tkLayout: a design tool for innovative silicon tracking detectors

G. Bianchi

European Organization for Nuclear Research, CERN, Genève 23, CH-1211, Switzerland
E-mail: giovanni.bianchi@cern.ch

ABSTRACT: A new CMS tracker is scheduled to become operational for the LHC Phase 2 upgrade in the early 2020’s. tkLayout is a software package developed to create 3d models for the design of the CMS tracker and to evaluate its fundamental performance figures. The new tracker will have to cope with much higher luminosity conditions, resulting in increased track density, harsher radiation exposure and, especially, much higher data acquisition bandwidth, such that equipping the tracker with triggering capabilities is envisaged.

The design of an innovative detector involves deciding on an architecture offering the best trade-off among many figures of merit, such as tracking resolution, power dissipation, bandwidth, cost and so on. Quantitatively evaluating these figures of merit as early as possible in the design phase is of capital importance and it is best done with the aid of software models.

tkLayout is a flexible modeling tool: new performance estimates and support for different detector geometries can be quickly added, thanks to its modular structure. Besides, the software executes very quickly (about two minutes), so that many possible architectural variations can be rapidly modeled and compared, to help in the choice of a viable detector layout and then to optimize it.

A tracker geometry is generated from simple configuration files, defining the module types, layout and materials. Support structures are automatically added and services routed to provide a realistic tracker description. The tracker geometries thus generated can be exported to the standard CMS simulation framework (CMSSW) for full Monte Carlo studies.

tkLayout has proven essential in giving guidance to CMS in studying different detector layouts and exploring the feasibility of innovative solutions for tracking detectors, in terms of design, performance and projected costs. This tool has been one of the keys to making important design

---

1On behalf of the CMS collaboration.
decisions for over five years now and has also enabled project engineers and simulation experts to focus their efforts on other important or specific issues.

Even if tkLayout was designed for the CMS tracker upgrade project, its flexibility makes it experiment-agnostic, so that it could be easily adapted to model other tracking detectors.

The technology behind tkLayout is presented, as well as some of the results obtained in the context of the CMS silicon tracker design studies.

**KEYWORDS**: Detector design and construction technologies and materials; Particle tracking detectors (Solid-state detectors); Trigger detectors
### Contents

1 Overview 1
   1.1 Towards HL-LHC 1
   1.2 The tkLayout tool 2
   1.3 tkLayout vs. full Monte Carlo simulation 2

2 Tracker modeling with tkLayout 3
   2.1 Module geometry 3
   2.2 Trigger window and sensor spacing dimensioning 3
   2.3 Module positioning 4
   2.4 Material 7
   2.5 Tracking performance 8

3 Conclusions 10

---

## 1 Overview

### 1.1 Towards HL-LHC

Over the next ten years, the Large Hadron Collider (LHC) [1] and its experiments have been earmarked for three technical stops (Long Shutdowns, or LS, 1, 2 and 3) during which maintenance and upgrade will be carried out to enable them to operate at higher performance (see table 1). The CMS tracker [2, 3] is a barrel-shaped detector sitting in the inner part of the Compact Muon Solenoid (CMS) experiment [4], which features detecting units (modules) based on silicon sensors. Data from all the modules is then correlated so that a particle’s track, as it leaves the tracker, can be reconstructed. The CMS tracker is foreseen to last in the current configuration till 2022.

