ottobre/october
2015

euro 10.00
Italy only
periodico mensile

A € 25,00 / **B** € 21,00 / **CH** CHF 25,00
CH Canton Ticino CHF 20,00 / **D** € 26,00
E € 19,95 / **F** € 16,00 / **I** € 10,00 / **J** ¥ 3,100
NL € 16,50 / **P** € 19,00 / **UK** £ 16,50 / **USA** $ 33,95

Poste Italiane S.p.A.
Spedizione in Abbonamento Postale D.L. 353/2003
(conv. in Legge 27/02/2004 n. 46), Articolo 1,
Comma 1, DCB—Milano

ISSN 0012-5377
50995
9 770012 537009

domus

995

LA CITTÀ DELL' UOMO

Collaboratori / Consultants

API/Paola Zanacca
Ilaria Bacciocchi
Cristina Moro
Wendy Wheatley

Traduttori / Translators

Marco Abrate
Paolo Cecchetto
Morgan Chiarella
Barbara Fisher
Kaon Ko
Annabel Little
Dario Moretti
Marcello Sacco
Aya Shigefuji
Edward Street
Rodney Stringer

Fotografi / Photographs

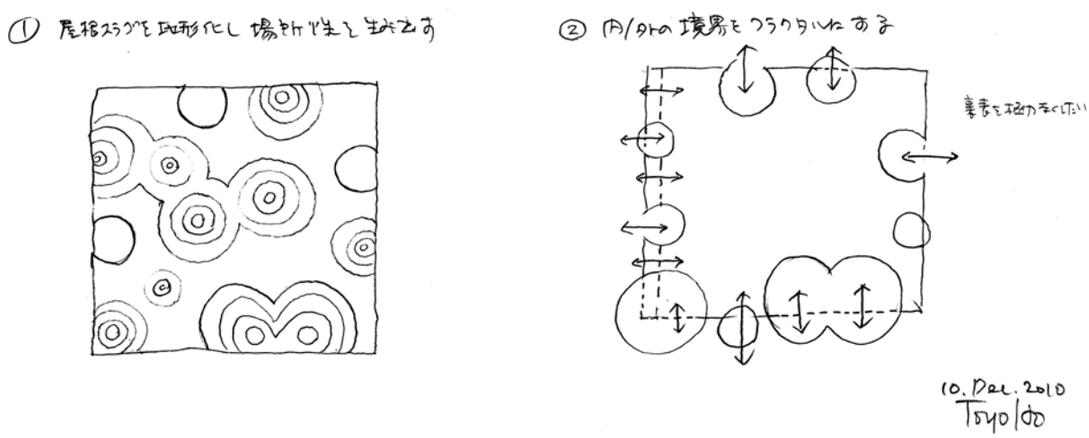
Daici Ano
Iwan Baan
Aldo Ballo
Vasily Babourov
Matteo D'Eletto
Omar Victor Diop
Roland Halbe
Werner J. Hannappel
Antonio Idini
Matteo Imbriano
Nelson Kon
Andrea Martiradonna
Fabrice Monteiro
Kai Nakamura
Yuri Palmin
Juan Rodriguez
Lorenzo Sivieri
Mikhael Subotzky
Patrick Waterhouse

Si ringraziano / With thanks to

Arianna Chiriatti, Galleria Vistamare
Benedetta Spalletti, Galleria Vistamare
Nicola Cavargini, Fondazione Burri
Lucrezia Pizzato, Patricia Urquiola Studio

In copertina: disegno tratto da una serie di schizzi di progetto (a destra) di Toyo Ito per l'edificio Minna no Mori Gifu Media Cosmos a Gifu (Giappone).
© Toyo Ito

▪ Cover: drawing based on design sketches (right) by Toyo Ito for Minna no Mori Gifu Media Cosmos, Gifu (Japan).
© Toyo Ito



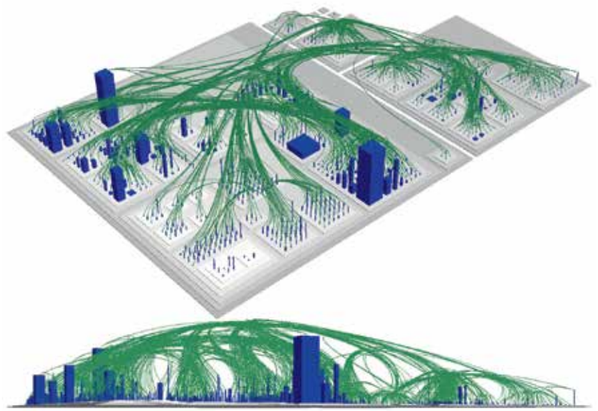
LA CITTÀ DEI SOFTWARE

La rappresentazione tridimensionale del sistema complesso di un software, sviluppato dal gruppo di ricerca diretto da Lanza, in analogia con la forma della città, mette a punto un innovativo strumento di lettura di sistemi altrimenti incomprensibili all'uomo, aprendo la strada al suo utilizzo in vari campi, tra cui anche l'urbanistica e l'architettura

