iPatRec Design

- Design Document exists.
- continuous state of evolution (fortran -> OO)
- order of modification determined by priority to improve speed/space/stability

- input (Detector description / Data input)
- track finding (Combinatorials / Dynamic storage)

- present design contains:

- actual clases and methods
- for design presentation classes and methods
- fortran / C++ routines

- Design Document consists of :

- Introduction (useage / algorithms)
- Requirements (existing + additional)
- OIGs (overall / specific)
- Classes and public Methods (from SNIFF , not anotated)

- iPatRec Home Page exists

The iPatRec Algorithm

- build detector description.

- build Module Directory to contain geometrical and status parameters of modules.

- a unique module index allows access to module parameters.

- seeds and roads.

- use external and internal seeds to construct roads.

- loop over roads, not necessarily 1:1 match with seeds.

- hits in road.

- find modules in road.

- input hits, configure and cluster them.

- make space points.

- store hits and space points with associated modules.

- track finding.

- complex algorithn, needs to be efficient and fast.

- use outer space points to initiate track.

- extrapolate and interpolate to pick up other hits.

- quality control at all stages.
- very large combinatorials.
- finally add trt hits to track.
- primary and secondary tracks.
- resolve ambiguities.

output.

- tracks, hits, fits, and statistics stored.

Requirements

- Global_Constraints.

- defaults and standards.
- compilers and execution.

- Global_System_Properties.

- fault rate and processing speed.

- iPatRec_Functional_Requirements.

2.4 Req: iPatRec_Functional_Requirements

iPatRec is a reconstruction program for the Atlas inner detector. It finds tracks in roads and outputs the tracks with their associated hits.

2.4.1 Find primary tracks - IPATREC shall find Inner Detector primary tracks and produce a list of them. Primary tracks are defined as tracks with a sufficient number of hits within the silicon pixel detectors, and have a high probability of originating within the 'beam spot'.

Satisfied by Class Operation look_for_primaries

2.4.2 Find secondary tracks - IPATREC shall find Inner Detector secondary tracks and produce a list of them. Secondary tracks are defined as tracks with insufficient hits within the silicon pixel detectors to be classed as primaries and are likely to have originated from a secondary interaction. These are produced by bremsstrahlung as well as hadronic interactions.

Satisfied by Class Operation look_for_secondaries

2.4.3 List of hits - IPATREC shall provide a list of hits used for each reconstructed track.

Satisfied by Class HitOnTrack

2.4.4 Track parameters - IPATREC shall produce a list of the fitted track parameters.

Satisfied by Class Track

2.4.5 Track errors - IPATREC shall produce a covariance matrix for the fitted track parameters.

Satisfied by Class Track

2.4.6 Hit clustering - IPATREC shall be able to cluster hits from both planar detectors and in 2 dimensions from pixel detectors. SATISFIED BY the class Clusterer. This is currently done in the fortran routine irBinary.

2.4.7 Make roads - IPATREC shall be able to construct roads, within which to look for tracks, from external and internal seeds. SATISFIED BY the class Road. This is currently done in routines, isecal, ismuons, etc. These are written in C++, but are not OO.

2.4.8 Hits in road - IPATREC shall be able to find all hits and their corresponding detector modules within a road.

Satisfied by Class HitContainer

2.4.9 Truth values - IPATREC shall be able to make comparisons with truth values from the simulation. SATISFIED. Comparisons are currently made in histograms showing pulls for fitted track parameters.

2.4.10 Detector description -IPATREC shall be able to obtain detector geometry and status information from a unique detector description source.

Satisfied by Class ModuleDirectory

2.4.11 Realistic operating conditions - IPATREC shall be able to work with realistic operating conditions. These must include the addition to the normal simulated input data of a reasonable level of: global random noise, noisy channels, reduced efficiency, dead readout chips, and dead channels.