After LS3, LHC will start to operate under High-Luminosity regime (HL-LHC) thus entailing an instantaneous luminosity six times as high as in 2012 ($5 \times 10^{34} \text{cm}^{-2}\text{s}^{-1}$). This last scenario represents a challenge for the CMS detector due to the increased radiation dose [5] and amount of data. Therefore, a complete redesign of the tracker is envisaged (Phase 2 tracker). In particular, the Phase 2 CMS tracker is planned to be instrumented with Level-1 (L1) triggering capabilities, to aid the triggering system in coping with the increased event rate. However, adding functionality must not cause a loss in the tracking performance, due to the additional material. It is very important, therefore, to find the right trade-off between a multitude of parameters, such as the aforementioned tracking performance, the triggering performance, material, cost, and so on. This search for the optimal trade-off goes through a thorough analysis of several different tracker architectures and the variations on each architecture. It is evident that the size of the task calls for a computer-aided approach.
Table 1. The evolution of LHC performance through LS1 (2013–14), LS2 (2018), LS3 (2021).

|                  | ≤ 2012 | 2015–2017 | 2019–2021 | ≥ 2023 |
|------------------|--------|-----------|-----------|--------|
| $E_{\text{CM}}$ [TeV] | 8      | 13–14     | 14        | 14     |
| $L$ [cm$^{-2}$s$^{-1}$] | $8 \times 10^{33}$ | $\leq 2 \times 10^{34}$ | $\leq 2.5 \times 10^{34}$ | $\simeq 5 \times 10^{34}$ |
| $\int L$ [fb$^{-1}$] | $\leq 30$ | $\leq 100–150$ | $\leq 100–150$ | $\leq 3000$ |

1.2 The tkLayout tool

tkLayout is a software package written in C++11, developed specifically to perform design studies of the new Phase 2 upgrade of the world’s largest silicon tracker. Currently in its fifth year of active development, it started out as a simple calculation tool and over the years evolved into a comprehensive modeling and performance analysis solution, expanding its feature set to take on each new design challenge as it shaped up.

tkLayout creates a 3d model of the tracker from simple configuration files. Material is then added to the model automatically, but according to user-defined rules. Once an architecture has been modeled, its performance can be estimated in terms of, for example, tracking and triggering performance, trigger data rate or power dissipation after irradiation. The analyses are based on \textit{a-priori} calculations or on \textit{input parameterizations}, such as distributions of track properties coming from the current CMS tracker. For ease of consultation by tracker designers and engineers, the output of the tool is a mini-website.

Finally, the internal geometry and the associated material model can be exported to the CMS Geant4-based Monte Carlo (MC) simulation framework CMSSW, for in-depth physics analysis.

1.3 tkLayout vs. full Monte Carlo simulation

tkLayout adopts a non-simulative, parameterization-driven approach to tracker performance evaluation, going in the opposite direction of event-based MC simulators. This has several advantages. Firstly, tkLayout shows the ideal performance figures of a tracker geometry, irrespective of the particular algorithm or technology used, for example, to perform track reconstruction. Once the final layout is decided, results from tkLayout can be used to benchmark different possible algorithms implemented in full simulations.

A second advantage to tkLayout’s approach is performance: by avoiding complex calculations stemming from full event-by-event simulation of particle interactions, tkLayout runs in around two minutes on a normal PC (for comparison, CMSSW may need hours to run on a computing grid).

Another important difference with respect to a general-purpose simulator is tkLayout’s \textit{specificity}. tkLayout automatically creates a model for the non-sensitive elements of a tracker geometry such as supports and services, by making assumptions specific to the architecture CMS Phase 2 tracker (for example, the way cables are routed), making it possible to quickly model complex geometries that would take weeks or months to be implemented from scratch in a general-purpose simulator.

For all these reasons, whereas a full MC simulation allows a detailed later-stage physics analysis, tkLayout works well as a design tool, allowing quick setup, modeling and comparative eval-
ual evaluation of the many possible tracker layouts and their variations, as well as the iterative layout optimization once the best candidate has been found.

2 Tracker modeling with tkLayout

tkLayout models the tracker architecture at varying levels of abstraction. Active elements (modules), supports and services are modeled and material is added automatically to them, according to a set of rules defined by the user.

2.1 Module geometry

Detector modules are the main objects in tkLayout. tkLayout supports rectangular as well as wedge-shaped modules. Modules can have one or two sensor surfaces. Each sensor is independently defined and can feature a different topology in terms of number of sensing elements in the longitudinal and transverse direction.

