rilasciato. Ciò consente di individuare eventuali difetti collaterali che possono involontariamente essere stati introdotti apportando modifiche o migliorie al programma.



In figura una schermata dello script durate l'esecuzione automatica dei test.



Modello di Reyleigh

La tematica della validazione è vastissima e non intendo certo parlarne qui. Qui intendo solo fornire alcuni spunti e informare della nostra ovvia ma spesso non nota maniacale attenzione a questi aspetti. Infatti un malfunzionamento ha un costo non solo per l'utilizzatore ma anche per noi e quindi evitarlo fa parte dei compiti di quella Ingegneria del Software che è una disciplina scientifica ma anche manageriale.

Quindi farò solo un cenno ai metodo statistici. La curva in figura è il modello di Rayleigh per valutare la difettosità (probabilistica) del software nel tempo. Come vedete la diminuzione ha un andamento asintotico e quindi non diventa mai nulla. E in effetti è dimostrabile che non può esistere un software senza difetti. Si può diminuire la probabilità che avvenga, intesa come intervallo di tempo in cui si manifesti il difetto. Gli informatici parlano del "baco dei 5000 anni" come una specie di essere sovrannaturale che sanno essere annidato nel loro software. Comunque l'attento monitoraggio della diminuzione della difettosità e le previsioni statistiche consentono di "rilasciare" il software quando la difettosità prevista nel periodo di utilizzo scenda sotto livelli prefissati. E in genere ci si indovina abbastanza.

Ultima osservazione. Servono benchmark (casi prova) nel modo dell'ingegneria civile ed edile. Non solo specializzati per gli elementi finiti applicati in tale campo, ma anche negli altri aspetti dell'ingegneria (progetto di armature, verifiche membrature metalliche etc.). Ciò consentirebbe di conseguire una migliore qualità. Devono essere resi disponibili da uno sforzo comune. Non possono essere certo predisposti solo dalle singole software house, non solo per i costi inaccessibili, ma anche perché sarebbe un po' come chiedere all'oste se il vino è buono…

Va notato che per poter eseguire i test il modello di calcolo adottato nel software deve essere completamente accessibile altrimenti i test veri non si possono fare. Un software che ha una immissione dati solo per "carpenterie" non consentirà certo di fare un patch test. E probabilmente neanche il produttore avrà facilità a farlo. Questo è un altro gravissimo limite dei programmi che "banalizzano" il problema.

Infine, oltre alla validazione del software, sarebbe utile “validare” gli utilizzatori. L’analisi strutturale oggi è un compito molto complesso al quale non si può pensare di essere preparati solo perché si ha una laurea o un’abilitazione. Ci vuole una preparazione specifica. il NAFEMS, ad esempio, fa dei corsi in questo senso e rilascia diplomi di analista.

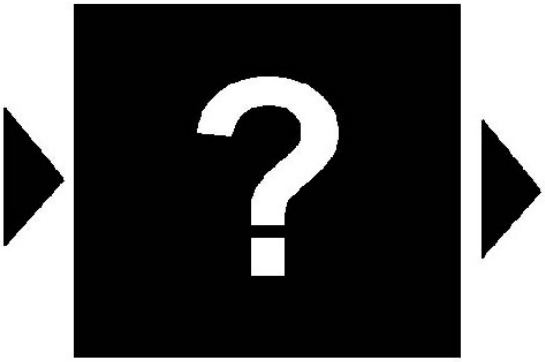
Dal sito NAFEMS i corsi specializzati per analisti

La nostra “filosofia”

Il software è un “prodotto”. Bisogna tenerlo presente. E come tale va progettato per rispondere al meglio a particolari esigenze. Chi si chiede soltanto “Quale software è il migliore?” non ha capito niente. Deve chiedersi “Quali sono le mie esigenze?” e quindi “Quale software risponde meglio alle mie esigenze?”. Quindi il progetto del software risponde a delle precise esigenze che sarebbe opportuno fossero sempre ben chiare.

