Chapter 1

Applications of Systems Architecture concepts [*]

[POSTER SESSION]

Fabio Roda[†], Alberto Costa, and Leo Liberti

Abstract *Systems Architecture* is a comprehensive discipline that aims to manage complex systems which can not be operated adopting conventional techniques due to the strong heterogeneity of their sub-elements. Architects exploit several techniques which help to get this purpose and, among them, we can mention the more important ones: *abstraction*, *concretization*, *decomposition* and *integration*. We propose two examples of application of the techniques proposed by Systems Architecture to a classical trasportation problem. The underlying idea is that Systems Architecture and Optimization are complementary. These examples are still very initial at this stage of our work and involves only *abstraction* and *concretization*. They are propedeutic to the development of a unified system modelling method which is in progress and which uses all the techniques provided by Systems Architecture on a coherent pipeline.

1.1 Introduction

Systems Architecture is a comprehensive discipline which aims to manage *complex systems* which can not be operated adopting conventional techniques due to the strong heterogeneity of their sub-elements. In order to precise what we mean with the term *complex system*, we sketch a (very) brief explanation of this term.

Roda, Costa, Liberti,
LIX, École Polytechnique, 91128 Palaiseau, France.
e-mail: `{roda,costa,liberti}@lix.polytechnique.fr`

[*] Partially supported by École Polytechnique-Thales “Engineering of Complex Systems” Chair.

[†] Corresponding author.

- Firstly, we recall that INCOSE defines informally a system as:

  1.1.1 Definition (System (INCOSE))
  "an interacting combination of elements to accomplish a defined objective. These include hardware, software, firmware, people, information, techniques, facilities, services, and other support elements"

- Secondly, we remark which is the meaning of "complex" in a Systems Architecture context.
  "*At a superficial level, complex refers here to the fact that the design and the engineering of these industrial systems are incredibly complicated technical and managerial operations. . . . At a deeper level, complex industrial systems are characterized by the fact that they are resulting of a complex integration process. This means that such systems are obtained by integrating in a coherent way, that is to say assembling through well defined interfaces altogether a tremendously huge number of heterogeneous sub-systems and technologies* [9]"

The design and management of complex systems is difficult and expensive. In order to facilitate the realisation of a new system we need methods which allow to "built it up" from known components. Conversely, to make a given system easier to manage and to understand we would profit techniques which "break it down" into its sub-systems.
Systems Architecture proposes some interesting techniques: *abstraction*, *concretization*, *integration*, *decomposition* [2, 3, 9].

- *Abstraction* is the process that allows to shift from a specific level of details to a properly lower one so that we can focus on the features which are really relevant and "hide" the others.
- *Concretization* is the opposite process, namely the precisation of the analysis, which is pushed to a finer-grained level producing a model that has more parameters, variables and, generally speaking, elements.
- *Decomposition* and *Integration* refers respectively to the action of modelling the original system as a set of subsystems, and to move again from the subsytems to the system after having done some operations with the subsystems.

The analysis of a system S, according to the System Architecture approach, consists in a recursive application of these operations in order to "reduce" S to its component subsystems s_k which are assumed well known and hence to reveal the "architecture" of the system, $S = I(\alpha(s_1),\ldots,\alpha(s_n))$, where I means *integration* and $alpha$ means *abstraction*. Thus, the process consists in the identification (when possible) of a recursive structure which shows that the main system S is obtained by means of an integration of abstracted sub-systems.

This approach is (or should be) general, hence we are trying to apply it to known classic problems rooted in the field of Operations Research. Thus, we report our experience in the application of these key concepts of Systems Architecture to an optimization problem. This example is still partial at this stage of our work and involves only *abstraction* and *concretization*, however we believe that Systems Architecture and Optimization are complementary and that it is worth to investigate possible synergies.

The rest of this work is organized as follows: section 1.2 introduces the Asymmetric Capacitated Vehicle Routing Problem, section 1.3 is decicated to an example of *abstraction* and 1.4 to an example of *concretization*. In the last section we sketch some conclusions.

1.2 Asymmetric Capacitated Vehicle Routing Problem

The problem we consider belongs to the "family" of the Vehicle Routing Problem. We can figure it in the following way. A group of delivery vehicles have to service known customers which demand for a certain quantity of a specific commodity. The vehicles have a fixed capacity and commodities arrive from a common depot. The objective is to find a set of m vehicle routes (a simple circuit for each vehicle) such that: each customer is visited by a single vehicle route, the starting point and the end of each circuit is the depot, the sum of the the demands of the customers visited by each vehicle is inferior than its capacity, the cost is minimum. The fixed capacity of the vehicles, and the orientation of the roads determines the specific variant, called Asymmetric Capacitated Vehicle Routing Problem (ACVRP) [11, 1], that we consider.