Dual-sensor modules are fundamental to enable triggering in the CMS tracker. On this kind of modules, specialized front-end electronics correlate the hits on the two sensors and measure their displacement to estimate the $p_T$ of the track. The lower a track’s $p_T$ the farther away the two hits will be, due to the particle’s trajectory bending in the CMS magnetic field. If the displacement between two hits is smaller than a configurable search window (trigger window), a stub is formed and sent out to the L1 trigger system. For a track with a given $p_T$, increasing or reducing the spacing between the two sensors respectively increases or reduces the hit displacement.

2.2 Trigger window and sensor spacing dimensioning

It is desirable to find a combination of trigger window and sensor spacing that achieve good triggering efficiency (meaning as many as possible high-$p_T$ tracks are detected) but also good low-$p_T$ rejection. The hit occupancy on each sensor puts a limit to the window size: if a window is made too large it will likely result in the erroneous correlation of hits belonging to different tracks, thus creating fake stubs. Additionally, sensors cannot be arbitrarily spaced apart, due to structural constraints.

The following formula relates hit displacement ($s$) with $p_T$:

$$s(p_T) = \frac{d_r}{\sqrt{a - 1}}$$

$$a = \left(\frac{2p_T}{B \cdot r \cdot 0.3 \text{GeV}/m}\right)^2$$

where $r$ is the radial coordinate of the module center and $d_r$ is the radial distance between the lower-sensor and upper-sensor hits, which is proportional to the sensor spacing.

This formula is used for the error propagation computation of the measurement of a track’s $p_T$ by the module itself. After trigger window and spacing values are assigned to modules, tkLayout can compute the probability of stubs being found as a function of $p_T$. This hinges on two hypotheses: (1) high-$p_T$ (i.e. passing the threshold) particles are mainly primaries, (2) the amount of low-$p_T$ particles detected by a module is much higher, due to combinatorics between the hit on the inner sensor and the hits within the search window on the outer sensor, and can be extracted from the predicted occupancy.
Figure 1. Distribution of particle rigidity ($p_T/Z$), fitted with 3 different sets of parameters (see table 2).

Table 2. Parameter set for the function fitting the rigidity distribution.

| Range ($p_T$) | $a_0$   | $a_1$   | $a_2$   | $a_3$   |
|---------------|---------|---------|---------|---------|
| $0.22 < x \leq 1$ | 25.2523 | −6.84183 | −12.0149 | 1.89314 |
| $1 < x \leq 4$   | 0.727638 | −1.04041 | 8.56495 | $6.52714 \times 10^{-3}$ |
| $4 < x \leq 10$  | 46.6514 | −2.88910 | −37.8716 | 0.126635 |

To quantify the average number of high-$p_T$ particles traversing a module, tkLayout uses the following function:

$$f(x) = \exp(a_0 + a_1 x + a_2 x^{-0.1} + a_3 x^2)$$

to fit a distribution of primaries coming from a MC simulation (see figure 1 and table 2).

Stubs originating from low-$p_T$ particles are instead estimated by a pure combinatorial of hits:

$$S_L = n \cdot W \cdot \nu^2,$$

where $\nu$ is the module occupancy (from a parameterization of the current tracker occupancies), $n$ is the number of sensing elements on the lower sensor and $W$ is the window size.

To help in the choice of the appropriate values for spacings and windows, tkLayout can plot stub-generation efficiency graphs in the form of turn-on curves (see figure 2) and several figures of merit such as: fake stubs rate, true stubs rate, purity (the ratio of true stubs over the total).

2.3 Module positioning

The CMS tracker is inherently symmetric around the axis formed by the beam pipe, so, to take advantage of this feature, layouts are traditionally composed of barrel and endcap sections. In a barrel section modules cover the surface of concentric cylindrical layers around the beam axis and, within a layer, they are hierarchically arranged in rods — strings of modules running along a layer’s length. An endcap section is instead composed of disks, acting as a “lid” for the barrel. Disks are composed of concentric rings of detector modules (see figures 3 and 4).
Figure 2. Example of a module turn-on curve. The curve becomes sharper as the window size ($W$) increases (here depicted $W = 1, 3, 5, 7, 9$ strips).