A 3D representation method developed in a research group led by Michele Lanza offers the opportunity to visualise the complex system of software programmes as if they were cities, making it an innovative tool to visualise applications that are otherwise incomprehensible, and paving the way for its use in fields such as urbanism and architecture

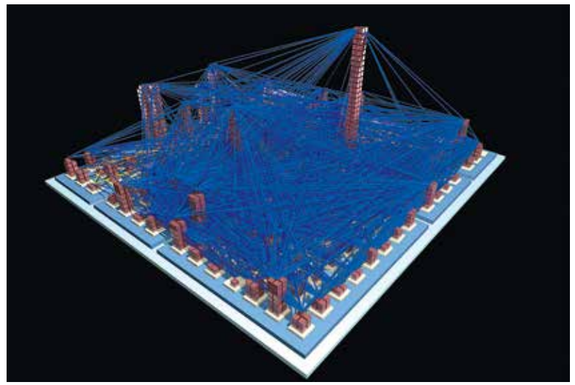
Michele Lanza

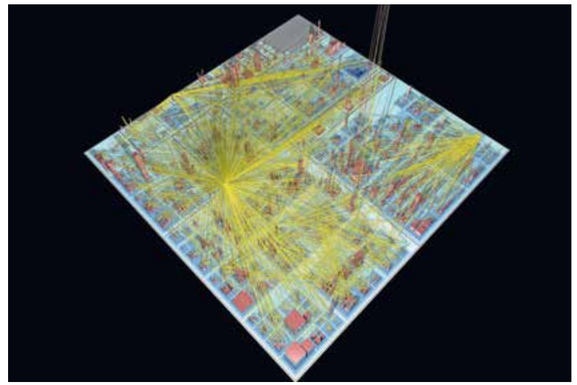
La comprensione e l'analisi dei sistemi informatici è al centro del lavoro del mio gruppo di ricerca Reveal all'Università di Lugano. Per cercare di renderla meno difficoltosa, abbiamo provato a creare una rappresentazione tridimensionale di quella che potremmo definire "l'architettura del software".
Usare la metafora della città per rappresentare il sistema complesso di un software ci ha permesso di mettere in evidenza le anomalie sulle quali sono appoggiate alcune sue parti, quelle che minano la stabilità dell'intero 'edificio' o dell'intera 'città'. Quando si sviluppa una parte del software, non c'è percezione del sistema totale ma si lavora a ogni file di testo come a un singolo edificio, senza avere la percezione dell'intera città. Al massimo si potranno vedere le connessioni con gli altri 'edifici', ma nulla più. Un programma è scritto in base a righe di codice, quindi ogni entità, ogni cuboide, rappresenta una 'classe'. Un sistema può avere milioni di righe di codice, che a loro volta creano decine di migliaia di classi. Ogni edificio rappresenta una classe, pertanto guardando queste rappresentazioni di città si può comprendere la complessità di un software, che è di per sé qualcosa di intangibile. Il vantaggio ricavato da questa operazione è enorme, perché se si dovesse decodificare un intero sistema informatico leggendo il codice riga per riga ci vorrebbero anni, senza peraltro riuscire a comprendere nulla. La modalità di rappresentazione che abbiamo adottato non consente solo di analizzare il sistema, ma anche di interagire modificandolo, muovendone gli elementi: è proprio partendo da questa nuova visualizzazione che oggi stiamo lavorando per creare una nuova modalità per scrivere i codici. I frutti di questa ricerca li potremo vedere solo tra qualche tempo.
Osservando nel dettaglio la visualizzazione del software a città si può notare la presenza di elementi più alti, una sorta di torri, che sono in verità solo entità del sistema il cui codice sorgente è più lungo; le piattaforme su cui poggiano gli altri elementi sono dei sottosistemi. È interessante notare come nelle rappresentazioni di città non vi siano strade, perché in un sistema informatico le distanze tra elementi non esistono. Sono invece interessanti gli elementi che appaiono come ponti, perché raccontano le connessioni tra elementi e giocano un ruolo importante nella visualizzazione del sistema.