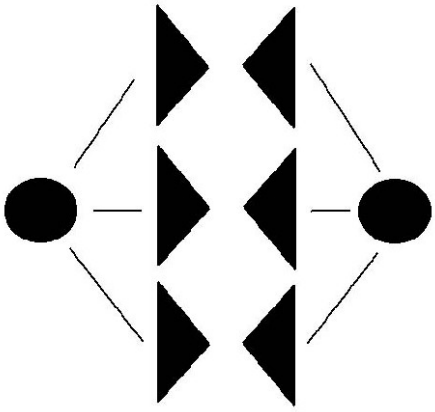
Tanti modelli di autovetture. Perché si crede che il software debba essre un pezzo unico?

Nel nostro caso la domanda fondamentale che ci siamo fatti è questa. Il progetto di ingegneria può essere definito in una procedura? La riposta che ci siamo dati è che solo il progetto di tipologie strutturali ben definite può essere procedurale. Il progetto strutturale nella sua generalità non è proceduralizzabile. Cioè non può essere descritto in una procedura predefinita e rigida. Il progetto strutturale ha una casistica vastissima e imprevedibile. Quindi il software NON PUO’ ESSERE PROCEDURALE se vuole rispondere a QUESTA ingegneria.



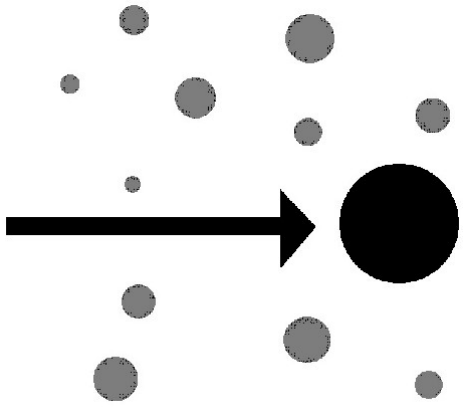
Software procedurale: La grande scatola nera

Il fatto è fondamentale. Noi della Softing progettiamo software per chi affronta casistiche variate di progettazione e per chi non ritiene di poter conseguire un progetto di qualità affidandosi ad una procedura predefinita e rigida.



Il software non procedurale: Idea – Analisi – Decisione - Progetto

Quindi diamo delle indicazioni molto precise su come è strutturato il nostro software e diamo al "consumatore" indicazioni precise su quello che troverà accostandosi alla nostra soluzione. Diamo anche l'indicazione, consequenziale, che chi vuole seguire la strada della scelta autonoma del percorso progettuale deve essere anche in grado di farlo. Non può pensare di avere libertà senza pagare il prezzo della responsabilità. Noi progettiamo software per queste persone. Ogni azienda ha un "target" di "consumatore" il nostro è quello del progettista attento, esperto, qualificato.

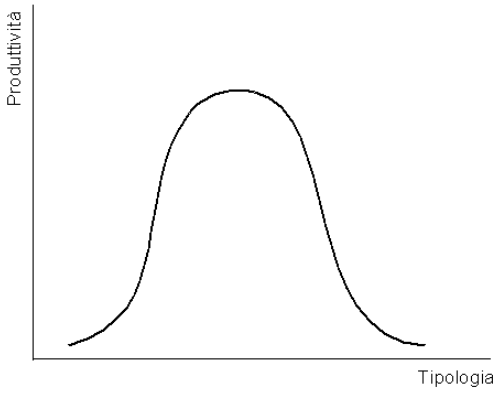


Ogni azienda ha un target preciso

Un piccolo ricordo personale che spero mi perdonerete. Quando nel 1983, su incarico di Apple Computer Spa (allora IRET Informatica) progettavo MacSap (oggi Nòlian) l'immagine che seguivo nella mente era quella del fabbro che ha un oggetto (il progetto) e che su di esso applica i suoi strumenti per forgiarlo come ritiene opportuno. Nulla di meno procedurale. La mia visione del progetto non è mai stata la catena di montaggio (procedura): la catena di montaggio è per la produzione, a mio avviso, la produzione che avviene DOPO il progetto e in conseguenza di esso. Progettare è EVITARE ogni condizionamento alle proprie decisioni.