Figure 3. Longitudinal section of a CMS Phase 2 tracker layout modeled with tkLayout. The beam pipe runs along the $z$ axis.

If we use the cylindrical coordinates $r$ (cylindrical radius), $\phi$ (rotation angle around $z$) and $z$ (beam axis), barrel modules are alternately staggered in $r$ to avoid collisions, by requiring a small radial gap $d_1$ between modules within the same rod, and rods of the same layer are staggered by a larger $d_2$.

tkLayout, given a set of parameters, such as the aforementioned $d_1$ and $d_2$, the number of layers ($n$), their length, and the inner and outer radius of the barrel ($r_m$ and $r_M$), creates and positions the modules automatically.

Given a certain radius $r$, the theoretical (non-integer) number of rods in $\phi$ to guarantee complete coverage can be calculated as $N(r) = 2 \cdot \lceil \pi / \alpha(r) \rceil$, where $\alpha(r)$ is the rods’ $\phi$-aperture, depending on the layer radius, the inter-rod radial stagger $d_2$, the module width and the rod $\phi$-overlap $d_3$. The first layer is always placed at $r_m$ with $N(r_m)$ rods, while, similarly, the last layer is at $r_M$ with $N(r_M)$ rods. The radius and number of rods of the middle layers is instead determined according to one of the following heuristics, selected by the user:

1. *Enlarge* causes layer $k = \{1, 2, ..., n - 1\}$ to be tentatively placed at radius $r_k = r_m + k \cdot (r_M - r_m)/n$, with $N(r_k)$ rods, then enlarged until the overlap between rods decreases to $d_3$;
To prevent particles from finding a gap between two adjacent modules from which to escape the barrel undetected, tkLayout positions the modules hermetically in $z$ to help software module alignment (see figure 5).

2. **Shrink** uses the same initial layer placement as Enlarge, but the number of rods is calculated as $N^* = 2 \cdot \left( \pi / \alpha(r) \right)$, so the layer radius is shrunk until the rods overlap by $d_3$;

3. **Fixed** forces the layer to be built at a user-specified radius;

4. **Auto** chooses Enlarge or Shrink on the basis of which is closer to the layer’s tentative radius.

To prevent particles from finding a gap between two adjacent modules from which to escape the barrel undetected, tkLayout positions the modules hermetically in $z$ so that no gap would be seen from $(0, 0, \pm \Delta z)$, where $\Delta z$ is the expected variance of the $z$ coordinate of the primary p-p interaction points. Additionally, tkLayout imposes that an overlap of at least $d_3$ be seen from the origin $(0, 0, 0)$. This is to make sure that two adjacent modules both register a hit on their overlapping edges, to help software module alignment (see figure 5).

Endcap modules can be either rectangular or wedge-shaped and are staggered alternately in $z$ within their ring by a small gap $d_1$. Rings are staggered in $z$ between each other by a larger gap $d_2$ and they are placed with an overlap in $r$ calculated with the same criterion as the $z$-overlap for barrel modules.
Modules can be rotated around their longitudinal axis by a skew angle $\alpha$. This effectively increases the resolution seen by tracks by a factor $\cos(\alpha)$, but, on the other hand, reduces the $\phi$-aperture by the same amount.

tkLayout also supports rotating barrel modules around their transverse axis by a tilt angle $\beta$, in a layout called “tilted barrel” (see figure 6), where barrel modules are increasingly tilted the further they are on the $z$ axis (up to $\beta = \pi/2$ when they effectively become endcap modules). The advantage of the tilted layout is that the modules’ aperture in $\theta$ (the angle relative to the beam axis) is increased, so the number of modules to cover the same volume is reduced, to the benefit of material amount and cost.