Dato che il software non ha un carattere fisico, non si può parlare di località o gravità, e quindi si possono spostare pezzi interi di un sistema enorme a un costo praticamente nullo. Naturalmente non è possibile fare lo stesso con una città, non si può spostare un quartiere da una parte all'altra. Se in una città viene abbattuta una casa, si libera uno spazio, mentre se in un sistema di software viene eliminato un pezzo non si può affermare di aver liberato un posto perché il pezzo appena tolto non ha mai avuto una locazione spaziale.
I sistemi vengono costruiti da esseri umani e, quindi, hanno la fisicità cognitiva della persona che deve mantenere e sviluppare il sistema: chi scrive il codice sorgente sa dove si trova una determinata parte del sistema, sa dove sta lavorando. È una questione mentale, non fisica, tutto si trova solo nella testa dello sviluppatore. Ecco perché è difficile mantenere sistemi così complessi, avere in mente un sistema intero. Inoltre, sono molte le persone che ci lavorano e ogni errore commesso rimarrà nel sistema e, dopo cinque o dieci anni, farà crollare l'intero sistema. Tutte le parti del software sono pesantemente interconnesse, perciò se si cambia una riga di codice si modifica potenzialmente tutto quello che è collegato in modo transitivo con quella riga, che di fatto spesso è il sistema intero. Per questo è difficile scovare il bug di un sistema, perché il bug, che è localizzato in una certa riga di codice sorgente, è lì, ma il suo bug, quello che fa crollare il sistema, certe volte può far sembrare che il sistema stia cascando a pezzi da un'altra parte.
Se si riuscisse a incanalare in un sistema di software tutte le regole che esistono dal punto di vista costruttivo e urbanistico, si potrebbero prospettare vari scenari, ovvero fare delle proiezioni di scenari futuri riguardo a quello che si potrebbe costruire. Il sistema potrebbe capire, e quindi mostrare, quale effetto verrebbe prodotto tra dieci anni se venisse realizzato un quartiere in una determinata area della città.
Si potrebbero inserire nel sistema le leggi e il sistema di regole che normano l'architettura, che sono tantissime, e il computer potrebbe calcolare quanto costerà, per esempio, mantenere un edificio in futuro, non solo la sua struttura ma anche tutte le infrastrutture che lo collegano alla città. Purtroppo, gli architetti attualmente utilizzano software che permettono di pianificare solo un edificio per volta e possono solo immaginare come potrebbe configurarsi la città dopo la sua realizzazione.
Questo è ancora più vero se il progettista di un determinato edificio non è del luogo, e quindi può non essere al corrente delle situazioni che coinvolgono indirettamente il suo progetto, che invece potrebbero essere individuate da software complessi come quelli a cui stiamo lavorando. ◑