La metafora del fabbro che forgia il progetto con liberi strumenti software è la metafora del software Softing

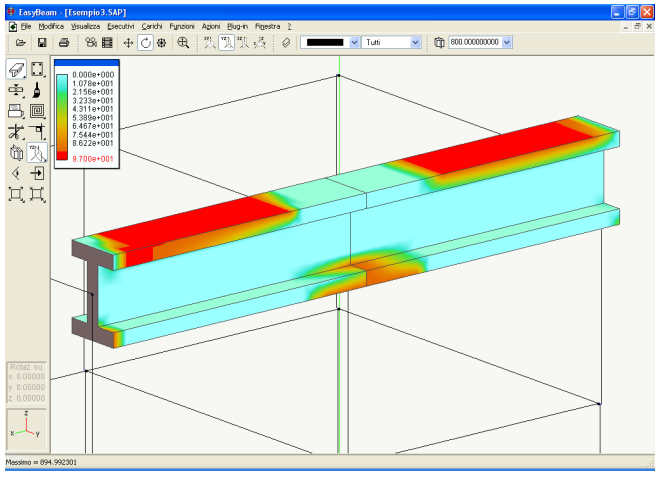
Un altro concetto che può aiutarvi a capire quanto sto dicendo si può esprimere in un grafico che metta in relazione la produttività del software con la tipologia di problema (in ascissa).



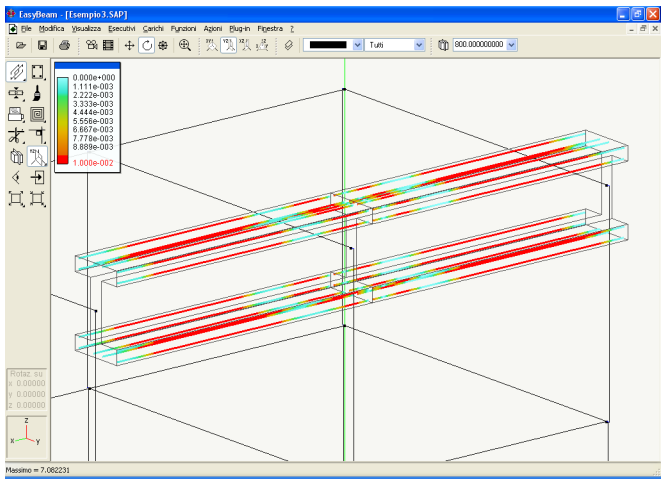
Immaginiamo (il grafico è puramente illustrativo) che si abbia una distribuzione gaussiana delle probabilità che si abbia un certo tipo di produttività per una certa tipologia strutturale. Ora, se la campana è "stretta" vi sarà la specializzazione del software. Se è più larga si avrà una produttività a più ampio spettro, cioè per più tipologie strutturali. Questo non solo vi aiuta a capire meglio il software che fa per voi ma vi aiuta anche a classificare il software in commercio. La gran parte dell'offerte sono dedicate all'edilizia pura e quindi hanno un'alta produttività per questa tipologia predefinita ma la loro produttività crolla se dovete fare un semplice serbatoio o, sia mai, se un contenzioso vi costringe a rispondere a delle domande e non solo a seguire un percorso progettuale rigido e predefinito.

Vediamo, con un solo e rapidissimo esempio degli innumerevoli che potremmo portare, quali sono gli strumenti che potete usare nel progetto delle armature, fermo restando che in EasyBeam con un solo bottone fate il progetto con gerarchia delle resistenze e per sezioni di qualsiasi geometria e comunque sollecitate (sollecitazioni deviate). Solo una mini panoramica su ben 12 (e forse ne