2.4 Material

For efficiency reasons, tkLayout models a module’s components (for example sensors, power converters or support mechanics) only in terms of the amount of material with which they contribute to the module’s volumetric material budget and never as true geometric objects placed on the module. Therefore, each component is defined only by its material composition ($1\text{ g Copper} + 0.5\text{ g PVC} \ldots$).

Furthermore, services running inside the detector to and from the end-flange (like power cables, cooling pipes or optical fibers) are also modeled as material assigned to the modules’ volumes, as these hermetically cover the detector volume. The actual assigned material depends on its position in the supporting structure: for example, for a barrel module on a given layer the amount of material $M$ is:

$$M(n) = n \cdot \sum_i A_i + \sum_j B_j$$

where $n$ is the module position on the rod ($n = 0$ for modules at $z = 0$), $A_i$ are the materials forming the services running through the rod and $B_j$ those forming a module’s components. This parameterization takes into account the accumulation of running services towards the end of a barrel layer: a module next to the end-flange will contain in its volume as many service lines as the number of modules before itself.
Figure 7. Distribution of material in an \((r,z)\) section of a tracker model. The accumulation of material along the barrel layers (a) is indicated by the big arrow. The accumulation of routed services (b) is shown by the line becoming increasingly dark.

A similar computation is done for endcap modules across rings, with a scaling factor applied to \(A_i\), such that the material amount for a module on ring \(n\) \((n = 0\) for the innermost ring\) is:

\[
M(n) = \sum_{k=0}^{n} \frac{N_k}{N_n} \cdot \sum_i A_i + \sum_j B_j
\]

with \(N_k\) being the number of modules on ring \(k\). This factors in the reduction in service density as they spread outward from one concentric ring to the next.

Additionally, each material \(A_i\) and \(B_j\) can be flagged as \textit{local} if it only contributes to the material inside the layers or disks (like support mechanics or silicon sensors) or \textit{exiting} if it implies the presence of other services coming from outside the detector (like cooling pipes or power lines).

Once the local material for modules and services is in place, additional volumes representing exiting services and their support structures are created. Materials for exiting services are automatically created depending on the amount of module material previously flagged as such, with several configurable conversion rules (for example, many small cooling pipes running inside a rod will join through a manifold into fewer larger exiting pipes). Part of these services are automatically routed up to the edge of the tracking volume with the same material accumulation mechanism described for services inside the detecting volumes (for example, the cooling pipes will be propagated, while the material of their manifolds will not). Furthermore, an additional fixed amount of material can be added to the service volumes to represent specific objects. The routing procedure is shown in figure 7.

Finally, the material assignment results in several summary plots, such as distribution of material, photon conversion and nuclear interaction probabilities, etc.

2.5 Tracking performance

The tracking performance analysis is the most important performance metric to qualify a tracker layout. It offers several figures of merit to estimate the accuracy achieved by a layout in reconstructing a track’s parameters: transverse momentum \(p_T\), longitudinal impact parameter \((z_0)\), transverse impact parameter \((d_0)\), polar angle \((\theta)\) and azimuthal angle \((\phi)\). \(p_T\) is expressed also in terms of the particle curvature radius \((R)\): 

\[
p_T = B \cdot R \cdot 0.3 \text{ GeV/}m.
\]
In tkLayout the tracking resolution is calculated taking into account the precision of the measurement points and the multiple scattering. The method is based on treating a particle’s trajectory in the CMS magnetic field independently as a circle in the \((r, \phi)\) plane and a straight line in the \((r, z)\) plane. This approximation was proven to be valid — a-posteriori — by modeling the current tracker and comparing it with the results from a full MC simulation \([6]\). These calculations will closely follow those published by Karimäki \([7, 8]\) with two main differences: first multiple scattering is taken into account here, while the author explicitly neglects it in the cited article; second we are only interested in the general solution of the problem, as tkLayout performs the computation for each particular case.