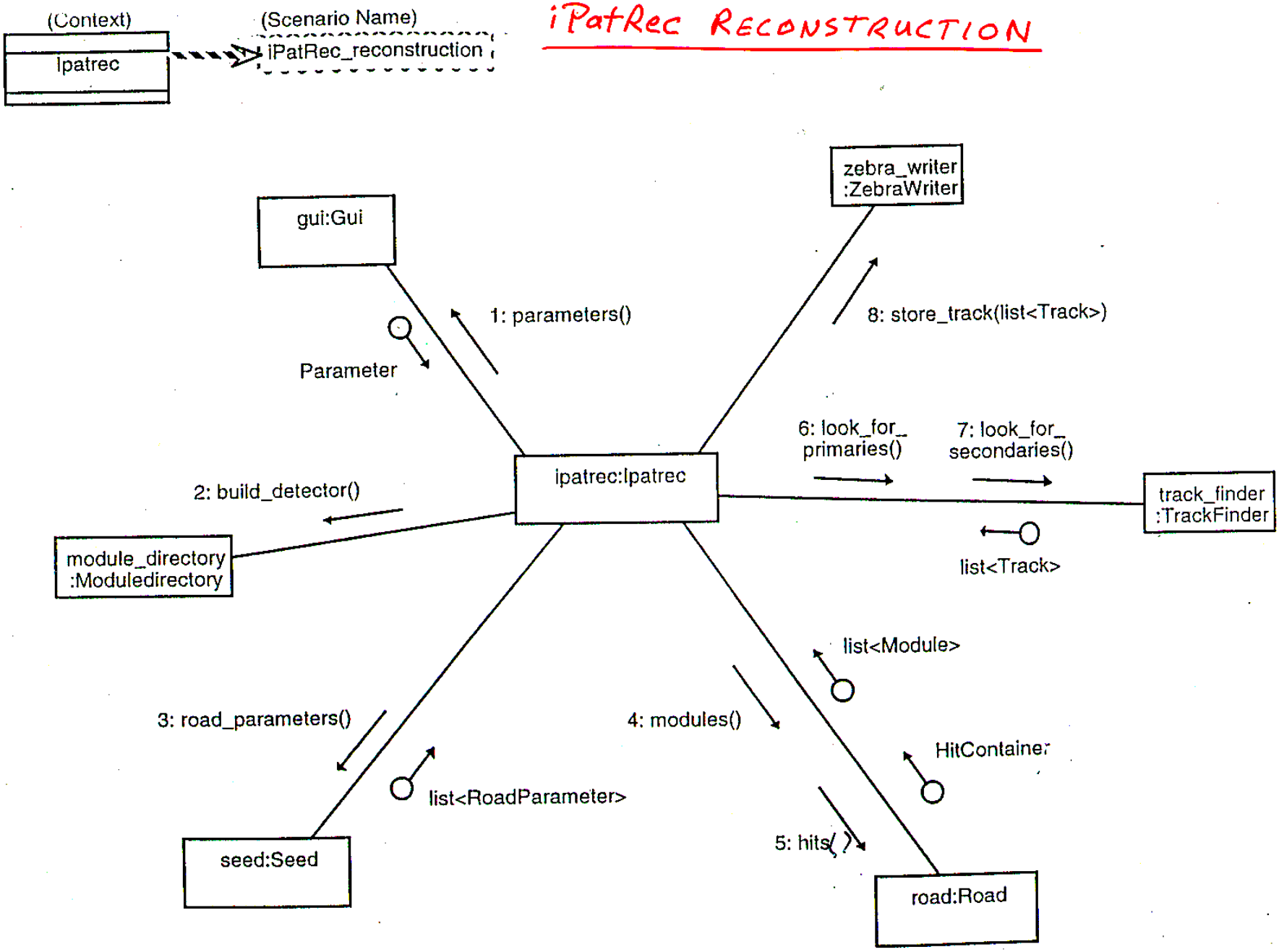
Satisfied by Class ModuleDirectory

iPatRec System Properties

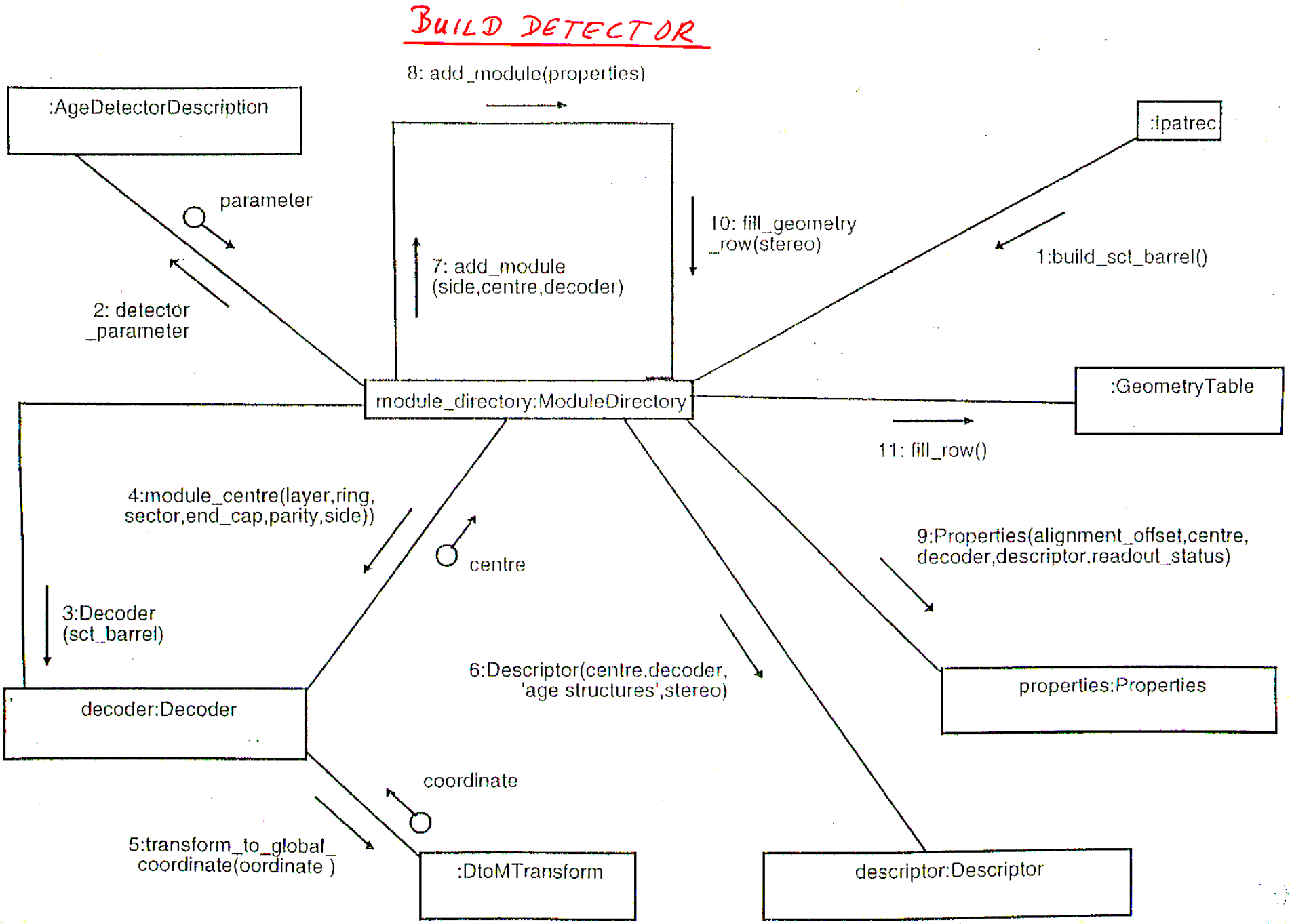
2.4.12 Minimum bias background - IPATREC shall be able to find tracks in the presence of a realistic rate of minimum bias background events. This rate must correspond to a pp luminosity of at least 10**34 cm-2 sec-1. SATISFIED. The current version of iPatRec operates in the presence of pile up consistent with the above luminosity, without problems. The speed is reduced by a factor ~3, and the track finding efficiency by < 5%.