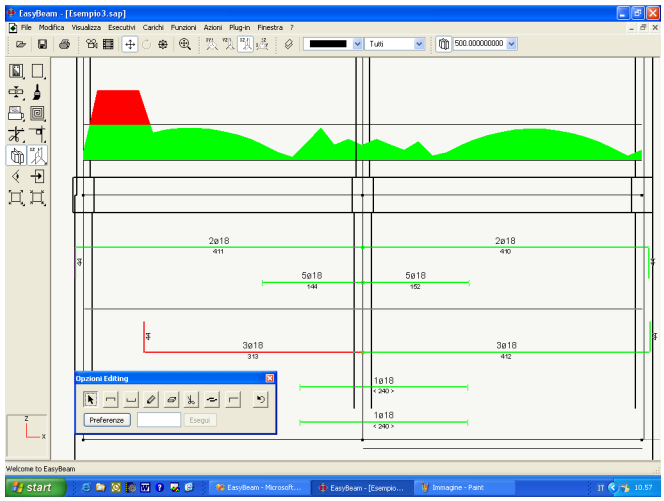
dimentichiamo qualcuno) strumenti soltanto per la verifica del progetto delle armature che vi consente di sapere tutto quello che volete del vostro progetto, di modificarlo e ricontrollarlo. La metafora del fabbro è quella del nostro software.



Tensioni nel calcestruzzo



Tensioni nelle armature



Modifica interattiva con diagramma in tempo reale del coefficiente tridimensionale di sicurezza

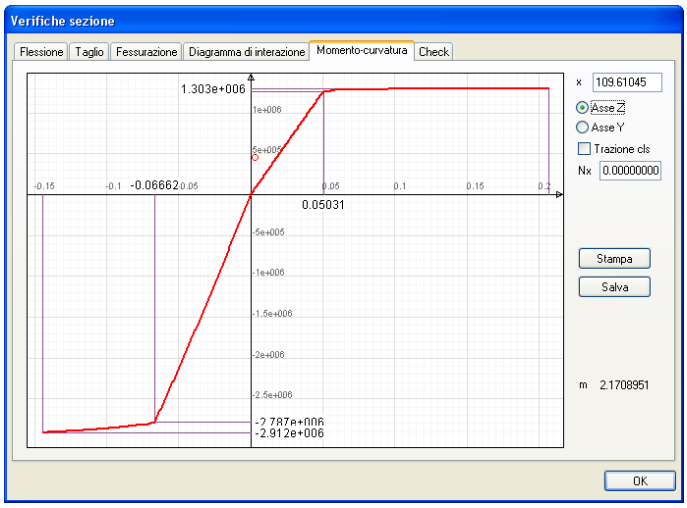
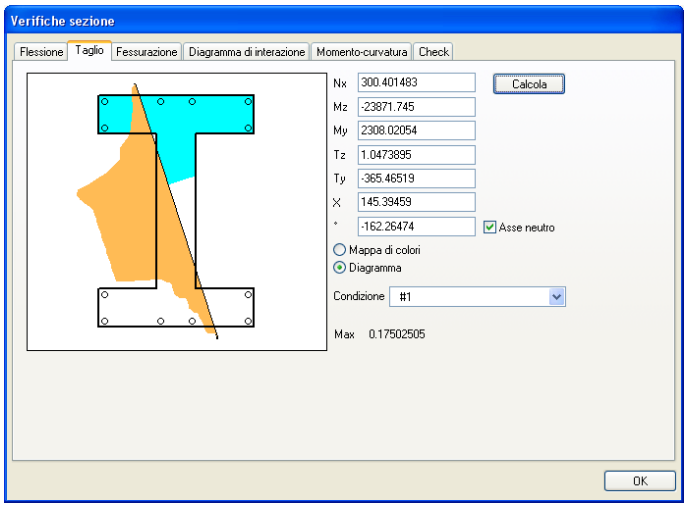


Diagramma momento curvatura e valutazione accurata di duttilità



Verifica delle tensioni tangenziali in sezioni qualsiasi sollecitate anche in modo deviato