1.2.1 First Formulation

We provide a (first) formulation of the ACVRP, introducing some formal elements.

Let G = (V,A) be a complete directed graph (modelling a road network) with vertex set $V = 0,\dots,n$ and arc set A. Every arc has an associated cost c_{ij}. Vertices j represent customers. Each customer requires a quantity d_j. The vertex v_0 is the depot. A single commodity is to be delivered to each customer $i \in N = \{1,\dots,n\}$ from a central depot using m independent delivery vehicles of identical capacity C. A binary variable x is associated with each arc of the graph G. Moreover, $\sigma(S)$ is the minimum number of vehicles necessary to satisfy all clients, $\forall\, S \subseteq V \setminus \{v_0\}$.

1. **Decision variables**

$$\forall (i,j) \in A\ x_{ij} = \begin{cases} 1 \text{ if the arc } (i,j) \text{ is in the solution} \\ 0 \text{ otherwise} \end{cases} \tag{1.1}$$

2. **Objective function**.

$$z = \min_x \sum_{i\in V} \sum_{j\in V} c_{ij} x_{ij}$$

3. **Constraints**.

$$\forall j \in V \setminus \{v_0\} \quad \sum_{i \in V} x_{ij} = 1 \tag{1.2}$$

$$\forall i \in V \setminus \{v_0\} \quad \sum_{j \in V} x_{ij} = 1 \tag{1.3}$$

$$\sum_{i \in V} x_{iv_0} = m \tag{1.4}$$

$$\sum_{j \in V} x_{v_0 j} = m \tag{1.5}$$

$$\forall S \subseteq V \setminus \{v_0\}, \quad S \neq \emptyset, \quad \sum_{i \notin S} \sum_{i \in S} x_{ij} \geq \sigma(S) \tag{1.6}$$

$$\forall i, j \in V \quad x_{ij} \in \{0,1\} \tag{1.7}$$

The model provided is the section above is the basis for the application of two of the operations suggested by System Architecture: *abstraction* and *concretization*. We applay the first one to the graph which represents the instance of the problem and the second one to the model itself.

1.3 Abstraction

If the road network and the representing graph are huge algorithms may fail to treat them efficiently. A reduction of the graph size would help. In order to achive this objective we should hide subsets of edges or vertices which represent information which is not necessary. We reduce the level of detail, dropping only the ones which are not useful. This operation is a realization of *abstraction* which is operated on the graph. This idea is already present in literature, but there is not a common framework.

- Holte et al. [8] applay abstraction to graphs which represent problem spaces. "*An abstraction of a problem space P is a mapping from P to some other problem space, P. . . . The abstract solution will not usually be a valid solution in the original space. The abstract solution serves as a skeleton for the final solution.*"
- Botea and Muller [4] propose, in the context of path-finding on grid-based maps, a method which "*abstracts a map into linked local clusters* [which] *return a complete path of sub-problems. The first sub-problem can be solved, giving . . . first few moves along the path. At* [this] *level clusters are traversed in a single big step.*"
- Bulitko et al. [5] refine the work of Botea and Muller and propose "*a clique-based abstraction mechanism*" and show that "*general clique computation is NP-*

complete, finding cliques in two-dimensional grid-based search graphs can be done efficiently".
- Harry and Lindquist [7] propose " *k-clique minimization with centrality reduction,* [which] *attempts to transform a complex graph into its abstract components, creating new graphs that are representative of the originals but of a structure which is* [simpler]"

Despite of this utilisation of the term *abstraction of the graph* in the mentioned works, there is not a shared, fully developed formalization of this concepts . We propose some initial distinctions.

1.3.1 Abstraction as graph contraction

The problem of graph abstraction reduces to the research of a different graph which shares some features with the original one. In others words, we look for two graphs which *match* one each other.

Graph Matching is a classical problem in Graph Theory and in Optimization. We can distinguish basically two kinds of *matching* between graphs:

- exact matching
- inexact matching

The *exact graph matching* requires that there is a perfect correspondence between the graphs. This is obtained by means of an isomorphism.