2.4.13 Maximum time - IPATREC shall be able to find tracks within a reasonable time per track. This is obviously a difficult number to specify, but it does imply that without sacrificing accuracy, every effort must be made to make the program run as fast as possible. Many factors enter into the equation including event type, road size, and luminosity. For the code developed to date, maximisation of the speed of operation has been the major objective.

iPatRec Reconstruction



BUILD DETECTOR



MODULEDIRECTORY

4.2.5.1 ModuleDirectory(void):

4.2.5.2 build_pixel_barrel(void): void

4.2.5.3 build_pixel_endcap(void): void

4.2.5.4 build_sct_barrel(void): void

4.2.5.5 build_sct_endcap(void): void

4.2.5.6 clear_decoder_map(void): void

4.2.5.7 find_module(int, int): int

4.2.5.8 find_module(int, Coordinate): int

4.2.5.9 module_index(const Descriptor*, int): int

4.2.5.10 operator[](int): Properties &

4.2.5.11 print_size(void): void

4.2.5.12 set_decoder_map(void): void

4.2.5.13 size(void): int

4.2.5.14 ~ModuleDirectory(void):

DECODER / PROPERTIES

4.2.1 Decoder

implicit copy constructor implicit destructor implicit assignment operator

Public operations:

4.2.1.1 Decoder(int):

4.2.1.2 clear_map(void): void

clear map ready for next event

4.2.1.3 decode_next_digit(const ModuleDirectory::Properties &): bool

4.2.1.4 get_key(void): int

generate key to map from current level

4.2.1.5 module_centre(int, int, int, int, int, int): CylindricalCoordinate

4.2.1.6 module_limit(Coordinate): CylindricalCoordinate

4.2.1.7 number_of_end_caps(void): int

4.2.1.8 number_of_module_sides(void): int

4.2.1.9 number_of_sector_parities(void): int

4.2.1.10 print_level(void): void

4.2.1.11 set_map(void): void

map digi banks for fast decoding

4.2.1.12 valid_volume(int, int, int, int, int, int): bool

make decoded quantities public ...

4.2.2 Properties

Public operations:

4.2.2.1 Properties(Coordinate, CylindricalCoordinate, int, const Descriptor*

(PROPERTIES)

4.2.2.2 alignment_offset(void): Coordinate

4.2.2.3 change_decoder_key(int): int

4.2.2.4 check_acceptance(Coordinate,double): HitStatus

4.2.2.5 cos_stereo(void): double

4.2.2.6 cos_tilt(void): double

4.2.2.7 decoder_key(void): int

4.2.2.8 descriptor(void): Descriptor *

4.2.2.9 length(void): double

4.2.2.10 phi_centre(void): double

4.2.2.11 phi_max(void): double

4.2.2.12 phi_min(void): double

4.2.2.13 pitch(void): double

4.2.2.14 r_centre(void): double

4.2.2.15 r_max(void): double

4.2.2.16 r_min(void): double

4.2.2.17 radiation_thickness(void): double

4.2.2.18 readout_status(void): unsigned int

4.2.2.19 sin_stereo(void): double

4.2.2.20 thickness(void): double

4.2.2.21 tilt_angle(void): double

4.2.2.22 z_max(void): double

4.2.2.23 z_min(void): double

4.2.2.24 ~Properties(void):

DESCRIPTOR

4.1.1 Descriptor

Abstract.

implicit copy constructor implicit assignment operator decoding stuff ...

Public operations:

4.1.1.1 Descriptor(void):
~Descriptor (void);

4.1.1.2 acceptance_status(unsigned int, CylindricalCoordinate, double): HitS

4.1.1.3 alignment_specification_phi(void): double

4.1.1.4 alignment_specification_r(void): double

4.1.1.5 alignment_specification_rphi(void): double

4.1.1.6 alignment_specification_z(void): double
intercept with a track ...

4.1.1.7 cos_stereo(void): double

4.1.1.8 cos_tilt(void): double

4.1.1.9 decoder(void): Decoder *

4.1.1.10 delta_phi_minus(void): double

4.1.1.11 delta_phi_plus(void): double

4.1.1.12 delta_r_minus(void): double

4.1.1.13 delta_r_plus(void): double

4.1.1.14 delta_z_minus(void): double
degradadation and alignment specifications ...

DESCRIPTOR

4.1.1 Descriptor

Abstract.

implicit copy constructor implicit assignment operator decoding stuff ...

Public operations:

4.1.1.1 Descriptor(void):
~Descriptor (void);

4.1.1.2 acceptance_status(unsigned int, CylindricalCoordinate, double): HitS

4.1.1.3 alignment_specification_phi(void): double

4.1.1.4 alignment_specification_r(void): double

4.1.1.5 alignment_specification_rphi(void): double

4.1.1.6 alignment_specification_z(void): double
intercept with a track ...

4.1.1.7 cos_stereo(void): double

4.1.1.8 cos_tilt(void): double

4.1.1.9 decoder(void): Decoder *

4.1.1.10 delta_phi_minus(void): double

4.1.1.11 delta_phi_plus(void): double

4.1.1.12 delta_r_minus(void): double

4.1.1.13 delta_r_plus(void): double

4.1.1.14 delta_z_minus(void): double
degradadation and alignment specifications ...

4.1.1.15 delta_z_plus(void): double (DESCRIPTOR)

4.1.1.16 fraction_of_dead_chips(void): double

4.1.1.17 fraction_of_dead_modules(void): double

4.1.1.18 length(void): double

4.1.1.19 local_coordinate(float*): Coordinate

4.1.1.20 local_phi_max(double, double): double

4.1.1.21 lorentz_correction(void): double

4.1.1.22 mean_radius(void): double

4.1.1.23 number_of_phi_channels(void): int

4.1.1.24 number_of_readout_chips(void): int

4.1.1.25 number_of_rz_channels(void): int

4.1.1.26 number_of_sectors(void): int

geometry table stuff ...