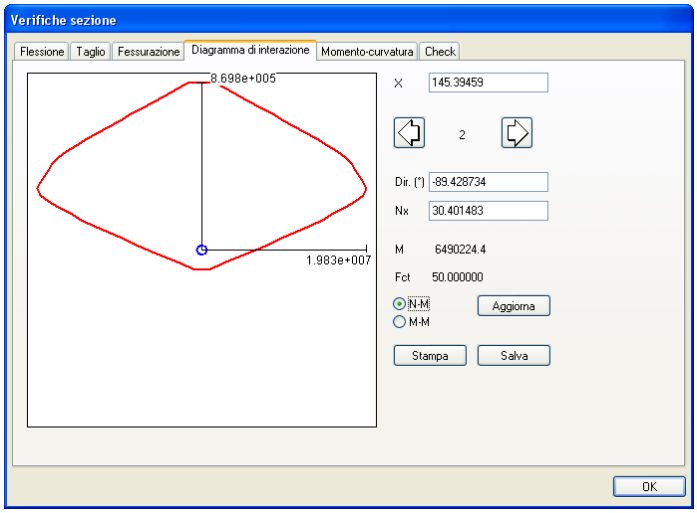


Diagramma di iterazione. Dominio di rottura

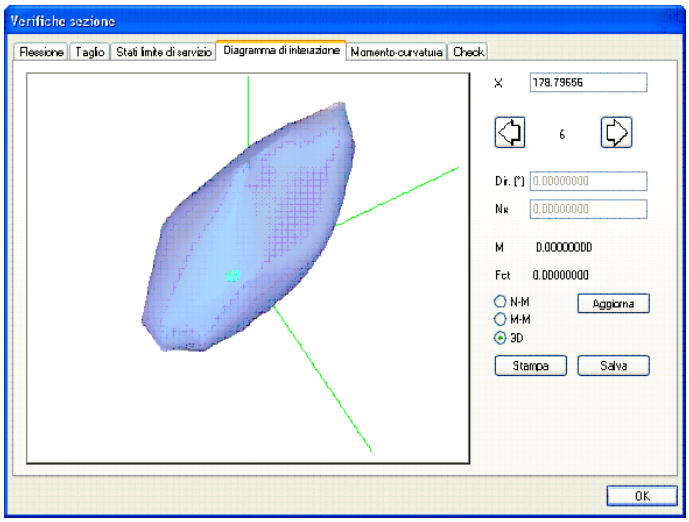
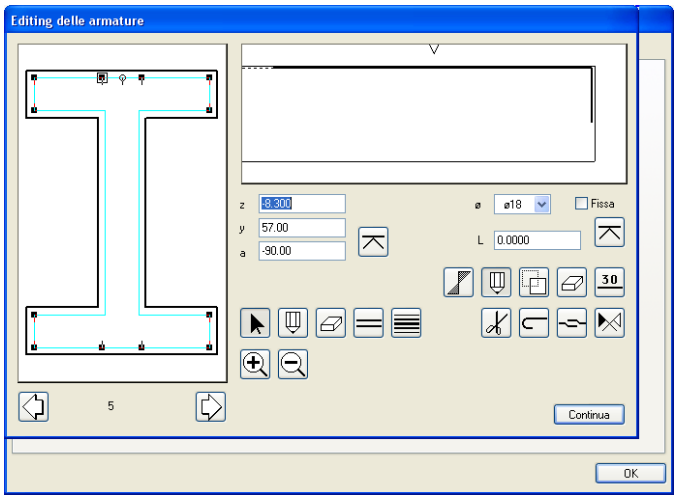
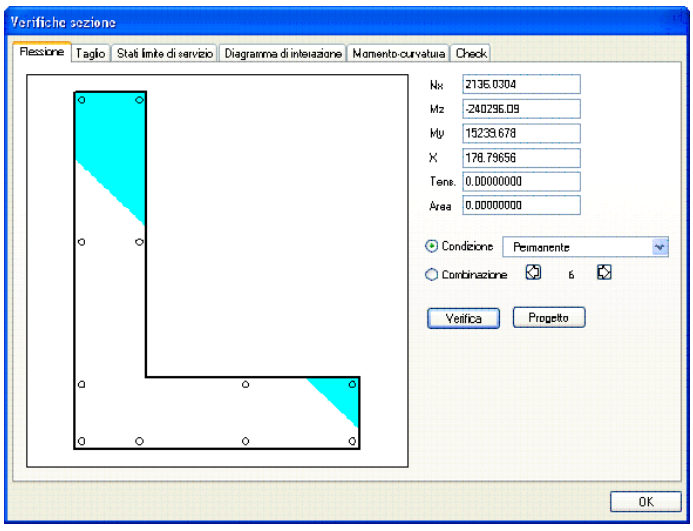