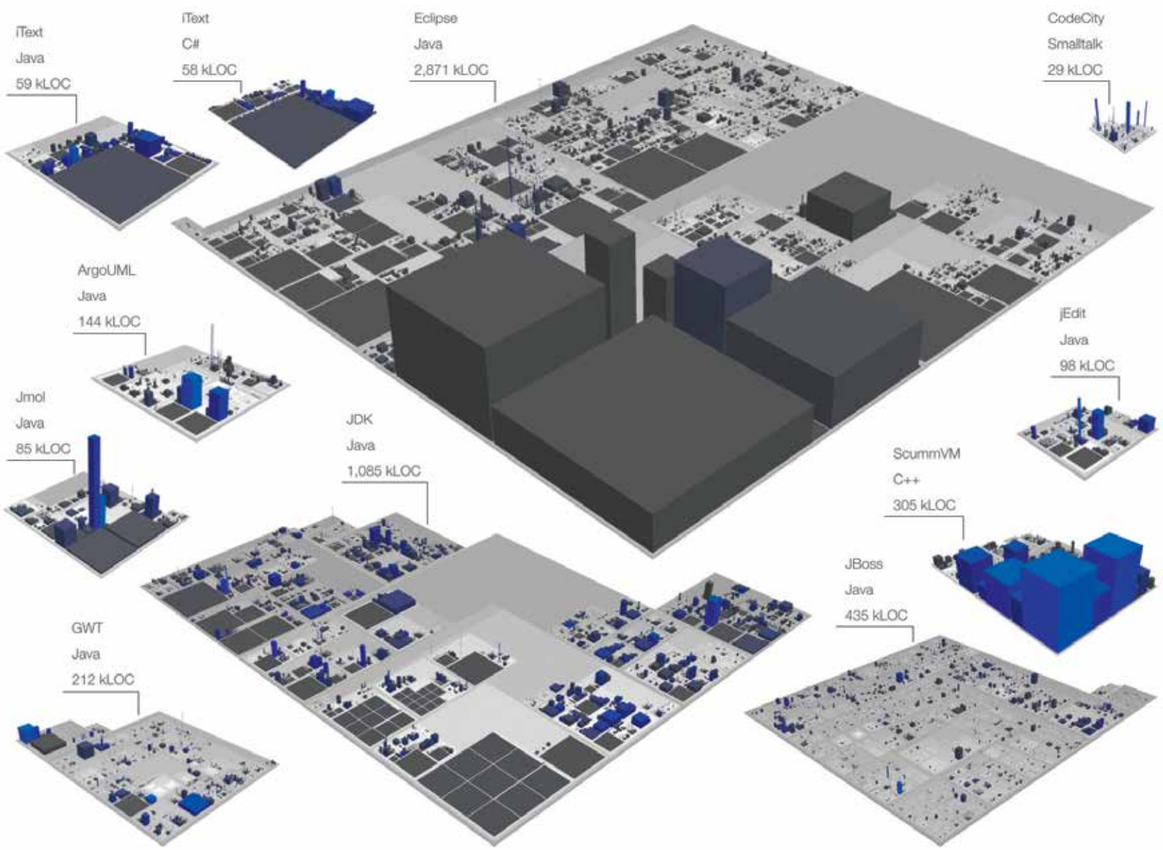


In questa pagina e in quella a fronte, in alto: configurazioni del sistema CodeCity messo a punto dal gruppo di ricerca Reveal, diretto da Michele Lanza dell'Università di Lugano. La rappresentazione tridimensionale del software consente di 'volare' attraverso il sistema e mettere in evidenza le tantissime relazioni che si instaurano tra gli elementi costruiti

▪ This page and opposite page, top: configurations of the CodeCity system developed by the Reveal research group headed by Michele Lanza at the University of Lugano. The three-dimensional representation of the software makes it possible to "fly" through the system and highlight the many connections that are established between the built elements







Sopra: visualizzazioni di sistemi informatici di dimensioni differenti che utilizzano diversi linguaggi (Java, C++, Smalltalk, CSharp). La rappresentazione a città consente di semplificarne la lettura e di comprendere velocemente i problemi che si verificano: ogni sub sistema diventa un quartiere, in cima al quale ci sono edifici che sono il file del sistema

▪ Above: visualisations of information technology programmes of different dimensions that use different languages (Java, C++, Smalltalk, CSharp). Representation in the shape of a city makes them easier to read and allows for quicker understanding of any problems that arise. Each subsystem is a city district, on top of which are buildings that represent the files of the system

THE CITY OF SOFTWARE

The understanding and analysis of information technology systems is at the heart of the work of Reveal, my research group at the University of Lugano. In our attempt to render the task less formidable, we tried creating a three-dimensional representation of what we could call the “architecture of software”. Using the city as a metaphor to represent the complex system of a software programme has allowed us to highlight the anomalies on which some of its parts rest, those that threaten the stability of the “building” or the “city” as a whole. When you develop part of a software programme, you do not perceive the system as a totality – you work on each file of text as if it were an individual building, without seeing the whole city. At most, the connections with the other buildings are visible, but nothing more. A programme is written using lines of code, so each entity, each cuboid, represents a “class”. A system can have millions of lines of code, which in turn create tens of thousands of classes. Each building represents a class, so by looking at these representations it is possible to understand the complexity of a programme, which is in itself intangible. The advantage of this procedure is enormous, because if you had to read an entire information technology system by reading the code line by line, it would take years, and you would not understand anything at all. The method of representation that we have adopted does not just allow us to analyse a system, but also to interact by modifying it, by moving its components. We have taken this new visualisation as a starting point and are now working on a new way of writing code. The results of this research will become clear in a while. Detailed observation of our city-based visualisation of software reveals the presence of taller components – towers, in a way – which are in fact only those components of the system where the source code is longer. The platforms on which the other components rest are the subsystems. It is interesting to note that there are no streets in the representation of the city, because there are no distances between the components in an information technology system. However, there are fascinating bridge-like elements that express the connections between components and play an important role in the visualisation of the system.