4.1.1.27 phi_channel(double): int

4.1.1.28 pitch(void): double

4.1.1.29 print_parameters(void): void

4.1.1.30 pulse_height(float): float

4.1.1.31 radiation_thickness(void): double

4.1.1.32 readout_chip(float*): int

4.1.1.33 readout_chip(double, double): int

4.1.1.34 rz_channel(double): int

4.1.1.35 set_limits(CylindricalCoordinate): void

4.1.1.36 sin_stereo(void): double

4.1.1.37 sin_tilt(void): double

4.1.1.38 status(unsigned int, double, double, double, double): HitStatus

4.1.1.39 thickness(void): double

4.1.1.40 tilt_angle(void): double

DESCRIPTORS

4.1.3 SctBarrelDescriptor

Inherits from Descriptor

Public operations:

4.1.3.1 SctBarrelDescriptor(CylindricalCoordinate, const Decoder*, SCTG&, SCTI&, SDRP&, SIDO&, float):

4.1.3.2 acceptance_status(unsigned int, CylindricalCoordinate, double): HitStatus

4.1.3.3 local_coordinate(float*): Coordinate

intercept with a track ...

4.1.3.4 print_parameters(void): void

4.1.4 PixelEndcapDescriptor

Inherits from Descriptor

Public operations:

4.1.4.1 PixelEndcapDescriptor(CylindricalCoordinate, const Decoder*, PXEI&, PEDI&):

4.1.4.2 acceptance_status(unsigned int, CylindricalCoordinate, double): HitStatus

4.1.4.3 local_coordinate(float*): Coordinate

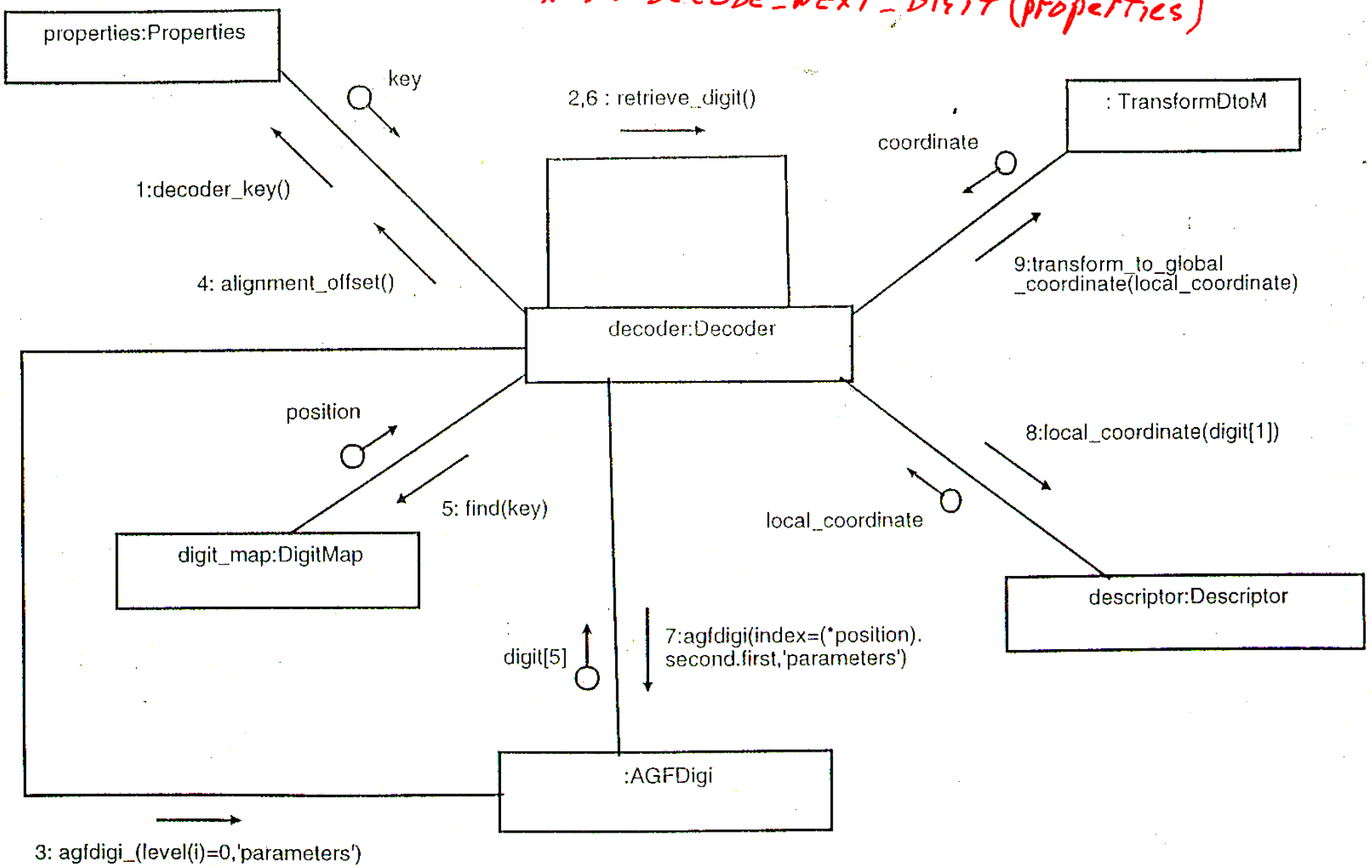
intercept with a track ...