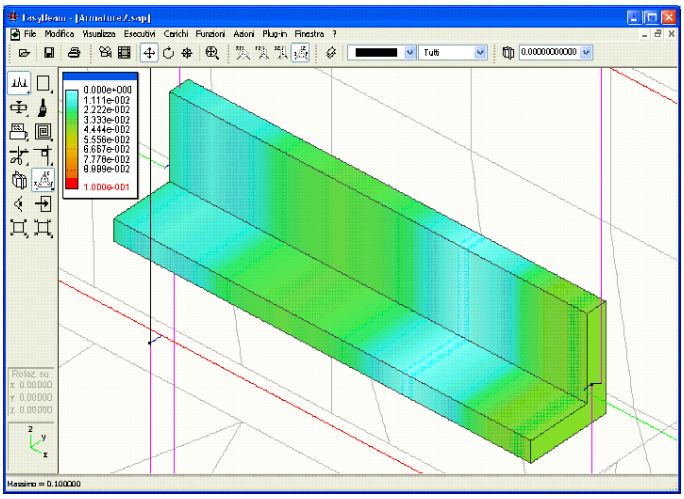
Diagramma di interazione tridimensionale



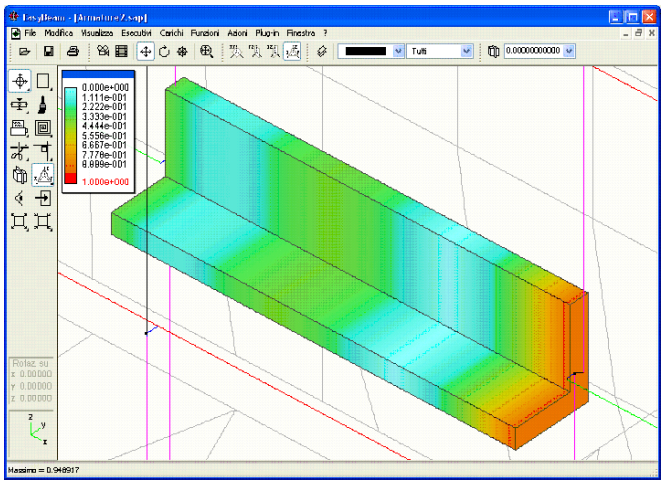
Gestione delle armature nelle tre dimensioni



Verifica a presso flessione deviata di sezioni generiche



Verifica sia a dialogo che a colori della fessurazione



Fattore di sicurezza tridimensionale (dominio di rottura) su tutta la trave

Il problema della normativa
Un falso problema

Chi mi ha seguito nell'esposizione del concetto di software non procedurale si renderà conto che un tale software risponde virtualmente ad ogni esigenza e ad ogni domanda del progettista. Poiché la normativa non è una procedura ma una serie dei requisiti, un software non procedurale è virtualmente indipendente dalla normativa.

Molti sono preoccupati dal metodo degli stati limite. Questo metodo con la normativa attuale non c'entra nulla. Esiste dall'ottocento, è noto e previsto anche dalla passata normativa. Dire che costituisce un problema introdotto dalla nuova normativa è errato. Può costituire un problema di mancanza di conoscenza da parte di alcuni progettisti. Ma non è un problema di normativa.

In effetti, grosse innovazioni la normativa antisismica non ne contiene. Oltretutto è fortemente ispirata agli Eurocodici noti da circa dieci anni.

Il problema semplicemente non c'è. E si noti che il nostro approccio non procedurale si è dimostrato e confermato vincente anche in questa situazione che per alcuni nostri concorrenti è stato uno stravolgimento.