Definition 1.1.
Given two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, with $|V_1| = |V_2|$, if exists a one-to-one mapping $I : V_1 \to V_2$ such that $(u,v) \in E_2 \;\; iff \;\; (f(u), f(v)) \in E_1$, then I is called an *isomorphism* and G_1 and G_2 are called *isomorphic*.

When it is not possible to find a perfect correspondence we can look for graphs which correspond "inexacly". This happens, for example, when the number of vertices and edges are different or when we consider attributed graphs (the same number of vertices and edges does not assure always that there is an isomorphism). Basically, we do not look for a perfect match but for the best possible one, assuming that between the two graphs there is a difference. An interesting way to measure this difference is the edit distance [6], namely the cost of the minimum number of operations (insert, delete, and relabel the vertices and edges) that are necessary to transform a graph in the other one (we assume that each operation has a cost).

Thus, we present below the operations on a graph which are relevant (in our opinion) in order to reduce the complexity of a graph saving some kind of matching (even if inexact) with the original one and to formalize the idea of "graph abstraction".

1. For two graphs $G = (V, E)$ and $G' = (V', E')$, we have $G \subseteq G'$ if $V \subseteq V'$ and $E \subseteq E'$; G is a *subgraph* of G'.

2. For a graph $G = (V,E)$ and $U \subseteq V$ let $E[U] = \{\{u,v\} \in E \mid u,v \in U\}$; $G[U] = (U,E[U])$ is the subgraph of G *induced* by U.
3. For $v \in V(G)$ let $\delta(v) = \{u \in V \mid \{u,v\} \in E\}$ be the *star* around v and $\bar{\delta}(v) = \{\{u,v\} \mid \{u,v\} \in E\}$ be the *cut* of v.
4. Stars and cuts can be extended to sets of vertices: for $U \subseteq V$ let $\delta(U) = \{v \in V \mid \exists u \in U\ (\{u,v\} \in E)$ and $\bar{\delta}(U) = \{\{u,v\} \in E \mid u \in U \wedge v \in V \smallsetminus U\}$.
5. $\forall \{u,v\} \in E(G)$ let $G - \{u,v\} = (V(G), E(G)/\ \{u,v\})$ the *edge deletion* of the edge $\{u,v\}$ in the graph G.
6. For $v \in V(G)$, let $G - v = (V(G)/\ \{v\}, E(G)/\ \bar{\delta}(v))$ the *vertex deletion* of the vertex v in the graph G.
7. $\forall\ U \subseteq V(G)$ let $G - U = G[V(G)/\ U]$ the *subgraph deletion* of the subgraph U in the graph G.
8. $\forall\ U \subseteq V(G)$ let $G/U = (V(G-U) \cup \{u\}, E(G-U) \cup \{\{u,v\} | v \in V(G-U) \wedge \exists w \in U(\{v,w\} \in E(G))\})$ the *contraction* of G relating to U.

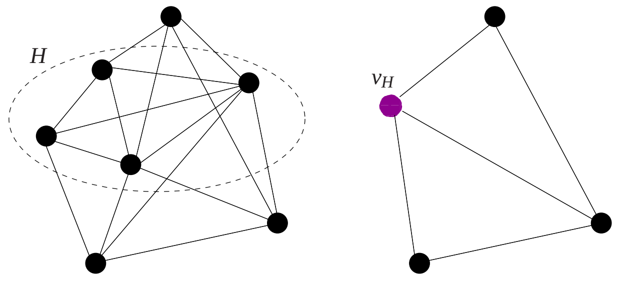
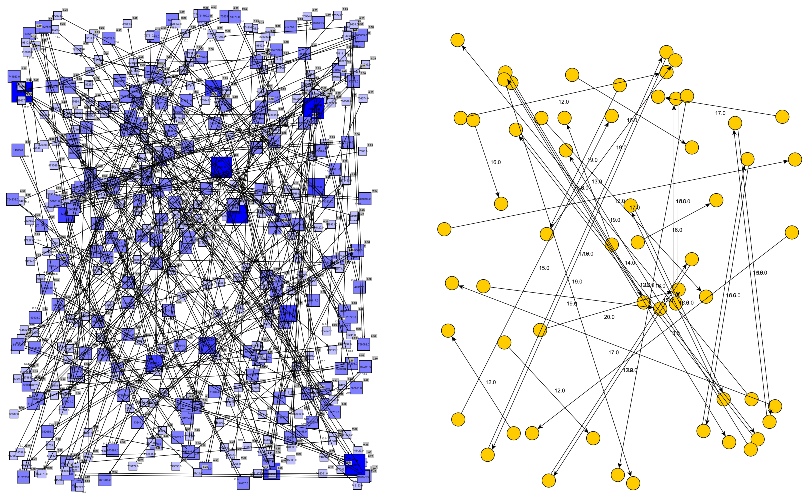


Fig. 1.1 The subgraph H is contracted to v_H, leaving the minor on the right.