In the \((r, \phi)\) plane, tracks are fitted with a circle, yielding a measurement \(\tilde{C}_i\) of the track parameters \(\alpha_i = \{\phi_0, d_0, \rho\}\) \((\rho = 1/R)\) from the set of \(N\) measured points \(P_i = (r_i, \phi)\).

If \(r_i\) is the module radial position, the uncertainty \(\epsilon_i\) of the \(i\)-th measurement point is:

\[
\epsilon_i = \frac{1}{2} \rho r_i^2 - (1 + \rho d_0) r_i \sin(\phi_i - \phi_0) + \frac{1}{2} \rho d_0^2 + d_0
\]

tkLayout builds the covariance matrix \(U_{ij} = \text{cov}[\tilde{C}_i, \tilde{C}_j]\). The uncertainties of measurement are \(\sigma(\tilde{C}_i) = \sqrt{U_{ii}}\).

If we rotate the reference frame by \(\phi_0\), the coordinate measured by the modules becomes \(x\). Furthermore, if we assume the sensor radial positions \(r_i\) to be known and errorless and, for high-\(p_T\) tracks, \(\rho \cdot d_0 \ll 1\), we have that: \(d\epsilon_i \simeq dx_i\) (approximation valid only for non-tilted and non-skewed modules, see below for other cases).

It is assumed here that the best fit of the trajectory to the measured points will be given by minimizing \(\chi^2\): \(\chi^2 = \sum_{i,j} \epsilon_i W_i \epsilon_j\) where \(W = U^{-1}\) is the weight matrix. This is given by \(W = D^T C^{-1} D\) where \(D\) is the matrix of derivatives \(D_{ij} = \partial \epsilon_i / \partial \alpha_j \simeq \partial x_i / \partial \alpha_j\) and \(C\) is the covariance matrix of the \(N\) measured points: \(C_{ij} = [\epsilon_i, \epsilon_j] \simeq [x_i, x_j]\).

Since tkLayout treats multiple scattering as a measurement error, considering it as a deviation from the track’s ideal trajectory, \(C\) can be written as a sum of two components: \(C = C^M + C^R\), where \(C^M\) is the covariance matrix of the multiple scattering and \(C^R_{ij} = \delta_{ij} \sigma(\epsilon_i)\) is the covariance matrix due to the intrinsic resolution of the \(N\) measurement points (due to the sensor characteristics).

To evaluate \(C^M\), tkLayout generates a number of sample tracks traveling in a straight line (mimicking high-\(p_T\) tracks) and registers all the \(M\) impact points tracks encounter on their way out of the detector, including inactive surfaces such as services and support structures. Hence, a larger covariance matrix \(C^M_{mn} = \text{cov}[\tilde{x}_m, \tilde{x}_n]\) is computed. Then, the \(M - N\) lines and columns corresponding to the multiple scattering contributions from non-sensitive elements are dropped to obtain in fact \(C^M\).

Given the radial positions of interactions \(r_n = r_1, r_2, \ldots, r_M\) and the associated scattering angles \(\vartheta_n = \vartheta_1, \vartheta_2, \ldots, \vartheta_M\), the deviation from the ideal path \(\tilde{x}_n\) is \(\tilde{x}_n \simeq \sum_{i=1}^{n} (r_n - r_i) \vartheta_i\) and, since the scattering angles \(\vartheta_n\) are uncorrelated:

\[
\tilde{C}_{mn} = \langle \tilde{x}_m, \tilde{x}_n \rangle \simeq \sum_{i=1}^{\min(n,m)} (r_m - r_i)(r_n - r_i) \langle \vartheta_i^2 \rangle
\]

Finally, the expected resolution of the track parameters can be obtained from the covariance matrix \(U = [D^T C^{-1} D]^{-1}\).
In the \((r,z)\) plane, assuming again \(\rho \cdot d_0 \ll 1\), tracks are fitted with straight lines to obtain the measurements of the last two parameters \(\theta\) and \(z_0\). Thus, the track equation is:

\[
    r_i = \frac{z_i - z_0}{\text{ctg}(\theta)}
\]

The resolutions of \(\theta\) and \(z_0\) can be evaluated with the same method described above on the simpler linear fit.