4.1.4.4 local_phi_max(double, double): double

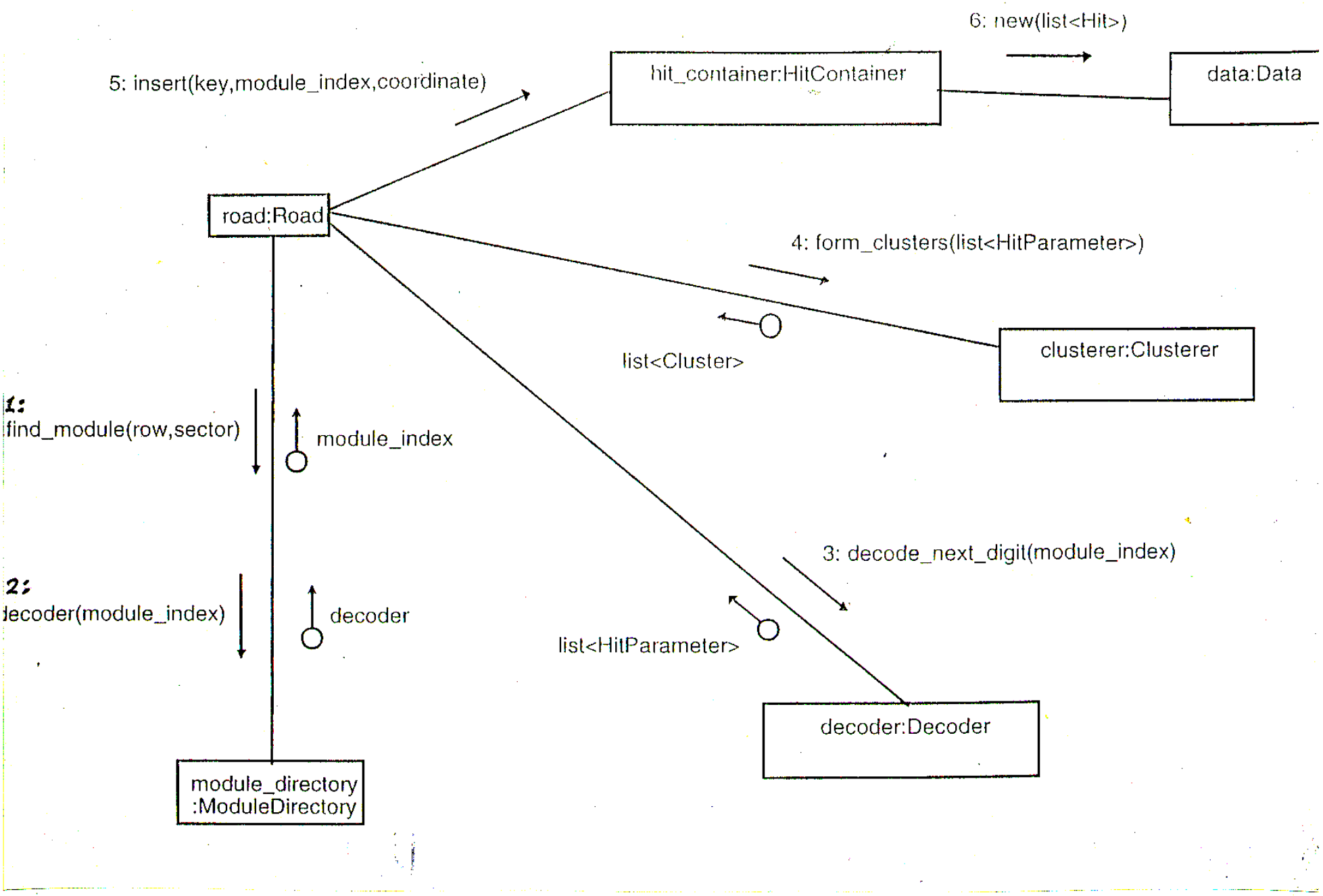
4.1.4.5 print_parameters(void): void

GET HITS IN MODULE

METHOD: DECODE_NEXT_DIGIT(properties)