Si può dire solo che certi adempimenti possono essere resi più speditivi o più facili. Ad esempio per la verifica degli spostamenti di piano si possono automatizzare le combinazioni di carico richieste. Ma un software non procedurale avrebbe permesso di farlo egualmente, magari solo impiegando qualche secondo di tempo in più.

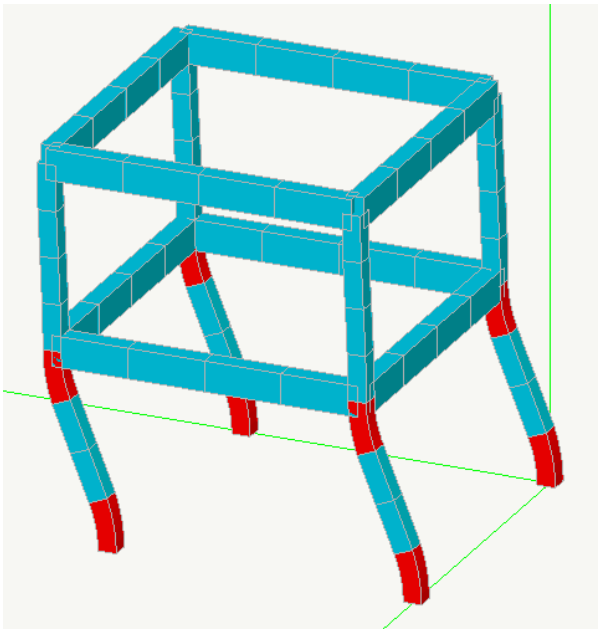
Quindi riaffermiamo il principio basilare: se il software è procedurale è legato alla normativa e la normativa, se cambia, lo stravolge e comporta tutta una serie di instabilizzazioni del codice. Se il software non è procedurale, la normativa rappresenta solo una serie di requisiti ai quali il progettista può SEMPRE e COMUNQUE rispondere.

Progettare è decidere
Il software come strumento decisionale

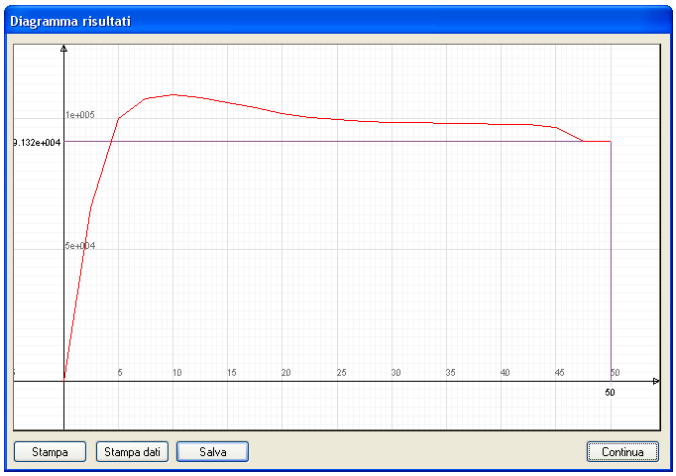
Qui di seguito, per immagini, il progetto con gerarchia delle resistenze e tutti i controlli e le valutazioni che si possono eseguire in Nòlian ed in EasyBeam per un progetto consapevole. Quello che ci preme illustrare, con queste immagini, è il "nostro" concetto di progetto e mettere in luce come gli strumenti progettuali siano tanti, nel nostro software, da consentire di avere un completo controllo sul proprio progetto.