Basically, when we contract a graph, we look at it “modulo” one of its subgraphs $H \subseteq G$. The results of the contraction is a *minor* G' of G where $V(G') = V(G) \smallsetminus V(H) \cup \{v_H\}$ and $E(G')$ is like $E(G)$ with $\bar{\delta}(V(H))$ replaced by a cut $\bar{\delta}(v_H)$. More precisely, if $\{u,v\} \in \bar{\delta}(V(H))$ with $v \in V(H)$, then $\{u,v\}$ is replaced by $\{u,v_H\}$.
In Fig. 1.1 we show a simple example of graph contraction.

We think that we could formalize the idea of abstraction by means of the ideas of (inexact) matching and edit distance, exploiting special operations such as the “graph contraction”.
Thus, we are working to fully develop this idea and to understand both the computational gain and the optimality loss we get from this method.
We plan to solve the ACVRP for graph istances representing a road networks which “inexactly match” (or, may say, are abstracted from the initial one) and to evaluate the outcome.
In fig. 1.2 we can sketch the outcome of the contraction of a portion of the road network of the metropolitan area of Paris.



(a) Complete graph (b) Abstracted graph

Fig. 1.2 The road network of Paris

1.4 Concretization

The second idea that we borrow from System Architecture is the *concretization*. We apply this technique to the model of the ACVRP proposed in a previous section, in order to get a different formulation. Despite of the fact that they both provide a proper model, the second formulation goes deeper in detail and represents a *concretization* of the first one. Different models of the same problem are not unusual in Operations Research. “*It is well known that several different formulations may share the same numerical properties (feasible region, optima) though some of them are easier to solve than others with respect to the most efficient available algorithms. Being able to cast the problem in the best possible formulation is therefore a crucial aspect of any solution process. When a problem with a given formulation P is cast into a different formulation Q, we say that Q is a reformulation of P.*” [10]. From this point of view, we may say that *concretization* is a reformulation technique.

1.4.1 Second Formulation

We introduce the second formulation of the ACVRP. A binary variable z is associated with each couple $(arc, vehicle)$ and a variable y with each couple $(vertex, vehicle)$.

1. **Decision variables**.

$$\forall (i,j) \in A, k \in K \; z_{ijk} = \begin{cases} 1 \text{ if the arc } (i,j) \\ \quad \text{is crossed by vehicle k} \\ 0 \text{ otherwise} \end{cases} \tag{1.8}$$

$$\forall (i) \in V, k \in K \; y_{ik} = \begin{cases} 1 \text{ if the vertex } (i) \\ \quad \text{is crossed by vehicle k} \\ 0 \text{ otherwise} \end{cases} \tag{1.9}$$

2. **Objective function**.

$$z = \min_{y,z} \sum_{i \in V} \sum_{j \in V} c_{ij} \sum_{k=1}^{m} z_{ijk}$$

3. **Constraints**.

$$\forall i \in V \setminus \{v_0\} \quad \sum_{k=1}^{m} y_{ik} = 1 \tag{1.10}$$

$$\sum_{k=1}^{m} y_{v_0 k} = m' \tag{1.11}$$

$$\forall k \in m \quad \sum_{i \in V} d_i y_{ik} \leq D \tag{1.12}$$

$$\forall i \in V, k \leq m \quad \sum_{j \in V} z_{ijk} = y_{ik} \tag{1.13}$$

$$\forall i \in V, k \leq m \quad \sum_{j \in V} z_{jik} = y_{ik} \tag{1.14}$$

$$\forall S \subseteq V \setminus \{v_0\}, h \in S, k \leq m, \quad \sum_{j \notin S} \sum_{j \in S} z_{ijz} \geq y_{hk} \tag{1.15}$$

$$\forall i \in V, k \leq m \;\; y_{ik} \in \{0,1\} \tag{1.16}$$

$$\forall i,j \in V, k \leq m \;\; z_{ijk} \in \{0,1\} \tag{1.17}$$

1.4.1.1 Relations between formulations

In this section we investigate the relations between the two formulations described above. We introduce a connection be the variables x and z.