Seeing that software has no physical character, locality or gravity, you can change the position of entire sections of an enormous system at practically no cost. Of course, it is not possible to do the same with a city. You cannot move a neighbourhood from one place to another. If you demolish a house in a city, you free up space. But if you remove a piece from a software system, you do not create a space because the piece never had a spatial location. Systems are built by human beings and so have the cognitive physicality of the person who has to maintain and develop the system. If you have written the source code, you know where to find a specific part of the system – you know where it is working. This is a mental question, not a physical one; everything resides only in the developer’s mind. In fact, it is difficult to maintain such complex systems, to keep a whole system in your head. Such complex systems are maintained by many people. Each error made will remain in the system and, after five or ten years will cause the whole system to crash. All the parts of a software programme are heavily interconnected. If you change a line of code, you are potentially changing everything that it is connected transitively with that line, which is often the entire system. For this reason, it is difficult to track down a bug in a system. The bug, which is located in a certain line of source code, is there, but its effect – making the system crash – can sometimes make it seem that the system is falling to pieces elsewhere. If we managed to channel into a software system all the construction and urban planning rules that exist, we could advance various scenarios, in other words project future scenarios about what we could construct.

The system could reveal what effect could be produced ten years after the development of a neighbourhood in a specific area of the city. We could insert into the system the laws and the many rules that regulate architecture and the computer could calculate what it would cost, for example, to maintain a building in the future – not just its structure, but also all the infrastructure connecting it to the city. Unfortunately, architects currently use software that allows for the planning of a single building alone, and they can only imagine how the city will look after the design has been realised.

This is true above all if the designer of a specific building is not from the place where it is being built, and so is not aware of situations that indirectly affect the design and that could have been identified by complex programmes like the ones we are working on. d

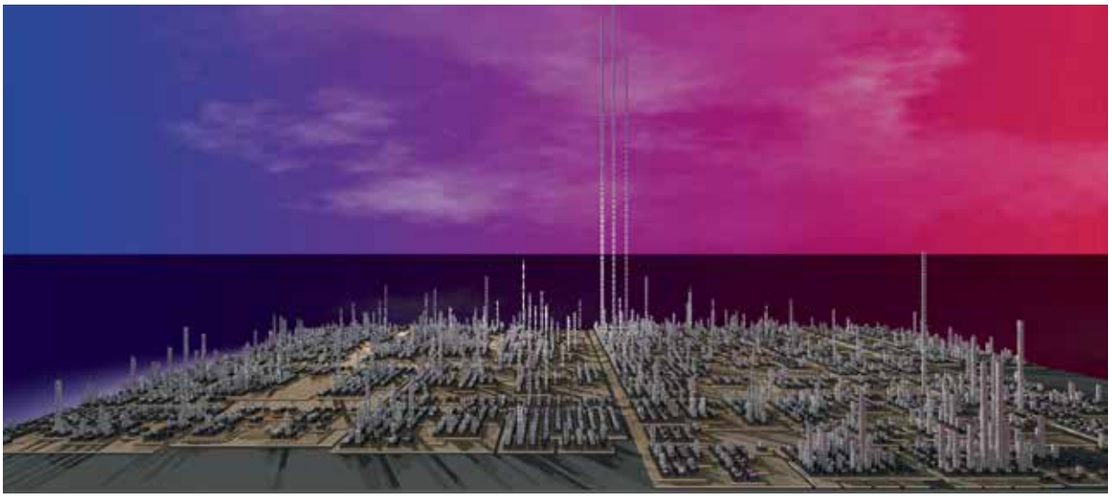
Michele Lanza

insegna alla facoltà di Informatica, che ha cofondato nel 2004 all’Università di Lugano, dove è direttore del gruppo di ricerca Reveal, che si occupa di visualizzazione, evoluzione e analisi del software.

▪ is a professor at the Faculty of Informatics, co-founded by him in 2004 at the University of Lugano, where he leads the Reveal research group in the areas of software visualisation, evolution and analytics.

Testo tratto da una conversazione tra Michele Lanza e Marcello Nasso, luglio 2015.

▪ Taken from a conversation between Michele Lanza and Marcello Nasso, July 2015.



Sopra: un software visualizzato con il CodeCity ideato dal team di ricerca di Lanza. I problemi dell’architettura del sistema, che portano alla sua esplosione, sono rappresentati dalle costruzioni più alte, quelle molto fuori scala rispetto agli altri edifici. Queste “mappe della salute” segnalano la necessità di riarchitettare il sistema

▪ Above: a software programme visualised with CodeCity, a representation system designed by the research team that Lanza directs. The problems that will cause the software to crash are represented by the tallest buildings, which are out of proportion compared to the others. These “health maps” indicate the need to redesign the programme’s architecture