Progetto con riferimento alla duttilità: il problema del "piano soffice"

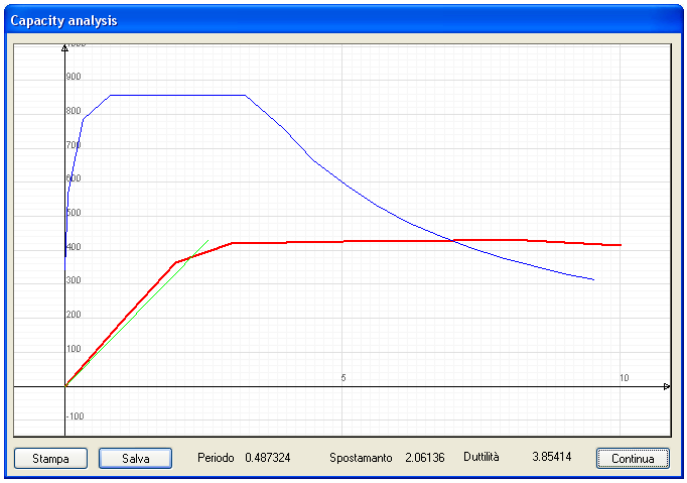
Nel seguito useremo un modello costituito da travi a "fibre"che impiega il legame costitutivo Pinto-Menegotto per l'acciaio e quello di Kent e Park per il calcestruzzo.



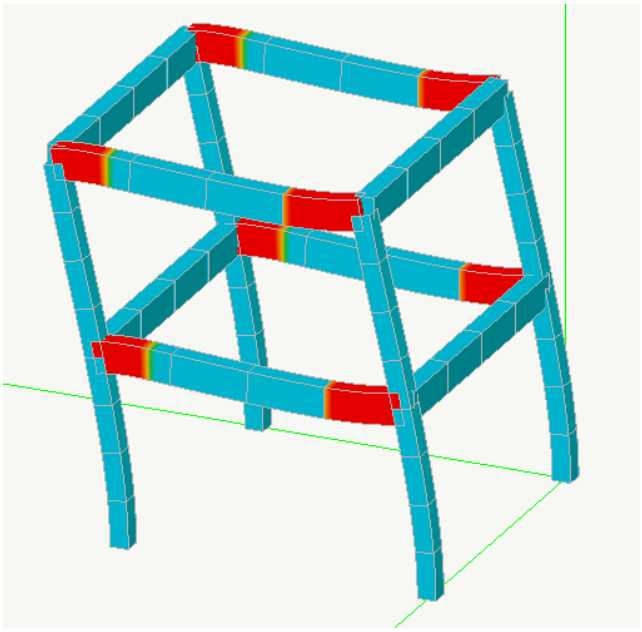
Simulazione in Nòlian. Travi a "fibre"e analisi non lineare. In rosso i tratti completamente plasticizzati.



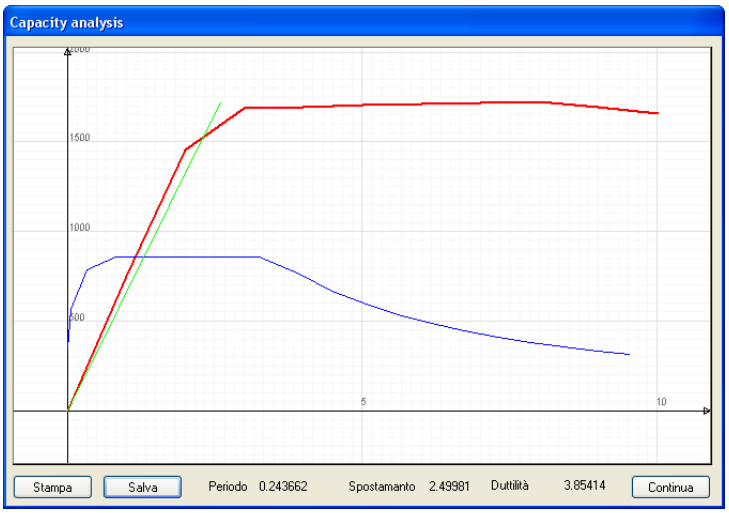
Curva "pushover" del telaio



Capacity design. Confronto tra lo spettro demand e quello capacity del telaio.



Riprogettazione delle armature secondo la gerarchia elle resistenze in EasyBeam



Nuove curve spettrali demand e capacity.

NOTA BENE: Le immagini relative alla analisi di capacità sono ottenute con funzioni di Nòlian non rilasciate nella versione standard.