The methods described above assume that \(r_i\) is known and \((x_i,y_i)\) are the only source of uncertainty, respectively, in the transverse and longitudinal planes. While this is a good approximation for regular, non-tilted, non-skewed barrel modules, this is not the case for modules with more complex placement. To tackle the latter, tkLayout performs a projection of the intrinsic measurement uncertainties on a virtual barrel module centered on the hit, such that, if \((x,y)\) are, respectively, the local transverse and longitudinal coordinates on the original module and \((a,b)\), respectively, the transverse and longitudinal coordinates on the virtual module, the variances \(\sigma^2_a\) and \(\sigma^2_b\) of the projected measurements are:

\[
    \sigma^2_a = \left(\sin(\alpha) \cos(\beta) + \cos(\alpha)\right)^2 \sigma^2_x + \left(B \cdot \sin(\beta)\right)^2 \sigma^2_y
\]

\[
    \sigma^2_b = \left((D \cdot \cos(\beta) + \sin(\beta)) \sin(\alpha)\right)^2 \sigma^2_x + \left(D \cdot \sin(\beta) + \cos(\beta)\right)^2 \sigma^2_y
\]

with \(B = A/\sqrt{1 - A^2}\), \(D = \text{ctg}(\theta)/\sqrt{1 - A^2}\) and \(A = r_i / (2R)\). This hinges on the assumption that the uncertainties on \(x_i\) and \(y_i\) can be treated as uncorrelated. It can be demonstrated that this assumption holds true for the range of track \(p_T\)'s and module geometries we are interested in. Hence, the measurement covariance matrix can be built as if the module were a regular barrel module and the method described above for them can be used for modules placed at any tilt and/or skew angles.

### 3 Conclusions

Designing a new tracker is a very complex endeavor. It requires making quantitative comparisons between many different architectures, in terms of their tracking and triggering performance, material amount, cost, etc. It involves seeking the right trade-off between different parameters and optimizing the final architecture. tkLayout was developed specifically to provide tracker designers with an easy and rapid way to evaluate the performance of layouts. tkLayout executes very quickly, so that performance-tuning a layout can be done in a trial-and-error, incremental manner. The accuracy of the a-priori, non-simulative approach the tool adopts has been proven to be within 20% with respect to a full MC simulation [6].

tkLayout was built to design the CMS tracker, but, since the vast majority of silicon trackers have similar defining characteristics, support for other trackers could easily be added.

Over the last five years, tkLayout has been a key tool in the design of the Phase 2 CMS tracker and geometries generated with it have made their way into the CMSSW simulation framework for physics analysis.
References

[1] LHC Study Group, The Large Hadron Collider: conceptual design, CERN-AC-95-05 (1995).
[2] CMS collaboration, The CMS tracker system project: technical design report, CERN-LHCC-98-006 (1998).
[3] CMS collaboration, The CMS tracker: addendum to the technical design report, CERN-LHCC-2000-016 (2000).
[4] CMS collaboration, CMS physics: technical design report, CERN-LHCC-2006-001 (2006).
[5] F. Hartmann, Evolution of silicon sensor technology in particle physics, Springer (2009).
[6] S. Mersi, D. Abbaneo, N. De Maio and G. Hall, Software package for the characterization of tracker layouts, in Astroparticle, particle, space physics and detectors for physics applications, vol. 7, World Scientific Publishing (2011).
[7] V. Karimäki, Effective circle fitting for particle trajectories, Nucl. Instrum. Meth. A 305 (1991) 187.
[8] V. Karimäki, Explicit covariance matrix for particle measurement precision, Nucl. Instrum. Meth. A 410 (1998) 284.