- **Variables Connection**.

$$\forall (i,j) \in A \; x_{ij} = \sum_{k \leq m} z_{ijk} \tag{1.18}$$

Let $\mathscr{F}_1$ and $\mathscr{F}_2$ the feasible regions of ACVRP(1) and ACVRP(2) respectively. The feasible points in $\mathscr{F}_2$ can be mapped into a feasible point in $\mathscr{F}_1$ by means of a projection π which uses (1.18).

$$x_{ij} = \pi(z_{ijk}) = \sum_{k \leq m} z_{ijk} \tag{1.19}$$

This relation forces a form of *concretization*. The first formulation of ACVRP (which uses decision variables x_{ij}) does not specify which vehicle crosses an arc. It only considers if an arc is crossed by a member of the whole fleet. The second formulation models the behavior of each single vehicle (by means of decision variables z_{ijk}). The focus on the fleet rather than on a single vehicle introduces a different level of granularity. One formulation ignores (conveniently) details which are considered by the other one. We may say that ACVRP(2) is a *concretization* of ACVRP(1).

In general, we think that we could formalize the idea of concretization by means of the idea of reformulation.

1.5 Conclusions and future work

In this work we provide some basic examples of the application of techniques suggest by Systems Architecture in order to manage complex systems. Almost unusually, we apply these techniques to a classic transportation problem with the aim of exploring the synergies between systems design and optimization. We suggest the implementation of *abstraction* on graphs as *contraction* and *concretization* of a mathematical programming model as *reformulation*.

Nevertheless, this work is very initial and many aspects need further developments. We list below the work we plan to do.

- Abstraction should e able to reduce the effort required to solve big problem instances because it reduces their size. We plan to perform computational tests to measure exactly the effective gain.
- Graph contraction requires methods to identify convenient subgraphs to be contracted (for example cliques and clusters). We plan to test algorithms which can this job and to evaluate their complexity.
- Apart of the impact on the computational side of the problem, *abstraction* and *concretization* have an influence on the optimality of the solutions found. We plan to further investigate this influence in order to better define the limit of application of these techniques. The aim is to understand when abstracted/concretized model saves the optima and when they provide a relaxation of the problem. This is probably the most important point we have to clarify.

This work is propedeutic to the development of a unified system modelling method which is in progress and which uses all the techniques provided by Systems Architecture on a coherent pipeline.

References

1. Baldacci, R., Hadjiconstantinou, E., Mingozzi, A.: An exact algorithm for the capacitated vehicle routing problem based on a two-commodity network flow formulation. Oper. Res. **52**, 723–738 (2004)
2. Bliudze, S., Krob, D.: Towards a functional formalism for modelling complex industrial systems. In: P. Bourgine, F. Kepes, M. Schoenauer (eds.) European Conference on Complex Systems (2005)
3. Bliudze, S., Krob, D.: Towards a functional formalism for modelling complex industrial systems. ComPlexUs, special Issue : Complex Systems - European Conference 2005 **2**(3-4), 163–176 (2006)
4. Botea, A., Mller, M., Schaeffer, J.: Near optimal hierarchical path-finding. Journal of Game Development **1**, 7–28 (2004)
5. Bulitko, V., Sturtevant, N., Lu, J., Yau, T.: State abstraction in real-time heuristic search. Tech. rep., Journal of Artificial Intelligence Research (2006)
6. Gao, X., Xiao, B., Tao, D., Li, X.: A survey of graph edit distance. Pattern Analysis and Applications **13**, 113–129 (2010)
7. Harry, D., Lindquist, D.: Graph abstraction through centrality erosion and kclique minimization. Tech. rep., Olin College (2004)
8. Holte, R., Mkadmi, T., Zimmer, R., MacDonald, A.J.: Speeding up problem solving by abstraction: A graph oriented approach. ARTIFICIAL INTELLIGENCE **85**, 321–361 (1996)
9. Krob, D.: Modelling of complex software systems: A reasoned overview. In: FORTE, pp. 1–22 (2006)
10. Liberti, L.: Reformulations in mathematical programming: Definitions. In: CTW08 Proceedings, pp. 66–70. G. Righini (ed.), New York, NY, USA (2008)
11. Toth, P., Vigo, D. (eds.): The vehicle routing problem. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA (2001)