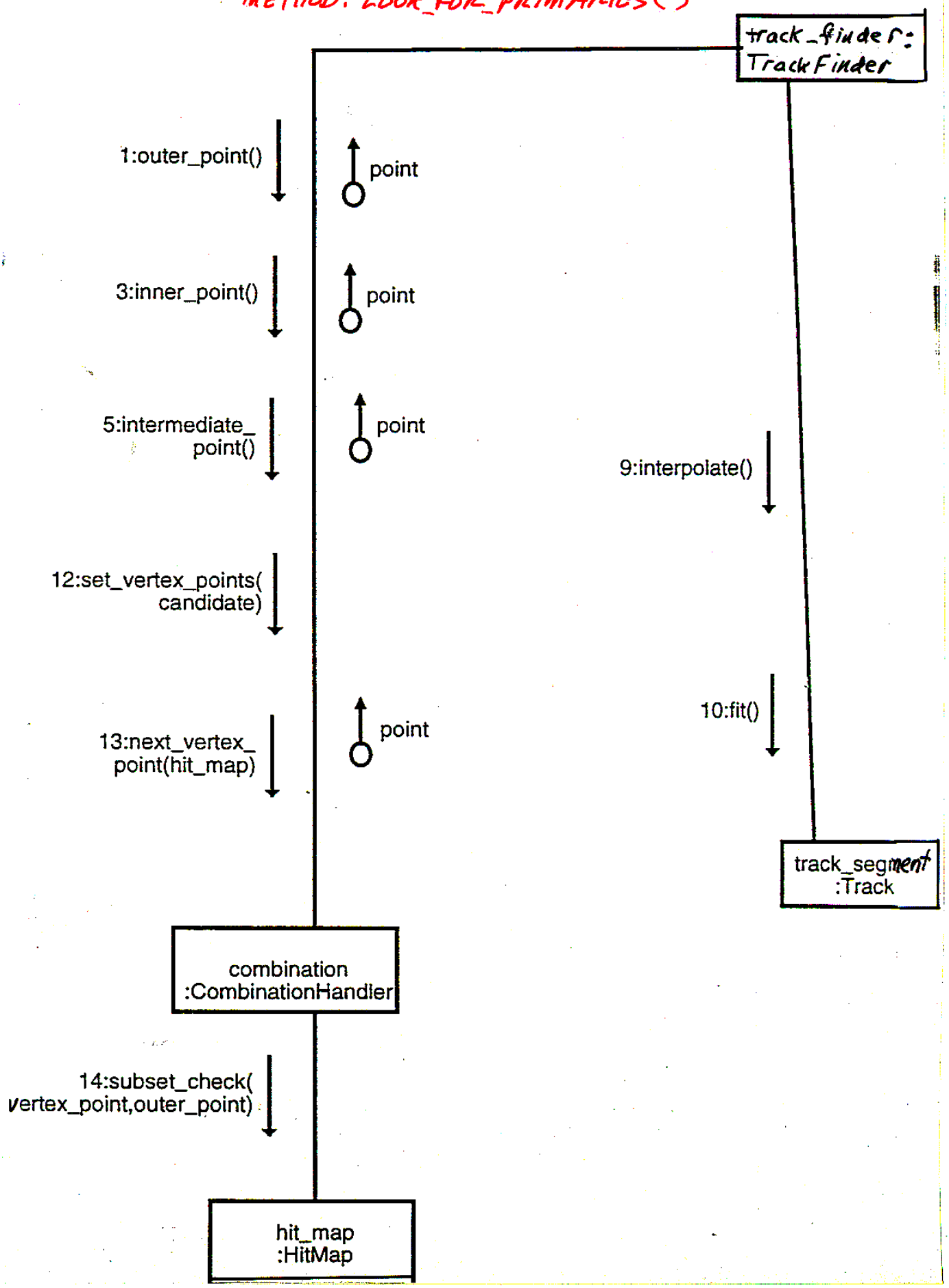


HITS IN ROAD


6: new(list<Hit>)
hit_container:HitContainer
data:Data
5: insert(key,module_index,coordinate)
road:Road
4: form_clusters(list<HitParameter>)
list<Cluster>
clusterer:Clusterer
1:
find_module(row,sector)
module_index
3: decode_next_digit(module_index)
2:
decoder(module_index)
decoder
list<HitParameter>
decoder:Decoder
module_directory
:ModuleDirectory

PRIMARY TRACK FINDER (7)

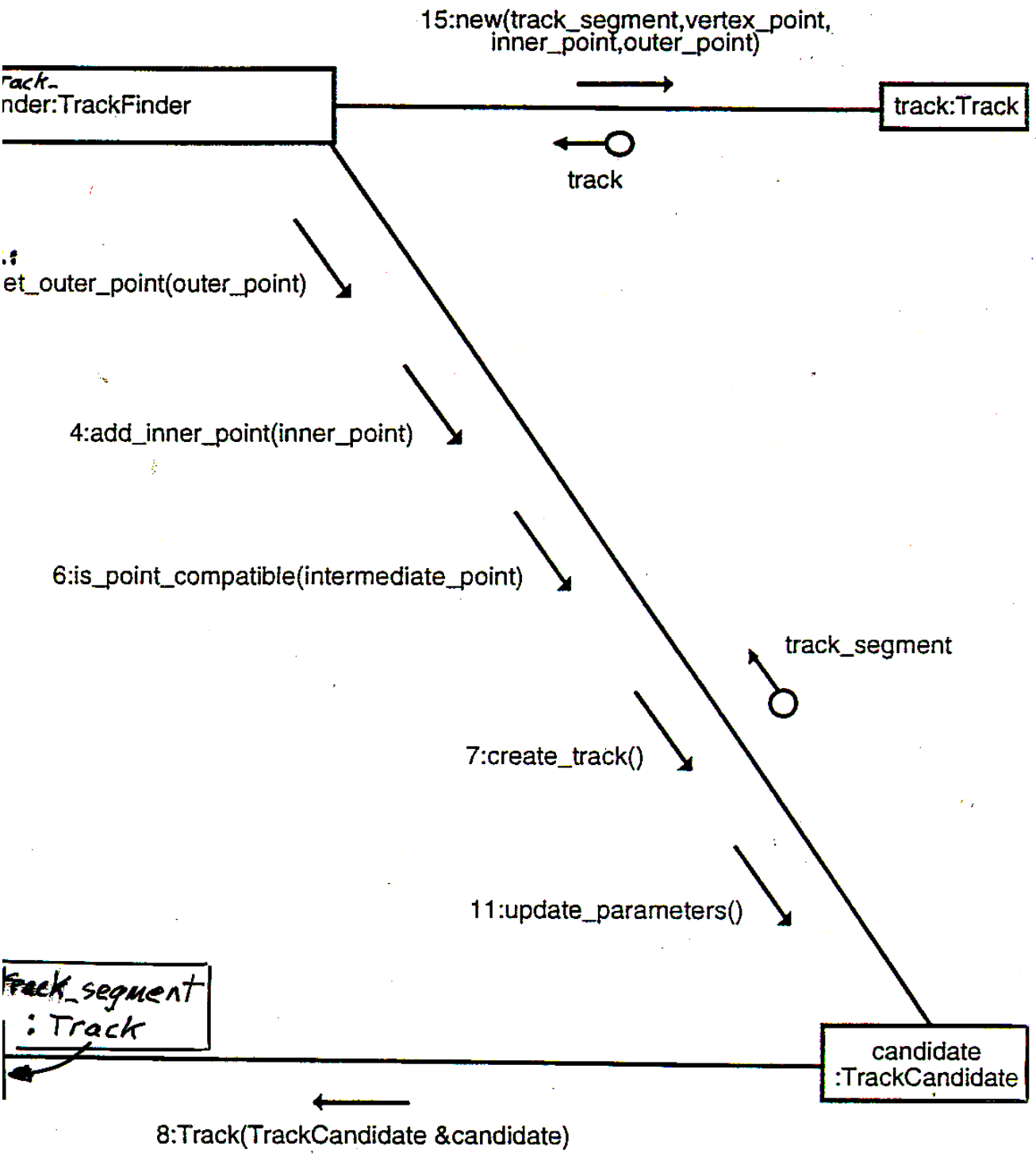
METHOD: LOOK_FOR_PRIMARIES( )



nder_1

PRIMARY TRACK FINDER (1)

METHOD: LOOK_FOR_PRIMARIES( )



COMBINATION HANDLER

4.2.11.1 CombinationHandler(void):

4.2.11.2 first_module(void): int

4.2.11.3 inner_point(void): SpacePoint *

4.2.11.4 intermediate_point(void): SpacePoint *

4.2.11.5 last_module(void): int

4.2.11.6 module_range(void): pair<int,int>

4.2.11.7 next_inner_point(HitMap*): bool

4.2.11.8 next_intermediate_point(void): bool

4.2.11.9 next_outer_point(HitMap*): bool

4.2.11.10 next_point_in_layer(int): bool

4.2.11.11 next_vertex_point(HitMap*): bool

4.2.11.12 outer_point(void): SpacePoint *

4.2.11.13 point_in_layer(void): SpacePoint *

4.2.11.14 set_vertex_points(const TrackCandidate &): void

4.2.11.15 vertex_point(void): SpacePoint *

4.2.11.16 ~CombinationHandler(void):

TRACK

4.2.12.1 Track(void):

4.2.12.2 Track(int):

4.2.12.3 Track(const Track &):

4.2.12.4 Track(const Track &, TrackStatus):

4.2.12.5 Track(const TrackCandidate &):

4.2.12.6 Track(const Track &, SpacePoint*, SpacePoint*, SpacePoint*):

4.2.12.7 associate_trt(void): bool

4.2.12.8 chi_squared(void): float

4.2.12.9 extrapolate_inwards(void): void

4.2.12.10 extrapolate_outwards(void): void

4.2.12.11 extrapolate_outwards2(void): bool

4.2.12.12 final_fit(void): int

4.2.12.13 fit(void): void

4.2.12.14 fit_probability(void): float

4.2.12.15 fit_with_rejection(void): bool

4.2.12.16 good_quality(void): bool

4.2.12.17 hit_statistics(void): HitStatistics &

4.2.12.18 include_trt_hits(void): void

4.2.12.19 interpolate(void): bool

4.2.12.20 interpolate_from_beam_spot(void): bool

4.2.12.21 is_better_than(const Track &): bool

(TRACK)

4.2.12.21 is_better_than(const Track &): bool

4.2.12.22 kine_reference(void): int

4.2.12.23 list_of_hits(void): list<HitOnTrack*>

4.2.12.24 missing_super_layers(void): int

4.2.12.25 momentum(void): float

4.2.12.26 number_dead(void): int

4.2.12.27 number_drift_hits(void): int

4.2.12.28 number_pixel_clusters(void): int

4.2.12.29 number_planar_clusters(void): int

4.2.12.30 number_planar_holes(void): int

4.2.12.31 number_superlayers(void): int

4.2.12.32 print(void): void

4.2.12.33 print_hits(void): void

4.2.12.34 print_parameters(void): void

4.2.12.35 reject_worst_hit(void): void

4.2.12.36 retrieve(void): void

4.2.12.37 seed_fit(void): int

(TRACK)

4.2.12.38 set_fit_parameters(void): void

4.2.12.39 set_hits_used(void): void

4.2.12.40 status(void): TrackStatus

4.2.12.41 vertex_fit(void): int

4.2.12.42 worst_hit_chi_squared(void): float

4.2.12.43 ~Track(void):

4.2.13 Trackfitter

Public operations:

HITMAP

4.2.14 HitMap

Public operations:

4.2.14.1 HitMap(void):

4.2.14.2 include_track(Track*): void

4.2.14.3 print(void): void

4.2.14.4 remove_track(Track*): void

4.2.14.5 shared_to_fortran(void): void

4.2.14.6 shares_final_hits(Track*): Track *

4.2.14.7 subset_check(SpacePoint*, SpacePoint*): bool

4.2.14.8 uniqueness_check(SpacePoint*): bool

TRACKCANDIDATE

4.2.33.1 TrackCandidate(void):

4.2.33.2 add_inner_point(SpacePoint*): bool

4.2.33.3 cos_phi0(void): double

4.2.33.4 create_track(void): Track *

4.2.33.5 fill_histogram(void): void

4.2.33.6 inner_point(void): SpacePoint *

4.2.33.7 inverse_pt(void): double

4.2.33.8 is_point_compatible(SpacePoint*): bool

4.2.33.9 is_point_compatible_hist(SpacePoint*): bool

4.2.33.10 outer_point(void): SpacePoint *

4.2.33.11 print(void): void

4.2.33.12 print_outer_point(void): void

4.2.33.13 set_outer_point(SpacePoint*): void

4.2.33.14 sin_phi0(void): double

4.2.33.15 update_parameters(void): void

4.2.33.16 z0(void): double

4.2.33.17 z_slope(void): double

4.2.33.18 ~TrackCandidate(void):

HITONTRACK

4.2.18 HitOnTrack

Public operations:

4.2.18.1 HitOnTrack(void):

4.2.18.2 HitOnTrack(int, int):

4.2.18.3 HitOnTrack(int, int, HitStatus):

4.2.18.4 HitOnTrack(int, int, int):

4.2.18.5 above_tr_threshold(void): bool

4.2.18.6 geometry_row(void): int

4.2.18.7 in_b_layer(void): bool

4.2.18.8 index(void): int

4.2.18.9 is_planar(void): bool

4.2.18.10 is_shared(void): bool

4.2.18.11 kine_reference(void): int

4.2.18.12 print(void): void

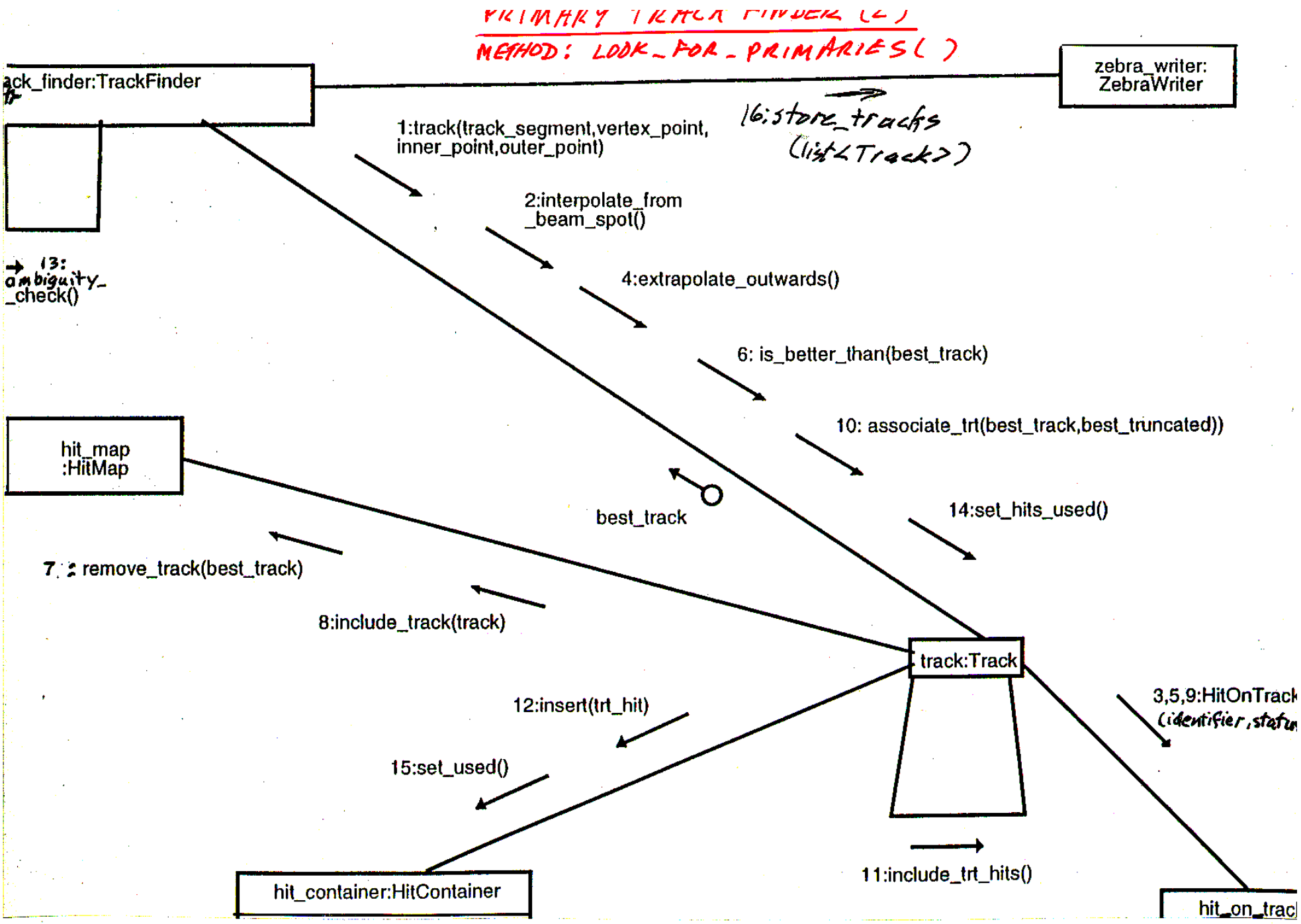
4.2.18.13 refresh(int, int): void

4.2.18.14 refresh(int, int, int): void

4.2.18.15 reject(void): void

4.2.18.16 set_used(void): void

4.2.18.17 status(void): HitStatus


PRIMARY TRACK FINDER (2)
METHOD: LOOK_FOR_PRIMARIES( )
ack_finder:TrackFinder
zebra_writer:
ZebraWriter
16:store_tracks
(list<Track>)
1:track(track_segment,vertex_point,
inner_point,outer_point)
2:interpolate_from
_beam_spot()
4:extrapolate_outwards()
13:
ambiguity_
_check()
6: is_better_than(best_track)
10: associate_trt(best_track,best_truncated))
hit_map
:HitMap
best_track
14:set_hits_used()
7: remove_track(best_track)
8:include_track(track)
track:Track
12:insert(trt_hit)
3,5,9:HitOnTrack
(identifier,statu
15:set_used()
11:include_trt_hits()
hit_container:HitContainer
hit_on_trac

4.2.16 HitContainer

HIT CONTAINER

Public operations:

4.2.16.1 HitContainer(void):

4.2.16.2 flush(void): void

4.2.16.3 flush_discrete(void): void

4.2.16.4 insert(int,int,double,double,double,int,int,int,int,
double,double,double,bool,bool): void

4.2.16.5 insert(int,double,double,double,double,int,double,double,double,int): void

4.2.16.6 operator[](int): Data &

4.2.16.7 print_size(void): void

DATA

4.2.34.1 Data(Cluster,Coordinate,int,double,double,double,HitType):

4.2.34.2 Data(Coordinate,double,int,double,double,double,HitType):

4.2.34.3 above_tr_threshold(void): bool

4.2.34.4 cluster(void): Cluster

4.2.34.5 coordinate(void): Coordinate

4.2.34.6 cos_stereo(void): double

4.2.34.7 cos_tilt(void): double

4.2.34.8 drift_distance(void): double

4.2.34.9 in_b_layer(void): bool

4.2.34.10 is_drift(void): bool

4.2.34.11 is_pixel(void): bool

4.2.34.12 is_planar(void): bool

4.2.34.13 is_shared(void): bool

4.2.34.14 is_strip(void): bool

4.2.34.15 kine_reference(void): int

4.2.34.16 set_used(void): void

4.2.34.17 sigma(void): double

4.2.34.18 sin_stereo(void): double

4.2.34.19 sin_tilt(void): double